
User Manual

BBEditTM

Professional HTML and Text Editor for the Macintosh

Bare Bones Software, Inc.

BBEdit™ 10.1

| | |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Product Design | <i>Jim Correia, Steve Kalkwarf, Rich Siegel, Patrick Woolsey</i> |
| Product Engineering | <i>Jim Correia, Seth Dillingham, Jon Hueras, Steve Kalkwarf, Rich Siegel, Steve Sisak</i> |
| Engineers Emeritus | <i>Chris Borton, Tom Emerson, Pete Gontier, Jamie McCarthy, John Norstad, Jon Pugh, Mark Romano, Eric Slosser, Rob Vaterlaus</i> |
| Documentation | <i>Philip Borenstein, Stephen Chernicoff, John Gruber, Simon Jester, Jeff Mattson, Jerry Kindall, Caroline Rose, Rich Siegel, Patrick Woolsey</i> |
| Additional Engineering | <i>Polaschek Computing http://www.polaschek-computing.com/</i> |
| Icon Design | <i>Byran Bell http://www.bryanbell.com/</i> |
| Packaging Design | <i>Ultra Maroon Design http://www.ultramaron.com/</i> |
| PHP keyword lists | <i>contributed by Ted Stresen-Reuter http://www.tedmasterweb.com/</i> |
| Exuberant ctags | <i>© 1996-2004 Darren Hiebert http://ctags.sourceforge.net/</i> |
| HTML Tidy Technology | <i>© 1998-2006 World Wide Web Consortium http://tidy.sourceforge.net/</i> |
| LibNcFTP | <i>© 1996-2010 Mike Gleason & NcFTP Software</i> |
| PCRE Library Package | <i>written by Philip Hazel and © 1997-2004 University of Cambridge, England</i> |

Bare Bones Software, Inc.

P. O. Box 1048

Bedford, MA 01730-01048

(978) 251-0500 main

(978) 251-0525 fax

<http://www.barebones.com/>

Sales & customer service: **sales@barebones.com**

Technical support: **support@barebones.com**

BBEdit and “It Doesn’t Suck” are registered trademarks of Bare Bones Software, Inc.

Exuberant Ctags is included under the terms of the GNU General Public License (GPL). Source is available within BBEEdit’s application package, or from the Bare Bones Software web site:

<http://www.barebones.com/support/develop/ctags.shtml>

Information in this document is subject to change without notice and does not represent a commitment on the part of the copyright holder. The software described in this document is furnished under a license agreement. Warranty and license information is included in printed form with the CD-ROM package or in electronic form for downloaded products, and is presented on the next page of this user manual.

The owner or authorized user of a valid copy of BBEEdit may reproduce this publication for the purpose of learning to use such software. No part of this publication may be reproduced or transmitted for commercial purposes, such as selling copies of this publication or for providing paid for support services.

Macintosh, Mac OS, Mac OS X, Power Macintosh, and AppleScript are trademarks of Apple, Inc. Intel is a registered trademark of Intel Corporation. All other trademarks are the property of their respective owners.

License Agreement:

You, the Licensee, assume responsibility for the selection of the program BBEdit to achieve your intended results, and for the installation, use, and results obtained from the program. Breaking the package seal and installing the program, or downloading and installing the program, constitutes your acceptance of these terms and conditions. If you do not accept these terms and conditions, then do not download and install the program or break the package seal and install the program, and contact Bare Bones Software, Inc., or return the package for a full refund.

License:

You may use the program and documentation on not more than two (2) machines and copy the program and documentation into any machine-readable or printed form for backup or support of your use of the program and documentation on either or both machines, provided that no copy of the program and documentation may be used by anyone other than you. You may not use or copy the program or documentation, or any copy thereof, in whole or in part, except as provided in this Agreement. You also may not modify or transfer (whether or not for consideration) the program or documentation, or any copy thereof, in whole or in part. If you use, copy, modify, or transfer the program or documentation, or any copy thereof, in whole or in part, except as expressly provided for in this agreement, your license is automatically terminated.

Special Provisions Applicable Only to Educational Users:

Bare Bones Software, Inc. offers the program and documentation at a discounted price to licensees that are "Educational Users". By downloading the Educational User program and documentation or purchasing the Educational User program package, you have represented to us that at the time of purchase you are a not-for-profit educational institution that is accredited by a nationally recognized accrediting agency, a full- or part-time student currently enrolled at such an educational institution, or a current member of the faculty or staff of such an educational institution (each, an "Educational User"). In reliance on this representation Bare Bones Software, Inc., has charged you fees that have been discounted from its standard fee. If your representation proves to have been untrue, Bare Bones Software, Inc. reserves the right to charge you the difference or to terminate your license, as well as any and all other rights and remedies to which it is entitled.

As an Educational User, your use of the program and documentation is limited solely to teaching, research, and other noncommercial activities consistent with your academic, teaching and research activities. Without limiting the generality of the foregoing, you may not use the program for the purpose of conducting, or in connection with, any commercial activity for your benefit or that of the educational institution. If your use of the program changes to that of commercial activity, Bare Bones Software, Inc. reserves the right to charge you the difference or to terminate your license, as well as any and all other rights and remedies to which it is entitled.

Term:

The license is effective on the date you accept this Agreement, and remains in effect until terminated as indicated above or until you terminate it. If the license is terminated for any reason, you agree to destroy the program and documentation, together with all copies thereof, in whole or in part, in any form, and to cease all use of the program and documentation.

Limited Warranty and Limitation of Remedies:

The program, documentation and any support from Bare Bones Software, Inc., are provided "as is" and without warranty, express and implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose. In no event will Bare Bones Software, Inc. be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages, even if Bare Bones Software, Inc. is advised of the possibility of such damages, or for any claim by you or any third party.

General Terms:

This Agreement can only be modified by a written agreement signed by you and Bare Bones Software, Inc. and changes from the terms and conditions of this Agreement made in any other manner will be of no effect. If any portion of this Agreement shall be held invalid, illegal, or unenforceable, the validity, legality, and enforceability of the remainder of the Agreement shall not in any way be affected or impaired thereby. This Agreement shall be governed by the laws of The Commonwealth of Massachusetts, without giving effect to conflict of laws provisions thereof. As required by United States export regulations, you shall not permit export of the program or any direct products thereof to any country to which export is then controlled by the United States Bureau of Export Administration, unless you have that agency's prior written approval. Use of the program and documentation by military and civilian offices, branches or agencies of the U.S. Government is restricted in accordance with the applicable Federal Acquisition Regulations (under which the program and documentation constitute "restricted computer software" that is "commercial computer software") or Department of Defense Federal Acquisition Regulations Supplement (under which the program and documentation constitute "commercial computer software" and "commercial computer software documentation") to that consistent with only those rights as are granted pursuant to the terms and conditions hereof.

Acknowledgment:

You acknowledge that you have read this agreement, understand it, and agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between you and Bare Bones Software, Inc. which supersedes all proposals or prior agreements, oral or written, and all other communications between you and Bare Bones Software, Inc. relating to the subject matter of this agreement.

Contents

| | | |
|-----------|---------------------------------------------|----|
| Chapter 1 | Welcome to BBEdit | 19 |
| | Getting Started | 19 |
| | What Is BBEdit? | 19 |
| | How Can I Use BBEdit? | 20 |
| | <i>Development Environments</i> | 20 |
| | <i>Writing HTML Documents</i> | 20 |
| | Human Interface Notes | 21 |
| | <i>Dynamic Menus</i> | 21 |
| | <i>Bypassing Options Dialogs</i> | 21 |
| | <i>Keyboard Shortcuts for Commands</i> | 21 |
| | <i>Contextual Menus</i> | 22 |
| | <i>Snappy Palettes</i> | 22 |
| | <i>Dialog Box and Sheet Key Equivalents</i> | 22 |
| | Feature Highlights | 23 |
| | <i>Info on New Features</i> | 23 |
| | Discussion Group | 23 |
| | Support Services | 24 |
| | <i>How to contact us</i> | 24 |
| Chapter 2 | Installing BBEdit | 25 |
| | Basic Installation | 25 |
| | <i>System Requirements</i> | 25 |
| | <i>Installing BBEdit</i> | 25 |
| | <i>Checking for Updates</i> | 26 |
| | <i>Upgrading from a Previous Version</i> | 26 |
| | <i>First Run Configuration</i> | 27 |
| | <i>Activating the Demo</i> | 28 |

| | |
|----------------------------------------------------------------------|----|
| BBEdit's Application Support Folders | 29 |
| <i>Using a Global Application Support Folder</i> – 29 | |
| <i>Using a Local Application Support Folder</i> – 29 | |
| <i>Application Support Folder Contents</i> – 29 | |
| <i>Attachment Scripts</i> – 30 | |
| <i>Auto-Save Recovery</i> – 30 | |
| <i>Clippings</i> – 30 | |
| <i>Color Schemes</i> – 30 | |
| <i>Completion Data</i> – 30 | |
| <i>HTML Templates</i> – 31 | |
| <i>Language Modules</i> – 31 | |
| <i>Menu Scripts</i> – 31 | |
| <i>Packages</i> – 31 | |
| <i>Plug-Ins</i> – 32 | |
| <i>Readme.txt [file]</i> – 32 | |
| <i>Scratchpad [file]</i> – 32 | |
| <i>Scripts</i> – 32 | |
| <i>Setup</i> – 33 | |
| <i>Shutdown Items</i> – 33 | |
| <i>Startup Items</i> – 34 | |
| <i>Stationery</i> – 34 | |
| <i>Text Filters</i> – 34 | |
| <i>Unix Worksheet.worksheet</i> – 35 | |
| <i>Web Site Projects</i> – 35 | |
| <i>Superseded App Support Folders</i> – 35 | |
| Preference Files and Folders | 35 |
| <i>BBEdit Preferences File</i> – 35 | |
| <i>BBEdit Preferences Folder</i> – 36 | |
| Sharing Application Support & Preferences Data via Dropbox | 36 |

Chapter 3 Working with Files 39

| | |
|----------------------------------------------------|----|
| Launching BBEEdit | 40 |
| <i>Startup Items</i> – 40 | |
| Creating and Saving Documents | 41 |
| <i>Saving a Copy of a File</i> – 43 | |
| <i>File Saving Options</i> – 43 | |
| <i>File State</i> – 44 | |
| <i>Emacs Local Variables</i> – 44 | |
| <i>Saving with Authentication</i> – 45 | |
| <i>Saving Compressed Files as bz2 or gzip</i> – 45 | |
| <i>Saving as Styled Text or HTML</i> – 45 | |
| Crash Auto-Recovery | 45 |

| | |
|----------------------------------------------------------------|----|
| Opening Existing Documents | 46 |
| <i>Choosing the Encoding for a Document</i> – 46 | |
| <i>Using the Open Command</i> – 48 | |
| <i>Reload from Disk</i> – 49 | |
| <i>Opening and Editing Files within Zip Archives</i> – 49 | |
| <i>Opening bz2, gzip, and tar Files and Binary plists</i> – 50 | |
| <i>Opening Hidden Files</i> – 50 | |
| <i>Using the Open from FTP/SFTP Server Command</i> – 50 | |
| <i>Using the Open Selection Command</i> – 50 | |
| <i>Using the Open File by Name Commands</i> – 51 | |
| <i>Using the Open Counterpart Command</i> – 52 | |
| <i>Using the Open Recent Command</i> – 53 | |
| <i>Using the Reopen using Encoding Command</i> – 53 | |
| Quitting BBEdit | 53 |
| An International Text Primer | 53 |
| <i>International Text in BBEdit</i> – 54 | |
| <i>Unicode</i> – 54 | |
| <i>Saving Unicode Files</i> – 54 | |
| <i>Opening Unicode Files</i> – 55 | |
| Accessing FTP/SFTP Servers | 56 |
| <i>Opening Files from FTP/SFTP Servers</i> – 56 | |
| <i>Saving Files to FTP/SFTP Servers</i> – 59 | |
| Using BBEdit from the Command Line | 61 |
| Using Projects | 62 |
| <i>Creating a Project</i> – 62 | |
| <i>Project Commands</i> – 63 | |
| <i>Using projects</i> – 64 | |
| <i>Removing Files from a Project</i> – 65 | |
| <i>Contextual Menu Commands</i> – 65 | |
| <i>Script Access to Project Contents</i> – 65 | |
| Using Stationery | 65 |
| Hex Dump for Files and Documents | 66 |
| Making Backups | 66 |
| Printing | 66 |
| <i>Text Printing Options</i> – 67 | |

Chapter 4 Editing Text with BBEdit 69

| | |
|------------------------------------------------|----|
| Basic Editing | 70 |
| <i>Moving Text</i> – 71 | |
| <i>Multiple Clipboards</i> – 71 | |
| <i>Drag and Drop</i> – 72 | |
| Multiple Undo | 72 |
| Window Anatomy | 73 |
| <i>The Toolbar</i> – 73 | |
| <i>The Split Bar</i> – 74 | |
| <i>The Navigation Bar</i> – 75 | |
| <i>The File List</i> – 79 | |
| <i>The Status Bar</i> – 80 | |
| <i>The Gutter and Folded Text Regions</i> – 81 | |

| | |
|--------------------------------------------------|-----|
| The View Menu | 83 |
| <i>Text Display</i> – 83 | |
| <i>Show/Hide Toolbar</i> – 84 | |
| <i>Show/Hide Navigation Bar</i> – 84 | |
| <i>Show/Hide Editor</i> – 84 | |
| <i>Show/Hide Files</i> – 84 | |
| <i>Show/Hide Open Documents</i> – 84 | |
| <i>Show/Hide Recent Documents</i> – 84 | |
| <i>Show/Hide Worksheet & Scratchpad</i> – 84 | |
| <i>Balance</i> – 84 | |
| <i>Balance & Fold</i> – 84 | |
| <i>Fold Selection</i> – 85 | |
| <i>Unfold Selection</i> – 85 | |
| <i>Collapse Enclosing Fold</i> – 85 | |
| <i>Collapse All Folds</i> – 85 | |
| <i>Expand All Folds</i> – 85 | |
| <i>Previous Document/Next Document</i> – 85 | |
| <i>Move to New Window</i> – 85 | |
| <i>Open in Additional Window</i> – 85 | |
| <i>Reveal in Finder</i> – 85 | |
| <i>Reveal in Project List</i> – 86 | |
| <i>Go Here in Terminal</i> – 86 | |
| <i>Go Here in Disk Browser</i> – 86 | |
| Cursor Movement and Text Selection | 86 |
| <i>Clicking and Dragging</i> – 86 | |
| <i>Arrow Keys</i> – 87 | |
| <i>CamelCase Navigation</i> – 87 | |
| <i>Rectangular Selections</i> – 87 | |
| <i>Working with Rectangular Selections</i> – 88 | |
| <i>Scrolling the View</i> – 90 | |
| <i>The Delete Key</i> – 91 | |
| <i>The Numeric Keypad</i> – 91 | |
| <i>Go To Line Command</i> – 92 | |
| <i>Function Keys</i> – 92 | |
| <i>Resolving URLs</i> – 92 | |
| Text Completion | 93 |
| Text Options | 93 |
| <i>Editing Options</i> – 94 | |
| <i>Display Options</i> – 95 | |
| How BBEEdit Wraps Text | 97 |
| <i>Soft Wrapping</i> – 98 | |
| <i>Hard Wrapping</i> – 99 | |
| The Insert Submenu | 101 |
| <i>Inserting File Contents</i> – 101 | |
| <i>Inserting File & Folder Paths</i> – 102 | |
| <i>Inserting a Folder Listing</i> – 102 | |
| <i>Inserting a Page Break</i> – 102 | |
| <i>Inserting Time Stamps</i> – 102 | |
| <i>Inserting an Emacs Variable Block</i> – 102 | |
| Comparing Text Files | 102 |
| <i>Compare Against Disk File</i> – 104 | |
| <i>Multi-File Compare Options</i> – 105 | |

| | |
|-----------------------------------------|-----|
| Using Markers | 106 |
| <i>Setting Markers</i> – 106 | |
| <i>Clearing Markers</i> – 106 | |
| <i>Using Grep to Set Markers</i> – 107 | |
| Spell Checking Documents | 107 |
| <i>Check Spelling As You Type</i> – 107 | |
| <i>Manual Spell Checking</i> – 108 | |
| <i>The Spelling Panel</i> – 108 | |
| – 109 | |

Chapter 5 Text Transformations 111

| | |
|--------------------------------------------------------|-----|
| Text Menu Commands | 111 |
| <i>Apply Text Filter</i> – 112 | |
| <i>Apply Text Filter <last filter></i> – 112 | |
| <i>Exchange Characters</i> – 112 | |
| <i>Change Case</i> – 112 | |
| <i>Shift Left / Shift Right</i> – 113 | |
| <i>Un/Comment Selection</i> – 113 | |
| <i>Hard Wrap</i> – 113 | |
| <i>Add Line Breaks</i> – 114 | |
| <i>Remove Line Breaks</i> – 114 | |
| <i>Convert to ASCII</i> – 114 | |
| <i>Educate Quotes</i> – 114 | |
| <i>Straighten Quotes</i> – 114 | |
| <i>Add/Remove Line Numbers</i> – 114 | |
| <i>Prefix/Suffix Lines</i> – 115 | |
| <i>Sort Lines</i> – 115 | |
| <i>Process Duplicate Lines</i> – 116 | |
| <i>Process Lines Containing</i> – 117 | |
| <i>Rewrap Quoted Text</i> – 118 | |
| <i>Increase and Decrease Quote Level</i> – 118 | |
| <i>Strip Quotes</i> – 118 | |
| <i>Zap Gremlins</i> – 119 | |
| <i>Entab</i> – 120 | |
| <i>Detab</i> – 120 | |
| <i>Normalize Line Endings</i> – 120 | |
| Text Factories | 121 |
| <i>Creating and Configuring Text Factories</i> – 121 | |
| <i>Applying Text Factories to Files</i> – 124 | |
| <i>Applying Text Factories to Open Documents</i> – 125 | |
| <i>HTML Processing Actions</i> – 125 | |
| Automator Actions | 126 |
| <i>Using BBEdit with Automator</i> – 126 | |
| <i>Available Actions</i> – 127 | |

| | |
|-------------------------------------------------|-----|
| Window Menu | 131 |
| <i>Minimize Window</i> – 131 | |
| <i>Bring All to Front</i> – 131 | |
| <i>Palettes</i> – 131 | |
| <i>Workspace</i> – 134 | |
| <i>Show Scratchpad</i> – 134 | |
| <i>Show Unix Worksheet</i> – 134 | |
| <i>Save Default <type of>Window</i> – 134 | |
| <i>Arrange</i> – 135 | |
| <i>Cycle Through Windows</i> – 135 | |
| <i>Exchange with Next</i> – 135 | |
| <i>Synchro Scrolling</i> – 135 | |
| <i>Window Names</i> – 135 | |
| <i>Zoom (key equivalent only)</i> – 135 | |

| | |
|-----------------------------------------------------|-----|
| Search Windows | 137 |
| Basic Searching and Replacing | 138 |
| <i>Search Settings</i> – 140 | |
| <i>Special Characters</i> – 140 | |
| Multi-File Searching | 141 |
| <i>Starting a Search</i> – 142 | |
| <i>Find All and Multi-File Search Results</i> – 143 | |
| <i>Specifying the Search Set</i> – 144 | |
| <i>Saved Search Sources</i> – 146 | |
| <i>Multi-File Search Options</i> – 146 | |
| <i>File Filters</i> – 147 | |
| <i>Searching SCM Directories</i> – 149 | |
| Multi-File Replacing | 149 |
| Live Search | 150 |

| | |
|--------------------------------------------------------|-----|
| Search Menu Reference | 151 |
| <i>Find</i> – 151 | |
| <i>Multi-File Search</i> – 151 | |
| <i>Search in ... (Project or Disk Browser)</i> – 151 | |
| <i>Live Search</i> – 152 | |
| <i>Find Next/Previous</i> – 152 | |
| <i>Find All</i> – 152 | |
| <i>Find Selected Text/Previous Selected Text</i> – 152 | |
| <i>Use Selection for Find</i> – 152 | |
| <i>Use Selection for Find (grep)</i> – 152 | |
| <i>Use Selection for Replace</i> – 152 | |
| <i>Use Selection for Replace (grep)</i> – 153 | |
| <i>Replace</i> – 153 | |
| <i>Replace All</i> – 153 | |
| <i>Replace to End</i> – 153 | |
| <i>Replace & Find Again</i> – 153 | |
| <i>Go to Line</i> – 153 | |
| <i>Go to Center Line</i> – 153 | |
| <i>Go to Previous/Next Error</i> – 153 | |
| <i>Go to Previous/Next Placeholder</i> – 154 | |
| <i>Go to Function Start/End</i> – 154 | |
| <i>Go to Previous/Next Function</i> – 154 | |
| <i>Jump Back</i> – 154 | |
| <i>Jump Forward</i> – 154 | |
| <i>Set Jump Mark</i> – 155 | |
| <i>Find Differences</i> – 155 | |
| <i>Compare Two Front Documents</i> – 155 | |
| <i>Compare Against Disk File</i> – 155 | |
| <i>Apply to New</i> – 155 | |
| <i>Apply to Old</i> – 155 | |
| <i>Compare Again</i> – 155 | |
| <i>Find Definition</i> – 155 | |
| <i>Find in Reference</i> – 155 | |

Chapter 8 Searching with Grep 157

| | |
|-------------------------------------------------------------------|-----|
| What Is Grep or Pattern Searching? | 158 |
| Recommended Books and Resources | 158 |
| Writing Search Patterns | 159 |
| <i>Most Characters Match Themselves</i> – 159 | |
| <i>Escaping Special Characters</i> – 159 | |
| <i>Wildcards Match Types of Characters</i> – 160 | |
| <i>Character Classes Match Sets or Ranges of Characters</i> – 162 | |
| <i>Matching Non-Printing Characters</i> – 163 | |
| <i>Other Special Character Classes</i> – 164 | |
| <i>Quantifiers Repeat Subpatterns</i> – 165 | |
| <i>Combining Patterns to Make Complex Patterns</i> – 166 | |
| <i>Creating Subpatterns</i> – 166 | |
| <i>Using Backreferences in Subpatterns</i> – 167 | |
| <i>Using Alternation</i> – 168 | |
| <i>The “Longest Match” Issue</i> – 168 | |
| <i>Non-Greedy Quantifiers</i> – 169 | |

| | |
|----------------------------------------------------|-----|
| Writing Replacement Patterns | 170 |
| <i>Subpatterns Make Replacement Powerful</i> – 170 | |
| <i>Using the Entire Matched Pattern</i> – 170 | |
| <i>Using Parts of the Matched Pattern</i> – 171 | |
| <i>Case Transformations</i> – 172 | |
| Examples | 173 |
| <i>Matching Identifiers</i> – 173 | |
| <i>Matching White Space</i> – 173 | |
| <i>Matching Delimited Strings</i> – 174 | |
| <i>Marking Structured Text</i> – 174 | |
| <i>Marking a Mail Digest</i> – 175 | |
| <i>Rearranging Name Lists</i> – 175 | |
| Advanced Grep Topics | 175 |
| <i>Matching Nulls</i> – 176 | |
| <i>Backreferences</i> – 176 | |
| <i>POSIX-Style Character Classes</i> – 177 | |
| <i>Non-Capturing Parentheses</i> – 178 | |
| <i>Perl-Style Pattern Extensions</i> – 179 | |
| <i>Comments</i> – 179 | |
| <i>Pattern Modifiers</i> – 180 | |
| <i>Positional Assertions</i> – 181 | |
| <i>Conditional Subpatterns</i> – 183 | |
| <i>Once-Only Subpatterns</i> – 184 | |
| <i>Recursive Patterns</i> – 186 | |

Chapter 9 Browsers 187

| | |
|---------------------------------------------------|-----|
| Browser Overview | 187 |
| <i>List Pane</i> – 187 | |
| <i>Toolbar</i> – 188 | |
| <i>Text View Pane</i> – 188 | |
| <i>Splitter</i> – 188 | |
| Disk Browsers | 189 |
| <i>Disk Browser Controls</i> – 189 | |
| <i>Contextual Menu Commands</i> – 190 | |
| <i>Dragging Items</i> – 190 | |
| <i>Using the List Pane in Disk Browsers</i> – 190 | |
| Search Results Browsers | 191 |
| Error Results Browsers | 192 |

Chapter 10 Preferences 193

| | |
|----------------------------------------|-----|
| The Preferences Window | 193 |
| <i>Searching the Preferences</i> – 195 | |
| <i>Restore Defaults</i> – 195 | |
| Appearance Preferences | 195 |
| <i>Toolbar</i> – 195 | |
| <i>Navigation Bar</i> – 196 | |
| <i>Editing Window</i> – 196 | |
| <i>Text Status Bar</i> – 197 | |
| <i>List Display Font</i> – 197 | |

| | |
|----------------------------------------------------------------------------------------------------------------------|-------|
| Application Preferences | 197 |
| <i>Open documents into the front window...</i> | – 198 |
| <i>Automatically refresh documents as they change on disk</i> | – 198 |
| <i>Remember the N most recently used items</i> | – 198 |
| <i>When BBEdit becomes active</i> | – 199 |
| <i>Automatically check for updates</i> | – 199 |
| Editing Preferences | 200 |
| <i>Show text completions</i> | – 200 |
| <i>Include dictionary words in completion list</i> | – 200 |
| <i>Use “hard” lines in soft-wrapped views</i> | – 200 |
| <i>Soft-wrapped line indentation</i> | – 200 |
| <i>Line spacing</i> | – 200 |
| Editor Defaults Preferences | 200 |
| <i>Auto-indent</i> | – 200 |
| <i>Balance while typing</i> | – 201 |
| <i>Use typographer’s quotes</i> | – 201 |
| <i>Auto-expand tabs</i> | – 201 |
| <i>Show invisible characters</i> | – 201 |
| <i>Check spelling as you type</i> | – 202 |
| <i>Default font</i> | – 202 |
| <i>Tab Width</i> | – 202 |
| <i>This option controls the default number of spaces that BBEdit uses to represent the width of a tab character.</i> | – 202 |
| <i>Soft Wrap Text</i> | – 202 |
| Keyboard Preferences | 202 |
| <i>Use Tab key to navigate Placeholders</i> | – 202 |
| <i>“Home” and “End” Keys</i> | – 203 |
| <i>Enter key generates Return</i> | – 203 |
| <i>Allow Tab key to indent text blocks</i> | – 203 |
| <i>Enable Shift-Delete for forward delete</i> | – 203 |
| <i>Option-¥ on Japanese keyboards</i> | – 203 |
| <i>Emulate Emacs key bindings</i> | – 203 |
| Languages Preferences | 204 |
| <i>Installed Languages</i> | – 204 |
| <i>Custom Extension Mappings</i> | – 204 |
| <i>Change clipping Set to match document’s language</i> | – 205 |
| Menus & Shortcuts Preferences | 205 |
| <i>Menu Key Equivalents and Item Visibility</i> | – 205 |
| <i>Allow menu key equivalents to autorepeat</i> | – 206 |
| Preview Helpers Preferences | 206 |
| <i>Web Browsers Available for Previewing</i> | – 206 |
| Printing Preferences | 207 |
| <i>Print using document’s font</i> | – 207 |
| <i>Printing font</i> | – 207 |
| <i>Frame printing area</i> | – 207 |
| <i>Print page headers</i> | – 207 |
| <i>Print full pathname</i> | – 207 |
| <i>Time stamp</i> | – 207 |
| <i>Print line numbers</i> | – 207 |
| <i>1-inch Gutter</i> | – 207 |
| <i>Print color syntax</i> | – 207 |

| | |
|-------------------------------------------------------|-----|
| Text Colors Preferences | 208 |
| <i>How to Change an Element's Color</i> – 208 | |
| <i>General</i> – 208 | |
| <i>Source Code</i> – 208 | |
| <i>Markup</i> – 209 | |
| Text Encodings Preferences | 209 |
| <i>Default text encoding for new documents</i> – 209 | |
| <i>If file's encoding can't be guessed, try</i> – 209 | |
| Text Files Preferences | 210 |
| <i>Line breaks</i> – 210 | |
| <i>Ensure file ends with line break</i> – 210 | |
| <i>Strip trailing whitespace</i> – 210 | |
| <i>Backups</i> – 210 | |
| Expert preferences settings | 211 |
| The Setup Window | 212 |
| <i>Bookmarks</i> – 212 | |
| <i>Filters</i> – 212 | |
| <i>Patterns</i> – 212 | |
| <i>Sites</i> – 212 | |

Chapter 11 BEdit HTML Tools 215

| | |
|-------------------------------------------|-----|
| Introduction to the HTML Tools | 215 |
| <i>Recommended Books</i> – 216 | |
| <i>Recommended Online Resources</i> – 216 | |
| <i>What You Need</i> – 216 | |
| Configuring the HTML Tools | 217 |
| Using the HTML Tools | 217 |
| <i>Creating a New Document</i> – 218 | |
| <i>File Addressing</i> – 220 | |
| <i>Checking Syntax</i> – 220 | |
| <i>Format Customization</i> – 222 | |
| <i>Previewing Pages</i> – 222 | |

| | |
|--------------------------------------|-----|
| HTML Tool Descriptions | 224 |
| <i>Edit Markup</i> – 224 | |
| <i>Close Current Tag</i> – 226 | |
| <i>Balance Tags</i> – 226 | |
| <i>Document Type</i> – 226 | |
| <i>Character Set</i> – 226 | |
| <i>CSS submenu</i> – 226 | |
| <i>Body Properties</i> – 233 | |
| <i>Head Elements</i> – 233 | |
| <i>Block Elements</i> – 234 | |
| <i>Lists</i> – 235 | |
| <i>Tables</i> – 236 | |
| <i>Forms</i> – 237 | |
| <i>Inline Elements</i> – 239 | |
| <i>Phrase Elements</i> – 242 | |
| <i>Font Style Elements</i> – 243 | |
| <i>Frames</i> – 243 | |
| <i>Check</i> – 244 | |
| <i>Update</i> – 245 | |
| <i>Includes</i> – 245 | |
| <i>Utilities</i> – 246 | |
| <i>Tidy</i> – 247 | |
| <i>Preview</i> – 249 | |
| The HTML Tools Palette | 250 |
| <i>HTML Tools Palette Tips</i> – 251 | |
| <i>HTML Tools Palette</i> – 251 | |
| <i>Other Palettes</i> – 252 | |
| HTML Translation | 253 |
| <i>Convert Paragraphs</i> – 253 | |
| <i>HTML Entities</i> – 253 | |
| <i>Remove Tags</i> – 253 | |
| Templates | 254 |
| <i>Template Setup</i> – 254 | |
| <i>Using a Template</i> – 254 | |

Chapter 12 Using Clippings 255

| | |
|----------------------------------------------------|-----|
| The Clippings Menu | 255 |
| The Clippings Palette | 256 |
| Managing Clipping Sets | 256 |
| <i>Installing New Clipping Sets</i> – 256 | |
| <i>Language Sensitivity of Clipping Sets</i> – 257 | |
| <i>Manually Sorting Clipping Sets</i> – 257 | |
| Creating and Editing Clippings | 257 |
| Inserting Clippings | 259 |
| Assigning Key Equivalents to Clippings | 260 |
| Clipping Substitution Placeholders | 261 |
| <i>Selection and Insertion Placeholders</i> – 263 | |
| <i>Jump Placeholder Format</i> – 264 | |
| <i>Optional-Argument Placeholder Format</i> – 264 | |
| <i>Date Formats</i> – 264 | |
| <i>Time Formats</i> – 265 | |
| Using Scripts in Clippings | 266 |

| | |
|-------------------------------------------------------|-----|
| AppleScript Overview | 269 |
| <i>About AppleScript</i> – 270 | |
| <i>Scriptable Applications and Apple Events</i> – 270 | |
| <i>Reading an AppleScript Dictionary</i> – 271 | |
| <i>Recordable Applications</i> – 276 | |
| <i>Saving Scripts</i> – 277 | |
| <i>Using Scripts with Applications</i> – 277 | |
| <i>Scripting Resources</i> – 278 | |
| Using AppleScripts in BBEdit | 279 |
| <i>Recording Actions within BBEdit</i> – 279 | |
| <i>The Scripts Menu</i> – 280 | |
| <i>The Scripts Palette</i> – 281 | |
| <i>Organizing Scripts</i> – 281 | |
| <i>Attaching Scripts to Menu Items</i> – 282 | |
| <i>Attaching Scripts to Events</i> – 283 | |
| <i>Filtering Text with AppleScripts</i> – 286 | |
| BBEdit’s Scripting Model | 287 |
| <i>Script Compatibility</i> – 287 | |
| <i>Getting and Setting Properties</i> – 289 | |
| <i>Performing Actions</i> – 290 | |
| <i>Arranging Documents and Windows</i> – 293 | |
| <i>Common AppleScript Pitfalls</i> – 295 | |

| | |
|-------------------------------------------------------------|-----|
| Configuring BBEdit for Development Environments | 298 |
| <i>Syntax Coloring</i> – 298 | |
| <i>Exuberant Ctags</i> – 298 | |
| <i>Switching Between Counterpart Files</i> – 300 | |
| BBEdit and the Unix Command Line | 300 |
| <i>Shell Worksheets</i> – 300 | |
| <i>Installing the Command Line Tools</i> – 302 | |
| <i>The “bbedit” Command Line Tool</i> – 302 | |
| <i>The “bbdiff” Command Line Tool</i> – 303 | |
| <i>The “bbfind” Command Line Tool</i> – 304 | |
| Unix Scripting: Perl, Python, Ruby, Shells, and more! | 305 |
| <i>Using Unix Scripts</i> – 305 | |
| <i>Language Resources</i> – 305 | |
| <i>Setting Environment Variables for GUI Apps</i> – 306 | |
| <i>Line Endings, Permissions and Unix Scripts</i> – 306 | |
| <i>Configuring Perl</i> – 307 | |
| <i>Configuring Python</i> – 307 | |
| <i>Configuring Ruby</i> – 307 | |
| <i>Shebang Menu</i> – 307 | |
| <i>Filters and Scripts</i> – 309 | |
| <i>Filters</i> – 310 | |
| <i>Scripts</i> – 310 | |
| <i>Additional Notes</i> – 310 | |

| | |
|--------------------------------------------------|---------------------------------------|
| Working with CVS | 311 |
| <i>Quick Start for SSH-accessed Repositories</i> | – 311 |
| <i>Other Configurations</i> | – 311 |
| <i>Performing Commit Operations</i> | – 312 |
| <i>CVS Menu Commands</i> | – 312 |
| Working with Perforce | 315 |
| <i>Perforce Menu Commands</i> | – 315 |
| Working with Subversion | 317 |
| <i>Configuring Subversion</i> | – 317 |
| <i>Command Line Integration</i> | – 317 |
| <i>Subversion Commands</i> | – 317 |
| | 319 |
| Chapter 15 | Language Modules and Packages |
| | 321 |
| Language Modules | 321 |
| <i>Installing Language Modules</i> | – 322 |
| <i>Overriding Existing Modules</i> | – 322 |
| <i>Codeless Language Modules</i> | – 322 |
| <i>Code-based Language Modules</i> | – 322 |
| <i>Language Module Compatibility</i> | – 322 |
| Packages | 323 |
| Plug-In Obsolescence | 323 |
| Appendix A | Command Reference |
| | 325 |
| Keyboard Shortcuts for Commands | 325 |
| Assigning Keys to Menu Commands | 326 |
| <i>Available Key Combinations</i> | – 326 |
| Listing by Menu and Command Name | 327 |
| Listing by Default Key Equivalent | 338 |
| Appendix B | Editing Shortcuts |
| | 343 |
| Mouse Commands | 343 |
| Arrow and Delete Keys | 344 |
| Emacs Key Bindings | 345 |
| <i>Using universal-argument</i> | – 346 |
| Appendix C | Placeholders and Include Files |
| | 347 |
| Placeholders | 347 |
| <i>Date Formats</i> | – 350 |
| <i>Time Formats</i> | – 350 |
| <i>Using the #RELATIVE# Placeholder</i> | – 351 |

| | |
|-------------------------------------------|-----|
| Include Files | 352 |
| <i>Include File Locations</i> – 352 | |
| <i>Simple Includes</i> – 352 | |
| <i>Persistent Includes</i> – 353 | |
| <i>Inline versus Block Includes</i> – 353 | |
| <i>Include Files with Variables</i> – 353 | |
| <i>Including AppleScripts</i> – 354 | |
| <i>Including Unix Scripts</i> – 355 | |
| <i>Other Include Notes</i> – 357 | |

Appendix D Codeless Language Modules 359

| | |
|---------------------------------------------------------|-----|
| Creating a Module | 359 |
| <i>Required Elements</i> – 359 | |
| <i>Installing Codeless Language Modules</i> – 360 | |
| <i>Function Scanning with Regular Expressions</i> – 360 | |
| <i>Spell Checking Code Runs</i> – 361 | |
| <i>Starting from a Template</i> – 361 | |
| Language Keys and Properties | 363 |

Index 373

Welcome to BBEdit

This chapter introduces you to BBEdit, a high-performance HTML and text editor for the Macintosh.

In this chapter

| | |
|--------------------------------------------------------------------------------|----|
| Getting Started | 19 |
| What Is BBEdit? | 19 |
| How Can I Use BBEdit? | 20 |
| <i>Development Environments</i> – 20 | |
| <i>Writing HTML Documents</i> – 20 | |
| Human Interface Notes | 21 |
| <i>Dynamic Menus</i> – 21 • <i>Bypassing Options Dialogs</i> – 21 | |
| <i>Keyboard Shortcuts for Commands</i> – 21 • <i>Contextual Menus</i> – 22 | |
| <i>Snappy Palettes</i> – 22 • <i>Dialog Box and Sheet Key Equivalents</i> – 22 | |
| Feature Highlights | 23 |
| <i>Info on New Features</i> – 23 | |
| Discussion Group | 23 |
| Support Services | 24 |

Getting Started

Thank you for selecting BBEdit, the premier HTML and text editor for the Macintosh.

If you are new to BBEdit, we recommend that you read at least Chapters 1 through 4 of this manual to familiarize yourself with the installation and basic operation of BBEdit. You may also wish to read or preview any other chapters that cover features you frequently use. After you have installed BBEdit, the best way to learn it is to use it. Complete online assistance is available from the Help menu.

If you have used earlier versions of BBEdit, we recommend that you read at least Chapter 1 for an overview of significant changes in this version, and Chapter 2 for information relevant to installation and upgrading.

What Is BBEdit?

BBEdit is a high-performance HTML and text editor. Unlike a word processor, which is designed for preparing printed pages, a text editor focuses on providing a means of producing and changing content. Thus, BBEdit does not offer fancy formatting capabilities, headers and footers, graphics tools, a thesaurus, or similar staples of feature-laden “office” software. Instead, it focuses on helping you manipulate text in ways that word processors generally cannot.

In service of this goal, BBEdit offers powerful regular expression–based (“grep”) search and replace, multi-file search, sophisticated text transformations, intelligent text coloring, and other features not usually found (or missed) in word processors.

BBEdit also has features that make it easier to edit specific kinds of text, such as source files for programming languages and HTML (Hypertext Markup Language) files for the World Wide Web. In fact, since the rise of the Web, BBEdit has been the tool of choice for Macintosh Web designers who need more flexibility than visual Web authoring tools can provide.

How Can I Use BBEdit?

Use BBEdit any time you need to create or edit Web pages, source files, or text documents of any kind. Whether you need to find (or change!) all the occurrences of some text in a set of files, or modify or reformat large text files of any sort, or quickly tweak a Web page, BBEdit is the right tool for the job.

Development Environments

BBEdit found its initial following among the Macintosh programming community with its core editing- and development-oriented tools. Although we have added countless other features to BBEdit since its first incarnation, its source code editing capabilities are now stronger than ever.

In addition to offering syntax coloring and function browsing for many different languages, BBEdit supports direct use of Perl, Python, and Ruby (as well as any other Unix scripting environment) and provides integrated support for the Subversion and Perforce source code control systems. Chapter 14 provides more information on how to set up BBEdit for this type of work.

Writing HTML Documents

BBEdit is an ideal tool for preparing and editing HTML documents (web pages). In addition to many options for preparing text content, such as wrapping, case changes, and searching, BBEdit offers a powerful set of tools to make editing web pages easier. BBEdit’s Edit Markup command allows you to quickly add tags or modify existing tags, while the HTML Tools palette lets you access commands with just a click.

Using BBEdit, you can easily preview your work in most Macintosh web browsers, including Safari, Chrome, Firefox, OmniWeb, and Opera, as well as via BBEdit’s native Preview feature and Windows browsers running under VMWare Fusion. For more information on using the HTML Tools to create, edit, and preview web pages, see Chapter 11.

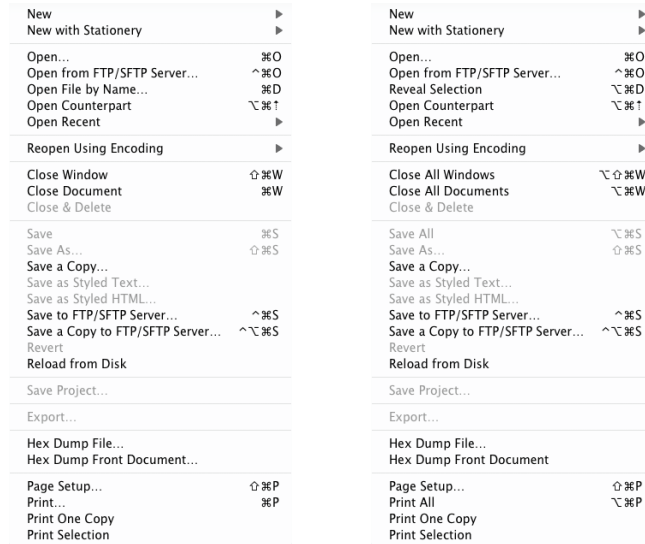
Human Interface Notes

BBEdit enhances the behavior of its menus and dialogs as described in this section.

Dynamic Menus

IMPORTANT

Many of BBEdit's pull-down menus are dynamic: if you hold down the Shift, Option, or Control key while a menu is open, you can see some of the items change. The illustration below shows what the File menu looks like normally (left) and when you hold down the Option key (right).



You can use the Shift, Option, or Control keys when you choose an item from a menu or when you use the Command-key equivalents.

Bypassing Options Dialogs

You may have noticed that commands that require additional settings to be made before they are performed appear on the menu with ellipses after their names. To bypass this step and use the command with its most recent settings, hold down the Option key while selecting the menu item. For example, “Zap Gremlins...” in the Text menu becomes “Zap Gremlins” when the Option key is pressed and, when chosen, will zap gremlins in the frontmost text document using the current settings.

Keyboard Shortcuts for Commands

Many of BBEdit's commands have keyboard shortcuts. BBEdit lets you reassign the shortcuts for any menu item, clippings entry, plug-in, or script to suit your own way of working.

To change the keyboard shortcut for any menu command as well as any available scripts and text filters, go to the Menus & Shortcuts preference panel.

Contextual Menus

When you Control-click on selected text or at the insertion point in a text window, BBEdit's contextual menu will display a set of commands relevant to that location or text, as well as some appropriate standard commands (such as Cut/Copy/Paste, or Check Spelling) so you do not have to hunt around in the menu bar for them.

Snappy Palettes

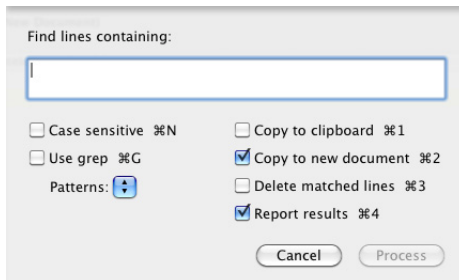
When you move or resize palettes (floating windows), they will “snap” to the edges of the screen and the edges of other palettes. You can override this behavior by holding down the Shift key while dragging or resizing.

Dialog Box and Sheet Key Equivalents

You can use key equivalents to click buttons or select options in most of BBEdit's dialog boxes and sheets. Certain keys have the same meaning in all dialogs and sheets:

- Pressing either the Return or Enter key is the same as clicking the default button.
- Typing Command-period or pressing the Escape key is the same as clicking the Cancel button.
- You can use the Cut, Copy, Paste, Clear, and Select All commands (either from the Edit menu or with their Command-key equivalents) in any text field.

To see the other key equivalents for a particular dialog or sheet, hold down the Command key. After a brief delay, labels appear next to any buttons or options which have key equivalents. For example, this is the Process Lines Containing sheet with Command-key equivalents visible:



BBEdit waits briefly before displaying the Command-key equivalents so that you can type a sequence quickly without encountering visible flicker.

Feature Highlights

BBEdit 10.0 offers many powerful features for editing and processing text and code, and for managing your work. Here are some highlights:

- Fully context-aware markup panel for adding & editing HTML tags
- Flexible multi-document editing windows
- Support for creating, editing and syntax checking HTML5 documents
- Enhanced project support for managing and editing related files
- Modeless Find and Multi-File Search windows provide a consistent, familiar interface to BBEdition's legendary search and replace capabilities
- In-window Live Search highlights and quickly jumps through matches
- Customizable text completion for easy insertion of words, syntax elements, and frequently-used material
- Extensive HTML and PHP completion support
- Find Differences detects sub-line differences
- Directly perform multi-file searches on Xcode projects
- Live display of document statistics: character, word, and line count
- Persistent global scratchpad for quick text cleanup and Unix worksheet
- Expanded ctags file format support
- Transparent support for viewing and editing files within bz2, gzip, and Zip archives.

as well as all the powerful core features that BBEdition is known for.

Info on New Features

In addition to these major features, BBEdition 10.0 also contains numerous additional convenience features and interface refinements, as well as performance enhancements and bug fixes. For a detailed summary of changes and bug fixes, please refer to the current release notes, which are available in the BBEdition Support section of our web site.

http://www.barebones.com/support/bbedit/current_notes.html

Discussion Group

We maintain a public Google Group where our customers can discuss and share knowledge about using BBEdition.

<http://groups.google.com/group/bbedit>

Support Services

If you need information about using BBEdit (or any of our other products) the Support area of our web site offers up-to-date details:

<http://www.barebones.com/support/>

You'll find a wide range of information there, including:

- Frequently Asked Questions (FAQ) — Information and answers for commonly encountered questions and problems. We strongly recommend you check the BBEdit FAQs before resorting to any other means of inquiry.
- Product Updates — The latest maintenance versions of our products are always available for download.

as well as access to plug-ins, sample scripts, developer info, and other materials.

How to contact us

If you have a registered copy of BBEdit (or any other Bare Bones product), and you can't find the information you need on our web site, or if you encounter any problems with the software, please use the contact form on our web site or send email to:

support@barebones.com

Note We do not offer telephone support. Please refer to the support resources available on our web site for information and assistance, or contact us via email.

Installing BBEdit

This chapter tells you how to install BBEdit on your Macintosh. It also describes the files BBEdit creates, where it puts them, and how to install or remove optional components of BBEdit.

In this chapter

| | |
|----------------------------------------------------------------------------------|----|
| Basic Installation | 25 |
| <i>System Requirements</i> – 25 • <i>Installing BBEdit</i> – 25 | |
| <i>Checking for Updates</i> – 26 • <i>Upgrading from a Previous Version</i> – 26 | |
| <i>First Run Configuration</i> – 27 • <i>Activating the Demo</i> – 28 | |
| BBEdit’s Application Support Folders..... | 29 |
| <i>Using a Global Application Support Folder</i> – 29 | |
| <i>Using a Local Application Support Folder</i> – 29 | |
| <i>Application Support Folder Contents</i> – 29 | |
| <i>Clippings</i> – 30 • <i>Clippings</i> – 30 • <i>HTML Templates</i> – 31 | |
| <i>Language Modules</i> – 31 • <i>Menu Scripts</i> – 31 • <i>Scripts</i> – 32 | |
| <i>Shutdown Items</i> – 33 • <i>Startup Items</i> – 34 • <i>Stationery</i> – 34 | |
| <i>Superseded App Support Folders</i> – 35 • <i>Upgrading</i> – 35 | |
| Preference Files and Folders | 35 |
| <i>BBEdit Preferences File</i> – 35 • <i>BBEdit Preferences Folder</i> – 36 | |
| Sharing Application Support & Preferences Data via Dropbox | 36 |

Basic Installation

BBEdit is supplied as a single application file. Specific system requirements and installation instructions are described below, and the organization of BBEdit’s supporting files is described in subsequent sections.

System Requirements

IMPORTANT

BBEdit 10 requires Mac OS X 10.6 or later (10.6.8, or 10.7.1 or later recommended). The software will not run on Mac OS 9 or any earlier versions of Mac OS X, and will not run on PowerPC-based machines.

Installing BBEdit

If you purchase BBEdit directly, you will receive a standard disk image (“.dmg”) file. Your web browser may automatically mount the disk image once the download is complete; otherwise, you should double-click on the disk image file to mount it. Once the disk image is mounted, drag the “BBEdit” application over the adjacent icon for the Applications folder and drop it there to copy BBEdit onto your Mac’s hard drive. You can then dismount (eject) the disk image and discard the “.dmg” file.

If you purchase BBEdit through the Mac App Store, you must install it via the App Store application.

Checking for Updates

BBEdit offers the option to automatically check for updates; this behavior is controlled by the “Software Update” option in the Application preferences panel. You can also directly check for updates at any time by choosing Check for Updates in the BBEdition (application) menu.

In order to update BBEdition when future versions become available, you need only apply the update when prompted. (Alternatively, you may quit BBEdition, and manually replace your existing copy with the updated version.) The first time you launch a newer version of the software, BBEdition will prompt you for any further actions which may be needed, such as updating the command-line tools.

Note Copies of BBEdition purchased through the Mac App Store do not include the update check feature; instead, updates will be delivered through the App Store app.

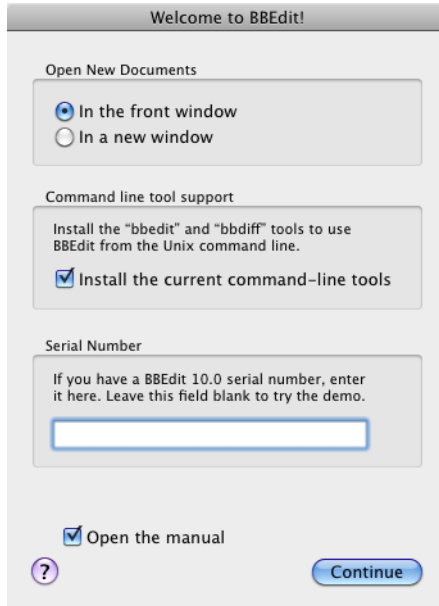
Upgrading from a Previous Version

IMPORTANT If you are upgrading from any version prior to BBEdition 8.5, in addition to installing the current application, you will need to manually copy over any items you wish to use from your existing “BBEdition Support” folder into BBEdition’s application support folder. You should **not** simply rename your existing “BBEdition Support” folder. (See “Sharing Application Support & Preferences Data via Dropbox” on page 36.)

Please carefully read the remainder of this chapter, since the organization of BBEdition 10’s supporting files has changed considerably. We have provided specific suggestions and tips for transferring your customized support items in each category.

First Run Configuration

The first time you launch BBEdit, it will display the “Welcome to BBEdit” dialog. This dialog allows you to choose whether to display multiple text documents in a single window versus one document per window, whether to install the command-line tools, and whether to immediately open the PDF user manual.



Opening New Documents

Since BBEdit supports opening multiple documents into a single text window, you must decide whether the application should work in this manner, or whether it should behave in the standard fashion by opening each document into its own window. (A document represents either a file which you open for editing, a text document created by the New Document command on the File menu, or any similar item, such as a text document created via the scripting interface.)

Choose “In the front window” to have BBEdit open all new documents into a single text window. Choose “In a new window” to have BBEdit open each new document into its own text window.

Note If you change your mind later, you can adjust this option in the Application preference panel.

Entering Your Serial Number

The dialog also enables you to enter the unique product serial number that you received at the time of purchase to activate BBEdit. Once you enter the serial number, your copy of BBEdit will be activated, and all demo restrictions will be removed.

Note Copies of BBEdit purchased through the Mac App Store do not use serial numbers; they are instead activated by and associated with your Apple ID account.

If you have not yet purchased the product and thus do not have a serial number, leave this space blank. BBEdit will operate as a fully functional demo for a limited trial period, after which you must purchase a license and enter a valid serial number in order to continue using it.

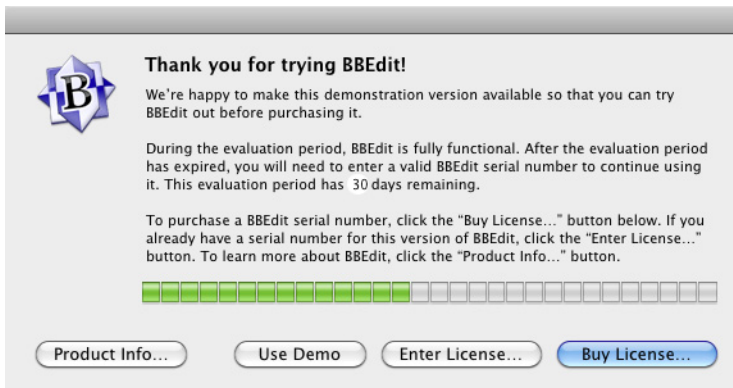
If BBEdit has already been activated, the dialog will display the associated serial number. You may continue using this serial number, or enter a different serial number if you wish to change the active license.

IMPORTANT

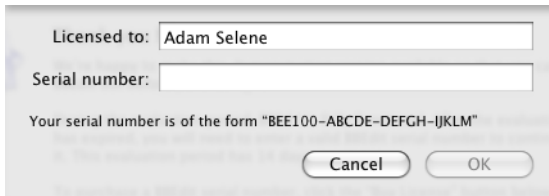
In order to activate a directly-purchased copy of BBEdit 10.0, you must have a BBEdit 10 product serial number (one beginning with a prefix of “BEE100-” or “BEC100-”). You cannot activate the application with a serial number from any older version of BBEdit.

Activating the Demo

While running in demo mode, BBEdit will operate with full functionality for a limited period of time. Upon expiration of this trial period, you must purchase the product and receive a serial number in order to continue using it.



To activate the demo, click Enter License and enter your name and product serial number in the activation sheet. (You can copy the serial number from your online order receipt and paste it into the Serial Number field of the sheet.)



You may review the active serial number at any time by choosing the License command from the BBEdit menu to bring up the license info dialog. In order to change activation to a different serial number, click Edit License to bring up the activation sheet and enter the new information.

BBEdit's Application Support Folders

BBEdit's application support folder contains items which define or extend BBEdition's capabilities, such as clippings sets, language modules, scripts, and more. These items are organized into subfolders according to their purpose (described below).

IMPORTANT

BBEdit's application support folder must be present in either or both of the following locations:

- Global (items available to all users):
/Library/Application Support/BBEdit/
- Local (user-specific items):
~/Library/Application Support/BBEdit/

By default, BBEdition creates only the local application support folder.

Note

Use of the ~ character in folder path descriptions is customary Unix shorthand for the location of your home directory. If written out in full, this path would be "/Users/<username>/Library/Application Support/BBEdit/".

Using a Global Application Support Folder

You can use a global application support folder to provide a common set of supporting items in BBEdition to each user of the machine.

Users whose accounts do not have admin privileges will not be able to modify the contents of a global application support folder, since it resides in the system hierarchy. This arrangement can be advantageous when configuring the software for use in shared-machine environments, such as labs or common-area workstations.

However, if such an arrangement is not desirable for your purposes, you should not create a global application support folder. Instead, each user can maintain their own local application support folder for BBEdition, which they may add items to, or remove items from, at will.

Using a Local Application Support Folder

If a local application support folder does not exist when BBEdition starts up, BBEdition will create this folder together with its standard subfolders, to which you can add any appropriate items. None of these folders are essential for doing basic tasks with BBEdition, and you can remove any or all of them that you don't use.

Application Support Folder Contents

BBEdit's application support folders contain various subfolders, each of which holds a specific type of support item. As indicated, items in some subfolders can be loaded from both the global and local application support folders; other items may only be used from a specific location.

If there are multiple copies of any language modules, BBEdition will use the latest version regardless of location. For all other items, BBEdition lists the global and local sets separately.

To prevent alias loops, BBEdit will not follow aliases to folders that are placed inside any of the subfolders within the application support folder. We also recommend that you do not try to share scripts between BBEdit and other applications, and that you not make aliases to any items located on remote (server) volumes.

Attachment Scripts

[Local only]

This folder does not exist by default, but you may create it at any time. The Attachment Scripts folder contains AppleScripts which are run at specific points: when BBEdit starts or quits; and when documents are open, saved, and closed.

Auto-Save Recovery

[Local only]

BBEdit will automatically create this folder. The Auto-Save Recovery folder contains information which BBEdit can use to recover the contents of unsaved documents after a crash, or to restore them at launch. Removing items from this folder can cause data loss.

Clippings

[Local only]

BBEdit will automatically create this folder if it does not exist. The Clippings folder contains clipping items. These items are text files which appear in the Clippings menu and palette, and whose contents can be inserted into a document by choosing them directly, or via text completion. Clippings may also contain special placeholders which insert varying or context-sensitive information—for example, a date or the name of the current file. (See Chapter 12 for more information on creating and using clippings.)

Upgrading

You should move or copy over any third-party clipping packages, or any custom clippings which you have created, that you wish to preserve.

Color Schemes

[Local only]

BBEdit will automatically create this folder when needed. The Color Schemes folder stores any custom color schemes which you have saved within the Text Colors preference panel (or which you have download and copied over). Each scheme is stored within a separate “.bbcolors” file.

Completion Data

This folder does not exist by default, but you may create it. The Completion Data folder contains tags files (or aliases to tags files) which can provide additional text completions for editing documents in the corresponding languages.

These tags files should be in the format generated by ‘bbedit --maketags’, and must be placed in subfolders corresponding to their languages.

Each subfolder should have the exact name of its language as that language appears in the list of installed languages (or on the Languages popup menu).

For example, the subfolder containing a Python tags file must be named “Python”, and the subfolder containing a tags file for ANSI C must be named “ANSI C”.

HTML Templates

BBEdit does not create this folder by default, but one may exist from previous versions. This folder contains HTML template files which are used by the New HTML Document command. In order to use these templates elsewhere, you must either choose this folder to be the Templates & Includes folder for an existing web site in the Sites panel of the Setup window, or you can copy the template files into an already-designated site templates folder. Please see Chapter 11 for more information on BBEdit’s HTML tools.

Upgrading You should move or copy over any customized template or include documents that you wish to preserve.

Language Modules

[Global, Local]

BBEdit does not create this folder by default, but will do so if necessary. The Language Modules folder allows you to add syntax coloring and function navigation support for additional languages by installing language modules.

IMPORTANT Do not attempt to extract or modify the language modules contained in the BBEdit application bundle.

A list of additional modules from third-party developers is available on our web site, or you may create your own compiled or codeless language modules (see “Codeless Language Modules” on page 322).

Upgrading You should move or copy over any compatible third-party language modules that you wish to preserve.

Menu Scripts

[Local only]

This folder does not exist by default, but you may create it. The Menu Scripts folder contains AppleScripts that are attached to BBEdit menu items. (For more details on using menu scripts, please see “Attaching Scripts to Menu Items” on page 282.)

Upgrading You should move or copy over any menu scripts that you wish to preserve.

Packages

[Local only]

This folder does not exist by default, but you may create it. The Packages folder contains pre-packaged sets of supporting items. (For information about creating packages, please see “Packages” on page 323.)

Plug-Ins

IMPORTANT

BBEdit 10.0 does not support plug-ins from older versions, and will not load any items present in this folder.

If you used any third-party commercial plug-in, please contact its developer for information on alternative solutions.

Readme.txt [file]

[Local only]

This file contains an abbreviated description of the default contents of BBEdition's application support folder.

Scratchpad [file]

[Local only]

BBEdit automatically creates this file, which contains the data for BBEdition's main Scratchpad. Removing this file will result in the loss of your Scratchpad data.

Scripts

[Global, Local]



BBEdit will automatically create this folder if it does not exist. The Scripts folder may contain AppleScript files, Automator workflows, text factories, and executable Unix files (scripts). Items placed in this folder will appear in the Scripts menu (left), and you may place items within subfolders (up to four levels deep) to organize them.

You may run these items from the Scripts menu, the floating Scripts palette, or via assigned key equivalents. (You may use the Menus & Shortcuts preference panel to assign a key equivalent to any item in the Scripts menu.)

BBEdit runs such items by simply loading the item and calling it directly, without providing any inputs. (Naturally, AppleScript scripts and Automator actions may query BBEdition for more information, and Unix scripts may obtain information from the environment variables that BBEdition sets, while text factories will use their stored target list if any.)

Upgrading

The first time you launch BBEdition 10, it will copy all of your existing Unix scripts into this folder.

If you are upgrading from a version prior to 8.5, you must instead manually move or copy over any customized scripts that you wish to preserve. Note also that scripts written for use with such older versions of BBEdition may no longer work. (Please see Chapters 13 and 14 for more details and tips on modifying your existing AppleScripts and Unix filters & scripts.)

Setup

[Local only]

BBEdit will automatically create this folder if it does not exist. The Setup folder contains configuration data which previous versions stored in either the preferences file proper or in the “com.barebones.bbedit.preferenceData” preference folder, including: stored file filters, FTP/SFTP bookmarks, key bindings, grep patterns, and web site settings. Thus, if you have placed your BBEdition application support folder in your Dropbox folder, these items will be synchronized.

The Setup folder may contain any or all of the following data files.

File Filters.filefilters

BBEdit stores all user-defined file filter patterns in this file. You should not attempt to directly edit the contents of this file; instead, please use the Filters panel of the Settings window to add, modify, or remove stored grep patterns.

FTP Bookmarks.xml

BBEdit stores user-defined FTP and SFTP bookmarks in this file. You should not attempt to directly edit the contents of this file; instead, please use the Bookmarks panel of the Settings window to add, modify, or remove bookmarks.

Grep Patterns.xml

BBEdit stores user-defined search patterns in this XML file. You should not attempt to directly edit the contents of this file; instead, please use the Patterns panel of the Settings window to add, modify, or remove stored grep patterns.

Upgrading

If you have created any custom grep patterns in a previous version of BBEdition, these patterns will be imported; otherwise, BBEdition will create a set of factory default patterns.

Menu Shortcuts.xml

BBEdit stores keyboard shortcuts for menu commands in this XML file.

Not Menu Shortcuts.xml

BBEdit stores other keyboard shortcuts in this XML file.

Web Site Settings.plist

BBEdit stores web site configuration details in this plist file.

Shutdown Items

[Local only]

This folder does not exist by default, but you may create it at any time. The items in this folder are opened when you quit BBEdition. Usually, this function is used to run scripts of some sort.

Shutdown items are run after all windows have been closed, and only if BBEdition is actually quitting. Thus, if you wish to run any items as the immediate result of a Quit command, you should write a menu script attached to BBEdition•Quit.

Note In some previous versions of BBEdit, shutdown items were run before all windows were closed, and were run whenever the application was told to quit (either by the Quit menu command or via the scripting interface), regardless of whether it actually quit or not.

Upgrading You should move or copy over any shutdown items that you wish to preserve.

Startup Items

[Local only]

This folder does not exist by default, but you may create it at any time. When launched, BBEdit will open any items it finds in this folder.

If the items present are documents of a type that BBEdit knows how to handle (such as text files or projects), BBEdit will open them directly. If you place a compiled AppleScript in this folder, BBEdit will execute the script. If you place a folder alias here, BBEdit will open a disk browser window based at that folder.

If you place other types of items in this folder, BBEdit will ask the Finder to open them. If you often edit HTML files, for instance, you may want to place an alias to your Web browser (or your visual HTML editor) in the BBEdit Startup Items folder so that it will start up automatically whenever you run BBEdit.

Upgrading You should move or copy any file or application aliases that you wish to preserve. If you have any AppleScripts startup items, please see the preceding upgrade note for the Scripts folder about script compatibility.

Stationery

[Global, Local]

This folder does not exist by default, but you may create it at any time. The Stationery folder contains stationery files for use with BBEdit's New with Stationery command, and the Stationery List palette. Stationery files may be placed within subfolders (up to four levels deep) to organize them.

You can hide, or show, all items included from the global folder by using the menu item "Hide/Show Library Stationery".

Upgrading You should move or copy over any stationery documents that you wish to preserve.

Text Filters

[Local]

This folder does not exist by default, but you may create it at any time. The Text Filters folder contains executable items, such as compiled AppleScripts, Automator workflows, text factories, and Unix filters, which you may apply to the frontmost document via the Apply Text Filter command in the Text menu.

When you apply such an item, BBEdit will pass either the selected text (if any) or the contents of the entire document on STDIN to Unix executables and filters, as a reference to a 'RunFromBBEdit' entry point in AppleScripts, as text input to Automator workflows, and as a source to text factories. (If an AppleScript script does not have a 'RunFromBBEdit' entry point, BBEdit will call its run handler, again passing a reference to the current selection range.)

AppleScript scripts and Automator workflows should return a string which BBEdit will use to replace the selection range, Unix filters should write to STDOUT, and the text emitted by a text factory will replace the selection range.

Upgrading The first time you launch BBEdit 10, it will copy all of your existing text factories and Unix filters into this folder.

Unix Worksheet.worksheet

[Local]

BBEdit automatically creates this file, which contains the data for BBEdit's global Unix worksheet window. (Choose Show Unix Worksheet in the Window menu to open this worksheet.) Removing this file will result in the loss of your global Unix worksheet data.

Web Site Projects

[Local]

This folder does not exist by default, but you may create it at any time. The Web Site folder contains project files corresponding to any web site configurations you have created in the Sites panel of the Setup window.

Upgrading The first time you launch BBEdit 10, it will create a web site project file for each existing website configuration.

Superseded App Support Folders

Upgrading BBEdit 10 no longer uses the Text Factories or Unix Support folders, though these folders may exist if they were created by a prior version. Instead, the first time you launch BBEdit 10, it will copy all existing Unix scripts into the Scripts folder, and all existing text factories and Unix filters into the Text Filters folder.

Preference Files and Folders

When you start up BBEdit, it may create the files and folders noted in this section.

BBEdit Preferences File

All of BBEdit's basic preference settings are stored in the file “~/Library/Preferences/com.barebones.bbedit.plist”, which is created and maintained using standard system services. In addition to the settings documented in Chapter 10, you may adjust additional expert preference settings outside of BBEdit by issuing suitable “defaults write” commands. For a complete list of available expert preference settings, please see the “Expert Preferences” page of the built-in Help book. (To open the Help book, choose “BBEdit Help” in the “Help” menu.)

Upgrading BBEdit 10.0 will import and use any relevant preference settings from BBEdit 8.5 or later, provided that prefs file is present.

BBEdit Preferences Folder

By default, BBEdition stores ancillary preference data in the folder “~/Library/BBEdit” so as to comply with current OS guidelines. (Earlier versions stored such data in the folder “~/Library/Preferences/com.barebones.BBEdit.PreferenceData/” and BBEdition 10 will continue to use that folder if it already exists.)

The standard contents of this folder are as follows.

Document State.plist

BBEdit stores state information for individual documents in this file.

Recent Files & Favorites

This folder is no longer used and may be deleted.

Recent Folders & Favorites

This folder is no longer used and may be deleted.

Save Application State.appstate

BBEdit stores application state info in this file.

Saved Sources.xml

BBEdit stores all user-defined search sources in this file.

Sleep State.appstate

BBEdit stores application state info in this file.

Sharing Application Support & Preferences Data via Dropbox

If you use Dropbox, you may relocate your BBEdition application support and preference data folders to your active Dropbox folder, and BBEdition will use their contents from that location rather than the default locations (within ~/Library/). In this way, you can easily share supporting files among multiple BBEdition installations.

In order to do this:

- Quit BBEdition if it is running.
- Move your BBEdition application support folder (~/Library/Application Support/BBEdit/) into a folder named “Application Support” within your Dropbox folder, (so its final location is “~/Dropbox/Application Support/BBEdit/”). (Since Dropbox does not create an “Application Support” folder by default, you may need to do so.)
- Locate your BBEdition preference data folder (either ~/Library/BBEdit/ or ~/Library/Preferences/com.barebones.bbedit.PreferenceData/)
- Rename your BBEdition preference data folder from “BBEdit” (or “com.barebones.bbedit.PreferenceData”) to “PreferenceData”.
- Move the “PreferenceData” folder into ~/Dropbox/Application Support/BBEdit/

Note Since the system does not support relocation of an application's core preferences data file (`~/Library/Preferences/com.barebones.bbedit.plist`), you cannot directly sync these settings. You may, however, copy the core preferences file to other machines to 'seed' them with your preferred prefs settings.

IMPORTANT The ability to share supporting files between multiple installations of BBEdit does not excuse you from the terms of BBEdit's license. You may use Dropbox to share your supporting files with whomever you wish, but you must continue to abide by the license agreement.

This chapter discusses how to use BBEdit to manipulate text files.

In this chapter

| | |
|------------------------------------------------------------------------|----|
| Launching BBEdit | 40 |
| <i>Startup Items</i> – 40 | |
| Creating and Saving Documents | 41 |
| <i>Saving a Copy of a File</i> – 43 • <i>File Saving Options</i> – 43 | |
| <i>File State</i> – 44 • <i>Emacs Local Variables</i> – 44 | |
| <i>Saving with Authentication</i> – 45 | |
| <i>Saving Compressed Files as bz2 or gzip</i> – 45 | |
| <i>Saving as Styled Text or HTML</i> – 45 | |
| Crash Auto-Recovery..... | 45 |
| Opening Existing Documents | 46 |
| <i>Choosing the Encoding for a Document</i> – 46 | |
| <i>Using the Open Command</i> – 48 • <i>Reload from Disk</i> – 49 | |
| <i>Opening and Editing Files within Zip Archives</i> – 49 | |
| <i>Opening bz2, gzip, and tar Files and Binary plists</i> – 50 | |
| <i>Opening Hidden Files</i> – 50 | |
| <i>Using the Open Recent Command</i> – 53 | |
| <i>Using the Reopen using Encoding Command</i> – 53 | |
| <i>Using the Open Selection Command</i> – 50 | |
| Quitting BBEdit | 53 |
| An International Text Primer | 53 |
| <i>International Text in BBEdit</i> – 54 • <i>Unicode</i> – 54 | |
| <i>Saving Unicode Files</i> – 54 • <i>Opening Unicode Files</i> – 55 | |
| Accessing FTP/SFTP Servers | 56 |
| <i>Opening Files from FTP/SFTP Servers</i> – 56 | |
| <i>Saving Files to FTP/SFTP Servers</i> – 59 | |
| <i>Using BBEdit from the Command Line</i> – 61 | |
| Using BBEdit from the Command Line..... | 61 |
| Using Projects | 62 |
| <i>Creating a Project</i> – 62 • <i>Project Commands</i> – 63 | |
| <i>Using projects</i> – 64 • <i>Removing Files from a Project</i> – 65 | |
| <i>Contextual Menu Commands</i> – 65 | |
| <i>Script Access to Project Contents</i> – 65 | |
| Using Stationery..... | 65 |
| Hex Dump for Files and Documents | 66 |
| Making Backups..... | 66 |
| Printing..... | 66 |
| <i>Text Printing Options</i> – 67 | |

Launching BBEdit

To launch BBEdit, double-click the BBEdit application icon or a BBEdit document. Holding down the following keys at launch has the indicated effects, overriding any startup options set in the Application preference panel. When one of these keys is held down, BBEdit will beep after it finishes launching.

| Modifier | Function |
|----------|--------------------------------------------------------------------------------|
| Option | Suppress startup items only |
| Shift | Disable all external services and startup items, and skip reopening documents. |

Startup Items

When launched, BBEdit will look for a folder named Startup Items in the its application support folder (see “Sharing Application Support & Preferences Data via Dropbox” on page 36). If this folder is found, BBEdit will open any items it finds in the folder.

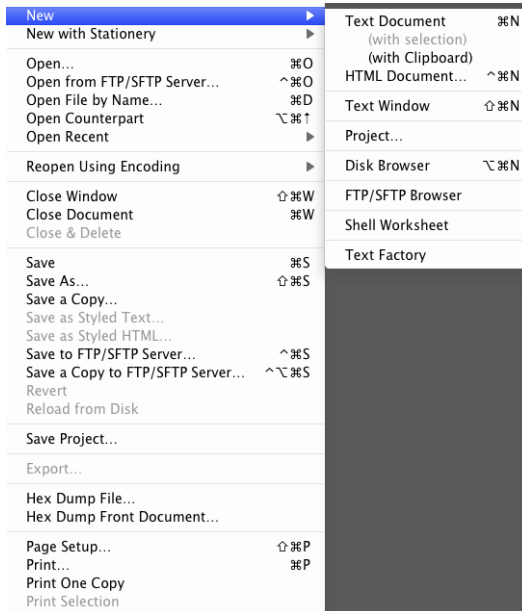
If the items present are documents of a type that BBEdit knows how to handle (such as text files or projects), BBEdit will open them directly. If you place a compiled AppleScript in this folder, BBEdit will execute the script. If you place a folder alias here, BBEdit will open a disk browser window based at that folder.

If you place other types of items in this folder, BBEdit will ask the Finder to open them. If you often edit HTML files, for instance, you may want to place an alias to your Web browser (or your visual HTML editor) in the BBEdit Startup Items folder so that it will start up automatically whenever you run BBEdit.

If you wish, you may place the actual Startup Items folder in any convenient location, create an alias to it, and place the resulting alias in BBEdit’s application support folder. Be sure to name the alias “Startup Items” so that BBEdit can locate it.

Creating and Saving Documents

To create a new text document or special-purpose window within BBEdit, pull down the File menu and open the New submenu. Since BBEdit uses different kinds of documents for specific purposes, you will see several options, as follows:



The available commands and their effects are as follows:

- Text Document: Opens an empty text document.
- (with selection): Opens a new text document containing any text selected in the active document and having the same display font, saving you the trouble of copying and pasting it.
- (with Clipboard): Opens a new text document and automatically pastes the contents of the current clipboard into it.
- HTML Document: Brings up a dialog with options for creating a new HTML document (see Chapter 11 for more information on working with HTML documents).
- Text Window: Opens a new text window (see “Text Windows” later in this chapter for more information).
- Project: Opens a new project window (see “Using Projects” later in this chapter for more information).
- Disk Browser: Opens a new disk browser (see Chapter 9 for more information).
- FTP/SFTP Browser: Opens a new FTP/SFTP browser (see later in this chapter for more information).

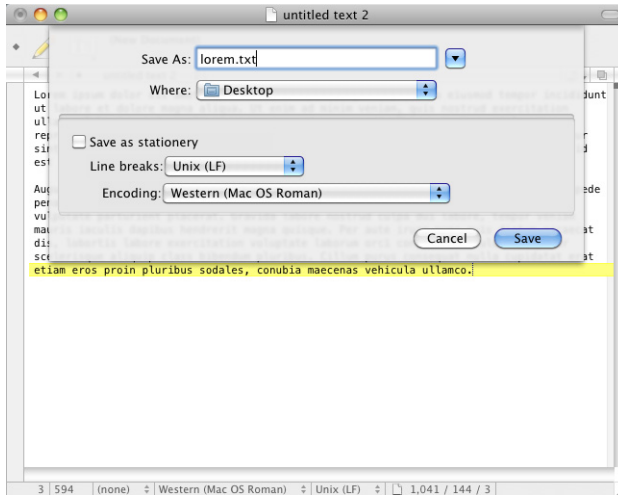
- Shell Worksheet: Opens a new shell worksheet using your default shell.
- Text Factory: Opens a new Text Factory window (see “Text Factories” in Chapter 5 for more information).

You can also create a new text document by selecting text in any application which supports the system Services menu, and choosing the New Window with Selection command from the BBEdit submenu of the Services menu. BBEdit will open a new text window containing a copy of the selected text.

When you want to save a new text document:

1 Choose the Save or Save As command from the File menu.

BBEdit opens a standard Save sheet:



2 Give the file a name.

3 Change any desired options (see below).

4 Click Save to save the file.

You can also create a new document from the selected text in any open window with BBEdit’s contextual menu. Simply Control-click the selected text and choose New (with selection) or Save Selection from the menu that appears. Depending on which command you choose BBEdit will either create a new editing window containing the selected text, or display the Save dialog and allow you to create a new file containing the selected text. The new file will use the same options (see “File Saving Options,” below) as those of the original parent document.

Saving a Copy of a File

You can save a copy of a file with BBEdit's Save a Copy command in the File menu. Just like the Save As command, the Save a Copy command displays a Save dialog and lets you choose a name and location for the file. However, unlike the Save As command, where BBEdit will start working with the new file you saved in place of the original, when you use Save a Copy, you create a new file in the designated location, but keep working with the original file.

For example, say you are editing a document called Test.c and use the Save a Copy command to save a document called Backup-Test.c. The next time you choose the Save command, BBEdit saves the changes to Test.c and not to Backup-Test.c.

File Saving Options

BBEdit's Save dialog is the standard Macintosh Save dialog with these additions:

Save As Stationery

When this option is on, BBEdit saves the document as a stationery pad file. When you later open the stationery file, BBEdit will use it as the basis of a new untitled document. The new document will inherit the contents and display settings of the stationery document, but BBEdit will prompt you for a name when you save it.

Line Breaks

The Line Breaks menu let you choose what kinds of line breaks BBEdit writes when you save the file. Choose:

- Unix line breaks (ASCII 10) for most purposes, including use with modern Mac applications, or for files being saved to a Unix file server. This is the default option.
- Classic Mac line breaks (ASCII 13) if you will be using the file with other applications which expect this format.
- Windows line breaks (ASCII 13/10) if the file resides on a Windows file server or if you will be sending it to someone who uses a Windows- or DOS-based system.

Encoding

BBEdit lets you save documents using any character set encoding supported by Mac OS X, including a variety of Unicode formats (see “Saving Unicode Files” on page 54). To select an encoding, choose its name from the Encoding pop-up menu. The list of available encodings is controlled by your preference settings (see “Text Encodings Preferences” on page 209).

When you select an encoding that requires a Unicode file format, you can also choose “Unicode” as an option from the Line Breaks pop-up menu in this dialog. (Unicode has its own line-ending standard.)

UTF-16 files created by BBEdit are given a type of ‘utxt’—the Mac standard type for Unicode text files. UTF-8 files are given a type of ‘TEXT’ for compatibility with other applications; however, BBEdit will also recognize such files with type ‘UTF8’.

Note You can choose which encodings appear in the Encoding pop-up menu in the Text Encodings preference panel.

File State

If you modify a document's window position or display settings and then save the document, BBEdit stores state information, which it will use to reopen that document in the same manner.

BBEdit captures only those settings which are fundamental to the document (window position, selection range, folds, splitter setting), or any settings which vary from the global preferences. (The latter ensures that changes to the global preferences are never inappropriately overridden by stored display options derived from prior global or default preference settings.)

For example, say BBEdit's default display font is Consolas for BBEdit, and you open (or create), save, and close a document which uses that font. If you then change BBEdit's default display font to Menlo before reopening that document, the document will display in Menlo.

Note The above example uses the display font option for illustration, but the same principle applies to any document display option which derives from BBEdit's global preferences.

Emacs Local Variables

Emacs (the popular Unix text editor) supports a convention in which you can define Emacs-specific settings in a block of text near the end of the file, or in the first line of the file. This convention helps maintain consistency when sharing files among a group of people, or across multiple systems.

For general information on Emacs variables, please see the GNU Emacs manual:

http://www.gnu.org/software/emacs/manual/html_node/emacs/Specifying-File-Variables.html

BBEdit will read and honor the “coding”, “tab-width”, and “x-counterpart” variables in any file which contains an Emacs variable block, and adjusts the value of the “coding” variable if you change the document's encoding by using the Encoding popup.

If a file contains an Emacs variable block (or line) having a “mode” variable, BBEdit will attempt to match the mode name against all currently recognized languages, before attempting to match the file name suffix or guess based on the file's contents.

You may add an Emacs variable block (or lines) to any document either directly, or by selecting the Emacs Variable Block command from the Insert submenu of the Edit menu.

Here is an example variable block from a plain text file:

```
Local Variables:  
coding: ISO-8859-1  
tab-width: 8  
End:
```

You may also add the BBEdit-specific variable “make-backup-files” to control whether or not BBEdit should back up a given file. For more details, please see “Controlling Backups with Emacs Variables” on page 211.

Saving with Authentication

BBEdit supports saving files that require administrator privileges, if you possess the necessary user and password information to enable this. For example, you can edit and save files that are owned by, and only readable by, the “root” user. Authenticated saving is particularly useful in conjunction with the “Show Hidden Items” option in the Open dialog, which allows you to see and open files in hidden folders (like /bin and /usr).

When you open a file for which you do not have write privileges, BBEdition will display a slash through the pencil icon in the toolbar. To edit the file, click the pencil icon. BBEdition will prompt you to confirm whether you wish to unlock the file. (Option-click the pencil icon to skip the confirmation dialog.)

When you are finished editing, simply choose Save from the File menu. BBEdition will prompt you to authenticate as a user with administrator privileges. Type a suitable user name and password to save the file.

Saving Compressed Files as bz2 or gzip

BBEdit transparently supports opening, browsing, and saving files compressed in the ‘bz2’ and ‘gzip’ formats. To save a file with gzip compression, simply append an filename extension of “.bz2”, “.gz”, or “.gzip” when creating it (or doing a Save As of an existing document). (For more information on these formats, issue the commands ‘man bz2’ or ‘man gzip’ in the Terminal.)

Saving as Styled Text or HTML

If syntax coloring is active, BBEdition can generate rich text (RTF) or HTML markup which replicates the visual appearance of the current document’s contents. You can save a file containing RTF via the Save as Styled Text command or a file containing HTML via the Save as Styled HTML command (both in the File menu), or copy the current selection to the clipboard as styled text via the Copy as Styled Text command, or as HTML markup via the Copy as Styled HTML command (both in the Edit menu).

Crash Auto-Recovery

IMPORTANT

BBEdit automatically saves auto-recovery information for all unsaved open documents at the specified interval. When you relaunch BBEdition after a system or application crash, BBEdition will reopen and restore the contents of any documents for which recovery information is available.

BBEdit’s auto-recovery mechanism can help minimize the chance of data loss in the event of unexpected system or application crashes. However, it may not protect against extraordinary events, and it will not protect against hardware failures or any other events that render your disk unreadable. You should always manually save a document after making any significant changes to it, and we strongly recommend that you take appropriate measures to back up your important files and other data.

Opening Existing Documents

There are several ways to open existing documents with BBEdit.

- Double-click any file with a BBEdit document icon.
- If BBEdit is running, choose the Open or Open Recent command from the File menu.
- Select the name of a file in a BBEdit editing window; then use the Open Selection command in the File menu.
- Double-click a file name in a browser's file list (see Chapter 9, "Browsers").
- Drag a file's icon to the Windows palette (see Chapter 6, "Working with Windows").
- Drag a file's icon to the file list of any editing window (see Chapter 4, "Window Anatomy").
- Drag a file's icon to the BBEdit icon or to an alias of the icon.
- Select a file in the Finder, and use the Open File command from the BBEdit submenu of the Services menu.

BBEdit can natively open files with type 'TEXT', 'utxt', and 'UTF8'. By default, BBEdit will attempt to display the contents of image files via QuickTime, but will open PDF files in a "raw" condition as if they were text documents. You can adjust how BBEdit should handle such files via its expert preferences. (See the "Expert Preferences" page of BBEdit's built-in Help book for complete details.)

Choosing the Encoding for a Document

When you open a document, BBEdit will automatically examine its contents for any indication of the proper encoding, and attempt to handle it appropriately. If BBEdit cannot determine the proper encoding, and you opened the file with the Open command, it uses the encoding specified in the Read As pop-up menu on the Open dialog. Otherwise, it uses the encoding specified by the "If the file's encoding can't be guessed, use" preference setting in the Text Encodings preference panel.

Note You can choose which encodings appear in the Read As pop-up menu by using the Text Encodings preference panel.

Here are the details of the steps that BBEdit goes through to determine the proper encoding for a file:

- 1 If the file is well-formed HTML or XML, BBEdit looks for an "encoding=" or <meta charset=> directive.**
- 2 If the file contains an Emacs variable specifying its encoding, BBEdit will use that encoding.**
- 3 If you have opened the file with BBEdit before, BBEdit will use the file's stored encoding info (if any).**
- 4 If the file contains a UTF-8 or UTF-16 (Unicode) byte-order mark (BOM), BBEdit opens it as that type of Unicode file.**

- 5 If the file has a resource that contains font information (such as a ‘styl’ resource) and that resource specifies a multi-byte font, BBEdit opens the file as a Unicode file.
- 6 If you are opening the file with the Open command, BBEdit uses the encoding specified Read As pop-up menu on the Open dialog.
- 7 If the file contains no other cues to indicate its text encoding, and its contents appear to be valid UTF-8, BBEdit will open it as UTF-8 without recourse to the below preferences option.
- 8 Finally, it uses the encoding chosen for the option “If the file’s encoding can’t be guessed, use” from the pop-up menu in the Text Encoding preference panel.

To change the encoding for a file after opening it, use the Text Encoding popup in the document’s status bar.

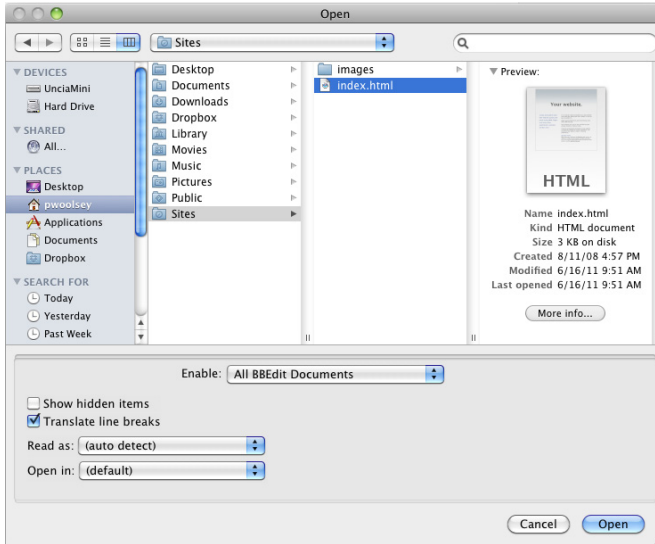
Note If an encoding change results in the conversion of a document’s contents from a single-byte script to a multi-byte script, BBEdit will mark the document as being “dirty” or changed.

Using the Open Command

To open a file with the Open command:

1 Choose Open from the File menu.

BBEdit displays the Open dialog box:



2 Select the file you want to open.

You can select (or deselect) multiple files by holding down the Command key or the Shift key as you click the files.

3 Change any desired options (see below).

4 Click Open to open the file.

You can use the options described below when you open a file.

Enable Menu

This popup menu lets you choose what kinds of files can be selected in the Open dialog. If you know a file contains text, but it does not appear in the Open dialog, this means the file does not have a recognized filename extension or any other property which would allow BBEdit or the system to recognize that it contains text. This is sometimes the case with files received from other computers or downloaded from the Internet. Choose “Everything” to view all available files without restriction.

BBEdit will first check the file’s suffix against its own language mappings (as shown in the Languages preference panel) and if the suffix matches up with any language (even if that language is “None”), BBEdit will assume that file to be a text file. Thus, you can use BBEdit’s suffix mappings to make it to treat any file as text which the system does not recognize. If BBEdit cannot match a file’s suffix (or its name) to a known language, BBEdit will next check to see if the system recognizes that file’s content type.

Show Hidden Items

When this option is selected, the Open dialog will display invisible files and folders. The setting you choose will persist until you change it.

Translate Line Breaks

When this option is selected, BBEdit translates Windows or Unix line breaks when opening a file. Otherwise, BBEdit leaves the original line breaks untranslated.

Unlike the other options in the Open dialog, the setting of this option is not preserved between uses of the Open command, since in general you will only need to use this operation temporarily, e.g. to read in a particular file.

Read As

When opening a file, you can tell BBEdit what encoding to use by choosing it from this pop-up menu. Usually, BBEdit will correctly auto-detect the encoding, but if it does not, you can try applying the Reopen Encoding command with an appropriate encoding. Chapter 5 includes more information on encodings.

Open In

When opening one or more files, you can use the options on this pop-up menu to override your default document opening preferences. These options have the following effect:

- (default): BBEdit will open the selected documents according to your preference settings.
- Front Window: BBEdit will open all of the selected documents into the frontmost text window. If there are no text windows open, or the frontmost text window contains an active sheet, this option will be disabled.
- New Window: BBEdit will open all of the selected documents into a new text window.
- Separate Windows: BBEdit will open each of the selected documents into its own text window.

Reload from Disk

When you choose this command, BBEdit will compare the contents of the current document in memory to those of its file on disk, and reload the document from its file if they differ. This is useful in situations where the file may have changed without BBEdit noticing, which can happen if, e.g. the “Automatically refresh documents” option in the Application prefs panel is turned off, or if the file is on a shared disk and has been modified from another workstation.

Opening and Editing Files within Zip Archives

BBEdit transparently opens and displays the contents of any Zip-compressed archives (“.zip”) both directly and during multi-file search. In addition, you can directly edit the contents of files within such an archive and save changes directly back to those files. (Zip archives must be in the format created by the Finder’s “Compress” command, or by applying ``ditto -k`` from the command line.)

Note If the Zip archive contains only one top-level item, and that item is a folder, BBEdit will “hoist” the rest of that package’s contents and not display the top-level item.

Opening bz2, gzip, and tar Files and Binary plists

BBEdit transparently opens and displays the contents of any bz2 or gzip-compressed files (“.bz2”, “.gz” and “.gzip” files), as well as tarballs (“.tar” files) and binary plists (“.plist” files), both directly and during multi-file search.

This is especially useful for viewing and working with system log files and similar automatically-generated files, as well as system and application preference files.

If you make any changes to such a file and save it, BBEdit will automatically re-compress or re-convert the file on save.

Opening Hidden Files

Turn on the “Show Hidden Items” option in the Open dialog to display hidden files (including both files whose invisible attribute has been set, and those whose names begin with a period) or files from a folder which is normally hidden by the system.

Using the Open from FTP/SFTP Server Command

See “Accessing FTP/SFTP Servers” on page 56.

Using the Open Selection Command

The Open Selection command lets you open a file that is referenced in the text of a document. It is particularly useful for opening include files or any document referenced by another file.

To open a file whose name is referenced in the text of a document:

- 1 Select the file name within the body of the document.**
- 2 Choose Open Selection from the File menu.**

If a suffix of the form “.x” follows the name, BBEdit will automatically expand the selection to include the suffix.

BBEdit also understands the Unix-style line number and character offset specifications “:line:offset” that can be appended to a file name, and will honor them when opening a file. If the specified file is already open, this command will simply select the designated location within the file. (These specifications are frequently generated by Unix command line tools.)

For example, selecting the text “main.cp:210” and choosing Open Selection will open the file “main.cp” and automatically select line 210, while selecting the text “foo.cp:398:43” will open the file “foo.cp” and automatically position the insertion point at the specified location.

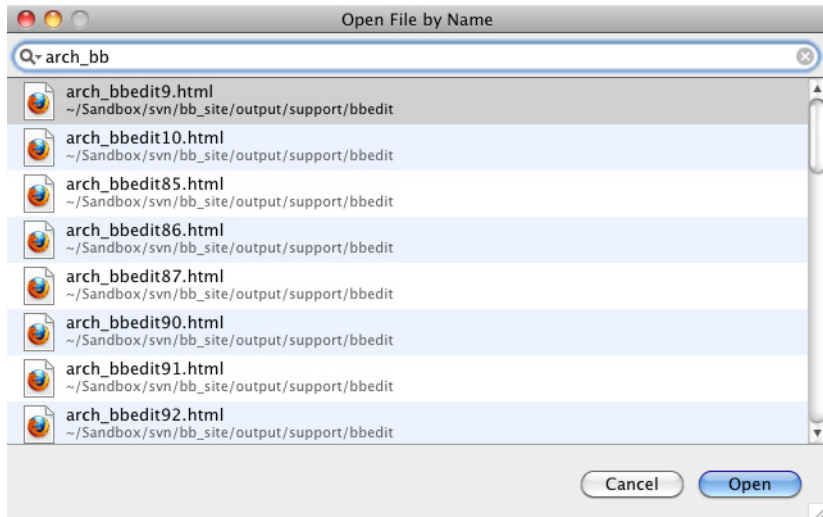
In searching for the requested file, BBEdit will look in the following locations, in order:

- If a project document is open, BBEdit will start its search within the project, looking first at discrete file entries and then (if necessary) searching any folders contained within the project. (If multiple projects are open, BBEdit will use the frontmost project.)
- If the Open Selection is being done from a shell worksheet, BBEdit will search the shell's current working directory, followed by any subdirectories within it.
- Otherwise, BBEdit will look first in the same folder as the file containing the selected file name, and then in any subfolders within that folder.
- If the file containing the selection is contained within one of the web sites defined in the Site panel of the Setup window, BBEdit will next search the “Templates & Includes” and site root folders, if any, associated with that site.
- If BBEdit cannot find the file in any of these places, it will display a Choose Folder dialog to allow you to locate the file manually.

In some cases, there may be more than one file with the same name in the various folders BBEdit looks in. Normally, BBEdit opens the first such file it encounters, and then stops.

Using the Open File by Name Commands

If there is no selection, or there is no text display view in the front window, Open Selection becomes Open File by Name. Choosing this command brings up the Open File by Name window.)



Activating the Open File by Name window, or choosing the menu command, will place keyboard focus in the search box and select its contents, so that you can just start typing. (To clear an existing entry, click the “clear” widget at the right-hand edge of the field.)

As you type, BBEdit will search for files matching the current string as well as wildcard matches, and present a list of possible matches in the bottom panel of the window. If the string you enter contains wildcard characters (see below) then BBEdit will treat it as a wildcard pattern. If the string you enter does **not** contain wildcards, BBEdit will instead use it as a basis for casting a wide net.

BBEdit will first look for matches in one of two possible places, in order of preference: all files in the frontmost (Z-order) project document; and all files in the active Xcode project (if applicable). Next, if Xcode is running, BBEdit will look for files in the system frameworks includes.

You can navigate the list of potential matches by using the up and down arrow keys or the mouse pointer, and open any listed file by selecting it and typing Return or Enter, or clicking the Open button.

If BBEdit does not locate any potential matches, you can still search for the file, as before. (The search will skip locations where such a file would have already been found, i.e. the frontmost project.)

If you type a string which appears to be an absolute or a home-relative path (e.g. “/path/to/some/file.txt” or “~/Documents/some/file.txt”, BBEdit will cease searching and when you type Return or Enter, or click the "Open" button, BBEdit will attempt to open the file at that path, if it exists.

If you type a string which appears to be a URL, BBEdit will attempt to open it directly, or hand it off to an application that can. (BBEdit supports a number of schemes, including ‘file’, ‘http’, ‘ftp’, and ‘sftp’.)

BBEdit also maintains a search history in the Open File by Name window: when you open a matched item, BBEdit will store the string you used, and the search history (magnifying glass) popup lists these recently used strings.

You may use the following wildcards as part of a search string:

| Wildcard | Meaning |
|----------|---------------------------------------------------------------------------------------------------------|
| ? | Any single character |
| * | Any number of characters |
| # | Any numeric character |
| \ | Escapes one of the above; for example, \? enters a question mark. To enter a literal backslash, use \\. |

Using the Open Counterpart Command

You can use this command or its default key equivalent of Command-Option-uparrow (configurable via the Menus & Shortcuts preference panel) to switch between counterpart files (from source to header and vice versa). In addition to intrinsic counterparts (e.g. C/C++ style header/source mapping), you can explicitly define counterparts for a language via the Suffix Mappings section of the Languages preference panel.

You can also override BBEdit's default rules for switching between counterpart files by setting a value for the (BBEdit-specific) "x-counterpart" variable in a file's Emacs variables. For example, if your file contains the following as part of its variable block:

```
-*- x-counterpart: ExampleStrings.R; -*-
```

when you type Command-Option-uparrow, BBEdit will look for the file "ExampleStrings.R".

Using the Open Recent Command

The Open Recent submenu contains a list of files you have opened recently. To open one of these files, choose it from the Open Recent submenu. To set the number of items displayed in the Open Recent list, use the "Remember the [] most recently used items" option on the Application preference panel.

Using the Reopen using Encoding Command

The Reopen using Encoding submenu contains a list of all available text encodings. To reopen the current text document and have its contents interpreted using a different encoding, choose the desired encoding from the Reopen using Encoding submenu. This command will only be available if the current document is unmodified.

Quitting BBEdit

By default, whenever you quit BBEdit or BBEdit automatically quits because of a system shutdown, restart, or user account logout, BBEdit will attempt to restore as much of its state as possible when starting back up. Thus, you may not be prompted to save new or unsaved documents, since BBEdit will automatically preserve the contents of all open documents before it exits.

You can control whether BBEdit should preserve and restore unsaved changes via the "Restore unsaved changes" option in the Application preference panel existing documents. If this option is on, BBEdit will automatically preserve unsaved changes. If this option is off, BBEdit will instead prompt you to save each document which has unsaved changes.

An International Text Primer

Mac OS X includes extensive support for working with international text, including Unicode. If you have enabled additional text input methods in the International section of the System Preferences, you will see the Input menu on the right-hand side of the menu bar. This menu allows you to change keyboard layouts or script systems as you work.

Note Actually, even if you have never used a non-Roman script system before, you may still have used this menu, if you have ever chosen an alternate keyboard layout such as Dvorak, or a keyboard layout for a Roman language such as French. However, since the Roman script is suitable for several languages, choosing one of these keyboard layouts still leaves you in the Roman script.

International Text in BBEdit

As a text editor, BBEdit supports only one font per document window, though it can display all available characters in the active font, including Unicode characters.

BBEdit supports editing in almost any language which uses left-to-right text input methods. To start entering text in any supported language, choose a suitable input method from the Input menu. The icon for that method will appear in the menu bar in place of either the American flag (for the U.S. English layout) or the icon for your usual Roman keyboard layout.

If you have turned off the “Try to match keyboard with text” option in the Options dialog of the International section of the System Preferences), you may also need to select a suitable display font via the Font panel. (We recommend leaving this option on, so that BBEdit can automatically switch to the correct input method when you change document windows.)

You can use international text throughout BBEdit—for example, in the Find window, in the HTML Tools, and everywhere else you would use Roman text. Likewise, BBEdit will provide the necessary style information so that if you copy and paste, or drag and drop, international text into another application, that application will have enough information to handle the text correctly (assuming it is capable of doing so).

BBEdit remembers the encoding used in a document when you save it, so the next time you open it, you will not need to choose the font. However, you may not be able to read files which do not have this stored information, for instance, files downloaded from the Internet, until you choose an appropriate encoding for them.

When performing a search, BBEdit respects any available information about each file’s encoding. If a file does not contain any information about its encoding, BBEdit will use the default encoding set in the Text Encodings Preferences panel.

Unicode

Unicode is an international standard for character encoding, which includes an extensive selection of characters from Roman, Cyrillic, Asian, Middle Eastern, and various other scripts. For more background information or complete details on Unicode, the Unicode Consortium web site is the best place to look.

<http://www.unicode.org/>

BBEdit fully supports and makes extensive use of Unicode, in addition to all other OS-supported text encodings. In particular, BBEdit internally represents all documents as Unicode, regardless of their on-disk encoding.

Saving Unicode Files

BBEdit lets you save documents that use character set encodings other than Mac Roman, even multi-byte character sets. When saving a file, you can choose to save text composed in any script with any encoding. In addition to the standard character set encodings, BBEdit also lets you save the files in a variety of plain Unicode files:

- **Unicode (UTF-8):** UTF-8 **without** a byte-order mark
- **Unicode (UTF-8, with BOM):** UTF-8 **with** a byte-order mark (BOM)

- UTF-16 Little-Endian
- UTF-16 Little-Endian, no BOM
- UTF-16
- UTF-16, no BOM

IMPORTANT

The naming convention BBEdit follows for UTF-8 documents has changed from that used by versions before 9.5: the encoding name “Unicode (UTF-8)” now refers to files **without** a byte-order mark (BOM), while the specific name “Unicode (UTF-8, with BOM)” refers to files which have a BOM.

Here are details about what each of the above options means:

- UTF-8: UTF-8 encoding is a more compact variant of Unicode that uses 8-bit tokens where possible to encode frequently used sequences from the file. (This format makes it easier to view and edit content in non-Unicode-aware editors.)
- UTF-16: UTF-16 encoding always uses 16-bit tokens.
- BOM: When saving Unicode files, you may include a byte-order mark (BOM) so that the reading application knows what byte order the file’s data is in. However, since many applications do not correctly handle files which contain BOMs, you may wish to use an encoding variant without a BOM for maximum compatibility. (For purposes of recognition when you use this option, the UTF-16 BOM is FEFF, and the UTF-8 BOM is EFBBBF.)
- Little-Endian: Since UTF-16 uses two bytes to represent each character, this leaves the question of which of the two bytes comes first—whether it is “little-endian” or “big-endian.” By default, BBEdit writes UTF-16 big-endian (the standard). By choosing one of the “Little-Endian” (or “byte-swapped”) encodings, you can write little-endian files instead, which some Windows software requires.

Files saved as Unicode from BBEdit are given a type of ‘utxt’—the standard for Unicode text files on the Mac. UTF-8 files are given a type of ‘TEXT’ for compatibility with other applications; however, BBEdit will also recognize such files with type ‘UTF8’.

Opening Unicode Files

When opening files, BBEdit will ordinarily determine the format of a file based on its file type and content, and automatically process Macintosh text, Unicode, and UTF-8. However, some files are structured such that BBEdit is unable to correctly determine their format based on their type or contents. The cases that we know of are:

- UTF-8 files whose type is ‘TEXT’ but which lack a byte-order mark and do not contain any encoding specification or any extended characters. (If a UTF-8 file is of type ‘TEXT’ but has a byte-order mark, it will be correctly interpreted as UTF-8.)
- Byte-swapped Unicode files which were written without a byte-order mark (usually by broken Windows software);
- Unicode files whose type is ‘TEXT’ (instead of the Macintosh standard ‘utxt’) *and* which lack a byte-order mark. (If a UTF-16 file lacks a BOM but is of type ‘utxt’, BBEdit will treat it as big-endian Unicode.)

If you know that a file you are trying to open is in Unicode but it displays as gibberish on your screen, close its window without saving. Then try reopening the file, using the Open As pop-up menu in the Open dialog to specify whether to treat the file as Unicode, byte-swapped (little-endian) Unicode, or UTF-8.

If you attempt to open a document which cannot be represented by either its declared encoding or any recognizable encoding, BBEdit will present an alert to warn you. Also, if BBEdit encounters such a file during a multi-file search, it will log a warning.

Accessing FTP/SFTP Servers

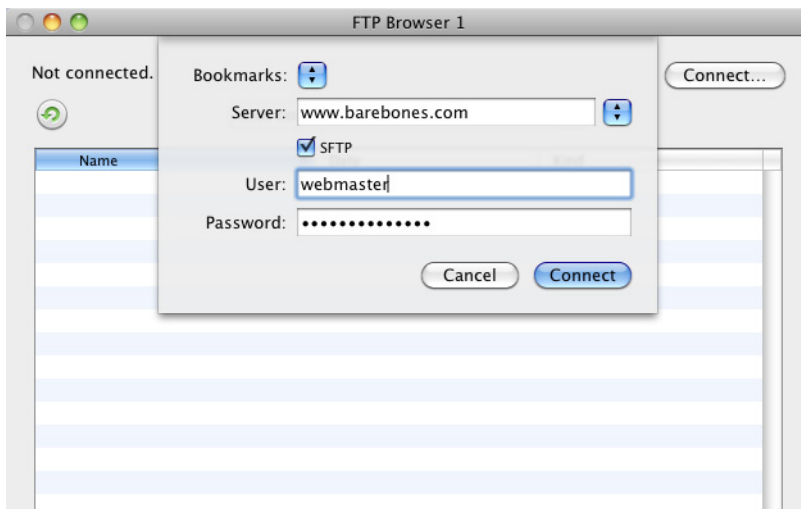
BBEdit can open files directly from, and save them to, any available FTP server. It can also open and save files directly via SFTP (SSH File Transfer Protocol). In order to access a server via SFTP, that server must be running a compatible version of `sshd`. (A great many machines, including Mac OS X systems for which “Remote Login” is turned on in the Sharing panel of System Preferences, satisfy these criteria.)

Aside from choosing the SFTP checkbox in the Open from.../Save to... dialogs, or the FTP/SFTP Browser, opening and saving files via SFTP works just like it does when using ordinary FTP. A file opened via SFTP will appear in the Open Recent submenu with an “sftp:” URL, and you can send a “get URL” event to BBEdit with an “sftp” URL as well.

Opening Files from FTP/SFTP Servers

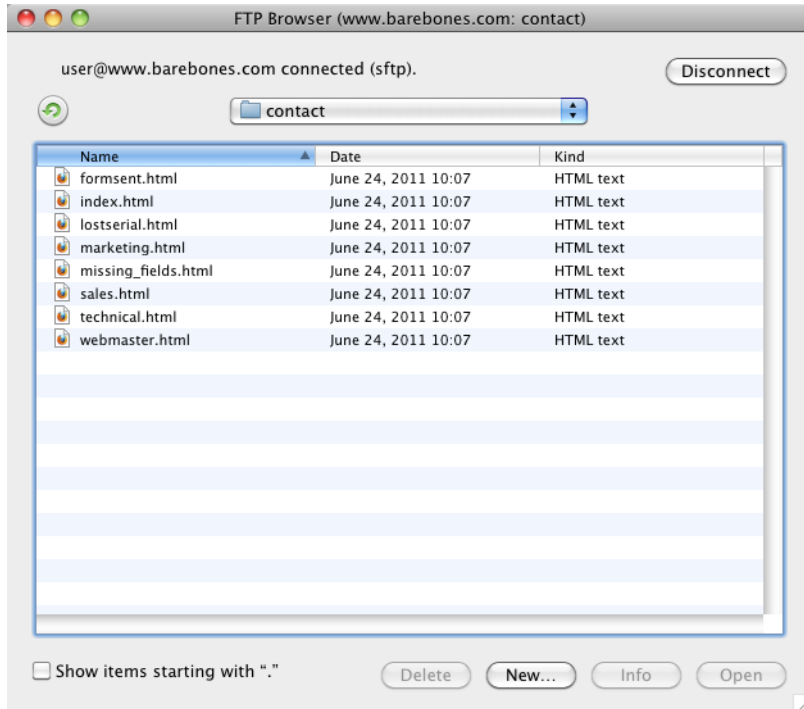
To directly open files from an FTP or SFTP server, choose Open from FTP/SFTP Server from the File menu. BBEdit will open a new FTP/SFTP Browser window. Like other browser windows, FTP/SFTP browsers will remain open until you close them, and once connected, they will maintain a persistent connection to the server for as long as they remain open.

Enter the server's name in the "Server:" field, or choose a local server advertised by Bonjour by clicking the popup menu to the right of the "Server:" field, specify your user name and password in the appropriate fields and choose the "SFTP" option if appropriate; then click the Connect button or press the Return or Enter keys to connect to the server.



Alternatively, you can choose a bookmark from the Bookmarks pop-up menu to fill in stored info for the server, user name, password, and connection options. You can create bookmarks by entering the appropriate information in the Open from... or Save to... dialogs and choosing Add from the Bookmarks pop-up menu, or via the Bookmarks list in the Bookmarks panel of the Setup window. You can modify or delete existing bookmarks via the Bookmarks panel of the Setup window.

Once you have connected to the server, you can open files by double-clicking them, or selecting them and clicking the Open button. You can double-click a folder to change directories. If you hold down the Option key when opening a folder, it will open in a new FTP/SFTP Browser window. You can select a range of files and directories by Shift-clicking, and you can select (and deselect) multiple items one at a time by Command-clicking them.



To refresh the directory listing, click the button with the circular arrow icon (located above the upper left corner of the listing). The checkbox below the listing labeled Show Files Starting with “.” tells BBEdit whether to display hidden or admin files in the chosen directory, such as .login, .forward, and .signature. (Starting a file name with a period is a convention used by Mac OS X and other Unix systems to make that file invisible in most directory listings.)

Once you have selected a file and opened it, BBEdit displays the file in a text editing window. The toolbar displays the URL of the file on the server, not the pathname of the file on your hard drive as it does for local files.

NEW You can drag items from FTP browser windows to other applications. BBEdit will include a URL in the drag event for each selected item in a form that applications which accept URLs may be able to use.

You can use the Info button to examine the size, modification date, and if applicable, file system permissions of the selected file. You can edit the file’s name and click the Rename button to rename the file on the server; you can also make changes to the permissions and click the Set button to change them. (Take care not to set the permissions such that the file becomes inaccessible to you!)

You can directly create a new file (or folder) on the server by clicking the New button, or remove files from the server by selecting them and pressing the Delete button.

Specifying Alternate Ports

BBEdit allows you to open an FTP or SFTP connection on ports other than the default. To specify an alternate port, place it at the end of the server name, separated by a colon—for example, ftp.example.com:1111.

Storing Passwords

As long as your Mac OS X Keychain is unlocked, BBEdition will use it to store the password for each server that you access, and to automatically fill in the corresponding password whenever you enter a server and user name pair for which there is a Keychain entry. If your Keychain is locked, you will need to retype your password every time you use the FTP browser.

Using SSH Key Files

In order to connect to an SFTP server which requires SSH keys (or certificates) rather than passwords, you must first create an appropriate entry for that server in your local account's .ssh/config. You may then type the server name, or shortcut name, into the Server field of the FTP/SFTP Browser and connect without entering a password.

Transfer Formats

When you open a file from an FTP or SFTP server, BBEdition downloads the file “raw” (in binary mode) and then performs a standard line ending conversion upon opening the (local temp) file.

Working with URL Clippings

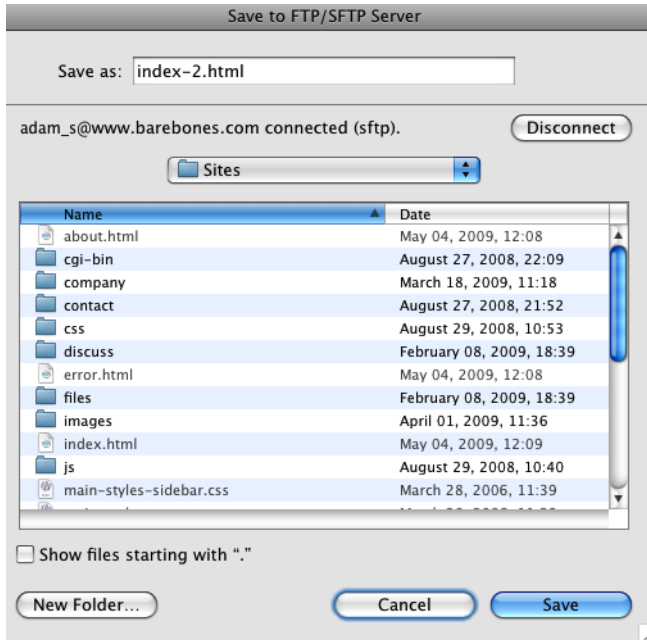
BBEdit also supports FTP and SFTP URL clippings. You can make a clipping of the FTP or SFTP URL for a file, add the clipping to a project, and double-click it, and BBEdition will open the specified file for editing. If the clipping contains the URL for a directory, BBEdition will open a new FTP/SFTP Browser at that location. Alternatively, you can double-click an FTP or SFTP clipping in a disk browser, or drop one on BBEdition's icon in the Finder, with the same results as just described.

Dragging the window proxy icon from the editing window of a file open from an FTP or SFTP server will create a clipping containing that file's URL.

Saving Files to FTP/SFTP Servers

After you have edited a file opened from an FTP or SFTP server, pressing Command-S or choosing Save from the File menu saves the new version back to the server. If you want to save the file in a different directory or under another name, choose Save to FTP/SFTP Server to open the Save to FTP/SFTP Server dialog (shown below).

This dialog works much like the standard Save dialog for saving a local file, with the addition of fields and controls similar to those in the FTP/SFTP browser allowing you to select or specify connection info, and to navigate and obtain info about other files.



Note When you save a file to an FTP or SFTP server using either Save or Save to FTP/SFTP Server, and the file has Unix (LF) or Windows (CR+LF) line endings, BBEdit uploads the file in binary mode, preserving its line endings exactly as they are on your local machine. However, if the file has Macintosh (CR) line endings, it is uploaded in text mode so that the server can convert the line endings as appropriate.

Finally, you can use Save a Copy to FTP/SFTP Server to upload a copy of your current file to an FTP server while keeping your local file open. This is especially useful when you maintain web site content on your local hard drive and only need to upload changes made in one or two files to the server.

Using BBEdit from the Command Line

You can use the “bbedit” command line tool to open files into BBEdit via the Unix command line. The first time you run BBEdit after installation, it will offer to install the command line tools for you. If you choose not to do so, you can choose “Install Command Line Tools” from the BBEdit (application) menu at any time to install (or re-install) the current version of the command line tools.

To open a file in BBEdit from the command line, type

```
bbedit filename
```

where *filename* is the name of the file to be opened. To launch BBEdit without opening a file (or activate it, if it is already running), type

```
bbedit -l
```

In addition to files, you can also specify FTP or SFTP URLs to files or directories, to have BBEdit open the specified files, or an FTP/SFTP Browser for each directory. You will be prompted to enter passwords if necessary.

You can also pipe STDIN to the “bbedit” tool, and it will open in a new untitled window in BBEdit: for example,

```
ls -la | bbedit
```

If you just type

```
bbedit
```

with no parameters, the tool will accept STDIN from the terminal; type Control-D (end-of-file) to terminate and send it to BBEdit.

The complete command line syntax for the “bbedit” tool is

```
bbedit [ -bcChlpSuvVw --(long_form_switches) ]  
[ -e <encoding_name> ] [ -t <string> ] [ +<n> ]  
[ file (or) <S/FTP URL> ... ]
```

See the “bbedit” tool’s man page (‘man bbedit’) for a complete description of the available switches and options.

Using Projects

If you frequently work with many related files, you may want to create a project for them. A project is a special kind of BBEdit file that contains references to other files and folders, including aliases and URL clippings.

You can use a project to gather and directly edit related files, as well as to serve as an organizational or navigational aid. For example, you can perform multi-file searches on the contents of projects, or process a project's contents with a text factory.

IMPORTANT

Due to a considerable expansion in the features now offered by project documents, the project document format used by BBEdit 10.0 and later is not backward compatible with the file group documents created by BBEdit 8.7.2 or prior.

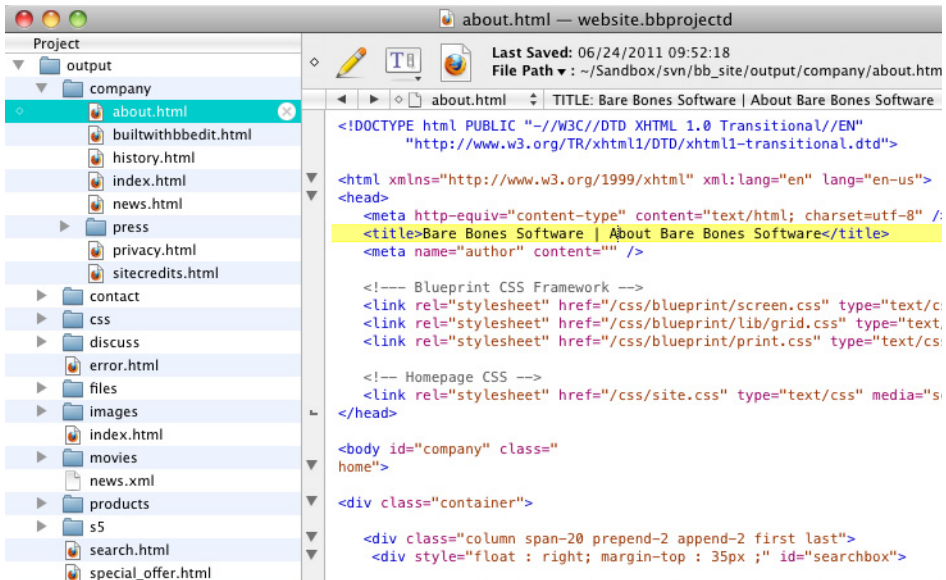
Creating a Project

To create a new project, choose the Project... command from the New submenu in the File menu. You will need to decide where to place the project file on disk; thereafter, the project document will autosave as necessary.

Alternatively, you can create a project from any open editing window by choosing the Save Project command in the File menu.

To add files or folders to a project, drag them from the Finder into the project window, or click the Add button. When you click Add, BBEdit presents the Open dialog in which you can choose one or more files and folders to add. The files and folders you select will be added directly to the project's file list, and you can drag them to reorganize their positions. You can also add a file by dragging its icon from a text window's toolbar or file list to the group window, or by dragging a file entry from any disk browser or results browser.

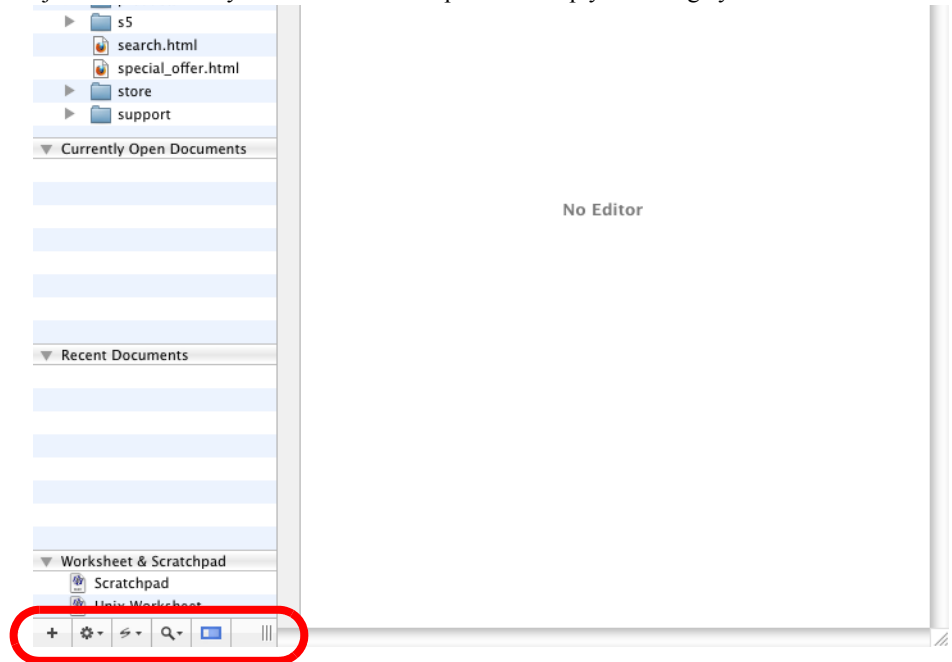
You can display the contents of a folder by clicking the adjacent disclosure control. Each folder's listing is a "live" representation of its contents on disk.



In addition to file and folder references, BBEdit supports URLs in projects. You can drag a URL (text or clipping file) to a project window, and the URL will be saved in the group. If you subsequently open the item, BBEdit will either open that URL directly if it is an FTP or SFTP URL, or hand the URL off to a system-designated helper.



Project Commands

Projects offer a variety of commands and options to help you manage your files.



The buttons and popup menus at the bottom of the file list are commands that give you quick access to project functions. You can use these commands to add new items, create new files and folders, open existing files and folders, reveal them in the Finder or navigate to them in the Terminal, limit the kinds of files which appear in the file list, and navigate through your disks and folders.

The following table explains each button.

| Icon | Meaning |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Click the Add button (plus sign) to bring up a standard file and folder selection dialog in which you can choose items to add to the project. |
|  | The commands on the Action (gear) popup menu at the bottom of the file list pane allow you to open the selected items, reveal them in the Finder, copy their paths, navigate to their location in the Terminal, move them to the Trash, or create new files and folders. |

Icon Meaning



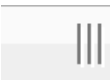
The Subversion popup menu allows you to apply various Subversion commands to the selected items. (If the currently selected item is not under Subversion control, the commands in this menu will be disabled.)



The Filter (magnifying glass) popup menu allows you to apply options to control what items BBEdition should display in the project's file list.



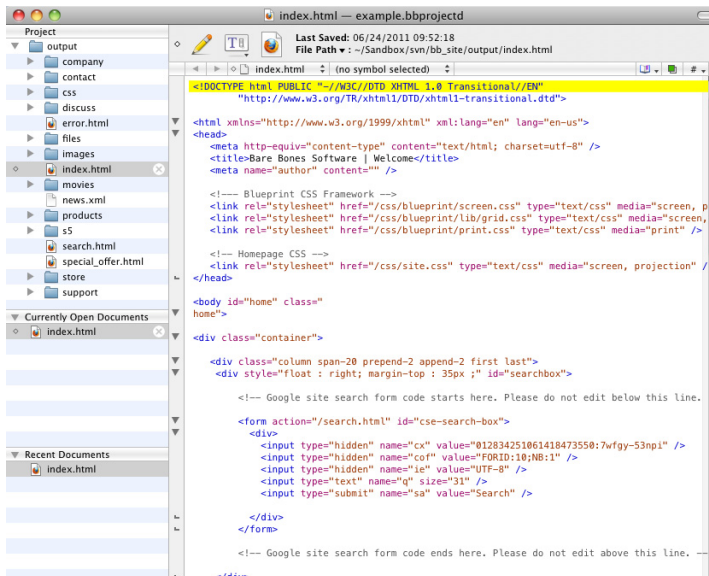
Click the Toggle Editor button to collapse or expand the project's text editing pane. (This button has the same effect as choosing the View/Hide Editor command in the View menu.)



Drag the Thumb button to expand or shrink the width of the file list pane.

Using projects

To open a text file within a project, just click on the file and BBEdition will display it in the editing pane. If you click on an FTP or SFTP URL clipping, BBEdition will open the remote file (or open an FTP Browser if the clipping points to a directory). Otherwise, BBEdition tells the Finder to open the file.



If you added nested folders, they appear in the project with disclosure triangles, as in a Finder list view. Click any folder's disclosure triangle to reveal the files and folders inside that folder.

You can use the project browser pane or the document list popup in the navigation bar to select any open file in the project.

You can use a project as the basis of a multi-file search. See Chapter 7, “Searching,” for more information

Removing Files from a Project

To remove an directly added file or folder from a project, select it and type Delete, or choose Remove from the Action popup.

You cannot directly remove files and folders which are part of a “live” folder listing from a project. However, you can move the original item(s) to the Finder trash by selecting them and choosing Move to Trash from the Action popup or contextual menu.

Contextual Menu Commands

If you select one or more items in the file list and bring up the contextual menu, BBEdit will offer commands to open the selection, copy the items’ paths (in the same fashion available via the "Copy Path" submenu), or apply selected Subversion commands (for items contained within a Subversion working copy).

Script Access to Project Contents

You can now access the items contained in a project document via AppleScript. Note that BBEdit enforces a strict containment hierarchy for such items. For instance, if you ask for “every item of project document 1”, BBEdit will return a list of every item that’s present at the top level of the project. You can recurse to explore the items contained within folders or collections.

Using Stationery

Like most Macintosh applications, BBEdit supports stationery pads. A stationery pad is a template document that, when opened, results in a new, untitled document with the content from the stationery file. In other words, you do not edit the stationery document itself; you use it as a starting point for a new document.

To create a stationery pad, click the Save As Stationery checkbox when saving the file from BBEdit. Alternatively, you can change any document into a stationery pad in the Finder by clicking the Stationery Pad checkbox in the document’s Get Info window.

You can create new documents from a stationery pad in any of these ways:

- Open the pad the same way you would open any other document.
- Choose New With Stationery from the File menu, and select the desired stationery pad from the contents of the Stationery folder (inside BBEdit’s application support folder).
- Use BBEdit’s Stationery List, which is available from the Window menu. The Stationery List is a palette that displays all the stationery pads you have placed inside the Stationery folder of BBEdit’s application support folder. You can create a new document from any of these pads by double-clicking them in this list.

To assign a keyboard shortcut to a stationery pad, select the pad in the Stationery List window; then, click the Set Key button, type the desired key in the Set Key dialog and click OK.

Manually Sorting the Stationery

By default, items in the Stationery List are displayed in alphabetical order. However, you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their names. For example, “00)Web template” would sort before “01)HTML Template”. For such files, the first three characters are not displayed in BBEdit. You can also insert a divider by including an empty folder whose name ends with the string “-***”. (The folder can be named anything, so it sorts where you want it.) These naming conventions are the same as those used by the utilities FinderPop and OtherMenu.

Hex Dump for Files and Documents

Choose the Hex Dump File command to generate a hex dump representation from a file that you choose. Choose Hex Dump Front Document to generate a hex dump representation of the frontmost document as it exists in memory.

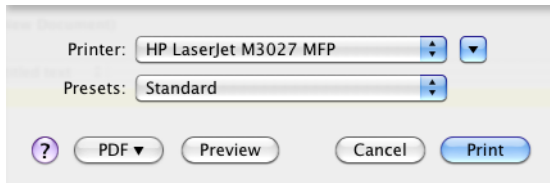
You should bear in mind that the result of performing the Hex Dump command against a disk file may differ from the result obtained by using it against an open document, since when a document is open in memory, even without any explicit edits being made, line-break translation and possibly character set encoding conversions have taken place.

Making Backups

BBEdit can automatically make a backup copy of each document you edit before saving it. To enable this feature, turn on the “Make backup before saving” option in the Text Files preference panel. For complete details on how this feature works, and optional behaviors, please see “Make backup before saving” on page 210.

Printing

To print a document, choose the Print command from the File menu. BBEdit will display a standard print sheet in that document’s window.

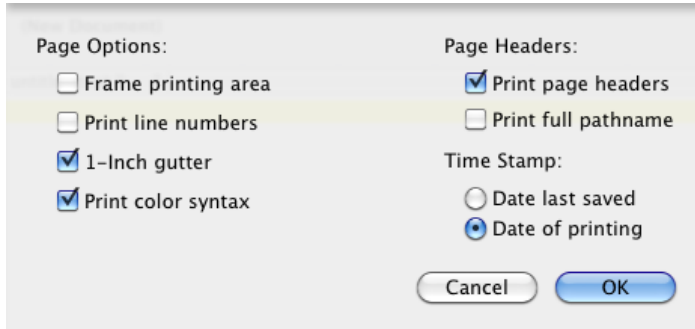


To print one copy of the active document without displaying the print sheet, hold down the Shift key and choose the Print One Copy command from the File menu.

To print only the selected range of text within the active document, choose the Print Selection command from the File menu.

Text Printing Options

You can access BBEdit's application-specific printing options for the current document by choosing the Text Printing command in the Edit menu. When you choose this command, BBEdit will open the printing options sheet.



Note You can set defaults for these options, as well as the printing font, in the Printing panel of BBEdit's Preferences window.

Page Options:

These options control how the printed pages will be laid out.

Frame printing area

When this option is selected, BBEdit draws a frame around the printed text.

Print line numbers

When this option is selected, BBEdit prints line numbers along the left edge of the paper.

1-Inch gutter

When this option is selected, BBEdit leaves a one-inch margin along the left edge of the paper. Use this option if you usually put your pages in three-ring binders.

Print color syntax

When this option is selected, BBEdit will print the document in color.

Page Headers:

These options control what information is included in the page headers.

Print page headers

When this option is selected, BBEdit prints the page number, the name of the file, and the time and date printed in a header at the top of each page.

Print full pathname

When this option is selected, BBEdit prints the full pathname of the file in the header.

Time Stamp

The Time Stamp option lets you choose whether the date that appears in the header is the date that the file was last modified or the date that the file was printed.

Editing Text with BBEdit

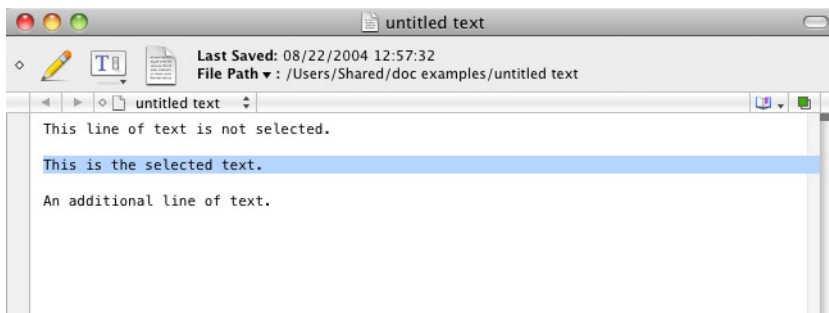
This chapter describes the basics of editing text with BBEdit, wrapping text, text manipulations, and file comparison.

In this chapter

| | |
|---------------------------------------------------------------------------------------|-----|
| Basic Editing | 70 |
| <i>Moving Text</i> – 71 • <i>Multiple Clipboards</i> – 71 | |
| <i>Drag and Drop</i> – 72 | |
| Multiple Undo | 72 |
| Window Anatomy | 73 |
| <i>The Toolbar</i> – 73 • <i>The Split Bar</i> – 74 • <i>The Navigation Bar</i> – 75 | |
| <i>The File List</i> – 79 • <i>The Status Bar</i> – 80 | |
| <i>The Gutter and Folded Text Regions</i> – 81 | |
| The View Menu | 83 |
| <i>Text Display</i> – 83 | |
| Cursor Movement and Text Selection | 86 |
| <i>Clicking and Dragging</i> – 86 • <i>Arrow Keys</i> – 87 | |
| <i>CamelCase Navigation</i> – 87 • <i>Rectangular Selections</i> – 87 | |
| <i>Working with Rectangular Selections</i> – 88 | |
| <i>Scrolling the View</i> – 90 • <i>The Delete Key</i> – 91 | |
| <i>The Numeric Keypad</i> – 91 • <i>Go To Line Command</i> – 92 | |
| <i>Function Keys</i> – 92 • <i>Resolving URLs</i> – 92 | |
| Text Completion..... | 93 |
| <i>Editing Options</i> – 94 | |
| Text Options..... | 93 |
| <i>Editing Options</i> – 94 • <i>Display Options</i> – 95 | |
| How BBEdit Wraps Text | 97 |
| <i>Soft Wrapping</i> – 98 | |
| <i>Hard Wrapping</i> – 99 | |
| The Insert Submenu..... | 101 |
| <i>Inserting File Contents</i> – 101 • <i>Inserting File & Folder Paths</i> – 102 | |
| <i>Inserting a Folder Listing</i> – 102 • <i>Inserting a Page Break</i> – 102 | |
| <i>Comparing Text Files</i> – 102 | |
| Comparing Text Files..... | 102 |
| <i>Compare Against Disk File</i> – 104 | |
| <i>Multi-File Compare Options</i> – 105 | |
| Using Markers | 106 |
| <i>Setting Markers</i> – 106 • <i>Clearing Markers</i> – 106 | |
| <i>Using Grep to Set Markers</i> – 107 | |
| <i>Using Grep to Set Markers</i> – 107 | |

Basic Editing

BEdit behaves like most Macintosh word processors and text editors. Characters that you type in an active window appear at the insertion point, a vertical blinking bar. You can click and drag the mouse to select several characters or words, and the selected text is highlighted using the system highlight color, which you can set in the Appearance panel of the System Preferences.



If you select some text and then type, whatever you type replaces the selected text.

To delete selected text, press the Delete key or choose Clear from the Edit menu. If you have a keyboard with a numeric keypad on it, you can press the Clear key on the keypad to delete the selected text.

In addition to clicking and dragging to select text, you can use the selection commands in the Edit menu.

| To select... | Choose this from the Edit menu... |
|--------------------------------------|---------------------------------------------------------|
| All text | Select All |
| No text (deselect) | (click anywhere in the document, or type any arrow key) |
| Line containing insertion point | Select Line |
| Paragraph containing insertion point | Select Paragraph |

You can then cut, copy, or perform any other action that affects the selected text.

Note BBEdit defines a paragraph as a block of text surrounded by blank lines (lines containing no characters other than tabs or spaces). The beginning and end of the document also mark the beginning and end of paragraphs.

Moving Text

To move text from one place to another, follow these steps:

1 Select the text you want to move.

2 Choose Cut from the Edit menu.

BBEdit removes the text from the window and stores it on the clipboard.

3 Use the scroll bars to move to the new place for the text if necessary; then click to set the insertion point where the text is to be inserted.

4 Choose Paste from the Edit menu.

You can paste the contents of the clipboard as many times as you want in any BBEEdit window or in any other application.

Pasting inserts the text stored on the clipboard at the insertion point. If there is a selection, pasting replaces the selection with the contents of the clipboard.

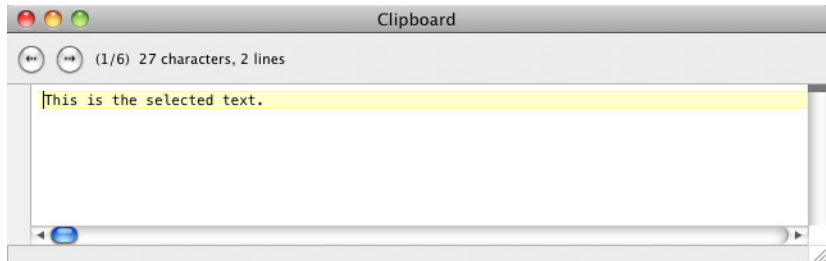
To place text on the clipboard without deleting it, choose Copy from the Edit menu.

Tip To add selected text to the existing contents of the clipboard, hold down the Shift key as you choose the Cut or Copy command. When you hold down the Shift key, BBEEdit changes these commands to Cut & Append and Copy & Append.

Multiple Clipboards

BBEdit supports six separate clipboards. Each time you use the Cut or Copy command, BBEEdit automatically switches to the next clipboard (wrapping back around to the first clipboard after the sixth). This way, the last six things you copied or cut are always available for pasting—sort of a “clipboard history.”

By default, the Paste command pastes text from the most recently used clipboard, so if you do nothing special, BBEEdit works just like any other Macintosh program. However, by using the Previous Clipboard command in the Edit menu you can access the previous clipboard contents. Next Clipboard moves forward through the clipboard history. There are also buttons in the Clipboard window (below) that let you move back and forth through the clipboards.



Once you have selected a clipboard using one of these methods, the next Cut, Copy, or Paste command will use the clipboard you chose. (Subsequent Cut or Copy commands will advance to the next clipboard; Paste never advances automatically.)

Holding down the Shift key changes the Paste command to Paste Previous Clipboard, or you can use the key equivalent Command-Shift-V. This command replaces the pasted text with the contents of the previous clipboard. The previous clipboard becomes current and will be used for any further paste operations; repeated applications of the command cycle backward through the available clipboards.

Note For compatibility with international text content, the Clipboard window displays text in the font (and font size) that it was put on the clipboard with. Changing the display font in the Clipboard window *does not* affect the underlying data.

Drag and Drop

Another way to move text from one place to another is by “drag and drop.” If you drag and drop text from one window to another, BBEdit copies the text to the target window without removing it from the original window.

In addition, you can drag and drop an item from the Finder onto an editing window in BBEdit. If the item is a text file, the file’s contents are inserted. If the item is a folder, a listing of the item’s contents is inserted. If you hold down the Command key while dragging a folder, the path of the item is inserted instead.

Multiple Undo

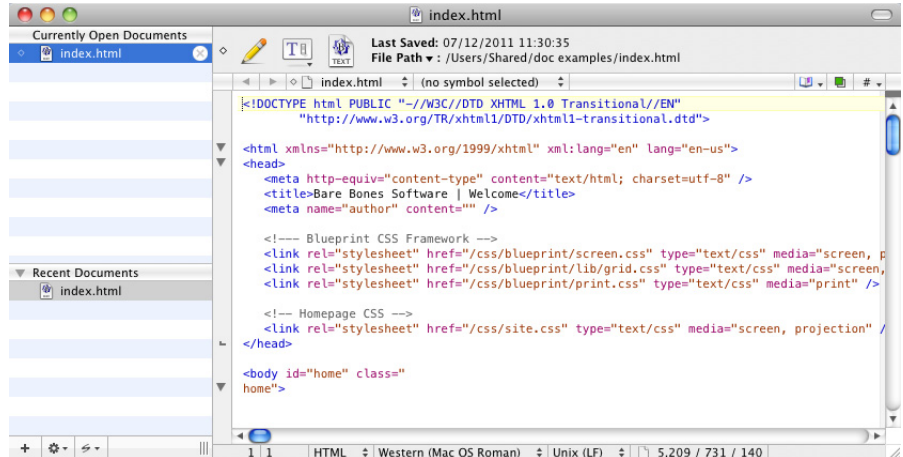
BBEdit provides the ability to undo multiple edits, one action at a time. The number of edits that may be undone is limited only by available memory. The practical limitation is determined by the extent of the edits and the amount of free memory.

BBEdit also supports multiple Redos. If you have not made any changes after performing an Undo, you can redo each action, in order, by choosing that Redo command from the Edit menu or typing Command-Shift-Z. However, once you perform a new action, you cannot redo any actions that you undid before you made that change.

There is also a Clear Undo History menu command (Command-Control-Z), which will clear the undo history for the current editing window. This command can be useful if you have performed many operations on a file and wish to recover memory stored by Undo state information (in the rare event that should become necessary). You can also script this operation via the "clear undo history" scripting command (see the scripting dictionary for details).

Window Anatomy

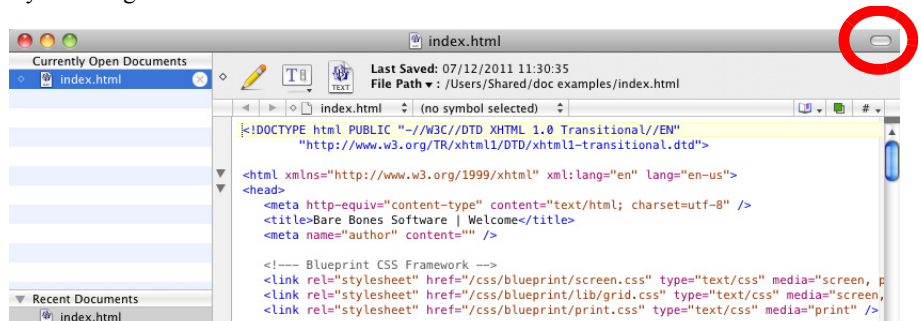
BBEdit text windows have the same controls you are familiar with from other Macintosh applications (for example, text windows are resizable and zoomable, and have both vertical and horizontal scroll bars). Some additional elements which may be less familiar are the toolbar, the split bar, the navigation bar, and the file list.



IMPORTANT You can choose whether BBEdit should display all new and opened documents in the frontmost window, or open each document into a new text window, by setting the “Open documents into the front window when possible” option in the Application preference panel (see page 198).

The Toolbar

The toolbar is a section at the top of each editing window containing buttons and controls that let you adjust display options for and provide info about the current document. You can toggle display of the toolbar by clicking the control in the top-right corner of the window, or by choosing Hide Toolbar/Show Toolbar in the View menu.







You can also use the options in the Appearance preference panel to hide or show individual items on the toolbar.

If the current document has a corresponding disk file, the toolbar displays the full path to the document's disk file and the last time the file was saved. If the document has not been saved to disk, the toolbar displays "(New Document)" instead of a file name.

Note Windows in which the toolbar is not directly below the window title bar (for example, disk browsers and search results) do not have a toolbar control, but do honor the global toolbar preference. You can also use the Text Options sheet to show and hide the toolbar on a per-window basis.

The icons on the toolbar are indicators, buttons, and popup menus that give you quick access to commonly used functions. The following table explains each icon.

| Icon | Meaning |
|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | A solid diamond indicates that the document has been modified. A hollow diamond means only the <i>state</i> of the document (window position, selection range, scrolling position, and so on) has changed. |
|  | The pencil icon indicates that the document can be modified. If the pencil has a slash across it, the document cannot be modified because the file is read-only, the disk is locked, or the file is part of a source-control system project (such as Perforce or CVS) and is checked out. If the file is not on a locked disk, you can click the pencil icon to toggle the document's editability. |
|  | The Text Options popup menu contains commands such as Soft Wrap Text, Show Page Guide, and Show Invisibles that let you control how text is displayed in the window. |
|  | The document proxy icon represents the current document. Clicking this icon is the same as choosing Reveal in Finder from the View menu: it opens a Finder window that contains the document. You can also drag the document proxy icon to any other application, or you can drag it to the Trash (which is the same as choosing Close & Delete from the File menu). |

Key Equivalents for Toolbar Menu Items

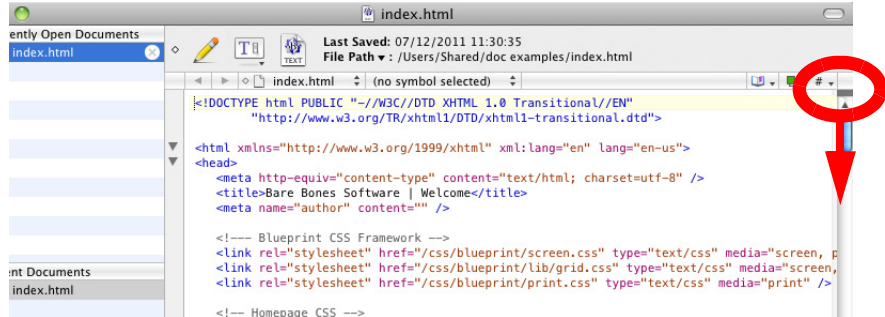
You can assign keyboard shortcuts to items on the Text Options popup menu from the Toolbar entry in the Menus & Shortcuts preference panel.

The Split Bar

Every text window and every browser text pane has a split bar, a small black bar above the scroll bar, that lets you split it into two active view regions. Splitting a text pane lets you view and edit a document's content in two places at the same time. Each region is independently scrollable.

Note Scrolling the non-active split region does not automatically change view focus.

To split the text pane, simply drag the split bar down and let go.

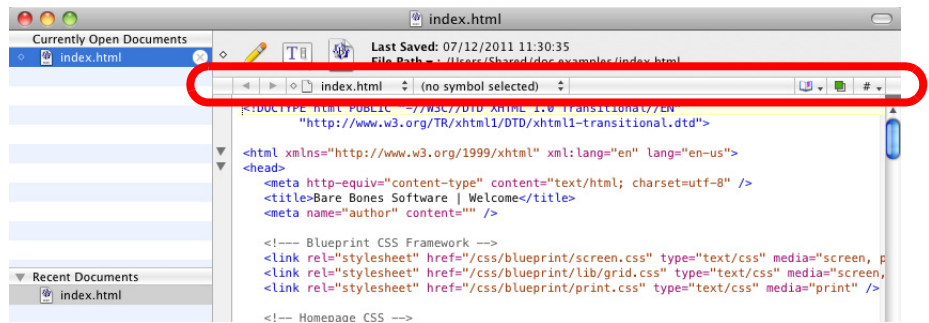


To collapse the text pane back down to a single region, drag the split bar (starting from anywhere along its length, not just at its right end) back up to its original position.

Tip Double-clicking the split bar unplits a split text pane or restores the last-used split position. If the text pane has never been split, it will be split 50-50. To force a 50-50 split for a previously split text pane, Option-double-click the split bar when it is in its original position.

The Navigation Bar

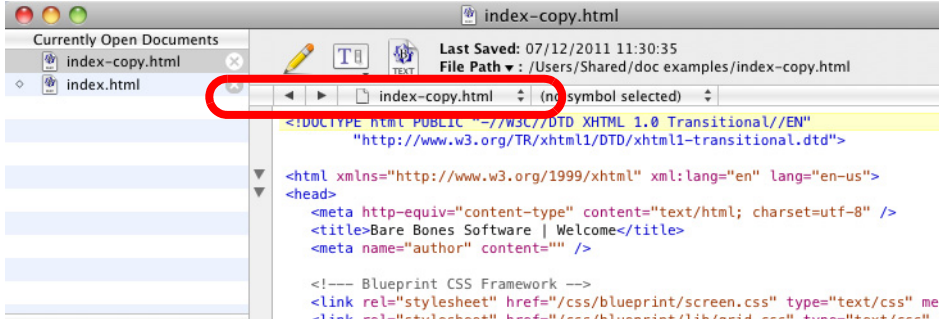
The navigation bar is a panel at the top of a text window which provides controls for selecting the active document and for moving to specific points with the current document. To hide the navigation bar, choose Hide Navigation Bar in the View menu, or turn off the Navigation Bar options in the Appearance preference panel.



You can also use the options in the Appearance preference panel to hide or show individual items on the navigation bar.

Choosing the Active Document

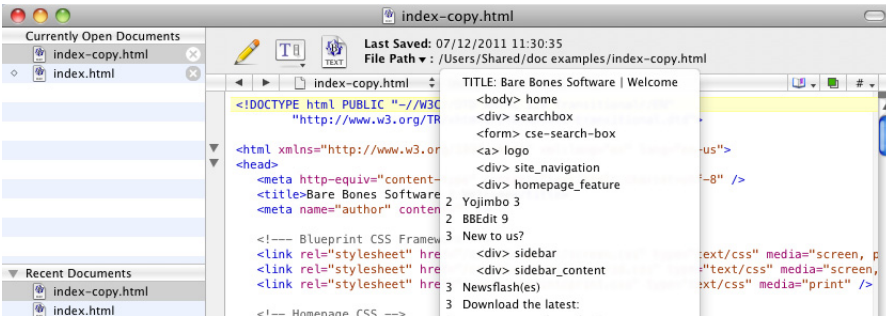
Click on any document in the file list to make that document active, click the Previous or Next buttons to move to the previous or next document in the window, or choose Previous Document/Next Document from the View menu. You can also choose a specific document from the adjacent popup menu to make it active.



The Previous and Next buttons in the Navigation bar, as well as the Previous Document/Next Document commands, select documents in most-recently used order, rather than alphabetical order.

Function Navigation

The Function popup menu lists the functions defined in a programming language source file or various specific tags present within an HTML document. If the current document's language does not support function scanning, the function popup will not be displayed in the navigation bar.



The following indicators appear in the function popup to show the type of function.

| Indicator | Meaning |
|--------------------|---------------------------------------------|
| • | The function containing the insertion point |
| + | C/C++ typedef |
| ◇ | C/C++ "#pragma mark" directive |
| <i>italic name</i> | C/C++ function prototype |
| 1-6 | Heading level (in HTML files) |

| Indicator | Meaning |
|-----------|-----------------------------------------------------------------------|
| tag name | Tag name for the indicated name or ID attribute value (in HTML files) |

Manually Defined Functions

For code written in several languages, including C/C++, PHP, Python, and Ruby, you can manually add customized entries to the function popup menu by inserting suitable “mark” directives within a document.

In C/C++ documents, BBEdit recognizes “#pragma mark” directives. For other languages, each directive consists of a line comment followed by a space and the string “#mark ”, plus the desired marker string.

For example, to add an function popup entry named “My item”:

In C/C++:

```
#pragma mark My item
```

In JavaScript:

```
// #mark My item
```

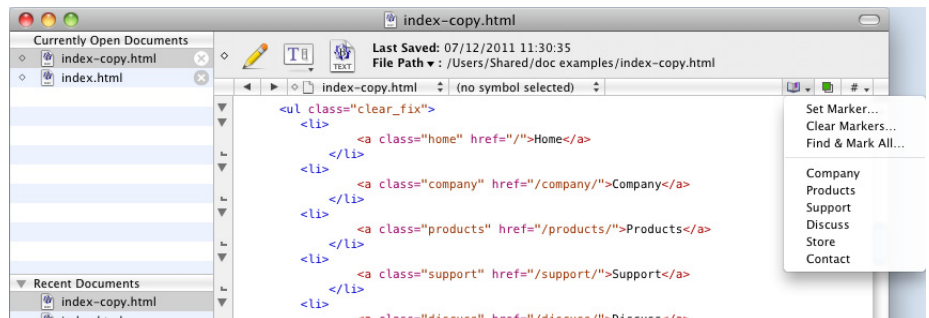
In HTML:

```
<!-- #mark My item -->
```

Note In Python files, each directive must be separated from the preceding content by at least one empty line.

Navigation with Markers

A marker is a selection range that you can name. If a document contains any markers, you can select them from the Marker popup menu to move quickly to the specified section of the file.

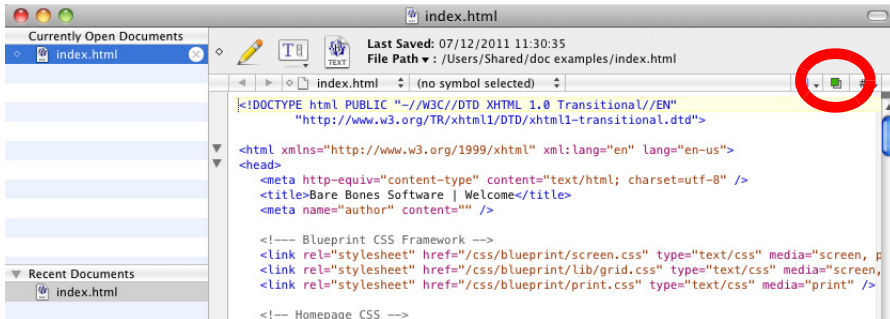


For more information on working with markers, please see “Using Markers” on page 106.

Note If you are programming, you may be tempted to use markers to mark functions in your source code. However, if BBEdit supports the language you are using, this is usually unnecessary; your functions will automatically appear in the Function popup menu.

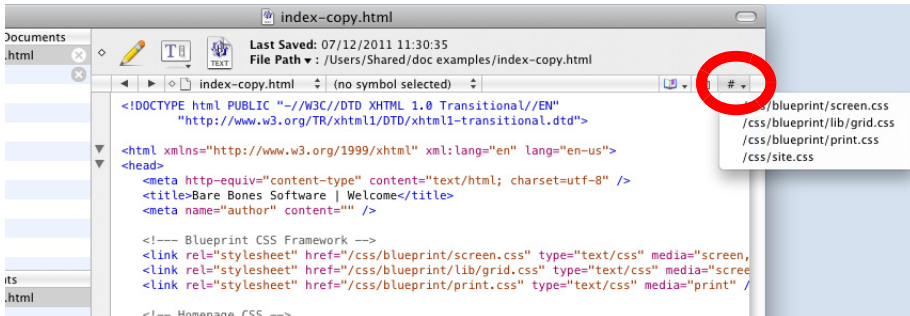
Opening Counterparts

You can press the Counterpart button next to the Marker popup to quickly open and/or switch back and forth between a file and its counterpart (source file to header, or vice versa). This button has the same effect as Open Counterpart in the File menu (see page 52).



Opening Included Files

You can use the Included Files popup to list or open any included files which the current document references.

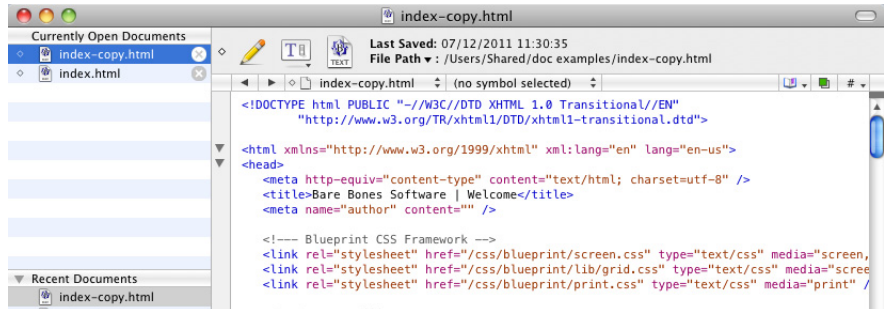


Key Equivalents for Navigation Bar Menu Items

You can assign key equivalents to the controls on the navigation bar from the Navigation Bar entry in the Menus & Shortcuts preference panel. So, for example, you can assign a key equivalent to Open Function Menu, then press that key combination and use the arrow keys to navigate the current document's function list directly from the keyboard.

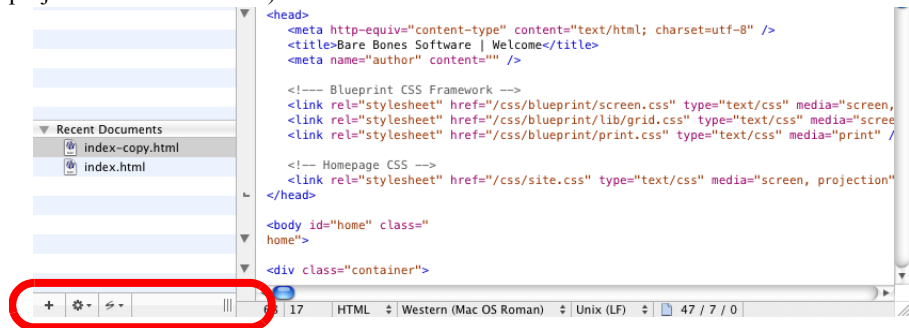
The File List

If BBEdit is configured to open documents into the front window, it will display a file list down the left-hand side of each editing window which shows all the documents currently open in that window. To hide (or show) the file list, choose the Show Files (or Hide Files) command in the View menu, or type its default key equivalent of Command-0. Click any document's name in the list to make that document frontmost in the text window.



Dragging a document's name from the file list has the same effect as dragging its proxy icon in the toolbar. You can also drag documents within the list to manually reorder them.

There are several buttons and popup menus below the file list, which you can apply to perform various additional actions. (These controls are very similar to the controls in a project window's file list.)



To open an existing file into the current text window, choose Open from the File menu, or drag and drop the file from the Finder into the window's file list.

To create a new document, click the Add (plus) button or choose New Text Document in the New submenu of the File menu.

To move a document from the current text window into its own text window, select it in the list and choose Move to New Window from the Action (gear) popup menu, or Control-click on the document in the list and choose this command from the contextual menu. To move multiple documents, select them and choose Move to New Window to create a new text window containing all the selected documents.

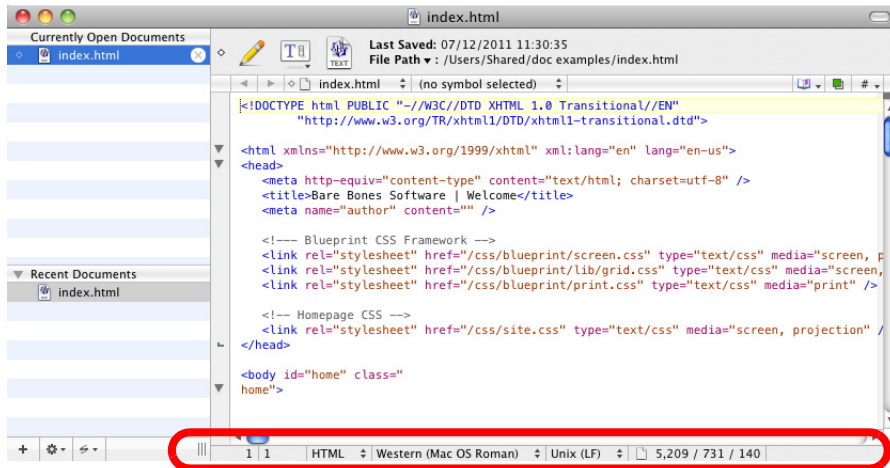
To close a document, you can choose Close Document from the File menu, click on the close box next to its name in the list, select it in the list and apply the Close command from the action (gear) menu, or Control-click on it in the list and select Close in the contextual menu. You can also choose the Close Others command from the action menu or in the contextual menu to close all documents except the selected document.

To move a document from one text window to another, drag its name from the first text window's file list into the second text window's file list. You can select and move multiple documents at once.

To save the current document, you can choose Save from the File menu or the action menu. To save multiple documents at once, select them and choose Save from the Action menu, or Control-click on them and select Save in the contextual menu. To save all documents in the window at once, hold down the Option key and choose Save All from the Action menu.

The Status Bar

The status bar is located directly to the left of the horizontal scrollbar. The status bar displays the current cursor position and contains popup menus showing the language, text encoding, and line break format of the current document. To hide the status bar, turn off the Show Status Bar option in the Appearance preference panel.



You can also use the options in the Application preference panel to hide or show individual items on the status bar.

Cursor Position

This section of the status bar shows the current line and character position of the insertion point.

Language

The Language popup menu displays the language mapping for the current document. You can change this mapping by choosing a different language from the popup.

Text Encoding

The Text Encoding popup menu displays the encoding used to open the current document. You can change the encoding in which the document will be saved by choosing a different encoding from the popup.

To choose an arbitrary encoding, even one not currently displayed, choose Other from the popup and pick your desired encoding from the resulting list.

Line Break Type

The Line Break Type popup menu shows the line break format of the current document's disk file. You can change the line break format with which the file will be saved by choosing it from the popup.

Document Statistics

This section of the status bar dynamically displays the number of characters, words, and lines in the document or the active selection (if any).

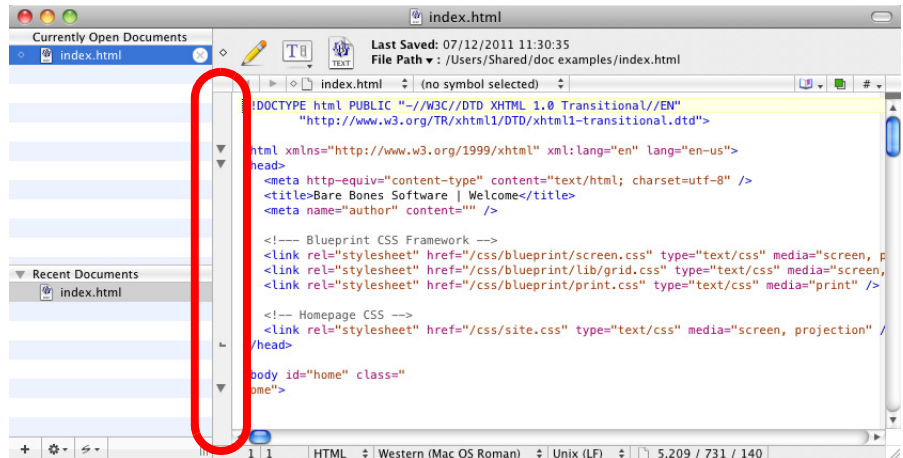
You may click on the statistics section at any time to toggle between displaying info for the whole document and for the selection range. (This icon will be white when BBEdit's displaying statistics for the whole document, and green when it's displaying statistics for the current selection.)

Key Equivalents for Status Bar Items

You can assign key equivalents to the items on the status bar from the Status Bar entry in the Menus & Shortcuts preference panel. For example, you can assign a key equivalent to the Line Breaks popup, then press that key combination and use the arrow keys to select the desired line break option directly from the keyboard.

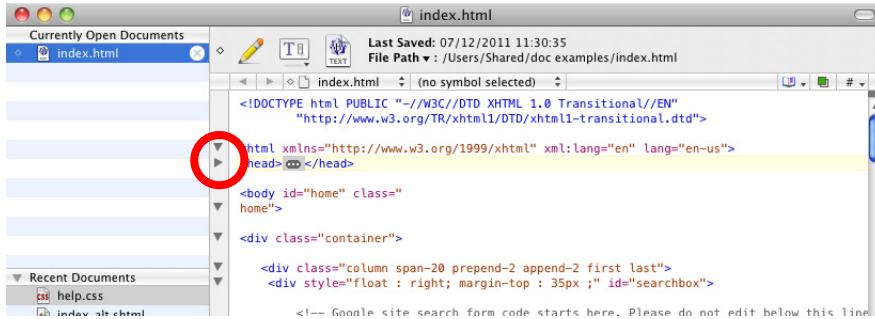
The Gutter and Folded Text Regions

The gutter is the vertical bar directly to the left of the text area, and immediately to the right of the line number display bar (not shown), which contains indicators for folded and foldable regions (automatically-generated folds).



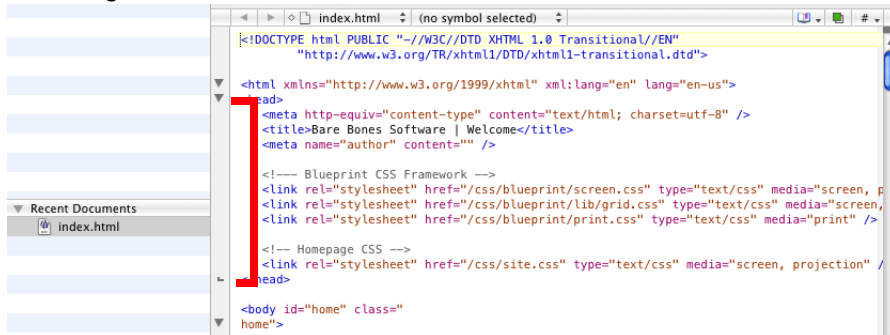
Folding Controls

The triangular controls displayed in the gutter are disclosure triangles; you can click on them to fold or expand regions within the document. If there are nested folds present, Option-clicking on the outermost fold will expand or collapse that fold and all subordinate folds.



You can also employ the commands on the View menu to expand or collapse folds, or fold manually-selected ranges of text. (See “The View Menu” on page 83.)

The linear bars displayed in the gutter are range end indicators. They show where each foldable range ends.



When you fold a range, BBEdit displays a fold indicator within the document to represent that range. To expand the range, you can either click on the disclosure triangle or double-click the fold indicator.



If you expand a range by double-clicking the fold indicator, the entire range will remain selected after expansion.

The View Menu

This menu contains commands which you can use to toggle the display of navigational elements in text windows, to fold and expand regions of text, to select documents, and to get information on documents and files.

Text Display

This submenu contains commands which control various text formatting and display options. You can set key equivalents for any of these commands under the Text Display entry of the View menu in the Menus & Shortcuts preference panel. You can also adjust many of the same options via the Text Options command in the Edit menu.

Show/Hide Fonts

This command toggles display of the standard system font panel, which you can use to set the font, font size, text style, and tab spacing for the active document.

IMPORTANT The chosen display style will be used for *all* text in the window; BBEdit does not support the use of selective text styles.

Note The font changes you make by using this command affect only the active document. To set the default font, size, style, and tab width for all documents, use the "Default Font" option in the Editor Defaults preference panel.

Soft Wrap Text

This command toggles the use of soft wrapping in the current document. (See "Soft Wrapping" on page 98.)

Show/Hide Page Guide

This command toggles display of the page guide in the current document. (See "Page guide" on page 95.)

Show/Hide Tab Stops

This command toggles display of tab stops in the current document. (See "Tab stops" on page 95.)

Show/Hide Line Numbers

This command toggles display of line numbers in the current document. (See "Line numbers" on page 95.)

Show/Hide Gutter

This command toggles display of the gutter in the current document. (See "The Gutter and Folded Text Regions" on page 81.)

Show/Hide Invisibles

This command toggles display of invisible characters in the current document. (See "Show invisibles" on page 95.)

Show/Hide Spaces

This command toggles display of invisible characters in the current document. (See "Show invisibles" on page 95.)

Show/Hide Toolbar

Choose this command to hide or show the toolbar in the frontmost text window. (See “The Toolbar” on page 73.)

Show/Hide Navigation Bar

Choose this command to hide or show the navigation bar in the frontmost text window. (See “The Navigation Bar” on page 75.)

Show/Hide Editor

Choose this command to hide or show the editing pane within a project window.

Show/Hide Files

Choose this command to hide or show the file list within the frontmost text window. (See “The File List” on page 79.)

Show/Hide Open Documents

Choose this command to hide or show the “Open Documents” section within a project window’s file list.

Show/Hide Recent Documents

Choose this command to hide or show the “Recent Documents” section within a project window’s file list.

Show/Hide Worksheet & Scratchpad

Choose this command to hide or show the “Worksheet & Scratchpad” section within a project window’s file list.

Balance

This command locates the pair of parentheses, braces, brackets, or smart (curly) quotes that surround the insertion point or the current selection. If there are unmatched delimiters within this area, BBEdit beeps. You can also double-click a delimiter character to invoke this command.

When syntax coloring is active for a document, Balance (including auto-balance) will ignore balance characters that appear inside strings or comments.

Balance & Fold

This command behaves identically to Balance (above) except that in addition to locating the paired delimiters, BBEdit will also generate a fold range including the delimiters and all the text they contain.

Fold Selection

This command generates a fold range from the currently selected text. You can use Unfold Selection (below) or double-click the fold indicator to expand the fold. When there is no active selection, this command is disabled.

Unfold Selection

This command will expand any text folds in the selection range. When there is no active selection, this command is disabled.

Collapse Enclosing Fold

This command will collapse the auto-generated fold that most closely surrounds the current insertion point (or the start of the selection range).

Collapse All Folds

This command will collapse all automatic fold points in the current document.

Expand All Folds

This command will expand all text folds in the current document, whether automatically generated or manually created.

Previous Document/Next Document

You may use these commands to switch between documents within the frontmost text window. (By default, BBEdit selects documents in most-recently viewed order, but you can choose to have it select documents in name order via an expert preference. For details, see the “Expert Preferences” page in BBEdit’s built-in Help.)

Move to New Window

Choose this command to open the active document of the frontmost text window or project window into its own text window. If the frontmost text window contains only one document, this command will be disabled.

Open in Additional Window

Choose this command to open the active document of the frontmost text window or project window into an additional text window, while leaving it open in the current window.

Reveal in Finder

Choose this command to open a Finder window which will display the active document’s file. If the active document is not associated with a file, this command will be disabled. Using this command is the same as clicking (without dragging) the document proxy icon in the toolbar.

If the selected text in a document is the name of a file, hold down the Option key as you open the File menu and choose the Reveal Selection command to have BBEdit open a Finder window which will display that file.

Reveal in Project List

When there are one or more projects open, choose this command to locate the frontmost project which contains the active document, and reveal that document's file in the project's file list.

Go Here in Terminal

This command is enabled when the active document has a corresponding disk file. Choose this command to open a Terminal window with the current working directory set to the document's parent directory.

Go Here in Disk Browser

This command is enabled when the active document has a corresponding disk file. Choose this command to open a disk browser in the document's parent directory.

Cursor Movement and Text Selection

BBEdit gives you several ways to move the insertion point and change the selection. You can click and drag using normal Macintosh text selection techniques or you can use various keys on the keyboard.

Clicking and Dragging

You can select text in an editing window in the normal Macintosh fashion, by clicking and dragging. Holding down the Shift key while clicking or dragging extends the selection.

| | No Modifier | Shift |
|--------------|----------------------|--------------------------|
| Click | Move insertion point | Extend selection |
| Double-click | Select word | Extend selection to word |
| Triple-click | Select line | -none- |

Triple-clicking is the same as clicking in a line and then choosing the Select Line command from the Edit menu.

You can hold down the Command or Option keys when clicking or double-clicking to trigger special actions:

| | Option | Command |
|--------------|------------------------------------------------|----------|
| Click | -none- | Open URL |
| Double-click | Look up selected word in programming reference | -none- |

BBEdit optionally allows you to select entire lines by clicking in the left margin of an editing window. (If you have line numbers displayed, via the Show Line Numbers option in the Appearance preference panel, you can click in the line number as well.) You can click and drag to select multiple lines, double-click to select an entire paragraph, or double-click and drag to select a range of paragraphs. A checkbox in the Editing preference panel, labeled Allow Single-Click Line Selection, controls this behavior. If the checkbox is turned off, clicking in the left margin simply moves the insertion point to the beginning of the clicked line.

Arrow Keys

You can use the arrow keys to move the insertion point right, left, up, and down, and augment these movements with the Command, Option, and Control keys:

| | No Modifier | Option | Command | Control |
|-------|---------------------|-----------------|-------------------|-------------------------------------------|
| Up | Up one line | Up one screen | Start of document | (scroll view up) |
| Down | Down one line | Down one screen | End of document | (scroll view down) |
| Left | Left one character | Left one word | Start of line | Previous case transition or word boundary |
| Right | Right one character | Right one word | End of line | Next case transition or word boundary |

Holding down the Shift key extends the selection. For example, pressing Shift-Option-Right Arrow selects the word to the right of the insertion point.

CamelCase Navigation

BBEdit supports CamelCase navigation. CamelCase (also “camel case”) is the practice of writing intercapitalized compound words or phrases; it is used as a standard naming convention in several programming languages, and as an automatic link creation method in wiki content.

You can move from one part of a CamelCase word to the next by holding the Control key down and tapping the right (or left) arrow key to jump to the next (or previous) transition from lower-case to upper-case or the next word boundary, whichever comes first.

Rectangular Selections

By holding down the Option key as you drag, or holding down the Shift and Option keys while clicking, you can select all text lying within a specified rectangular area (column). You can then perform all of the normal editing operations on this “rectangular selection,” such as Cut, Copy, Paste, or drag and drop, as well as text transformations such as Change Case, Shift Left, Shift Right, Entab, Detab, Increase Quote Level, Decrease Quote Level, Strip Quotes, and Zap Gremlins.

IMPORTANT

Rectangular selection and soft wrapping are mutually incompatible. When soft wrapping is enabled, dragging the mouse to make a selection will always result in a normal (non-rectangular) selection even if you hold down the Option key. Conversely, if you have made a rectangular selection in a hard wrapped document, the Soft Wrap Text option in the Text Options popup menu or sheet will be disabled.

Working with Rectangular Selections

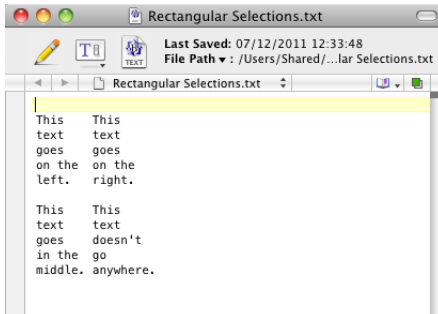
Commonly, while working with text, you will be performing actions on a line-by-line basis; for example, when making a selection, you will start by selecting the contents of one line before moving on to the next. However, if you need to deal with tabular data, it can be useful to think in terms of rectangles or blocks of text that include parts of several lines. This is where you can make use of BBEdit's ability to manipulate rectangular selections.

IMPORTANT

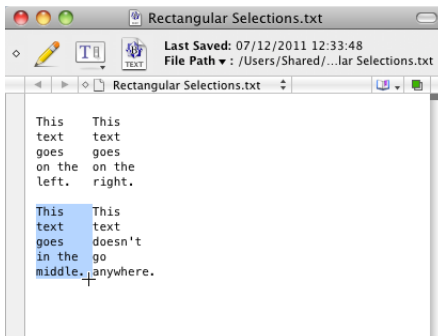
You cannot make or insert rectangular selections into a document which is soft wrapped, so you must turn off soft wrap before using this technique. (See “Soft Wrapping” on page 98.)

Example: Moving a Column

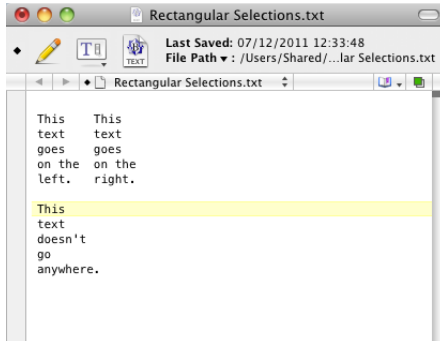
Consider you have the document shown below, and you want to move only the bottom left column (the one that says “This text goes in the middle”) and move it in between the top left and top right columns. To do this using standard selection methods, you would have to perform five separate cut-and-paste operations. However, by using rectangular selections, you can move the whole column in one operation.



To start, hold down the Option key while dragging over the bottom left column, until you get a selection that looks like this:



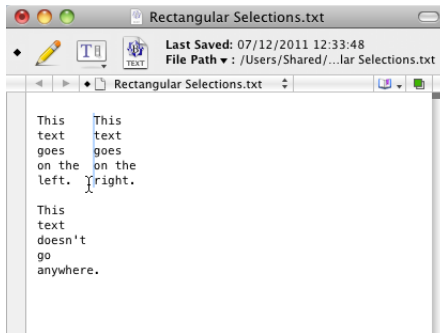
Choose Cut from the Edit menu (or press Cmd-X) to cut the selected text out of the document and place it on the Clipboard.



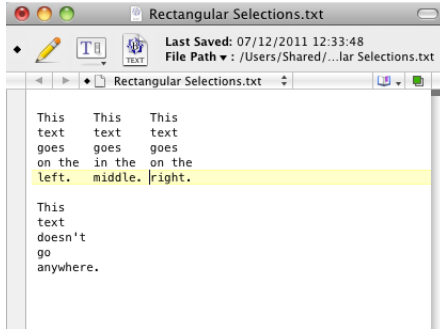
Next, you must paste in the text you just cut. You can do this in either of two ways:

- Use the Paste Column command, which will “paste down” from the current insertion point. This allows you to directly insert text without needing to make a rectangular selection first.
- Make a rectangular selection as described below, and then use the standard Paste command. This procedure is less efficient for moving columnar data than using the Paste Column command, but it allows you to select and replace a region of text as well as simply inserting text.

To manually make a rectangular selection prior to pasting text, position the arrow pointer just to the left of the top right column, press and hold the Option key, press the mouse button, and drag straight down until you have a very thin vertical selection just to the left of the whole column, as shown below.



Now, paste the text you previously cut back in, and the task is finished.



Filling Down

When you apply the Paste Column command and the pasted text contains no line break (or only a single line break at the end), BBEdit will perform a “fill down”, placing a copy of the pasted text on each line within the selected column.

Further Details

Some word processors also provide support for rectangular selections which works a little differently than BBEdit's, so you may wish to keep this difference in mind. Typically, when you copy a rectangular selection of text to the clipboard in these programs, they handle that piece of text differently than text copied from a line-by-line selection. Then, when you paste, the text will be entered in a block, even when you have not made a rectangular selection to paste into.

BBEdit does not do this. Instead, when you copy a rectangular selection to the clipboard, BBEdit turns the selection into a series of individual lines, which is why you must make a rectangular selection before pasting, so BBEdit will know it should paste the text in block fashion. Though this method does require an extra step, it is more flexible, because you can select a set of lines and then paste it as a block, or vice versa.

Scrolling the View

When holding down the Control key, the arrow keys will scroll document windows without moving the insertion point.

Accelerated Scrolling

When clicking the arrows in a scroll bar, you can use the Command and Option keys to accelerate the scrolling. These shortcuts also apply if you use a mouse with a built-in scroll wheel.

| Modifier | Scroll Speed |
|----------------|----------------|
| none | Normal |
| Command | 2x accelerated |
| Option | 3x accelerated |
| Command+Option | 6x accelerated |

The Delete Key

The Delete key deletes the character to the left of the insertion point. If you have selected text, the Delete key deletes all the text in the selection. You can use the Command and Option keys to modify the way the Delete key works:

| Modifier | Action |
|----------------|-------------------------------------------------------------------------|
| none | Deletes character to the left of the insertion point |
| Option | Deletes to the beginning of the word to the left of the insertion point |
| Command | Deletes to the beginning of the line |
| Command+Option | Deletes to the beginning of the document |

Holding down the Shift key with the Delete key makes the Delete key work the same way as the Forward Delete key on extended keyboards.

The Numeric Keypad

Some keyboards have a numeric keypad on the right side. Normally, you use the keys on the keypad to enter numbers.

To toggle the behavior of the keypad between moving the cursor and entering numbers, hold down the Option key and press the Clear key in the upper-left corner of the keypad. (This key is also labeled Num Lock on some keyboards.)

When keypad navigation is active, BBEdit will perform the following actions:

| | | |
|---------------------------|----------------------------|-------------------------|
| start of line 7 | up 8 | Scroll up 9 |
| left 4 | show selection 5 | Right 6 |
| end of line 1 | down 2 | Scroll down 3 |

You can use the Shift key with the keys on the numeric keypad to extend a selection. You can use the Command and Option keys with the 2, 4, 6, and 8 keys as you would the arrow keys.

Go To Line Command

To move the insertion point to a specific line, use the Go To Line command in the Search menu. When you choose this command, BBEdit opens a Go To Line sheet in the frontmost document. Type the number of the line you want to move to and click Go To.

The Go to Line command will also accept relative inputs. Entering a value prefixed with +/ - will add that value to the current line number. For example, with the insertion point in line 100, "+75" will move to line 175; "-75" will go to line 25. (As always, when you enter an unsigned number, BBEdit will move to the specified line number.)

Note The Go To Line command honors the Use "Hard" Lines in Soft-Wrapped Views option in the Editing preference panel.

Function Keys

If your keyboard has function keys, you can use the following key equivalents for cutting and pasting, to scroll, and to move the insertion point.

| | No Modifier | Option | Command | Shift |
|-------|---------------------------|-----------------------|-------------------------------------------|-------|
| del | forward delete | delete to end of word | delete to end of line | |
| Home | scroll to top of document | | move insertion point to start of document | |
| End | scroll to end of document | | move insertion point to end of document | |
| Pg Up | scroll page up | | | |
| Pg Dn | scroll page down | | | |

Note Holding down the Command and Option keys as you press the forward delete key deletes to the end of the document.

Resolving URLs

To resolve a URL (Uniform Resource Locator), you can Command-click anywhere in the URL text, or Control-click to bring up the contextual menu and choose Open URL from the menu. BBEdit will examine the URL and launch the appropriate helper application. If the URL is not valid or the helper application cannot be found, BBEdit will beep.

Note Some Web browsers cannot resolve URLs if the request is sent when the browser is starting up. If your Web browser does not properly resolve the URL, wait until the browser has finished starting up and then try again.

Bare Bones Software gratefully acknowledges John Norstad for providing the URL parsing code.

Text Completion

NEW BBEdit will keep track of and offer completions for words and symbols as you edit. You can control when BBEdit offers completions via the “Show Text Completions” option in the Editing preferences panel:

- After a delay in typing: If you pause briefly while typing, BBEdit will figure out the possible completions for what you just typed and display them.
- Only manually: BBEdit will only display possible completions when you invoke the Complete command (see below).

You can also enable (or disable) automatic completion on a per-language basis by adding a custom language setting in the Languages preferences panel.

Completions are derived from a variety of sources, including (in no particular order):

- clippings (both language-specific and universal);
- ctags symbols computed by running the current document through 'ctags';
- ctags symbols found in 'tags' files in the current document's hierarchy;
- predefined names for the source code language at the point of completion;
- language-specific completions;
- dictionary words provided by the system spelling service

You may trigger a completion at any time (whether or not automatic completion is enabled) by using the “Complete” command in the Edit menu or pressing its default key equivalent of F5. (You can change this equivalent via the Menus & Shortcuts preference panel.)

Note Text completion treats clippings in the same way that the “Insert Clipping” command used to (and still does). So, the behavior of F5 should be indistinguishable if you were used to using it to complete clippings.

You can also optionally use the Escape key to invoke text completion. By default, this key is not used for completion, but you may enable it by issuing the following Terminal command:

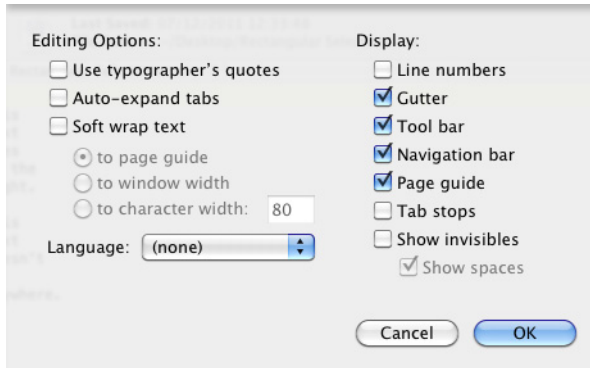
```
# allow the Escape key to invoke the "Complete" command  
  
defaults write com.barebones.bbedit UseEscapeKeyAsCompletionTrigger -  
bool YES
```

Note Because the Escape key has a special meaning when the “Use Emacs key bindings” option is on (in the Keyboard preference panel), if this option is active and you choose to use the Escape key as a completion trigger, you will have to press Escape twice to invoke completion.

Text Options

You can use the Text Options command to change the way BBEdit edits text and the way it displays text and additional elements in its windows. When you choose this command, BBEdit will drop a Text Options sheet in the current text window.

The controls on the Text Options sheet are divided into two parts: the Editing options on the left control the way BBEdit behaves while you type, and the Display options on the right control the appearance of the BBEdit window.



Note You can also change many of these options using the commands in the Text Display submenu of the View menu.

Changes you make in the Text Options sheet affect only the active document or window. To set options which will apply to all text windows you open, use the Editor Defaults and Appearance preference panels.

Editing Options

These options control the way BBEdit behaves as you type text in the active document window. Changes you make here affect only that document. To change the default editing options for documents that you will open in the future, use the Editor Defaults preference panel.

Use typographer's quotes

When this option is on, BBEdit will automatically replace straight quotes (" ') with typographer's quotes (“ ” ‘ ’) in the current document. If you need to type a straight quote when this option is selected (or to type a typographer's quote when the option is not selected), hold down the Control key as you type the " or ' key.

Note We recommend against using this option if you are editing HTML content, email content, or program code.

Auto-expand tabs

When this option is selected, BBEdit inserts an appropriate number of spaces when you press Tab, rather than inserting a tab character.

Additionally, when there are only spaces (and tabs) between the insertion point and the start of the current line (or the first non-whitespace character on the line), BBEdit will delete a tab stop's worth of spaces when you press Delete (Backspace).

Soft wrap text

When this option is selected, BBEdit soft-wraps the text in the file to the right margin that you choose: the page guide, the window width, or a specific number of characters. The page guide is an arbitrary visual boundary whose width you can set in the Appearance preference panel. (See “Soft Wrapping” on page 98 to learn how wrapping works in BBEdit.)

Language

The Language menu lets you specify which source code language the file uses. The file’s language setting affects how BBEdit performs syntax coloring and parses function names for the function popup menu. BBEdit generally determines the file’s language from its filename extension, using the mapping table in the Languages preference panel.

For example, .cp files are C++, and .m files are Objective-C. You can use this menu to override those settings for a specific file. To quickly check the language for a file, click the Text Options popup menu in the toolbar and look at the Languages item.

Display Options

These options determine which controls appear in the frontmost text window, regardless of whether that window contains one or more documents. Changes you make here affect only that window. To change the display characteristics for text windows that you will open in the future, use the Appearance preference panel.

Line numbers

This option displays line numbers along the left edge of the window.

Gutter

This option shows or hides the gutter in the window.

Toolbar

This option shows or hides the toolbar in the window.

Navigation bar

This option shows or hides the navigation bar in the window.

Page guide

This option shows or hides the page guide in the window.

Tab stops

This option shows or hides tab stop indicators in the window.

Show invisibles

This option shows or hides non-printing characters in the window. Select this option when you want to see line breaks, tabs, and “gremlins” (other invisible characters). BBEdit uses these symbols:

| Symbol | Meaning |
|--------|---------|
| △ | tab |
| ◇ | space |

| Symbol | Meaning |
|--------|------------------------------------------|
| • | non-breaking space |
| ↵ | line break |
| ¶ | page break |
| ¿ | other non-printing or special characters |

If you turn on Show Invisibles, the Show Spaces option will become available, allowing you to enable display of the visually “noisy” space characters if you desire.

Syntax Coloring

When this option is selected and the editing window contains a document in a programming language BBEdit recognizes, BBEdit displays keywords and other language elements in color.

BBEdit uses several methods to determine what language (if any) to use for a particular file. The primary way to activate syntax coloring in a document is simply to save it with a file name extension that indicates what programming or markup language the file contains. For example, if you save your file with “.html” at the end of the file name, BBEdit will color your HTML tags and anchors. Other common suffixes are “.tex” for TeX files and “.c” for C files.

For any file whose name does not have an extension, or whose name has an extension that does not match any of the mappings in BBEdit’s Languages preference panel, BBEdit will attempt to guess what language the file contains and apply the appropriate syntax coloring. If BBEdit guesses wrong (or is unable to guess), you can resort to the Language submenu of the Text Options popup menu in the toolbar or the Language popup menu in the Text Options sheet, which gives you the ability to manually select *any* installed language to be applied to the document, regardless of its name. If the file is saved with “BBEdit” state, the manual language selection will persist and override any suffix mapping.

By default, BBEdit recognizes over 20 different languages and several dozen suffix mappings. You can add new suffixes to map to existing languages or (by installing third-party language plug-ins) add syntax coloring support for new languages as well. All the specific languages that BBEdit recognizes, and the suffixes or extensions it expects for them, are listed in the Languages preference panel, and suffix mappings can also be changed there. You can choose the colors that BBEdit uses for syntax coloring in the Text Colors preference panel.

Note BBEdit will recognize and syntax-color VBScript embedded within HTML via the `<%...%>` and `<SCRIPT>...</SCRIPT>` tags.

How BBEdit Wraps Text

BBEdit wraps text in one of two ways: soft wrapping or hard wrapping.

Soft wrapping is like the word wrapping found in most word processors. When the insertion point reaches a right margin as you type, the word processor automatically moves the insertion point to the beginning of the next line. You never need to type a carriage return (that is, press the Return key) at the end of a line, but only to start a new paragraph. If you place the insertion point in the middle of a paragraph and start typing, the text reflows so that words that are pushed out beyond the right margin end up on the next line. Usually, you use soft wrapping when you are editing memos, mail messages, and other prose. It is also useful for HTML documents. With soft wrapping, you generally do not have to scroll the window horizontally to see all the text in the file.

Unlike soft wrapping, hard wrapping requires a carriage return at the end of every line. When soft wrapping is turned off, BBEdit lets you type as far as you like on a line, and never automatically moves the insertion point to the beginning of the next line. You have to manually type a carriage return to start a new line. You usually use hard wrapping to write programs, tabular data, resource descriptions, and so on. With hard wrapping, each line of source code or data appears on its own line in the window, although you may have to scroll the window horizontally to see the entire line if it is long.

Note When you use the Hard Wrap command on a rectangular selection, lines will be padded with spaces as necessary.

Tip If you open a file in BBEdit that appears to consist of a few very long lines, you should select the soft wrapping option for that file.

This table summarizes the commands to soft-wrap and hard-wrap text. The sections that follow give details about using the wrapping commands.

| To do this... | Do this... |
|---------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Soft-wrap text as you type | Choose Soft Wrap Text from the Text Display submenu of the View menu or select the Soft Wrap Text option from the Text Options sheet |
| Convert hard-wrapped text to soft-wrapped text | Use the Remove Line Breaks command in the Text menu, and activate soft wrapping |
| Convert soft-wrapped text to hard-wrapped text | Use the Add Line Breaks command in the Text menu |
| Hard-wrap text to a specific margin, reflowing paragraphs as needed | Use the Hard Wrap command in the Text menu |

Note Users of very old versions of BBEdit or BBEdit Lite will note that the Wrap while Typing option (which hard-wrapped text automatically by inserting a Return when you reach the right margin) has been relegated to the dustbin of history. It has been superseded by soft wrapping.

Soft Wrapping

To turn on soft wrapping for the active window do one of the following:

- Choose Soft Wrap Text from the Text Display submenu of the View menu.
- Select the Soft Wrap Text option from the Text Options sheet by choosing Text Options from the Edit menu.

To specify the wrapping margin, use the Text Options command. You can have text wrap at the Page Guide, the edge of the window, or a specific character position.

IMPORTANT

Soft wrapping and rectangular selection are mutually incompatible. When soft wrapping is enabled, dragging the mouse performs normal (non-rectangular) selection even if the Option key is held down; when there is a rectangular selection, the Soft Wrap Text option is unavailable in the Text Options popup menu and dialog box.

To make soft wrapping the default for new windows, select the Soft Wrap Text option in the Editor Defaults preference panel. You can also use the settings in that panel to specify the default wrapping margin.

To “freeze” the current line endings and hard-wrap the text at the current soft wrapping settings, use the Add Line Breaks command to insert a carriage return at the end of each line.

While BBEdit prefers to break lines at white space when soft-wrapping, lines will be broken as close as possible to the designated wrap width if they do not contain any white space. This way, long URLs and other extended strings of characters are visible without requiring horizontal scrolling.

Soft Wrapping with Indentation

You can control how BBEdit indents soft wrapped text by means of the Soft Wrapped Line Indentation option in the Editing preference panel. Choose Flush Left to have all lines of each paragraph below the first wrap flush to the left margin of the window. Choose First Line to have all subsequent lines of a paragraph wrap to the same indent level as its first line. Choose Reverse to have all subsequent lines of each paragraph wrap indented one level deeper than its first line.

Exporting Soft-Wrapped Text

BBEdit will not insert hard line breaks into softwrapped files upon saving them. If you wish to add hard line breaks to a softwrapped file, use the Hard Wrap or Add Line Breaks command.

Soft Wrapping in Browsers

Use the Text Options command from the Edit menu to control soft wrapping (and other display options) for files viewed in a browser window.

Soft Wrapping and Line Numbers

The preference Use “Hard” Lines in Soft-Wrapped Views controls how line numbers are displayed when you use soft wrapping. If this option is turned on, the line number bar, cursor position display, and Go To Line commands in editing views will use line numbers that correspond to “hard” carriage returns in the document, rather than to soft-wrapped line breaks. To restore the behavior of previous versions of BBEdit, turn this preference off.

Hard Wrapping

The easiest way to hard-wrap text is to type a carriage return (by pressing the Return key) whenever you want to start a new line. If you are editing program source code, it is generally best to turn off soft wrapping altogether.

To turn off soft wrapping for the active window, do one of the following:

- Choose Soft Wrap Text from the Text Options popup menu in the toolbar.
- Deselect the Soft Wrap Text option from the Text Options sheet box by choosing Text Options from the Edit menu.

To turn off soft wrapping for new windows, deselect the Soft Wrap Text option in the Editor Defaults preference panel.

BBEdit provides two ways to convert soft-wrapped text into hard-wrapped text. The first is a simple technique that uses a single command; the second is a bit more complicated but gives you much more control over wrapping.

Hard-Wrapping Soft-Wrapped Text

To convert soft-wrapped text to hard-wrapped text, use the Add Line Breaks command in the Text menu. This command inserts a carriage return at the end of every line of the text as it appears in the window. If your wrapping margin is the edge of the window, you will get different results depending on the width of the window.

If the current document contains a selection range, Add Line Breaks will affect only the selected text; if there is no selection, this command will affect the entire contents of the current document.

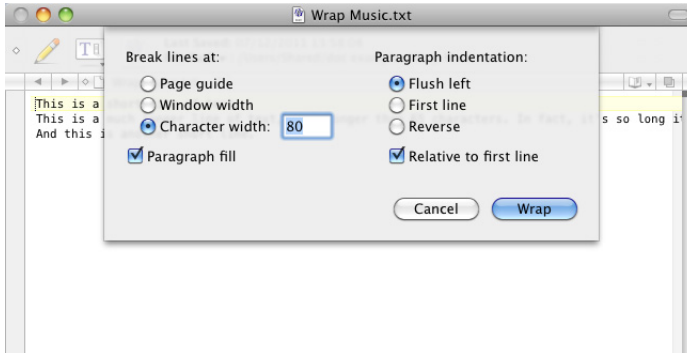
Note The Add Line Breaks command does not turn off soft wrapping.

Hard Wrapping and Filling Text

The Hard Wrap command in the Text menu offers more flexibility for hard-wrapping text than the Add Line Breaks command. Whereas Add Line Breaks merely “freezes” the line breaks displayed in a document by inserting carriage returns, the Hard Wrap command allows you to wrap text to any arbitrary width, while also reflowing or indenting paragraphs.

If the current document contains a selection range, Hard Wrap will affect only the selected text; if there is no selection, this command will affect the entire contents of the current document.

When you choose the Hard Wrap command, BBEdit opens a sheet in the frontmost document:

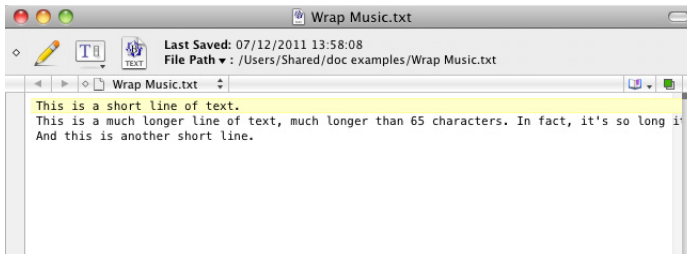


The controls in the left half of the sheet control the maximum width of lines after hard wrapping, and whether wrapped lines should be consolidated to fill paragraphs to the specified width. The controls in the right half determine how paragraphs should be indented.

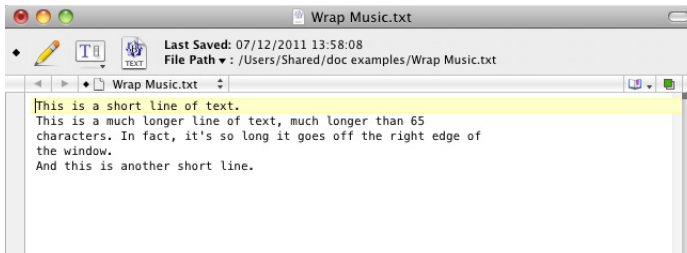
The “Break Lines at” buttons let you specify the wrapping margin.

If the Paragraph Fill option is selected, BBEdit forms the lines into paragraphs before wrapping the lines. An example is the best way to illustrate this option.

Suppose you start with this text:

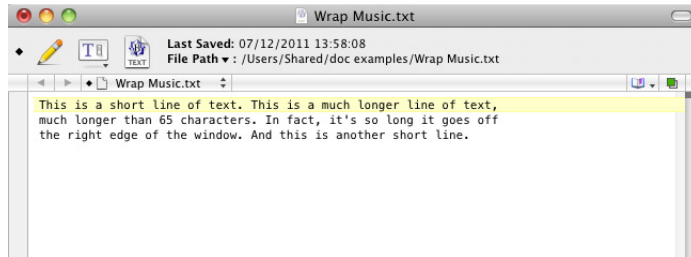


This is what happens when you wrap to 65 characters with Paragraph Fill off:



BBEdit breaks the long line at a width of 65 characters (twice, because the line was so long) and leaves the short lines alone.

This is what happens to the same text when you wrap with Paragraph Fill on:



BBEdit joins all the lines together to form a single paragraph and then wraps the text to a width of 65 characters.

The Paragraph Indentation buttons let you indent paragraphs after they have been wrapped.

- Flush Left does not indent paragraphs at all.
- First Line indents all lines in the paragraph by one tab stop.
- Reverse places the first line in the paragraph flush against the left edge of the window and indents all other lines in the paragraph by one tab stop.

Mark the Relative to First Line checkbox to make any paragraph indents relative to the original indent of the first line of the selection or document. If you want paragraph indents to be relative to the left margin of the document, make sure this checkbox is not marked.

Click the Wrap button to perform the Hard Wrap command. Click the Don't Wrap button to save the settings without changing the text.

Tip If you hold down the Option key as you choose the Hard Wrap command, BBEdit uses the last Hard Wrap settings to perform the operation, without displaying a sheet.

The Insert Submenu

In addition to typing, you can use the commands in the Insert submenu of the Edit menu to insert text into the active window. These commands, which are also available in the Insert popup menu (left) in the document toolbar, let you insert the contents of other files, folder listings, Macintosh Toolbox templates, and page break characters.

Inserting File Contents

The File Contents command inserts the contents of one or more files into the document you are editing. When you use this command, BBEdit displays an Open sheet in which you can choose the files to insert. To select more than one file hold down the Shift key or Control key as you click the files. BBEdit then inserts the contents of the selected files at the insertion point or replaces the selected text. If you select more than one file, the files will be inserted in alphabetical order, according to file name.

Tip You can also drag a file's icon from the Finder into a BBEdit editing window to insert the contents of that file.

Inserting File & Folder Paths

The File/Folder Paths command inserts the full path information for the selected files and folder into the document you are editing. When you use these commands, BBEdit displays a sheet that lets you select the files and/or folders. BBEdit inserts the path information at the insertion point or replaces the selected text.

Inserting a Folder Listing

The Folder Listing command inserts a textual listing of a folder hierarchy. When you use this command, BBEdit displays a sheet that lets you select a folder to insert, and inserts that folder's listing at the insertion point or replaces the selected text.

Tip You can also drag a folder's icon from the Finder into a document to insert a folder listing.

Inserting a Page Break

To insert a page break, choose the Page Break command from the Insert submenu of the Edit menu. This will place a form feed character (ASCII 12) at the location of the insertion point. BBEdit uses this character to indicate the start of a new page when printing.

Inserting Time Stamps

To insert the current time, choose Short Time Stamp or Full Time Stamp from the Insert submenu of the Edit menu. These commands will insert short and long forms (respectively) of the current date and time at the location of the insertion point.

Inserting an Emacs Variable Block

To insert an Emacs variable block describing the option settings for the current document, choose Emacs Variable Block from the Insert submenu of the Edit menu. This will bring up a sheet which you can use to review and confirm the desired options. (Since depending on what options are set, the resulting block can be rather verbose, you may wish to prune the resulting text.)

These options specified in this block will take precedence over saved document state when BBEdit opens the document. (Inserting these explicit settings can be useful when sharing the document with others.)

Comparing Text Files

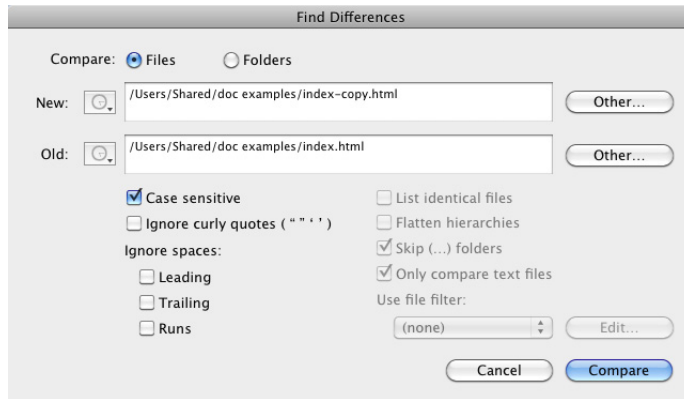
If you have ever had to reconcile changes between two different versions of a file, or even larger numbers of documents, you know how laborious this task can be. BBEdit's Find Differences command is a powerful tool for doing such comparisons faster and more effectively. Using Find Differences, you can compare any two files, or the contents of two folders. You can also specify options to eliminate minor variations in document content, such as different amounts of white space, from being considered.

If you have two or more text documents open, choose the Compare Two Front Documents command on the Search menu to quickly compare the topmost two documents. (BBEdit will automatically determine which document is newer and which older based on their modification dates.)

To compare two arbitrary files or folders:

1 Choose the Find Differences command from the Search menu.

BBEdit opens the Find Differences dialog box.



2 Click the Compare Files radio button.

3 Use the New and Old popup menus to select the documents you want to compare.

If the files you want to compare are already open, they will appear in the popup menus; otherwise, you can select them by clicking the Other button next to one of the popup menus, or by dragging the files' or folders' icons from the Finder into the New and Old boxes in the Find Differences dialog.

You can also select recently opened files from the Recent Files item on the New and Old popup menus.

The terms “new” and “old” are used for convenience since most often you will want to find changes between two versions of the same file saved at different times. However, the Find Differences command can be used to compare any two files or folders.

4 Select the Compare options that apply.

When the Case Sensitive option is selected, BBEdit distinguishes uppercase from lowercase letters; deselect this option if you want BBEdit to consider uppercase and lowercase letters the same.

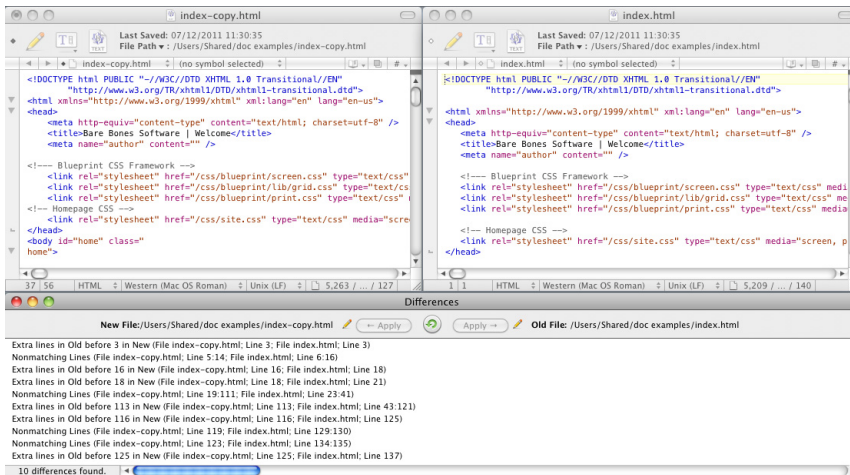
When Ignore Curly Quotes is selected, BBEdit treats typographers' quotes the same as straight quotes.

When one or more of the Ignore Spaces options is selected, BBEdit will ignore the corresponding presence of whitespace at the specified positions while comparing files.

5 Click Compare to perform the comparison.

Alternatively, you can use the ‘bdiff’ command line tool to specify two files or folders, and have BBEdit perform a Find Differences on them.

If the two files are different, BBEdit tiles the documents and opens a Differences window below them.



The Differences window lists all the differences between the new file and the old file. To see the differences in context, click a line in the Differences window; BBEdit scrolls and selects that spot in both files.

The entire range of difference in each file is drawn with a grey background, while individual differences within the range are highlighted with the standard selection color.

To view and apply individual differences within a line or region (i.e. sub-line differences), just click on the grey triangle to expand the list and select the appropriate character difference.

Use the Apply to New and Apply to Old buttons in the Differences window to transfer the differing text from the new file to the old file, or vice versa. After you use one of these buttons, BBEdit italicizes the entry in the Differences window to indicate that you have already applied that change.

You may also apply all differences by clicking in the differences list, then choosing Select All in the Edit menu, and using the Apply to New or Apply to Old button to apply the differences to the desired file.

If a Differences window is open and is the frontmost window, the Compare Again command in the Search menu will recompare the two files being compared and refresh the list of differences accordingly. The small button (with the circular icon) between the Apply to New and Apply to Old buttons performs the same function.

Compare Against Disk File

You can use the Compare Against Disk File command to compare the contents of the active document against the disk file for that same document. This capability makes it easy to locate in-progress changes to a document.

Multi-File Compare Options

You can compare multiple files at once by selecting the Folders button in the Find Differences dialog; BBEdit lists all the files and marks those that are different with a bullet. You have the additional options described below.

List identical files

Normally, when you compare folders using the Find Differences command BBEdit presents you with three lists: one list of the items that are in the first folder but not in the second folder, another list of the items that are in the second folder but not in the first one, and another list of the items that appear in both folders.

The list of items that appear in both folders generally displays a bullet next to items that are not identical. For example, if you have an archived mail folder that you are comparing against a current mail folder, mailbox files that appear in both the old and new file will all be listed together; however, if there have been any changes to the contents of particular mailbox files, the changed mailbox files will be listed with bullets next to them.

If you are comparing very large folders, however, the list of common items can be extremely long, making the flagged items hard to find. When you deselect the List Identical Files checkbox, BBEdit will list only the flagged items (the ones that have been changed) in the list of items that appear in both folders.

Flatten hierarchies

Normally, BBEdit retains the hierarchy of the files being compared in a folder. In other words, when comparing folders, it looks in each subfolder of the first folder you select and tries to match it with a file of the same name in the same subfolder of the second folder, and so on down for all subfolders. If you choose Flatten Hierarchies, BBEdit considers the files in the folders as a single flat list, allowing a file in one folder to match a file of the same name in the other folder, regardless of whether they are in the same subfolder in both hierarchies.

Skip (...) folders

If this option is set, BBEdit skips subfolders whose names are enclosed in parentheses when comparing folders.

Only compare text files

If this option is set, BBEdit does not include non-text files when comparing folders.

Use file filter

File filters allow you to select files for comparison with great precision. If either file in a compared pair matches the filter, the files are eligible for comparison; if *neither* file matches the filter, the files will not be compared. See Chapter 7, “Searching,” for more information on creating, editing, and using file filters.

Note When comparing folders with the Find Differences command, BBEdit applies any specified file filter to the contents of the resulting “Only in new” and “Only in old” lists, so that only those files that match the filter criteria will appear in the lists.

Using Markers

A marker is a selection range that you can name. If a document contains any markers, you can select them from the Mark popup menu to move quickly to the specified section of the file. (The navigation bar must be visible in order to access the Mark popup menu. Choose Show Navigation Bar from the View menu to display the Navigation bar if it's hidden.)

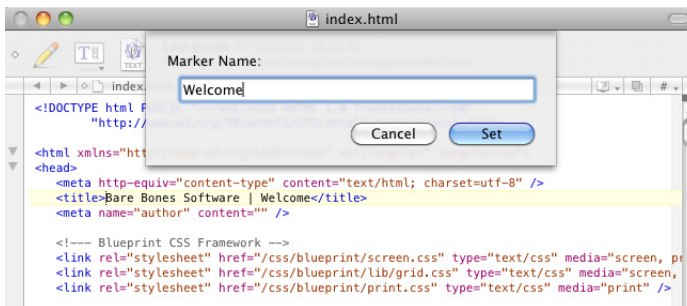
Note If you are programming, you may be tempted to use markers to mark functions in your source code. However, if BBEdit supports the language you are using, this is usually unnecessary; your functions will automatically appear in the Function popup menu in the document window.

Setting Markers

To set a marker:

- 1 Select the text you want to mark.
- 2 Choose the Set Marker command from the Mark popup menu (identified by the icon shown at left), or Control-click the selected text and choose Set Marker from the contextual menu.

BBEdit opens a sheet so that you can name the marker. If you have selected a range of text, the sheet will contain the first characters of the selection.



- 3 Click Set to set the marker.

Tip If you hold down the Option key as you choose Set Marker, BBEdit sets the marker using the leading characters of the selected text as the name of the marker, without displaying a dialog box.

Clearing Markers

To clear a marker:

- 1 Choose the Clear Markers command from the Mark popup menu.

BBEdit displays the list of markers.

- 2 Select the marker you want to delete.
- 3 Click Clear to clear the marker.

BBEdit also offers a Clear All Markers command, which clears all the markers in the document in one fell swoop. You can access this command by holding down the Option key and using the Mark popup menu.

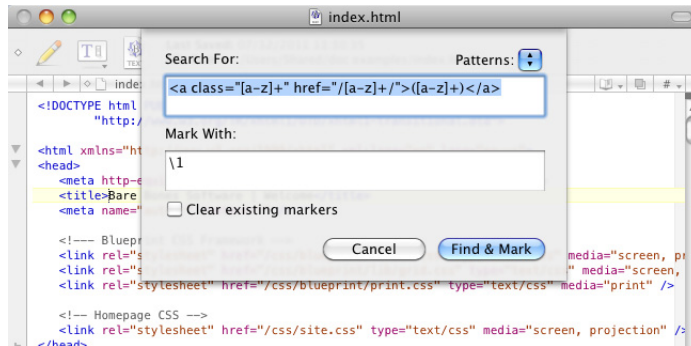
Using Grep to Set Markers

You can use the Find & Mark All command in the Mark popup menu to mark text that matches a grep pattern. To learn more about using grep patterns, see Chapter 8, “Searching with Grep.”

To use a grep pattern to mark text:

1 Choose the Find & Mark All command from the Mark submenu.

BBEdit opens the Find & Mark All sheet.



2 Type the pattern in the Search For field and the marker names in the Mark With field.

You can also choose stored patterns from the Patterns popup menu.

3 Click Find & Mark to mark the matching text.

BBEdit searches the current document for text that matches the pattern and marks it the way you specified.

Spell Checking Documents

The Check Spelling command in the Text menu lets you check the spelling of the text in your documents using the system’s built-in spelling checker.

Check Spelling As You Type

To have BBEdit automatically check spelling as you type for the current document, select Check Spelling as You Type in the Text menu. To have BBEdit always check spelling as you type, turn on the corresponding option in the Editor Defaults preference panel.

When BBEdit encounters a word which is either misspelled or not in the checker’s dictionary, it will draw a heavy red underline beneath the word. You can either type a correction, or Control-click on the word and select a suggested correction from the contextual menu.

To skip the identified word and continue checking, use the Check Spelling command again. To ignore all further instances of the word, Control-click on it and choose Ignore Spelling from the contextual menu. To add the word to the dictionary, Control-click on it and choose Learn Spelling from the contextual menu.

Manual Spell Checking

Choose the Find Next Misspelled Word command from the Text menu, or type its key equivalent (Command-;) to start checking a document's spelling. BBEdit will check every word in the document in order, starting from the current insertion point.

To check the spelling of all words in the document at once, choose the Find All Misspelled Words command, or type its key equivalent (Command-Option-;). BBEdit will draw an underline under every questioned word in the document. You can then correct the spelling of any questioned word by typing, or by using the contextual menu to select a suggested correction or to skip, ignore, or add the word to the dictionary.

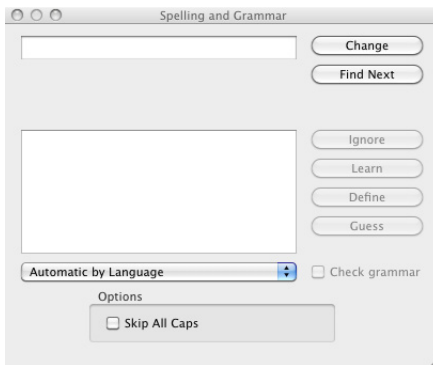
To clear the underline from all questioned words, choose the Clear Spelling Errors command.

The Spelling Panel

In addition to allowing you to correct, ignore, or learn identified words, the Spelling panel allows you to choose which spelling dictionary BBEdit will use, and to forget learned spellings. To use the Spelling panel:

1 Choose the Show Spelling Panel command from the Text menu.

BBEdit opens the Spelling panel.



2 Set spelling options.

Choose a dictionary to use by selecting it from the Dictionary popup menu. Select Skip All Caps to avoid checking words consisting of only capital letters. (Note that these settings persist across runs of the application.)

3 Click Find Next to begin checking.

BBEdit scans the document, and stops at the first misspelled or unrecognized word. This word is displayed in the text field to the left of the Correct button. Possible corrections for the questioned word are listed in the Guess box above.

4 If the questioned word is misspelled, choose the correct spelling from the Guess list or type it yourself in the Correct field.

5 Click one of the Spelling panel's action buttons to handle the questioned word.

Click Ignore to ignore further instances of the questioned word, without adding it to the active dictionary.

Click Guess to display a list of possible corrections.

Click Find Next to ignore this instance of the questioned word and continue checking.

Click Correct to replace this instance of the questioned word with the text in the adjacent text field.

Click Learn to add the questioned word to the active dictionary.

Click Forget to remove the questioned word from the active dictionary.

This chapter describes the range of powerful text transformation commands offered by BBEdit. Beyond applying individual commands to the current document, you can define and save Text Factories, which are sequences of commands that can be applied to one or more documents.

In this chapter

| | |
|-----------------------------------------------------------------------------------------------|-----|
| Text Menu Commands..... | 111 |
| <i>Apply Text Filter</i> – 112 • <i>Exchange Characters</i> – 112 | |
| <i>Change Case</i> – 112 • <i>Shift Left / Shift Right</i> – 113 | |
| <i>Un/Comment Selection</i> – 113 • <i>Hard Wrap</i> – 113 | |
| <i>Add Line Breaks</i> – 114 • <i>Remove Line Breaks</i> – 114 | |
| <i>Convert to ASCII</i> – 114 • <i>Educate Quotes</i> – 114 | |
| <i>Straighten Quotes</i> – 114 • <i>Add/Remove Line Numbers</i> – 114 | |
| <i>Prefix/Suffix Lines</i> – 115 • <i>Sort Lines</i> – 115 • <i>Process Duplicate Lines</i> – | |
| 116 • <i>Process Lines Containing</i> – 117 • <i>Rewrap Quoted Text</i> – 118 | |
| <i>Increase and Decrease Quote Level</i> – 118 • <i>Strip Quotes</i> – 118 | |
| <i>Zap Gremlins</i> – 119 • <i>Entab</i> – 120 • <i>Detab</i> – 120 | |
| <i>Normalize Line Endings</i> – 120 | |
| Text Factories..... | 121 |
| <i>Creating and Configuring Text Factories</i> – 121 | |
| <i>Applying Text Factories to Files</i> – 124 | |
| <i>Applying Text Factories to Open Documents</i> – 125 | |
| <i>HTML Processing Actions</i> – 125 | |
| Automator Actions..... | 126 |
| <i>Using BBEdit with Automator</i> – 126 | |
| <i>Available Actions</i> – 127 | |

Text Menu Commands

BBEdit provides a variety of commands which you can use to transform text in different and useful ways. Most of these commands are situated in the Text menu, and described in this section. You can also use BBEdit’s search and replace capabilities, or additional plug-in tools, to transform text; each of these topics is covered in a separate chapter.

Unless otherwise specified, each of these commands will be applied to the active text selection in the frontmost document range, or if there is no active selection, to the entire contents of the document.

Hold down the Option key when selecting any command from the menu in order to quickly re-invoke it with its last-used option settings. (These “short form” commands are also available in the Menus & Shortcuts preference panel, so that you can set key equivalents for them.)

Apply Text Filter

This command presents a submenu listing all currently available text filters. (These filters consist of any executable items contained in the Text Filters folder of BBEdit's application support folder. See "Text Filters" on page 34.)

When you choose a filter, BBEdit will pass either the selected text (or the contents of the active document, if there is no selection) on STDIN to Unix executables or filters, as a string to text factories, as a reference to a 'RunFromBBEdit' entry point in AppleScripts, as text input to Automator workflows, and as a source to text factories. (If an AppleScript script does not have a 'RunFromBBEdit' entry point, BBEdit will call its run handler, again passing a reference to the current selection range.)

AppleScript scripts and Automator workflows should return a string which BBEdit will use to replace the selection range, Unix filters should write to STDOUT, and the text emitted by a text factory will replace the selection range.

Apply Text Filter <last filter>

This command will reapply the two characters according to the following rules:

Exchange Characters

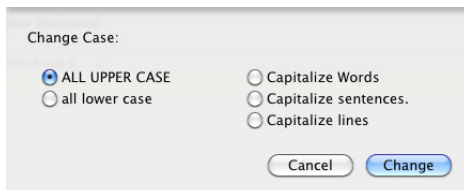
This command swaps two characters according to the following rules:

- If there is no selection and the insertion point is not at the beginning or end of a line or of the document, this command transposes the two characters on either side of the insertion point.
- If the insertion point is at the beginning of a line or document, this command transposes the two characters following the insertion point.
- If the insertion point is at the end of a line or document, this command transposes the two characters before the insertion point.
- If there is a selection, this command transposes the characters at *either end* of the selection.

If you hold down the Option key as you choose this command, Exchange Characters becomes Exchange Words. Exchange Words behaves like Exchange Characters except that it acts on entire words rather than individual characters.

Change Case

This command lets you change between uppercase and lowercase characters, or capitalize word, line, or sentence starts. When you choose the Change Case command, the following sheet appears:



The radio buttons let you choose how to change the case of the text. The following table explains the function of each option in this dialog. The radio buttons let you choose how to

| This button... | Changes the text like this... |
|----------------------|-------------------------------------------------------------------------------------------------------|
| ALL UPPER CASE | Every character changes to uppercase. |
| all lower case | Every character changes to lowercase. |
| Capitalize Words | The first character of every word changes to uppercase; all other characters change to lowercase. |
| Capitalize sentences | The first character of every sentence changes to uppercase; all other characters change to lowercase. |
| Capitalize lines | The first character of every line changes to uppercase; other characters are unaffected. |

In addition to using the Change Case sheet, you can also select individual case change actions from the Change Case submenu immediately below the Change Case... command.

Shift Left / Shift Right

These commands indent or outdent the selected text by one tab stop. If there is no selection, this command works on the current line. Hold down the Shift key while choosing these commands, to have BBEdit indent or outdent the text by one space instead of one tab stop.

BBEdit also entabs and detabs on the fly as you shift text. For example, if the selected text is indented one tab stop and you apply Shift Left One Space, the tab will be converted to spaces and the text will be outdented one space. If you then apply Shift Right One Space, the spaces will be converted back to a single tab.

Un/Comment Selection

This command automates the task of commenting and uncommenting sections of code in various programming languages. Choose a range of text and apply this command to add or remove comments to it, depending on its initial comment state. If there is no selection, this command is disabled.

You can use the Options sheet of the Installed Languages list in the Languages preference panel to modify or set comment strings for any available languages.

Note If you have set custom comment delimiters for HTML in the Languages preference panel, those delimiters will be honored when you use the Un/Comment command. However, they will not affect the operation of the HTML-specific comment commands on the Markup menu.

Hard Wrap

This command wraps long lines by inserting hard line breaks and can reflow (fill) paragraphs if desired. See “How BBEdit Wraps Text” on page 97 for more information.

Add Line Breaks

This command inserts a hard line break at the end of each line of text as displayed. See “How BBEdit Wraps Text” on page 97 for more information.

Remove Line Breaks

This command removes carriage returns and spaces from sections of text. Use this command to turn text that has hard line breaks into text that can be soft-wrapped. See “How BBEdit Wraps Text” on page 97 for more information.

Convert to ASCII

This command will convert certain eight-bit Mac Roman characters (characters whose decimal values are greater than 128 and less than 255) to 7-bit (printable ASCII range) equivalents. Converted characters include unlauded and accented vowels, ligatures, typographer's quotes, and various specialized punctuation forms. This conversion may entail expansion to multiple characters; for example, in the case of ligatures.

Educate Quotes

This command converts straight quotes (" and ') to typographer's quotes (“ ” and ‘ ’).

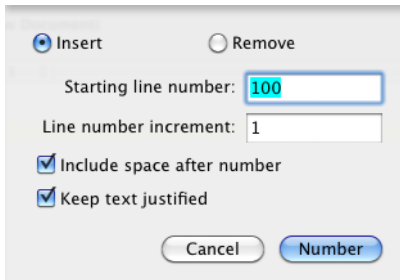
Note You should not use this plug-in to prepare text for use in a web page or an email, as typographer's quotes in the Mac character set will generally not be properly displayed by applications on other platforms.

Straighten Quotes

This command performs the reverse of Educate Quotes; it converts typographer's quotes (“ ” and ‘ ’) to straight quotes (" and ').

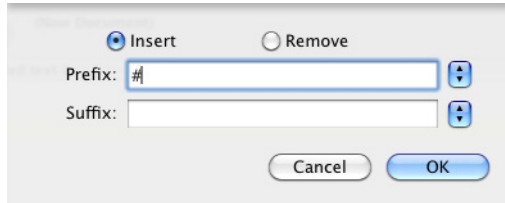
Add/Remove Line Numbers

This command displays a sheet which allows you to add or remove line numbers for each line of the selected text or of the document. You can set the starting number and numbering increment, as well as whether to include a trailing space, and whether to right-justify the inserted numbers, by choosing the appropriate options.



Prefix/Suffix Lines

This command displays a sheet which allows you to insert (or remove) the specified prefix and/or suffix strings on each line of the selected text or of the document.

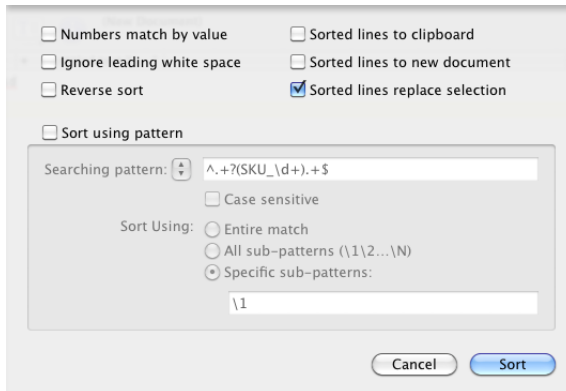


If you define both a prefix and a suffix string, BBEdit will apply them to the text at the same time.

Note When using the “add prefix”, “add suffix”, “remove prefix”, or “remove suffix” scripting commands, the string direct parameter is required.

Sort Lines

This command displays a sheet which allows you to sort lines of text by collating them in alphanumeric order. The sorted lines can be copied to the clipboard, be displayed in a new untitled window, replace the selection within the original document, or any combination of the three.



There are also options for ignoring white space at the beginning of lines, taking case distinctions into account, sorting strings of digits by numerical value instead of lexically, and sorting in descending rather than ascending order.

By checking the Sort Using Pattern option, you can specify a grep pattern to further filter the lines to be sorted. If the pattern contains subpatterns, you can use them to control the sort order based on the contents of the strings they match. When you sort using a grep pattern, the Case Sensitive option controls the case sensitivity of the pattern match in the same manner as the equivalent option in the Find dialog.

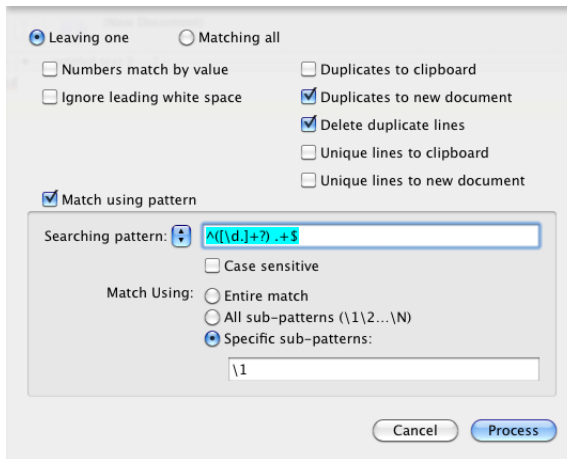
For example, suppose you are sorting a list of cities together with their two-letter state abbreviations, separated by a tab character. The pattern and subpatterns shown in the figure will sort the results first by city name and second by state abbreviation. Changing the contents of the Specific Sub-Patterns field from “\1\2” to “\2\1” will instead sort the results by state first and by city second.

IMPORTANT

When you use a grep pattern with this command, matches are not automatically anchored to line boundaries, so ambiguous patterns may produce unpredictable results. To avoid this problem, you should use the line start `^` and line end operators as necessary. Also, keep mind that the pattern will only be tested against a single line at a time. So, if the pattern matches one or more sets of multiple lines within in the document, but does not match any individual lines, BBEdit will not sort the contents of the document.

Process Duplicate Lines

This command displays a sheet which allows you to locate duplicate lines within a body of text and operates on them in various ways.



The Matching All option processes all duplicate lines; Leaving One ignores the first of each set of duplicate lines and processes only the additional ones.

The Numbers Match by Value and Ignore Leading White Space options allow you to choose whether strings of digits should be evaluated numerically or compared as strings, and whether white space at the beginnings of lines should be considered.

The Match Using Pattern option allows you to use a grep pattern to further filter the lines to be processed. You can enter a pattern in the Searching Pattern field, or choose a stored pattern from the popup menu. The Match Using: radio buttons control what part of the specified pattern should be used to determine duplication.

IMPORTANT

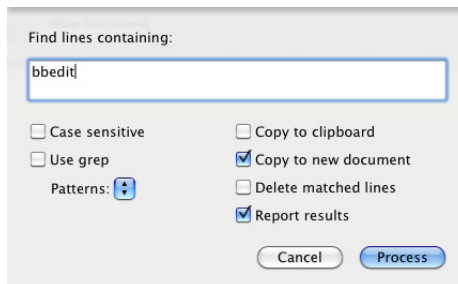
When you use a grep pattern with this command, matches are not automatically anchored to line boundaries, so ambiguous patterns may produce unpredictable results. To avoid this problem, you should use the line start `^` and line end operators as necessary.

The options on the right-hand side of the sheet allow you to specify how duplicate lines should be handled once they have been identified. You can copy duplicate lines to the clipboard (Duplicates to Clipboard), copy them to a new document (Duplicates to New Document Window), and/or delete them from the current document (Delete Duplicate Lines). You can likewise specify how to handle the lines that are *not* duplicated by choosing Unique Lines to Clipboard and/or Unique Lines to New Document).

Since each of these options is an independent checkbox, you can select any combination of them that you wish. For example, selecting both Delete Duplicate Lines and Unique Lines to Clipboard would delete the duplicate lines from the document and copy them to the clipboard for pasting elsewhere.

Process Lines Containing

This command displays a sheet which allows you to search the active window for lines containing a specified search string and then removes those lines or copies them to the clipboard. The options on the left side of the dialog box control how the search is performed and the options on the right side control what happens to the lines that are found.



To specify a search pattern, enter it in the Find Lines Containing field. If you do not want BBEdit to match text when the letters in the text differ from the letters in the search string only by case (upper-case versus lower-case), select Case Sensitive.

To search using a grep pattern, select Use Grep and enter the pattern in the text field. You can also select a predefined search pattern from the Patterns popup menu.

Note If the selection ends in a trailing carriage return, the carriage return will be omitted from the search string copied into the text field.

The checkboxes on the right of the sheet control the way lines containing the specified search pattern will be processed. By selecting the appropriate combinations of these options, you can achieve the effect of applying various editing commands to each line:

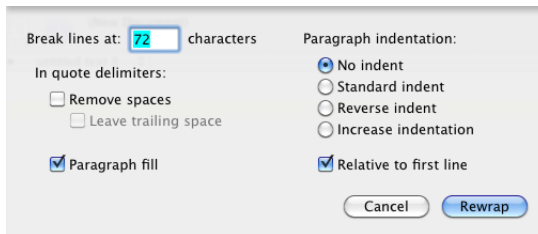
- Setting both Copy to Clipboard and Delete Matched Lines on is equivalent to applying the Cut command.
- Setting Copy to Clipboard on and Delete Matched Lines off is equivalent to applying the Copy command.
- Setting Copy to Clipboard off and Delete Matched Lines on is equivalent to applying the Clear command.

The Copy to New Document option opens a new, untitled document containing copies of all lines matching the search pattern, whether or not they are deleted from the original window. By using this option and turning Copy to Clipboard off, you can collect all matching lines without affecting the previous contents of the clipboard.

The Report Results option causes BBEdit to display a dialog reporting the total number of lines matched, regardless of their final disposition. With all of the other options turned off, this can be useful for pretesting the extent of a search operation without affecting the clipboard or the contents of the original window.

Rewrap Quoted Text

This command rewraps hard-wrapped text having Internet-style quoting, while retaining the quoting characters and quote level.



In Internet messages, it is common to use the “>” symbol to indicate that part of a message is quoted from a message that is being replied to. As a message gets batted back and forth in a discussion, the oldest bits of text will end up having several “>” symbols in front of them. Each line of text in the message has a carriage return at the end, making rewrapping the text to a different width somewhat problematic.

When you apply this command, BBEdit first extracts each chunk of quoted text (a successive set of lines with the same number of markers), and temporarily removes the markers and any hard line breaks from the chunk of text, forming it into a soft-wrapped paragraph. BBEdit then hard-wraps that paragraph according to your chosen settings, which are the same as for the Hard Wrap command (see “Hard Wrap” on page 113), and reinserts the quote markers.

Note When you use this command on a rectangular selection, BBEdit will pad lines with spaces as necessary.

Increase and Decrease Quote Level

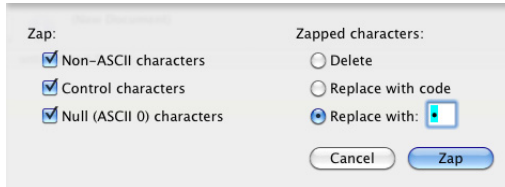
These commands respectively insert or delete a standard Internet quote character (“>”) from each line of the selected hard-wrapped text, or for the current line if there is no selection.

Strip Quotes

This command removes all Internet-style quoting from the selected hard-wrapped text, or from the current line if there is no selection.

Zap Gremlins

This command displays a sheet which allows you to remove or replace various non-printing characters, often known as “gremlins”. Use this command when you have a file that may contain extraneous control characters, or any non-ASCII characters, which you wish to identify or remove.



The checkboxes on the left-hand side of the sheet determine which types of characters the Zap Gremlins command affects, while the radio buttons on the right-hand side determine what to do with gremlins that are found.

Zap Non-ASCII characters

When this option is selected, Zap Gremlins zaps *all* characters in the file that do not fall in the 7-bit (or ASCII) range. Examples of such characters include special Macintosh characters such as bullets (•) and typographer’s quotes (“ and ”, ‘ and ’), as well as all multi-byte characters. In general, such special characters are those that you type by holding down the Option key.

Zap Control characters

When this option is selected, Zap Gremlins zaps a specific range of invisible low-ASCII characters, also known as control characters. Control characters can cause compilers and other text-processing utilities to malfunction, and are therefore undesirable in many files.

Zap Null (ASCII 0) characters

When this option is selected, Zap Gremlins zaps all instances of the null character (ASCII 0). Like other control characters, nulls can cause many programming tools and text-processing utilities to malfunction. This specific option is included in case you want to remove only nulls without affecting other control characters that may be present in a file.

Delete

This option removes the zapped character completely from the text. It is useful if you are only interested in destroying gremlins and you do not care where they were in the text.

Replace with code

This option replaces the gremlin character with any other character specified in escaped hexadecimal format. The escape code is formed via the same convention used by the C programming language: `\0x` followed by the character code in hexadecimal (base 16). This option is useful for identifying both the value and the location of gremlin characters. Later, you can search for occurrences of `\0x` to locate the converted characters. (Searching for the grep pattern of `“\0x..”` will select the entire character code for easy modification or deletion.)

Replace with <character>

This option replaces the gremlin with the character you type in the text field next to the radio button. It is useful for identifying the location of gremlins, but not their value. The replacement character can be specified not only as any typeable character, but also by using any of the special characters defined for text searches, including hex escapes. (See “Special Characters” on page 140.)

Note In some cases, this option could be counterproductive, since hex escapes (`\xNN`) can themselves be used to insert unprintable characters.

Entab

This command displays a sheet which allows you to set the number of consecutive space characters which should be converted into tabs. This transformation is useful when you are copying content from many online sources, which use spaces to line up columns of text. If you do not use a monospaced font, columns usually will not line up unless you entab the text first.

Detab

This command displays a sheet which allows you to set the number of consecutive spaces which should replace each tab. This command is useful when you are preparing text for use in a program which has no concept of tabs as column separators, for email transmission, and similar purposes.

Normalize Line Endings

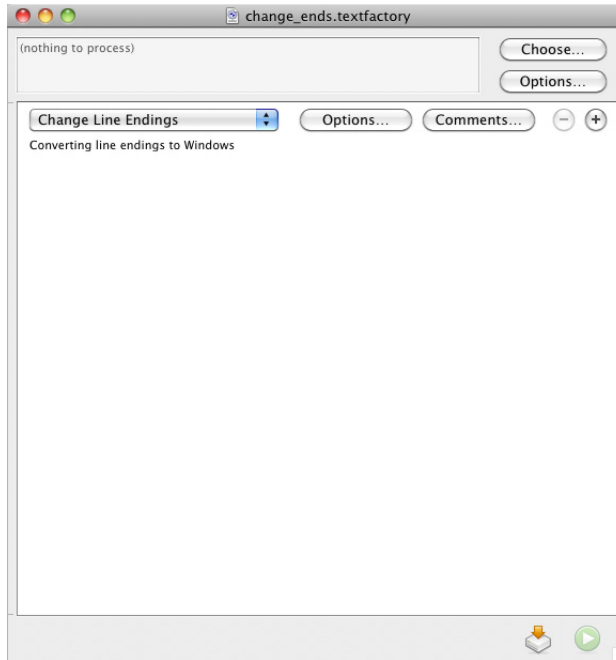
This command converts a document containing mixed line endings to have a uniform set of line endings.

If you open a file which contains a mixture of Mac, Unix, and DOS/Windows line endings, the “Translate Line Breaks” option may not suffice to properly convert the document for viewing and editing. After conversion, the document may appear to not have any line breaks at all (this usually happens if the first line break in the file is a Mac line break, and all the rest are Unix), or to have an invisible character at the beginning of each line.

Should this happen, use Normalize Line Breaks to convert the remaining line endings, and save the document. Once you have done this, the document’s line endings will be consistent, and BBEdit’s line-break translation will suffice when you next open it.

Text Factories

A text factory document enables you to apply BBEdit’s powerful text transformation commands in the order and fashion that you decide, to whatever collection of files you choose. So, for example, if you routinely need to process a folder full of server logs by reducing them to lines which don’t contain “error”, prefixing each line with a line number, and converting each file’s line endings to Macintosh, you can assemble and save a text factory to do that work for you, and apply it at any time.



Creating and Configuring Text Factories

To create a new text factory, choose the Text Factory command from the New submenu of the File menu. BBEdit will create a text factory document window which contains a single action step (corresponding to a single operation).

You may save a text factory document to disk any time by using the Save command, or clicking the Save button at the bottom of the window.

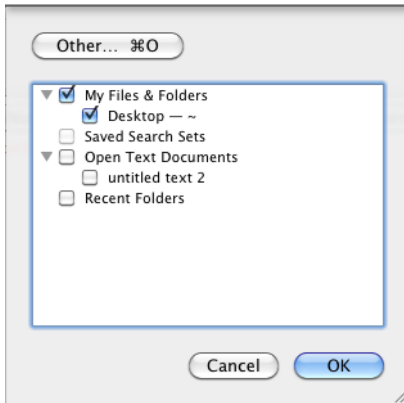
Text factories saved in the Text Filters folder of BBEdit’s application support folder will appear in the Apply Text Filter submenu, and you may apply such a factory to process the current selection (if any) or the contents of the frontmost document.

Text factories saved into the Scripts folder will appear in the Scripts menu, and you may run such a factory to process any designated set of target files and folders.

Note Text factory documents are plist files, which are distinguished from editable text files by having either a file type of “TxEN” and a creator type of “R*ch”, or a filename extension of “.textfactory”. If you store documents in any manner which does not preserve file type info, you must name your text factory documents accordingly.

Choosing Targets

At the top of a text factory window is a summary area which displays information about the target files and folders you have chosen for processing. Click Choose to present a sheet containing a list of selected and available target items.

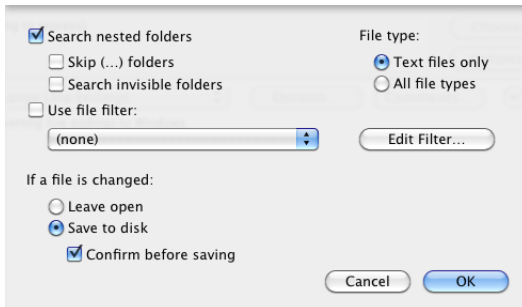


Available targets include:

- open text documents
- individual files
- the files in any selected or recently-searched folder
- the files listed in any results browser (such as a search results browser, an HTML syntax errors browser, or a compile errors browser)
- the files and folders contained in any open project
- the files in any defined web site project
- any Finder “Smart Folders” which you have defined (BBEdit will automatically list such items from the “Saved Searches” folder for your login account)

To select (or deselect) an item as a target, click the checkbox next to its name. To add a file or folder to the list, click Other and select it in the resulting Choose Object dialog.

Click Options to select additional options for controlling which target items will be processed. To process all the files in subfolders of each target folder, mark the Search Nested Folders checkbox. The Skip (...) Folders checkbox skips the contents of folders whose names are enclosed in parentheses. The “Search Invisible Folders” checkbox allows you to process the contents of invisible folders.



You can also choose to process only text files or to process all file types. If you have graphics or other types of files in the target folders, you should restrict processing to only text files. This setting works in addition to any file filter (see “File Filters” on page 147) and is in fact applied *before* the filter.

The last group of options controls how BBEdit treats processed documents. Choose Leave Open to have BBEdit leave all the documents open so that you can inspect the results of the operation. Choose Save to Disk to have BBEdit automatically save changes to each file after processing it. When the Confirm Saves setting is active, you will have an opportunity to approve the changes before BBEdit saves them to disk. You should not turn this off unless you are sure that the processing being applied is what you want.

Defining Actions

Each action in a text factory contains the following elements:

- a popup menu from which you select the operation to apply. You are not limited to a single use of an operation; for example, you can apply multiple Replace All operations in a single factory.
- an Options... button, which is used to configure any settable options for the command. If the command has no settable options, this button is disabled.
- a Comments button, which brings up a dialog you can use to record comments about the configuration or purpose of the action.
- Add (“plus”) and Remove (“minus”) buttons. Clicking the Remove button in an action will remove that action; clicking the Add button will insert a new action after the selected action. Hold down the Option key while clicking the Add button to create a new action which duplicates the action next to that button. If there is only one action in the factory, its Remove button is disabled.
- a summary line which describes the chosen command and any parameters.

Note It is up to you to make sure that your actions do not work at cross purposes. For example, it would not be useful to have an “Educate Quotes” operation followed immediately by a “Straighten Quotes” operation.

You can re-order actions by clicking the line containing an action (in any place not occupied by a button or popup menu) and dragging it to a new location.

Each operation available on an action's pop-menu behaves similarly to its counterpart command on the Text menu. When choosing and configuring operations, you should keep the following differences in mind.

- Change Line Endings and Change Text Encoding will only affect the line endings and text encoding of the file that BBEdit writes out when you choose Save to Disk from the factory's Options sheet. Neither operation changes any file contents in memory, so they will have no visible effect if the document is left open (either by choosing the Leave Open button in the Options sheet, or by opting to leave the document open when confirming a save).
- When you use the Run AppleScript Filter operation, your script should be written with an entry point named "ApplyTextTransform". The input parameter to this entry point is a Unicode string containing the file's contents. This entry point should return the file's contents as a Unicode string (or something which can be directly coerced to one):

```
on ApplyTextTransform (fileData
-- do something to fileData

return fileData -- or some reasonable facsimile thereof
end
```

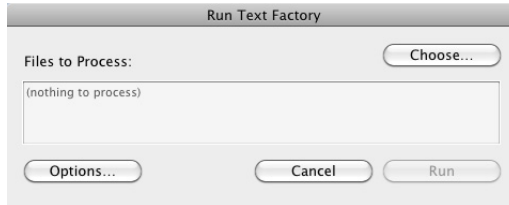
- The "Run Unix Filter" action can reference any executable file; such files need not contain a shebang line.
- Unix filters run from text factories get their input on STDIN, and should write any content they want passed to the next filter stage to STDOUT. (Unlike Unix filters run from the Shebang menu, text factories do not pass a temp file in argv[1].)
- When you use the "Run Unix Filter" operation, you can choose to have BBEdit convert the file's contents and the output to UTF-8 by setting the "Use UTF-8 for I/O" option. If you do not set this option, BBEdit will use the system encoding instead.

Applying Text Factories to Files

Once you have configured a text factory and selected the files and folders to process, click the Apply button in the lower right-hand corner of the window. BBEdit will apply the actions in the order you specify, to each file in the target set. This processing happens in the background, so you can keep using BBEdit while it's underway (similar to a multi-file search operation).



To run a stored text factory, choose it from the Scripts menu. If the text factory has any saved targets, BBEdit will apply the text factory to them. Otherwise, BBEdit will display the Run Text Factory dialog so you may choose a set of target items to process.



Applying Text Factories to Open Documents

You can apply a stored text factory to the current document, or to the contents of the current selection, by choosing it from the Apply Text Filter submenu of the Text menu, or by double-clicking it in the Text Filters palette.

HTML Processing Actions

Text factories support the following actions based on the equivalent Markup menu commands. (See Chapter 11 for details.)

- Tidy: Clean
- Tidy: Reflow
- Tidy: Convert to XHTML
- Tidy: Convert to XML.
- Translate Text to HTML
- Translate HTML to Text
- Format Markup
- Optimize Markup

Automator Actions

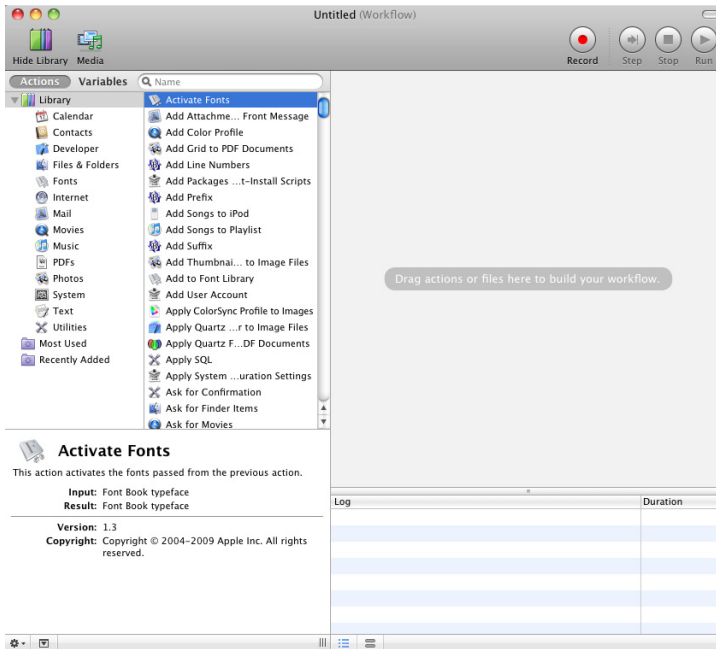
Automator is a system feature which enables you to create and reuse actions to easily perform common or time-consuming tasks. The Mac OS X Automation web site offers a good overview of Automator.

<http://www.macosxautomation.com/automator/>

Using BBEdit with Automator

BBEdit offers Automator actions which correspond to most of the text transformation commands on the Text menu. In order to use any of BBEdit's actions, BBEdit must be running.

After you launch Automator, it will create an Untitled workflow as shown below. Choose any application in the Library section to see what actions it offers. To add an action to your workflow document, you can double-click on the action, or drag it into the right-hand panel. After you add an action to your workflow, you can set options (if any) in the action's tile. To expand or collapse an action tile, click the triangle next to the action's name.



The Automator actions which BBEdit offers allow you to make use of BBEdit's unique transformations in the context of a larger Automator workflow. Please keep in mind that BBEdit currently does not include any actions which support working with files (other than opening them), as it's expected that you will be applying BBEdit actions to work on text acquired from other sources in the workflow. If your main need is to apply BBEdit's text transformations to files or sets of files, you would be better served by using a Text Factory.

Available Actions

Add/Remove Line Numbers

This action will prepend a line number to each line processed, or remove the leading line number from each line processed.

Input: (Text)

Result: (Text) Text with leading line numbers added or removed.

Change Case

This action will change the case of the provided text according to the specified rules.

Input: (Text)

Result: (Text) Text with the case transformation applied.

Convert Spaces to Tabs

This action will convert each run of N space characters into a tab character.

Input: (Text)

Result: (Text) Text with runs of space characters converted to tabs.

Convert Tabs to Spaces

This action will convert each tab character into the specified number of spaces.

Input: (Text)

Result: (Text) Text with tab characters converted to runs of spaces.

Convert to ASCII

This action will convert the input text into an ASCII-only representation.

Input: (Text)

Result: (Text) Text with non-ASCII characters “translated” into ASCII

Delete Lines Containing

This action will return all lines in the input string which do not match the specified string.

Input: (Text)

Result: (Text) Text with the complement of the matching lines.

Related Actions: Extract Lines Containing

Educate Quotes

This action will convert ASCII quotes into their typographer equivalent.

Input: (Text)

Result: (Text) Text with ASCII quotes converted to typographer’s quotes

Related Actions: Straighten Quotes

Extract Lines Containing

This action will return all lines in the input string which contain the specified string.

Input: (Text)

Result: (Text) Text with the matching lines.

Related Actions: Delete Lines Containing

Get Contents of BBEdit Document

This action will get the contents of the frontmost text document in BBEdit according to the options you set.

Requires: A text document must be open.

Result: (Text) A BBEdit document object

Related Actions: Set Contents of BBEdit Document, New BBEdit Document

New BBEdit Document

This action creates a new BBEdit document from the input text.

Input: (Text)

Result: (com.barebones.bbedit.document-object) A BBEdit document object

Related Actions: Get Contents of BBEdit Document, Set Contents of BBEdit Document

Normalize Line Breaks

This action will change all line breaks into the format specified.

Input: (Text)

Result: (Text) Text with the chosen line breaks.

Prefix/Suffix Lines

This action will insert (or remove) the specified prefix and/or suffix strings on each line processed.

Input: (Text)

Result: (Text) Text with the specified strings inserted on (or removed from) each line.

Remove Duplicate Lines

This action will eliminate duplicates from the provided lines according to the specified rules.

Input: (Text)

Result: (Text) Unique Lines of Text

Related Actions: Sort Lines

Remove Prefix

This action will remove the supplied string from the beginning of each line processed.

Input: (Text)

Result: (Text) Text with the specified prefix removed from each line.

Related Actions: Add Suffix, Remove Suffix

Remove Suffix

This action will remove the supplied string from the end of each line processed.

Input: (Text)

Result: (Text) Text with the specified suffix removed from each line.

Related Actions: Add Suffix, Remove Prefix

Search and Replace

This action will search the input string, and replace all matches with specified string.

Input: (Text)

Result: (Text) Text with the specified replacements done.

Set Contents of BBEdit Document

This action will replace or insert the passed text into the frontmost text document in BBEdit according to the options you set.

Requires: A text document must be open.

Input: (Text)

Result: (com.barebones.bbedit.document-object) A BBEdit document object

Related Actions: Get Contents of BBEdit Document, New BBEdit Document

Sort Lines

This action will sort the provided lines according to the specified rules.

Input: (Text)

Result: (Text) Sorted Lines of Text.

Related Actions: Remove Duplicate Lines

Straighten Quotes

This action will convert typographer quotes into their ASCII equivalent.

Input: (Text)

Result: (Text) Text with typographer's quotes converted to ASCII quotes

Related Actions: Educate Quotes

Zap Gremlins

This action will convert each non-ASCII character as specified in the configuration.

Input: (Text)

Result: (Text) Text with non-ASCII characters removed.

Related Actions: Convert to ASCII

Windows & Palettes

This chapter describes the commands in the Window menu. These commands allow you to arrange and access editing and browser windows quickly, and also to access BBEdit's extensive set of tool and utility palettes.

In this chapter

| | |
|-------------------------------------------------------------------------------|-----|
| Window Menu | 131 |
| Minimize Window | 131 |
| Bring All to Front..... | 131 |
| Palettes | 131 |
| <i>ASCII Table – 132 • Clippings – 132 • 1Clippings – 132 Scripts – 133 •</i> | |
| <i>Stationery – 133 • HTML Markup Tools – 134 • Workspace – 134</i> | |
| Workspace | 134 |
| Save Default <type of>Window | 134 |
| Arrange..... | 135 |
| Zoom (key equivalent only)..... | 135 |
| Cycle Through Windows | 135 |
| Exchange with Next..... | 135 |
| Synchro Scrolling..... | 135 |
| Window Names | 135 |

Window Menu

The Window menu provides easy, centralized access to all of BBEdit's tool and utility palettes, in addition to offering commands that you can use to access and organize editing and results windows on screen.

Minimize Window

This command puts the frontmost window into the Dock. Click the window icon in the Dock to restore the window. Hold down the Option key and this command becomes Minimize All Windows.

Bring All to Front

This command brings all un-minimized BBEdit windows to the front.

Palettes

The Palettes submenu provides quick access to all of BBEdit's numerous tool palettes and utility windows. Choosing an item from this submenu toggles display of the corresponding palette.

When moved or resized, palettes now automatically “snap” to the edges of the screen and the edges of other palettes. You can override this behavior by holding down the Shift key while dragging or resizing.

ASCII Table

The ASCII Table command opens a palette that contains the 127 entries of the ASCII character set plus all of the standard extended (8-bit) Macintosh character set (Mac Roman).

The decimal value for each character is displayed in the left-hand column, while in the right-hand column, the character value is displayed in either hexadecimal “escape” format, or in URL-encoded format, based on the language mapping of the frontmost text window. (The values shown for all extended characters are their Unicode values, rather than the equivalent Mac Roman values.)

Depending on the modifier keys you hold down, the Insert button inserts the selected character in different formats:

| Clicking Insert while holding... | Inserts in this format... |
|----------------------------------|--------------------------------------------------------------------------|
| None | Escape code appropriate to the front window—for example, (\x69) or (%69) |
| Option | Decimal value—for example, (105) |
| Command | Literal character—for example, (i) |

Note You can also double-click on a line in the ASCII table to insert the corresponding character or character code into the editing window.

Clicking the Show button in the ASCII Table window displays the ASCII value of the character to the right of the insertion point or the first character of the selection.

Clippings

BBEdit’s powerful Clippings feature provides an easy way to store and access frequently used text of any sort. For details on using Clippings, please see “Using Clippings” on page 255.

Colors

This command opens the system color picker, which you can use to insert hex color values into HTML and XML files.

Filters

The Filters palette displays all the text filters currently present in the Text Filters subfolder of BBEdit’s application support folder. See Chapter 2, “Text Filters”, for more information about using text filters in BBEdit.

Scripts

The Scripts palette displays all the scripts currently installed in the Scripts subfolder of BBEdit's application support folder. See Chapter 2, "Scripts", for more information about using scripts in BBEdit.

Stationery

The Stationery List is a palette that displays all the stationery pad files present within the Stationery folder of BBEdit's application support folder. You can create a new document from any of these pads by double-clicking it in this list. Although the document created will have the content and all the state information from the stationery pad, it is a new untitled document separate from the stationery pad.

To create a stationery pad, click the Save As Stationery checkbox when saving the file from BBEdit. Alternately, any document can be changed into a stationery pad in the Finder by clicking the Stationery Pad checkbox in the document's Get Info window.

By default, items in the Stationery List are displayed in alphabetical order. However, you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example "(00)Web template" would sort before "(01)HTML Template.") For such files, the first three characters are not displayed in BBEdit. You can also insert a divider by including an empty folder ending with the string "-***". (The folder can be named anything, so it sorts where you want it.) These conventions are the same as those used by the utilities FinderPop and OtherMenu.

Note In the Clippings, Filters, Scripts, and Stationery palettes, the Set Key button allows you to assign key equivalents to any item contained in that window. You can use combinations of the Command, Shift, Option, and Control keys, plus any single other key, to create such equivalents, except that any equivalent must contain either the Command or Control keys (or both). You can also map Function keys directly to items, with or without the use of a modifier.

Windows

The Windows palette displays the names of all open windows ordered by name and kind. You can open a file by dragging its icon from the Finder or from a project window into the Windows window.

Document windows, which correspond to text files, have a document icon next to them; display windows, such as browsers and search results windows, do not. A solid diamond to the left of a window's name means that the window's contents have been modified and have not yet been saved, while a hollow diamond indicates that the window's state has been modified but not yet saved.

To bring any window to the front, click its name in the Windows palette. You can select one or more windows in the list, and choose the Save, Close, or Print commands from the action menu at the top of the palette. Holding down the Option key changes these commands to Save All, Close All, and Print All, which apply to all listed windows for which the given command is possible. You can also Control-click on any selected windows and apply the Save, Close, or Print commands from the resulting contextual menu.

“Hovering” the mouse over a window name displays a tool tip showing the full window title; this is useful for names that have been truncated with ellipses (...) because they are too long to fit within the width of the window. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Holding down the Command key displays the full pathname for document windows (or other relevant windows such as disk browsers and FTP browsers).

HTML Markup Tools

The main HTML Markup Tools palette is a comprehensive listing of BBEdit’s numerous HTML markup commands. See “HTML Tools Palette” on page 251 in Chapter 11 for details on what these commands do.

Several other HTML palettes are available, each with a specific focus: CSS, Entities, and Utilities. For more information on these tools, please see Chapter 11, “BBEdit HTML Tools.”

Workspace

The Workspace submenu gives you the ability to save and reopen sets of palettes. You can choose the Save Workspace command to create a named workspace, the Delete Workspace command to delete a named workspace. If you choose a named workspace from this submenu, BBEdit will restore all persistent windows (all palettes, and the Preferences window) to the visibility state and positions they had when the workspace was created.

For example, you might create an “HTML” workspace with the HTML Tools and Windows palettes open, and a “Perl” workspace with the Unix Scripts and Unix Filters palettes open. Then, starting with only the Windows palette displayed, when you choose the “HTML” workspace, BBEdit will open the HTML Tools palette and position both palettes according to their saved locations. If you then choose the “Perl” workspace, BBEdit will close both these palettes, and open the Unix Scripts and Unix Filters palettes.

Show Scratchpad

This command opens BBEdit’s persistent Scratchpad window. The scratchpad provides a workspace where you can store text to manipulate without needing to manage files. It’s ideal as a ‘worktable’ where you can accumulate and modify text from one source (by performing transforms, manual edits, or batches of copy/paste operations) before gathering the text and using it elsewhere.

You can use the Save a Copy command in the File menu to save the scratchpad’s contents into a stand-alone text file.

Show Unix Worksheet

This command opens BBEdit’s persistent Unix Worksheet window.

Save Default <type of >Window

The Save Default Window command stores the position and size of the front window in BBEdit’s preferences, and BBEdit will create all new windows of the same type with the stored position and size.

By default, new windows always stack down and right 20px. If you have saved a default window size, and that window is of full screen height, new windows will just stack to the right, and preserve their saved height.

Each type of window has its own default position and size. For instance, the default position and size for project windows is different from the default position and size for text windows.

Window position and size preferences are also keyed to the active screen configuration, so if you frequently switch screen layouts (as when connecting an external display to a portable), you can save separate default window preferences which will be applied depending on which screen configuration is active.

Arrange

The Arrange command cascades all open editing windows in the default fashion: each successive window will be moved incrementally down and to the right (as described above).

When you hold down the Option key, this command toggles to Tile Two Front Windows, and choosing it will produce the specified outcome.

Cycle Through Windows

This command sends the front window behind all the other windows. Hold the Shift key down when choosing this command to Cycle Through Windows Backwards, i.e. to bring the rearmost window to the front.

Exchange with Next

This command makes the second window the active window. Choose this command repeatedly to alternate between the front two windows.

Synchro Scrolling

When you have two or more windows open, Synchro Scrolling makes both files scroll when you scroll one. This feature is useful to look over two versions of the same file.

Window Names

The last items in the Window menu are the names of all the open documents, browsers, and other editing windows. Choose a window's name from this menu (or use its numbered Command key equivalent, if applicable) to bring that window to the front.

Tip You can also use the Windows palette to quickly select any open window.

Zoom (key equivalent only)

There is no longer a Zoom command in the Window menu, but the key equivalent Command-/still works. Zoom will produce the same effect as clicking a window's zoom box: it makes the active window larger if it is small, or returns it to its original size if it was previously enlarged by a Zoom command.

When zooming windows, BBEdit will move the window as little as possible (consistent with maximizing the window's size). This behavior is similar to what the Finder does when zooming a window.

This chapter describes BBEdit’s powerful Find and Multi-File Search commands, now enhanced with a flexible file filtering mechanism. It tells you how to search for text in the active window or within a set of files. BBEdit can also do advanced pattern, or grep, searching. To learn about pattern searching, you should read this chapter first and then read Chapter 8, “Searching with Grep.”

In this chapter

| | |
|--------------------------------------------------------------------|-----|
| Search Windows..... | 137 |
| Basic Searching and Replacing | 138 |
| <i>Search Settings</i> – 140 • <i>Special Characters</i> – 140 | |
| Multi-File Searching | 141 |
| <i>Find All and Multi-File Search Results</i> – 143 | |
| <i>Specifying the Search Set</i> – 144 | |
| <i>Multi-File Search Options</i> – 146 • <i>File Filters</i> – 147 | |
| <i>Searching SCM Directories</i> – 149 | |
| Multi-File Replacing | 149 |
| Live Search..... | 150 |
| Search Menu Reference..... | 151 |

Search Windows

BBEdit’s Find and Multi-file Search windows provide a consistent modeless interface to BBEdit’s powerful text search and replace capabilities.

If you are familiar with the Find dialog used in older versions, you’ll generally feel at home, but there are some important differences and improvements of which you should be aware:

The Find dialog has been split in two, with a Find window for searching only the front document, and a Multi-File search window for searches which span more than one document (including folders, arbitrary open documents, BBEdit and Xcode projects, etc.).

The set of search options which configure how text is actually searched (for single-file searches) has been condensed down to a single pair of options: “Selected text only” and “Wrap around”.

- “Selected text only” affects **only** the Find All and Replace All operations: if there is a selection range in the front document, these operations will affect search only the contents of the selection range if this option is on, or the entire document (starting from the top) if this option is off.

- “Wrap around” affects **only** the “Next”, “Previous”, “Replace”, and “Replace & Find” operations: if this option is on and the search reaches the end (or the beginning) of the document, then BBEdit will continue the search from the appropriate end of the document.

Keyboard navigation is considerably different due to the Find and Multi-File Search windows’ modeless nature.

- Pressing the Return or Enter key with focus in the Find field will perform “Next” in the Find window or “Find All” in the Multi-File Search window.
- Pressing the Escape key will close the window.
- Choosing an appropriate command in the Search menu will trigger the corresponding action in the front Find window.
- BBEdit supports the following key equivalents to control (toggle) the search options contained in the Find and Multi-File Search windows. The factory default key equivalents for these options are as follows:

| | |
|---------------------|-----------------|
| Case sensitive | Control-shift-N |
| Entire word | Control-Shift-E |
| Grep | Control-Shift-G |
| Selected text only | Control-Shift-S |
| Wrap around | Control-Shift-W |
| Open search history | Control-Shift-H |
| Open saved patterns | Control-Shift-P |

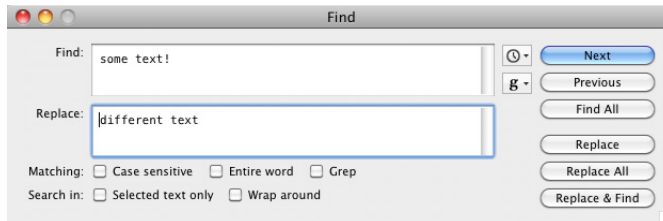
If these assignments overlap with any keyboard equivalents for clippings that you have set, or if you just wish to change them, you can do so via the “Find Windows” section of the Menus & Shortcuts preference panel.

Note The “Replace All” command replaces **all** occurrences of the search string within the document (or in the selection if there is one and “Search Selection Only” is checked). If you wish to replace only occurrences between the current insertion point and the end of the document, you can instead apply the Replace to End command in the Search menu.

Basic Searching and Replacing

This section describes the basic steps for searching and replacing text in a document. Later sections in this chapter cover more advanced techniques. To search and replace text in the front document, follow these steps:

- 1 Choose Find from the Search menu. BBEdit opens the Find window.



2 Type the string you are looking for in the Find text field.

You can use special characters in the Find text field to search for tabs, line breaks, or page breaks. See “Special Characters” later in this section.

3 Type the replace string (if any) in the Replace text field.

BBEdit persistently remembers the pairs of search and replace terms that you have most recently used. If you want to repeat a previous search or replace, you can choose the appropriate entry from the Search History popup menu at the right of the Find text field to fill in the Find and Replace fields.

Note The size of both the search and replace terms is limited only by available memory.

4 Turn on any options that you want to apply to your search.

For more info about these options, see “Search Settings” later in this section.

5 Click one of the buttons along the right side of the dialog box.

The following table explains what each of the buttons does.

| This button... | Does this... |
|----------------|--------------------------------------------------------------------------------------------------------------------------|
| Next | Finds the first occurrence of the text in the active window after (below) the current insertion point. |
| Previous | Finds the first occurrence of the text in the active window before (above) the current insertion point. |
| Find All | Finds all the occurrences of the search string and displays the results in a search results browser. |
| Replace | Replaces the current selection with the replace string. |
| Replace All | Replaces every occurrence of the search string in the active window with the replace string. |
| Replace & Find | Replaces the current selection with the replace string, then finds the next occurrence of the text in the active window. |

Once you have entered a search string (and also, if desired, a replace string), you can also use the commands in the Search menu to find and replace matches (see “Search Menu Reference” later in this chapter). The table below summarizes the most common commands you can use at this point.

| This command... | Does this... |
|-----------------|---------------------------------------------------------------------------------------------------|
| Find Next | Finds the next occurrence of the search string. To reverse the search direction, hold down Shift. |
| Replace | Replaces the selection with the replace string. |
| Replace All | Replaces all occurrences of the search string within the document with the replace string. |

| This command... | Does this... |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Replace to End | Replaces every occurrence of the search string from the current insertion point to the end of the document with the replace string. |
| Replace & Find Again | Replaces the selection with the replace string and looks for the search string again. |

Search Settings

The checkboxes in the Find window lets you control how BBEdit searches your document for the indicated text.

Case Sensitive

When this checkbox is selected, BBEdit treats upper- and lowercase letters as different letters. Otherwise, BBEdit treats upper- and lowercase letters as if they were the same.

Entire Word

When this checkbox is selected, BBEdit matches the search string only if it is surrounded in the document text by word-break characters (white space or punctuation). Otherwise, BBEdit matches the search string anywhere in the text.

Grep

When this checkbox is selected, BBEdit treats the search and replace strings as grep patterns. Otherwise, BBEdit searches the document for text that matches the search string as it appears literally, and will replace any matched text with the replace string. To learn more about pattern searching see “Searching with Grep” on page 157.

Selected Text Only

When this checkbox is selected, BBEdit searches only the selected text. Otherwise, BBEdit searches the entire document.

Wrap Around

When this checkbox is selected, BBEdit continues searching from the beginning of the document if a match is not found (or from the end of the document if searching backwards). Otherwise, BBEdit stops searching when it reaches the end (or the beginning if searching backwards) of the file.

Special Characters

You can use the following special characters to search for line breaks and other non-printing characters, as well as hexadecimal escapes to search for any desired 8-bit character.

| Character | Matches... |
|-----------|------------------------------|
| \r | line break (carriage return) |
| \n | Unix line break (line feed) |
| \t | tab |

| Character | Matches... |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\f</code> | page break (form feed) |
| <code>\xNN</code> | hexadecimal character code NN (for example, <code>\x0D</code> for CR) |
| <code>\x{NNNN}</code> | any number of hexadecimal characters NN... (for example, <code>\x{0}</code> will match a null, <code>\x{304F}</code> will match a Japanese Unicode character) |
| <code>\\</code> | backslash (<code>\</code>) |

The form of a hex escape is “\xNN”, where “N” is any single hex digit [0-9,A-F]. The “x” may be upper or lower case. (You can use the ASCII Table in the Window menu to find the hex value for any 8-bit Macintosh character.) You can perform a literal search for any character, including a null, using this option. Malformed escapes are treated as literal strings.

Multi-File Searching

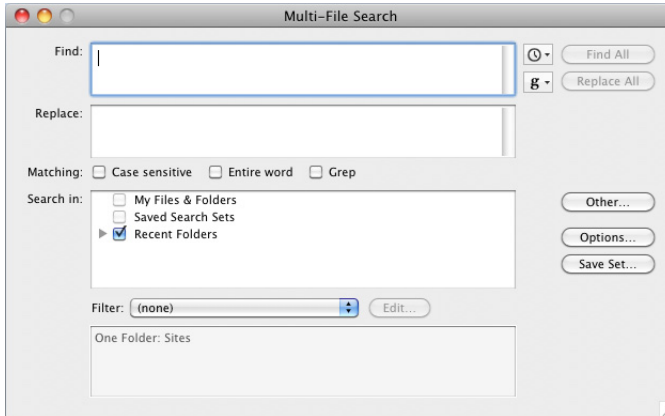
The main difference between single-file searching and multi-file searching is that to perform a multi-file search, you must specify the files to be searched. BBEdit gives you a great deal of flexibility in how to do this. You can search all the files in a given folder or defined web site, in a project, in open editing windows, or in an existing search results browser. For even greater control, you can select a diverse set of search sources, or apply BBEdit’s advanced file filtering capabilities.

When you start a search, BBEdit will display a search progress window and return control, so that you can continue working. You can perform more than one multi-file searches at a time; each search will have its own progress window. Closing a search’s progress window or clicking Cancel in the progress window will stop the operation, and BBEdit will display a search results browser containing any matches found up to that point.

Starting a Search

To search for a string in multiple files, do the following steps:

- 1 Choose Multi-File Search from the Search menu, or type Command-Shift-F, to open the Multi-File Search window (if it is not already open).**



- 2 Type the string you are looking for in the Find text field.**

- 3 Type the replace string (if any) in the Replace text field.**

Be sure to read the section “Multi-File Replacing” later in this chapter if you want to perform replace operations.

- 4 Turn on any search options that you want to apply to your search.**

To learn more about these options, see “Search Settings” earlier in this chapter.

- 5 Drag a folder to the search target area to search its contents, or select any of the available search sources in the Sources list to specify the set of files to search.**

See “Specifying the Search Set” later in this chapter for more information.

- 6 Click one of the buttons along the right side of the dialog box to begin the search.**

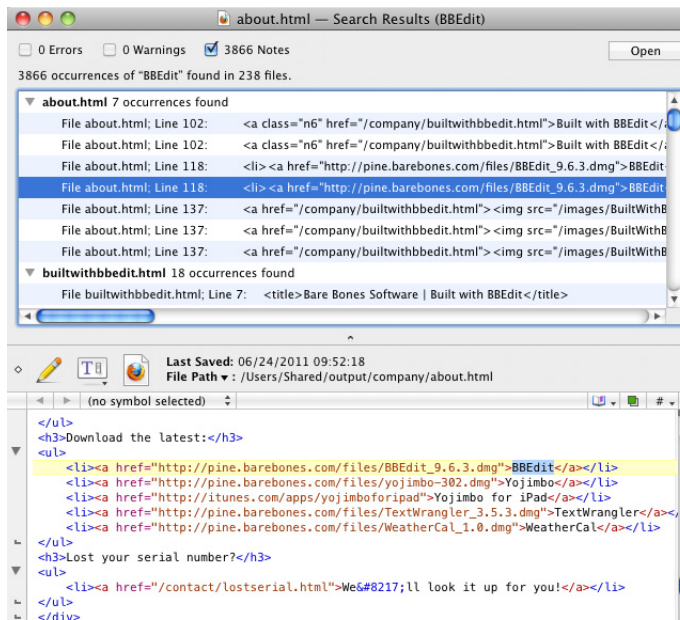
The table below tells you what each of the buttons does.

This button... Does this...

| | |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Find All | Finds all occurrences of the search string in all the files in the selected search sources. BBEdition displays the results in a search results browser. |
| Replace All | Finds all occurrences of the search string in all the files in the selected search sources and replaces them with the replace string. |
| Options | Brings up the Search Options sheet. |
| Save Set | Creates an entry under the "Saved Search Sets" heading in the search sources list which you can later choose to reselect the same search sources. |
| Other | Select arbitrary file(s) or folder(s) to add to the search sources. |

Find All and Multi-File Search Results

When you perform a Find All search, either on a single file or across multiple files, BBEdition will open a search results browser which lists every occurrence of the search string in the selected file(s)



The information at the top of the window tells you how many matches BBEdition found in the set of files you specified, as well as specifying whether there were any error conditions or warnings generated during the search. You can display or hide any combination of errors, warnings, and matches, by checking the appropriate options.

The middle panel lists each line that contains the matched text. Every match is identified by file name and line number. You can toggle between the standard Finder-style hierarchical list, where each match in a file is listed under the file's name, or a flat list where each occurrence is displayed in order, by pressing the File List button next to the Open button.

Click any match in the list of occurrences to have BBEdit display the part of the file that contains the match in the text pane.

IMPORTANT

You can edit text directly within a search results browser, or double-click any line that contains a match to open the corresponding file at the point of the match.

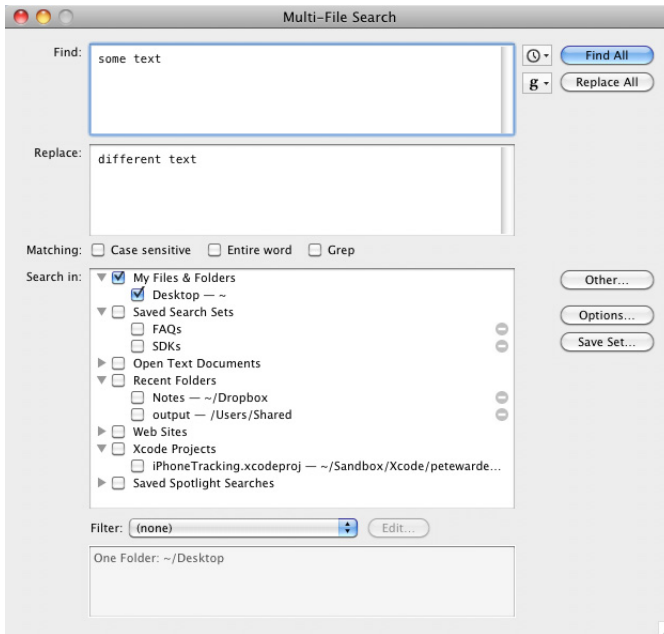
After you have opened a file, you can use the Find Again, Replace, Replace All, and Replace & Find Again commands in the Search menu to continue searching it, as if you had chosen a File by File search. See the next section for information on File by File searching.

Note

You can use a search results window as the basis of another multi-file search. See "Specifying the Search Set" below.

Specifying the Search Set

To specify which files and folders BBEdit should examine when performing a multi-file search, just select items from the Search In list of the Multi-File Search window.



You can choose multiple sources for a multi-file search, and you can mix different types of sources. Available sources include:

- specified individual files
- the files in any selected or recently-searched folder
- open text documents

- the frontmost project
- the files listed in any results browser (such as a search results browser, an HTML syntax errors browser, or a compile errors browser)
- the files and folders contained in a project
- the files and folders contained within any Zip archives
- any Xcode projects in your home directory
- any “Smart Folders” which you have saved in the Finder (BBEdit will automatically list any such items present in the “Saved Searches” folder for your login account)

To add a file, folder, or other suitable item to the Search In list, click Other in the Multi-File Search window, and choose the item using the resulting selection sheet. (You can select multiple items to be added.)

To designate any item in the list as a search source, click on the box next to its name, or double-click on the name, to add a checkmark. To deselect a search source, click the box next to its name, or double-click its name, to turn off the checkmark. To select a single source only, and deselect all other sources, Command-click on the checkbox next to the desired source’s name. To remove a search source from the list, click on the minus sign (–) to the right of its name. (Doing so removes only the entry from the list, **not** the original item.)

BBEdit will display a summary of the selected sources in the information box at the bottom of the Multi-File Search window. Here are some common scenarios.

Searching the files in a folder

To search the files in a folder, click on the box next to the folder’s name, or double-click its name, in the Sources list. If the folder you want to search is not in the Sources list, click the Other button at the right of the dialog and pick the folder using the resulting selection sheet.

You can also drag a folder from the Finder directly into the search items box of the Find & Replace dialog to choose it as the source.

Searching the frontmost project

You can directly search all files contained within the frontmost project document. To choose the frontmost project, click the box next to the Frontmost Project item, or double-click on the item in the list.

If the active document is also a project document, you can directly target it by choosing the Search in [project name] command (in the Search menu).

Searching all open documents

You can choose any or all open text documents as search sources. This option allows you to search documents that have not yet been saved to a file, or which contain unsaved changes. To choose all open documents, click the box next to the Open Text Documents item, or double-click on the item in the list.

Searching the contents of compressed archives

You can control whether BBEdit should search within the contents of compressed archives (“.bz2”, “.gz”, and “.zip”) via the “Search compressed files” option in the Multi-File Search window’s “Options” sheet. When this option is off, BBEdit will skip all bz2, gz, and Zip while searching, even if they may contain compressed text files.

Searching the files contained in a results browser

If a previous multi-file search found many files that contain your search string, you may want to narrow the search. To search the files listed in any results browser window, click the box next to that browser’s name, or double-click on its name, in the Sources list. You can also click the box next to the Results Browsers item, or double-click on this item, to search the files listed in all results browsers.

Searching the files in a project

If the files you are working with are all listed in a BBEdit project, you can choose that group in order to search the files. To choose a project, click the box next to that group’s name, or double-click on its name, in the Sources list.

Searching the files on a web site

You can limit your multi-file search to the files within a specific web project as defined in the Sites panel of the Setup window.

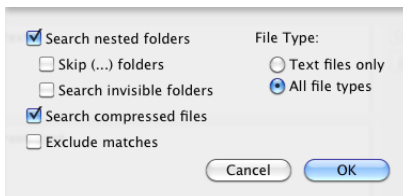
Note The Choose a Folder dialog will display any packages it encounters as folders (rather than just as single files, the way they appear in the Finder). This allows you to navigate their internal structure just as you would any other folder. Similarly, you can drag a package from the Finder into the path box in the Find & Replace dialog and it will be treated as a true folder rather than as a single file.

Saved Search Sources

You can use the Saved Search Sources popup menu to store specific sets of search sources for later reuse. To save a set of search sources, choose Remember this Set from the popup menu and give the set a name in the resulting dialog. To select a saved set of search sources, choose that set’s name from the pop-menu.

Multi-File Search Options

Click the Options button to display the search options sheet.



To search the contents of all subfolders within the folders you choose, select the Search Nested Folders option in the resulting sheet. You can also choose to skip any folders whose names are enclosed in parentheses here by selecting the Skip (...) Folders option, or whether to search the contents of invisible folders by selecting the Search Invisible Folders option.

You can also choose to search only text files or to search all file types. If you have image files or other non-text files in search source folders, it may be a good idea to restrict the search to only text files. This setting is applied in addition to any file filter (see next section) and in fact takes effect *before* the filter.

To find only files whose contents do **not** contain the search string, select the Exclude Matches option.

You can further restrict which files from the chosen sources will be searched by applying a file filter. See “File Filters” (below) for more details.

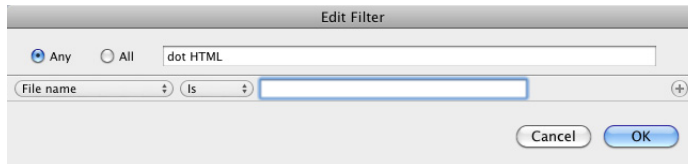
File Filters

If you do not want to search every file in the set you selected, but want to include only those that meet certain criteria (such as those created on a certain date, or only those created by BBEdit and not some other program, or those that are HTML or Perl documents), you can use a file filter.

To apply a file filter, just choose it from the Filters popup menu in the Multi-File Search window. If none of the available filters meets your needs, you can define a new one, or create a temporary filter.

New Filter

To define a new saved file filter, select New... from the popup menu. BBEdit will ask you for a filter name, and then display the Edit Filter dialog (below). You can also define new file filters in the Filters panel of the Setup window (see page 212).



Note If the Setup window is open, any filters you define in the Multi-File Search window will not be available in the Filters panel of the Setup window until you close and reopen the Setup window.

The Edit Filter dialog lets you specify multiple criteria that determine whether a given file is selected by the filter. You can choose whether these criteria are exclusive (that is, whether a file must meet *every* listed test to be selected) or inclusive (that is, whether a file that meets *any* of the tests is selected) using the Every (AND) and Any (OR) radio buttons at the top of the dialog.

To add a test, click the Add (+) button, and a new row will appear in the dialog.

Within each row (criterion), the left-hand popup lets you specify which attribute of a file you wish to test. BBEdit lets you test a file’s name, the name of its enclosing folder, its creator or type, its creation and modification date (or both date and time), or its Finder label, visibility, or the programming or markup language it is written in. You can also test the content of a file, using the “Contents” criterion.

The center popup lets you choose the test to be applied to the selected attribute. The available options here change depending on what attribute you selected. If you choose Visibility in the first column, for instance, your only choices are whether the file is or is not visible. However, if you choose File Name in the first column, the middle column lets you test to see if the name does or does not exactly match, contain, begin with, or end with a particular string. You can also test file names to see if they match wildcard or Grep patterns.

Note In wildcard patterns, the asterisk (*) and question mark (?) characters have special meanings. The asterisk matches any number of characters, such that "*.c" matches any file whose name ends with ".c". The question mark matches a single character, so that "foo?" matches "food", "fool", "foot", and many other words. Both the asterisk and the question mark can be used anywhere in a wildcard pattern, and any number of either can be used in a single pattern.

Grep patterns, also known as regular expressions, are a powerful method of selecting file names based on classes of text or repeating text. They are covered in great detail in the next chapter.

The right-hand text field specifies the match criterion. For example, when filtering by File Name, you type the text you want the name to match, contain, begin with, or end with (or not). When filtering by Language, you choose a supported language from a popup menu.

Specifying Time and Date Criteria

When using a time or date criterion, you can use the special words below to specify dates and times relative to the current date and time.

| Word | Means... |
|-----------|--------------------------------------|
| now | current date and time |
| today | midnight on the current date |
| yesterday | current date and time minus 24 hours |
| tomorrow | current date and time plus 24 hours |

You can add any number of criteria using the Add (+) button. To delete any criterion, click the Remove (-) button next to it.

Click Save to save the file filter and use it for this search. BBEdit will ask you to name the filter, and it will then appear in the Filters popup menu in the Find & Replace dialog (and in the Filter panel of the Setup window). Click Cancel to discard any changes you have made to the filter. (Hold the Option key when you click Cancel to skip the confirmation alert.)

Filtering by Name

In order to provide the greatest possible flexibility, BBEdit offers several different criteria for filtering based on file names

File Name: Tests the complete string corresponding to the file name.

File Name Root: Tests only the "root" portion of the file name. Given a name of the form "foo.txt", the root is the string which occurs before the period, in this case "foo".

File Name Suffix: Tests only the file name suffix. In the above example, the suffix is “txt”. Note that the suffix does not include the period.

Temporary Filters

Choose “(current criteria)” from the popup menu in the Find & Replace dialog to reuse the last set of criteria applied (either from using a saved filter, or from using the Edit button to define criteria). Thus, you can use filter criteria on the fly, without the need to create and store a throwaway filter.

Editing and Deleting Filters

To edit a file filter you have already defined, choose it from the Filters popup menu, change it as desired, and click Save. Since each filter must have a unique name, saving it will replace the old version of the filter. To delete a filter entirely, visit the Filters panel of the Setup window. (You can also create or modify filters there.)

Searching SCM Directories

When scanning folders for various purposes (multi-file search, Find Differences, and other batch operations), BBEdit ignores all directories which contain administrative data for source-control management (SCM) tools: CVS, .svn, .git, .hg, .bzi. This behavior prevents any inadvertent modifications to such data which might otherwise occurs during a multi-file search or other batch operation. If you must search the contents of such directories, you can enable BBEdit to do so by issuing the following Terminal command:

```
defaults write com.barebones.bbedit SkipSCMAdminDirsWhenScanningFolders -bool NO
```

Note The “Search Invisible Folders” option no longer enables BBEdit to search within such directories.

Multi-File Replacing

If you want to replace only some occurrences of text in multiple files, you can simply search those files, select the instances you want to change in the search results browser to open the files to those points, and perform the replacements individually. However, BBEdit can also change *all* occurrences of a string in a group of files with one command.

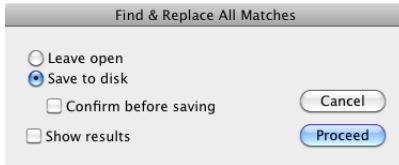
Globally replacing text in more than one file works the same as replacing it in a single file. The only possible complication is that, if you make a mistake, it can have much wider consequences. If you are not sure what effect a replace operation will have, test it out on a few sample files (or a copy of your data) first!

To do a multi-file search and replace, replacing all occurrences:

- 1 Enter your desired find and replace strings in the Multi-File Search window as described in the section “Multi-File Search.”**
- 2 Choose the files to be searched as described in “Specifying the Search Set”.**

3 To start the operation, click Replace All in the Multi-File Search window, choose the Replace All command, or type its key equivalent of Command-Option-R.

BBEdit displays the Find & Replace All Matches dialog box:



This is what each of its options does:

| | |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| This option... | Replaces all occurrences of the search string with the replace string and... |
| Leave Open | Leaves all the files open so that you can inspect the replacements. If there are many files that contain the search string, BBEdit may run out of memory. |
| Save to Disk | Saves each file with the changes. When the Confirm Saves setting is active, you will have an opportunity to approve the changes before BBEdit saves them to disk. You should <i>not</i> turn this off unless you are sure that the replace operation being done is what you want. |
| Show Results | Opens a results browser listing each of the files which was changed, and the number of changes in each file. |

Live Search

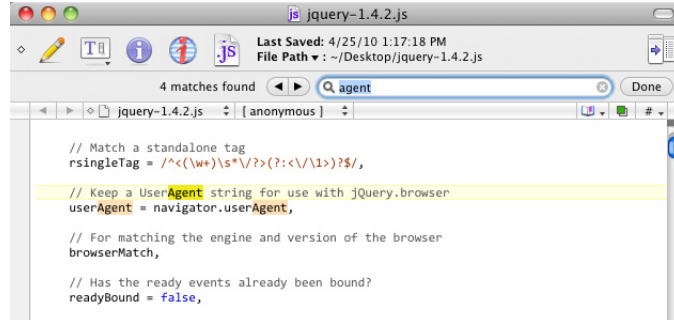
The Live Search command performs an incremental search. In other words, it shows the matching text as you type the search string, so you only have to type until you find the text you want.

Live Search always searches in the text view of the frontmost window; if that window has no text view, the Live Search command will be disabled.

To use Live Search:

- 1 Choose Live Search from the Search menu, or type Command-Option-F.**

2 Type the string you are looking for into the Live Search field.



As you type, BBEEdit selects the first occurrence of what you have typed so far.

3 To find the next occurrence of the matching text, click the Next (right) arrow, or type Return or Enter.

4 To find the previous occurrence of the matching text, click the Previous (left) arrow, or type Shift-Return or Shift-Enter.

If Emacs key bindings are enabled, you can also type Control-S to start a Live Search, and then type Control-S or Control-R to search forward or backward respectively.

To clear the most recent word of the search string, you can type Option-Delete, or click on the “delete” button (the “X”) within the search field to delete the entire search string.

To cancel Live Search, you may click the “Done” button in the search bar or type the Escape key.

Note The Live Search bar replaces the Quick Search window present in older versions.

Search Menu Reference

This section describes all of the commands in the Search menu.

Find

Opens the Find window (or the Find & Replace dialog). See “Basic Searching and Replacing” on page 138.

Multi-File Search

Opens the Multi-File Search window. See “Multi-File Searching” on page 141 and “Multi-File Replacing” on page 149.

Search in ... (Project or Disk Browser)

If the frontmost window is an editing window, this command’s name will reflect the name of the current document’s parent folder (if any).

If the frontmost window is a project or disk browser, this command’s name will reflect the name of the project, or the name of the directory currently visible in the disk browser.

Choosing this command will open the Multi-File Search window with the described target (the current document's parent folder, the current project, or the selected directory) set as the search source.

If the Multi-File Search window is frontmost, this command will target the project or disk browser which is closest to the front (Z-order).

This command is also available in the Action ('gear') menu of projects and disk browsers.

Live Search

Opens the Live Search bar. You can use this feature to interactively search for text strings, as described in the previous section.

Find Next/Previous

Searches the current document for the next occurrence of the search string. Hold down the Shift key to find the previous occurrence.

Find All

Finds all instances of the search string in the current document or search set, and displays a search results browser.

Find Selected Text/Previous Selected Text

Uses the selected text as the search string and finds the next occurrence of the selected text. Hold down the Shift key to find the previous occurrence of the selected text.

When you invoke this command, BBEdit will add the current search string to its Search History list of recently-used search strings.

Tip You can also hold down the Option and Command keys as you double-click on a selection to search for the next occurrence of the selected text.

Use Selection for Find

Sets BBEdit's search string to the currently selected text but does not perform a search.

When you invoke this command, BBEdit will add the current search & replace strings to its Search History list.

Use Selection for Find (grep)

When you hold down the Shift key, Use Selection for Find becomes Use Selection for Find (grep). This command sets BBEdit's search string to the currently selected text and turns on the Grep option, but does not perform a search. When you invoke this command, BBEdit will add the current search & replace strings to its Search History list.

Use Selection for Replace

Sets BBEdit's replace string to the currently selected text but does not perform a search operation. When you invoke this command, BBEdit will add the current search & replace strings to its Search History list.

Use Selection for Replace (grep)

When you hold down the Shift key, Use Selection for Replace becomes Use Selection for Replace (grep). This command sets BBEdit's replace string to the currently selected text and turns on the Grep option, but does not perform a search operation. When you invoke this command, BBEdit will add the current search& replace strings to its Search History list.

Replace

Replaces the selected text (usually an occurrence of the search string) with the replace string.

Replace All

Replaces **all** occurrences of the search string in a file with the replace string, or, starts a multi-file search & replace operation.

Replace to End

Replaces each occurrence of the search string from the current insertion point (or the start of the current selection range) to the end of the document.

Replace & Find Again

Replaces the selected text with the replace string and searches for the next occurrence of the search string.

Go to Line

When you choose this command, BBEdit opens the Go To Line dialog box. Type in a line number and the frontmost text window will jump to display that line.

This command does not follow the usual convention of applying the last-used setting when invoked with the Option key pressed. Instead, if you select a number within the current document, then hold down the Option key and choose "Go to Line", BBEdit will go to that numbered line.

Note The Go To Line command honors the "Use 'Hard' Line Numbering in Soft-Wrapped Text Views" option in the Editing preference panel.

Go to Center Line

Will move the insertion point to the beginning of the middle or center line of the displayed text.

Go to Previous/Next Error

If an error browser is open, this command will open the listed error which came before or after the selected error. See Chapter 9 for more information on error browsers.

Go to Previous/Next Placeholder

When you apply a clippings item that contains multiple #INSERTION# cookies, the second and subsequent cookies are replaced with special jump placeholders. You can also manually insert jump placeholders at any desired points within a clipping or a document.

These placeholders are strings of the form “<#...#>”, where the content “...” between the two # signs may be any alphanumeric text, or empty.

You can use the Go To Previous Placeholder and Go to Next Placeholder commands to jump back and forth between these special strings from the keyboard. For example, you might use this command when filling in the parameters of a function call, or a series of tag attributes. For additional details, see “Selection and Insertion Placeholders” on page 263.

If the “Use Tab key to navigate Placeholders” option in the Keyboard preferences panel is turned on, you can press the Tab key (or Shift-Tab) to navigate between placeholders.

In addition to jump placeholders, you can also insert “optional” placeholders of the form <#?#>. When the “Go to Next Placeholder” command would select such a placeholder, BBEdit will place the insertion point at the specified position and remove the optional placeholder.

Go to Function Start/End

When you choose one of these commands, BBEdit will move the insertion point to a position immediately before the start or immediately after the end of the current function, where a function is any item which appears on the function popup menu. If you anticipate using these commands often, you may wish to assign them key equivalents in the Menus & Shortcuts preference panel.

Go to Previous/Next Function

When you choose one of these commands, BBEdit will select the name of the previous or next function in the document, where a function is any item which appears on the function popup menu. If you anticipate using these commands often, you may wish to assign them key equivalents in the Menus & Shortcuts preference panel.

Jump Back

When you choose this command, BBEdit will go to the last selection you made in the document which was outside the current view (an automatic jump mark), or the last location you marked with the Set Jump Mark command (a manual jump mark--see below).

Jump Forward

When you choose this command after choosing Jump Back, BBEdit will go to the next later jump mark, or return to the most recent position of the insertion point. If you have not jumped back to a jump mark, this command is disabled.

Set Jump Mark

Choose this command to define the current insertion point location or selection range as a manual jump mark within the active document. You can navigate to jump marks using the Jump Back and Jump Forward commands.

Find Differences

Finds the differences between two files, or all of the files contained in two folders. See “Comparing Text Files” in Chapter 4 for more details.

Compare Two Front Documents

Performs a Find Differences on the two frontmost text documents, using the same settings currently active for the Find Differences command.

Compare Against Disk File

Performs a Find Differences between the contents of the front document and the disk file for that same document. This capability makes it easy to locate in-progress changes to a document.

Apply to New

Applies the currently selected difference to the “New” version of two files which are being compared. See “Comparing Text Files” for more details.

Apply to Old

Applies the currently selected difference to the “Old” version of two files which are being compared. See “Comparing Text Files” for more details.

Compare Again

Find the differences between two files, using the same settings that were used in the last time you used the Find Differences command. See “Comparing Text Files” for more details.

Find Definition

Looks up definitions for the selected word using ctags information if available. If there is no selection, BBEdit will attempt to determine the symbol name by inspection of the text around the insertion point, rather than requiring you to type a name. (See “Exuberant Ctags” in Chapter 14 for more details about working with ctags.)

Find in Reference

Performs a search for the selected symbol using an appropriate language-specific online resource. As for Find Definition, if there is no selection, BBEdit will attempt to determine the symbol name by inspection around the insertion point.

For example, Find in Reference in a PHP document will look up the selected symbol on php.net; in a Ruby document, it will use the ‘ri’ interactive reference; in a Unix Shell Script, it will open the appropriate Unix man page.

For languages which don’t have a pre-defined resources, lookups will performed on the Apple Developer Connection web site.

You can modify the URL template which BBEdit uses to perform the lookup for a particular language by bringing up the Options sheet for that language in the Languages preference panel and editing the template directly. In the template, use “__SYMBOLNAME__” to indicate where the selected symbol name should be placed in the lookup string.

Searching with Grep

This chapter describes the Grep option in BBEdit's Find command, which allows you to find and change text that matches a set of conditions you specify. Combined with the multi-file search and replace features described in Chapter 7, BBEdit's grep capabilities can make many editing tasks quicker and easier, whether you are modifying Web pages, extracting data from a file, or just rearranging a phone list.

In this chapter

| | |
|--------------------------------------------------------------------------------------|-----|
| What Is Grep or Pattern Searching? | 158 |
| Recommended Books and Resources | 158 |
| Writing Search Patterns | 159 |
| <i>Most Characters Match Themselves</i> – 159 | |
| <i>Escaping Special Characters</i> – 159 | |
| <i>Wildcards Match Types of Characters</i> – 160 | |
| <i>Character Classes Match Sets or Ranges of Characters</i> – 162 | |
| <i>Matching Non-Printing Characters</i> – 163 | |
| <i>Other Special Character Classes</i> – 164 | |
| <i>Quantifiers Repeat Subpatterns</i> – 165 | |
| <i>Combining Patterns to Make Complex Patterns</i> – 166 | |
| <i>Creating Subpatterns</i> – 166 • <i>Using Backreferences in Subpatterns</i> – 167 | |
| <i>Using Alternation</i> – 168 • <i>The “Longest Match” Issue</i> – 168 | |
| <i>Non-Greedy Quantifiers</i> – 169 | |
| Writing Replacement Patterns | 170 |
| <i>Subpatterns Make Replacement Powerful</i> – 170 | |
| <i>Using the Entire Matched Pattern</i> – 170 | |
| <i>Using Parts of the Matched Pattern</i> – 171 | |
| <i>Case Transformations</i> – 172 | |
| Examples | 173 |
| <i>Matching Identifiers</i> – 173 • <i>Matching White Space</i> – 173 | |
| <i>Matching Delimited Strings</i> – 174 • <i>Marking Structured Text</i> – 174 | |
| <i>Marking a Mail Digest</i> – 175 • <i>Rearranging Name Lists</i> – 175 | |
| Advanced Grep Topics | 175 |
| <i>Matching Nulls</i> – 176 • <i>Backreferences</i> – 176 | |
| <i>POSIX-Style Character Classes</i> – 177 | |
| <i>Non-Capturing Parentheses</i> – 178 | |
| <i>Perl-Style Pattern Extensions</i> – 179 • <i>Comments</i> – 179 | |
| <i>Pattern Modifiers</i> – 180 • <i>Positional Assertions</i> – 181 | |
| <i>Conditional Subpatterns</i> – 183 • <i>Once-Only Subpatterns</i> – 184 | |
| <i>Recursive Patterns</i> – 186 | |

What Is Grep or Pattern Searching?

Grep patterns offer a powerful way to make changes to your data that “plain text” searches simply cannot. For example, suppose you have a list of people’s names that you want to alphabetize. If the names appear last name first, you can easily put these names in a BBEdit window and use the Sort tool. But if the list is arranged first name first, a simple grep pattern can be used to put the names in the proper order for sorting.

A grep pattern, also known as a regular expression, describes the text that you are looking for. For instance, a pattern can describe words that begin with C and end in l. A pattern like this would match “Call”, “Cornwall”, and “Criminal” as well as hundreds of other words.

In fact, you have probably already used pattern searching without realizing it. The Find window’s “Case sensitive” and “Entire word” options turn on special searching patterns. Suppose that you are looking for “corn”. With the “Case sensitive” option turned off, you are actually looking for a pattern that says: look for a C or c, O or o, R or r, and N or n. With the “Entire word” option on, you are looking for the string “corn” only if it is surrounded by white space or punctuation characters; special search characters, called metacharacters, are added to the search string you specified to indicate this.

What makes pattern searching counterintuitive at first is how you describe the pattern. Consider the first example above, where we want to search for text that begins with the letter “C” and ends with the letter “l” with any number of letters in between. What exactly do you put between them that means “any number of letters”? That is what this chapter is all about.

Note Grep is the name of a frequently used Unix command that searches using regular expressions, the same type of search pattern used by BBEdit. For this reason, you will often see regular expressions called “grep patterns,” as BBEdit does. They’re the same thing.

Recommended Books and Resources

Mastering Regular Expressions, 3rd Edition

by Jeffrey E.F. Friedl. O’Reilly & Associates, 2006. ISBN 0-596-52812-4

Although it does not cover BBEdit’s grep features specifically, *Mastering Regular Expressions* is an outstanding resource for learning the “how-to” of writing useful grep patterns, and the second edition is even better than the original.

BBEdit Talk

The BBEdit Talk discussion group covers a wide range of topics and questions about using BBEdit, which frequently include searching and the use of grep patterns.

<http://groups.google.com/group/bbedit>

Note BBEdit’s grep engine is based on the PCRE library package, which is open source software, written by Philip Hazel, and copyright 1997-2004 by the University of Cambridge, England. For details, see: <http://www.pcre.org/>

Writing Search Patterns

This section explains how to create search patterns using BBEdit’s grep syntax. For readers with prior experience, this is essentially like the syntax used for regular expressions in the Perl programming language. (However, you *do not* need to understand anything about Perl in order to make use of BBEdit’s grep searching.)

Most Characters Match Themselves

Most characters that you type into the Find window match themselves. For instance, if you are looking for the letter “t”, Grep stops and reports a match when it encounters a “t” in the text. This idea is so obvious that it seems not worth mentioning, but the important thing to remember is that these characters *are* search patterns. Very simple patterns, to be sure, but patterns nonetheless.

Escaping Special Characters

In addition to the simple character matching discussed above, there are various special characters that have different meanings when used in a grep pattern than in a normal search. (The use of these characters is covered in the following sections.)

However, sometimes you will need to include an exact, or literal, instance of these characters in your grep pattern. In this case, you must use the backslash character `\` before that special character to have it be treated literally; this is known as “escaping” the special character. To search for a backslash character itself, double it `\\` so that its first appearance will escape the second.

For example, perhaps the most common “special character” in grep is the dot: “.”. In grep, a dot character will match any character except a return. But what if you only want to match a literal dot? If you escape the dot: “\.”, it will only match another literal dot character in your text.

So, most characters match themselves, and even the special characters will match themselves if they are preceded by a backslash. BBEdit’s grep syntax coloring helps make this clear.

Note When passing grep patterns to BBEdit via AppleScript, be aware that both the backslash and double-quote characters have special meaning to AppleScript. In order to pass these through correctly, you must escape them in your script. Thus, to pass `\r` for a carriage return to BBEdit, you must write `\\r` in your AppleScript string.

Wildcards Match Types of Characters

These special characters, or metacharacters, are used to match certain types of other characters:

Wildcard Matches...

| | |
|----|----------------------------------------------------------------|
| . | any character except a line break (that is, a carriage return) |
| ^ | beginning of a line (unless used in a character class) |
| \$ | end of line (unless used in a character class) |

Being able to specifically match text starting at the beginning or end of a line is an especially handy feature of `grep`. For example, if you wanted to find every instance of a message sent by Patrick, from a log file which contains various other information like so:

```
From: Rich, server: barebones.com
To: BBEdit-Talk, server: lists.barebones.com
From: Patrick, server: example.barebones.com
```

you could search for the pattern:

```
^From: Patrick
```

and you will find every occurrence of these lines in your file (or set of files if you do a multi-file search instead).

It is important to note that `^` and `$` do not actually match return characters. They match zero-width *positions* after and before returns, respectively. So, if you are looking for “foo” at the end of a line, the pattern “foo\$” will match the three characters “f”, “o”, and “o”. If you search for “foo\r”, you will match the same text, but the match will contain four characters: “f”, “o”, “o”, and a return.

Note `^` and `$` do not match the positions after and before soft line breaks.

You can combine `^` and `$` within a pattern to force a match to constitute an entire line. For example:

```
^foo$
```

will only match “foo” on a line by itself, with no other characters. Try it against these three lines to see for yourself:

```
foobar
foo
fighting foo
```

The pattern will only match the second line.

Note In versions of BBEdit prior to 6.5, the `#` character was a wildcard that matched any digit (0–9); however, this is no longer the case. If you have `grep` patterns written with an old version of BBEdit that use `#` for this purpose, you will need to change these patterns. The easiest way to do this is to use the `\d` character class, which has the exact same meaning that `#` used to—it matches any character from 0–9. Character classes are explained in the next section.

Other Positional Assertions

BBEdit's grep engine supports additional positional assertions, very similar to `^` and `$`.

| Escape | Matches |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\A</code> | only at the beginning of the document (as opposed to <code>^</code> , which matches at the beginning of the document and also at the beginning of each line) |
| <code>\b</code> | any word boundary, defined as any position between a <code>\w</code> character and a <code>\W</code> character, in either order |
| <code>\B</code> | any position that is <i>not</i> a word boundary |
| <code>\z</code> | at the end of the document (as opposed to <code>\$</code> , which matches at the end of the document and also at the end of each line) |
| <code>\Z</code> | at the end of the document, or before a trailing return at the end of the doc, if there is one |

Examples (the text matched by the pattern is underlined)

Search for: `\bfoo\b`
Will match: bar foo bar
Will match: foo bar
Will not match: foobar

Search for: `\bJane\b`
Will match: Jane's
Will match: Tell Jane about the monkey.

Search for: `\Afoo`
Will match: foobar
Will not match: This is good foo.

Character Classes Match Sets or Ranges of Characters

The character class construct lets you specify a set or a range of characters to match, or to ignore. A character class is constructed by placing a pair of square brackets [...] around the group or range of characters you wish to include. To exclude, or ignore, all characters specified by a character class, add a caret character ^ just after the opening bracket [^...]. For example:

| Character Class | Matches |
|-----------------|-----------------------------------|
| [xyz] | any one of the characters x, y, z |
| [^xyz] | any character except x, y, z |
| [a-z] | any character in the range a to z |

You can use any number of characters or ranges between the brackets. Here are some examples:

| Character Class | Matches |
|-----------------|---------------------------------------------------|
| [aeiou] | any vowel |
| [^aeiou] | any character that is not a vowel |
| [a-zA-Z0-9] | any character from a-z, A-Z, or 0-9 |
| [^aeiou0-9] | any character that is neither a vowel nor a digit |

A character class matches when the search encounters any *one* of the characters in the pattern. However, the contents of a set are only treated as separate characters, not as words. For example, if your search pattern is [beans] and the text in the window is “lima beans”, BBEdit will report a match at the “a” of the word “lima”.

To include the character] in a set or a range, place it immediately after the opening bracket. To use the ^ character, place it anywhere except immediately after the opening bracket. To match a dash character (hyphen) in a range, place it at the beginning of the range; to match it as part of a set, place it at the beginning or end of the set. Or, you can include any of these character at any point in the class by escaping them with a backslash.

| Character Class | Matches |
|-----------------|-----------------|
| []0-9] | any digit or] |
| [aeiou^] | a vowel or ^ |
| [-A-Z] | a dash or A - Z |

| Character Class | Matches |
|-----------------|----------------------------------------|
| [--A] | any character in the range from - to A |
| [aeiou-] | any vowel or - |
| [aei\ -ou] | any vowel or - |

Character classes respect the setting of the Case Sensitive checkbox in the Find window. For example, if Case Sensitive is on, [a] will only match “a”; if Case Sensitive is off, [a] will match both “a” and “A”.

Matching Non-Printing Characters

As described in Chapter 7 on searching, BBEdit provides several special character pairs that you can use to match common non-printing characters, as well as the ability to specify any arbitrary character by means of its hexadecimal character code (escape code). You can use these special characters in grep patterns as well as for normal searching.

For example, to look for a tab or a space, you would use the character class [\t] (consisting of a tab special character and a space character).

| Character | Matches |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------|
| \r | line break (carriage return) |
| \n | Unix line break (line feed) |
| \t | tab |
| \f | page break (form feed) |
| \a | alarm (hex 07) |
| \cX | a named control character, like \cC for Control-C |
| \b | backspace (hex 08) (<i>only in character classes</i>) |
| \e | Esc (hex 1B) |
| \xNN | hexadecimal character code NN (for example, \x0D for CR) |
| \x{NNNN} | any number of hexadecimal characters NN... (for example, \x{0} will match a null, \x{304F} will match a Japanese Unicode character) |
| \\ | backslash |

Use \r to match a line break in the middle of a pattern and the special characters ^ and \$ (described above) to “anchor” a pattern to the beginning of a line or to the end of a line. In the case of ^ and \$, the line break character is not included in the match.

Other Special Character Classes

BBEdit uses several other sequences for matching different types or categories of characters.

| Special Character | Matches |
|-------------------|------------------------------------------------------------------------------------------------------|
| <code>\s</code> | any whitespace character (space, tab, carriage return, line feed, form feed) |
| <code>\S</code> | any non-whitespace character (any character not included by <code>\s</code>) |
| <code>\w</code> | any word character (a-z, A-Z, 0-9, <code>_</code> , and some 8-bit characters) |
| <code>\W</code> | any non-word character (all characters not included by <code>\w</code> , including carriage returns) |
| <code>\d</code> | any digit (0-9) |
| <code>\D</code> | any non-digit character (including carriage return) |

A “word” is defined in BBEdit as any run of non-word-break characters bounded by word breaks. Word characters are generally alphanumeric, and some characters whose value is greater than 127 are also considered word characters.

Note that any character matched by `\s` is by definition not a word character; thus, anything matched by `\s` will also be matched by `\W` (but not the reverse!).

Quantifiers Repeat Subpatterns

The special characters `*`, `+`, and `?` specify *how many times* the pattern preceding them may repeat. `{}`-style quantifiers allow you to specify exactly how many times a subpattern can repeat. The preceding pattern can be a literal character, a wildcard character, a character class, or a special character.

| Pattern | Matches |
|--------------------------|--------------------------------------------------------------------------------|
| <code>p*</code> | zero or more <code>p</code> 's |
| <code>p+</code> | one or more <code>p</code> 's |
| <code>p?</code> | zero or one <code>p</code> 's |
| <code>p{COUNT}</code> | match exactly <i>COUNT</i> <code>p</code> 's, where <i>COUNT</i> is an integer |
| <code>p{MIN, }</code> | match at least <i>MIN</i> <code>p</code> 's, where <i>MIN</i> is an integer |
| <code>p{MIN, MAX}</code> | match at least <i>MIN</i> <code>p</code> 's, but no more than <i>MAX</i> |

Note that the repetition characters `*` and `?` match *zero or more* occurrences of the pattern. That means that they will *always* succeed, because there will always be at least zero occurrences of any pattern, but that they will not necessarily select any text (if no occurrences of the preceding pattern are present).

For this reason, when you are trying to match more than one occurrence, it is usually better to use a `+` than a `*`, because `+` *requires* a match, whereas `*` can match the empty string. Only use `*` when you are sure that you really mean “zero or more times,” not just “more than once.”

Try the following examples to see how their behavior matches what you expect:

| Pattern | Text | Matches |
|----------------------|--------------------------------------------|-------------------------------------------------------------|
| <code>.*</code> | Fourscore and seven years | Fourscore and seven years |
| <code>[0-9]+</code> | I've been a loyal member since 1983 or so. | 1983 |
| <code>\d+</code> | I've got 12 years on him. | 12 |
| <code>A+</code> | BAAAAAAB | AAAAAA |
| <code>A{3}</code> | BAAAAB | AAA (first three A's) |
| <code>A{3, }</code> | BAAAAB | AAAA |
| <code>A{1, 3}</code> | BAAAAB | AAA on the first match, the remaining A on the second match |
| <code>c?andy</code> | andy likes candy | "andy" on the first match, "candy" on the second |

| Pattern | Text | Matches |
|---------|--------------------------|-----------------------------------------------------------------|
| A+ | Ted joined AAA yesterday | "AAA" on the first match; "a" from "yesterday" on the second |

Combining Patterns to Make Complex Patterns

So far, the patterns you have seen match a single character or the repetition of a single character or class of characters. This is very useful when you are looking for runs of digits or single letters, but often that is not enough.

However, by combining these patterns, you can search for more complex items. As it happens, you are already familiar with combining patterns. Remember the section at beginning of this discussion that said that each individual character is a pattern that matches itself? When you search for a word, you are already combining basic patterns.

You can combine any of the preceding grep patterns in the same way. Here are some examples.

| Pattern | Matches | Examples |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| <code>\d+\+\d+</code> | a string of digits, followed by a literal plus sign, followed by more digits | 4+2 1234+5829 |
| <code>\d{4} [\t]B\.C\.</code> | four digits, followed by a tab or a space, followed by the string B.C. | 2152 B.C. |
| <code>\\$?[0-9,]+\.\d*</code> | an optional dollar sign, followed by one or more digits and commas, followed by a period, then zero or more digits | 1,234.56 \$4,296,459.1 9 \$3,5,6,4.0000 0. (<i>oops!</i>) |

Note again in these examples how the characters that have special meaning to grep are preceded by a backslash (`\+`, `\.`, and `\$`) when we want them to match themselves.

Creating Subpatterns

Subpatterns provide a means of organizing or grouping complex grep patterns. This is primarily important for two reasons: for limiting the scope of the alternation operator (which otherwise creates an alternation of everything to its left and right), and for changing the matched text when performing replacements.

A subpattern consists of any simple or complex pattern, enclosed in a pair of parentheses. You can optionally specify a simple string to identify a subpattern, making it a named subpattern.

| Pattern | Matches |
|------------------|---------------------------------------|
| <code>(p)</code> | the pattern <i>p</i> and remembers it |

| Pattern | Matches |
|--------------------------------|---------------------------------------------------------------------------|
| <code>(?P<NAME>p)</code> | the pattern <i>p</i> and remembers it by the specified string <i>NAME</i> |

You can combine more than one subpattern into a grep pattern, or mix subpatterns and other pattern elements as you need.

Taking the last set of examples, you could modify these to use subpatterns wherever actual data appears:

| Pattern | Matches | Examples |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| <code>(\d+)\+(\d+)</code> | a string of digits, followed by a plus sign, followed by more digits | 4+2 1234+5829 |
| <code>(\d{4})[\t]B\C\.</code> | four digits, followed by a tab or a space, followed by the string B.C. | 2152 B.C. |
| <code>\\$(?([0-9,]+)\.(\d*))</code> | an optional dollar sign, followed by one or more digits and commas, followed by a period, then zero or more digits | 1,234.56 \$4,296,459.1 9 \$3,5,6,4.0000 0. |

Using Backreferences in Subpatterns

What if we wanted to match a series of digits, followed by a plus sign, followed by the exact same series of digits as on the left side of the plus? In other words, we want to match “1234+1234” or “7+7”, but *not* “5432+1984”.

Using grouping parentheses, you can do this by referring to a backreference, also known as a captured subpattern. There are two kinds of backreferences: numbered backreferences, and named backreferences. You can use both types of backreference within the same grep pattern.

Each subpattern within the complete pattern is numbered from left to right, starting with the opening parenthesis. Later in the pattern, you can refer to the text matched within any of these subpatterns by using a backslash followed by the number of that subpattern; this is a numbered backreference. Unlike numbered backreferences, which are automatically identified from the pattern, named backreferences are only available after you define them.

| Pattern | Matches... |
|-------------------------------|---------------------------------------------------------------------|
| <code>\1, \2, ..., \99</code> | the text of the <i>n</i> th subpattern in the entire search pattern |
| <code>(?P=NAME)</code> | the text of the subpattern <i>NAME</i> |

Names may include alphanumeric characters and underscores, and must be unique within a pattern.

Here are some examples of numbered backreferences:

| Pattern | Matches | Examples |
|---------------------------|-----------------------------------------------------------------------------------------------------|-----------------------------------|
| <code>(\d+)\+\1</code> | a string of digits, followed by a plus sign, followed the same digits | 7+7 1234+1234 |
| <code>(\w+)\s+\1</code> | double words, or, a pair of identical character runs separated by whitespace | the the tire return (oops!) |
| <code>(\w)(\w)\2\1</code> | a word character, a second word character, followed by the second one again and the first one again | abba |

We will revisit subpatterns in the section on replacement, where you will see how the choice of subpatterns affects the changes you can make.

Using Alternation

The alternation operator `|` allows you to match any of several patterns at a given point. To use this operator, place it between one or more patterns `x|y` to match either `x` or `y`.

As with all of the preceding options, you can combine alternation with other pattern elements to handle more complex searches.

| Pattern | Text is... | Matches... |
|-----------------------------|-----------------------------------------------------------|--------------------------------------|
| <code>a t</code> | A cat | each "a" and "t" |
| <code>a c t</code> | A cat | each "a", "c", and "t" |
| <code>a (cat dog) is</code> | A cat is here. A dog is here. A giraffe is here. | "A cat is", "A dog is" |
| <code>A b+</code> | Abba | "A", "bb", and "a" |
| <code>Andy Ted</code> | Andy and Ted joined AAA yesterday | "Andy" and "Ted" |
| <code>\d{4} years</code> | I've been a loyal member since 1983, almost 16 years ago. | "1983", "years" |
| <code>[a-z]+ \d+</code> | That's almost 16 years. | "That", "s", "almost", "16", "years" |

The "Longest Match" Issue

IMPORTANT

When creating complex patterns, you should bear in mind that the quantifiers `+`, `*`, `?` and `{}` are "greedy." That is, they will always make the longest possible match possible to a given pattern, so if your pattern is `E+` (one or more `E`'s) and your text contains "EEEE", the pattern matches all the `E`'s at once, not just the first one. This is usually what you want, but not always.

Suppose, for instance, that you want to match an HTML tag. At first, you may think that a good way to do this would be to search for the pattern:

```
<.+>
```

consisting of a less-than sign, followed by one or more occurrences of a single character, followed by a greater-than sign. To understand why this may not work the way you think it should, consider the following sample text to be searched:

```
<B>This text is in boldface.</B>
```

The intent was to write a pattern that would match both of the HTML tags separately. Let's see what actually happens. The < character at the beginning of this line matches the beginning of the pattern. The next character in the pattern is . which matches any character (except a line break), modified with the + quantifier, taken together, this combination means one or more repetitions of any character. That, of course, takes care of the B. The problem is that the next > is also "any character" and that it *also* qualifies as "one or more repetitions." In fact, all of the text up to the end of the line qualifies as "one or more repetitions of any character" (the line break does not qualify, so grep stops there). After grep has reached the line break, it has exhausted the + operator, so it backs up and sees if it can find a match for >. Lo and behold, it can: the last character is a greater-than symbol. Success!

In other words, the pattern matches our entire sample line at once, *not the two separate HTML tags in it as we intended*. More generally, the pattern matches all the text in a given line or paragraph from the first < to the *last* >. The pattern only does what we intended when there is only one HTML tag in a line or paragraph. This is what we meant when we say that the regular quantifiers try to make the longest possible match.

Non-Greedy Quantifiers

IMPORTANT

To work around this "longest match" behavior, you can modify your pattern to take advantage of *non-greedy* quantifiers.

| Quantifier | Matches... |
|-------------|------------------------------------------------|
| +? | one or more |
| *? | zero or more |
| ?? | zero or one |
| {COUNT}? | match exactly COUNT times |
| {MIN, }? | match at least MIN times |
| {MIN, MAX}? | match at least MIN times, but no more than MAX |

Astute readers will note that these non-greedy quantifiers correspond exactly to their normal (greedy) counterparts, appended with a question mark.

Revisiting our problem of matching HTML tags, for example, we can search for:

```
<.+?>
```

This matches an opening bracket, followed by one or more occurrences of any character other than a return, followed by a closing bracket. The non-greedy quantifier achieves the results we want, preventing BBEdit from “overrunning” the closing angle bracket and matching across several tags.

A slightly more complicated example: how could you write a pattern that matches all text between `` and `` HTML tags? Consider the sample text below:

```
<B>Welcome</B> to the home of <B>BBEdit!</B>
```

As before, you might be tempted to write:

```
<B>.*</B>
```

but for the same reasons as before, this will match the entire line of text. The solution is similar; we will use the non-greedy `*?` quantifier:

```
<B>.*?</B>
```

Writing Replacement Patterns

Subpatterns Make Replacement Powerful

We covered subpatterns earlier when discussing search patterns and discussed how the parentheses can be used to limit the scope of the alternation operator. Another reason for employing subpatterns in your grep searches is to provide a powerful and flexible way to change or reuse found information as part of a search-and-replace operation. If you do not use subpatterns, you can still access the complete results of the search with the `&` metacharacter. However, this precludes reorganizing the matched data as it is replaced.

| Pattern | Inserts... |
|-------------------------------|---------------------------------------------------------------------|
| <code>&</code> | the text matched by the entire search pattern |
| <code>\1, \2, ..., \99</code> | the text matched by the nth subpattern of the entire search pattern |
| <code>\P<NAME></code> | the text matched by the subpattern <i>NAME</i> |

Note BBEdit will remember up to 99 backreferenced subpatterns. Versions prior to 6.5 were limited to 9 subpatterns.

Using the Entire Matched Pattern

The `&` character is useful when you want to use the entire matched string as the basis of a replacement. Suppose that in your text every instance of product names that begin with the company name “ACME” needs to end with a trademark symbol (™). The following search pattern finds two-word combinations that begin with “ACME”:

```
ACME [A-Za-z]+
```

The following replacement string adds the trademark symbol to the matched text:

```
&™
```

For example, if you start with

```
ACME Magnets, ACME Anvils, and ACME TNT are all premium
products.
```

and perform a replace operation with the above patterns, you will get:

```
ACME Magnets™, ACME Anvils™, and ACME TNT™ are all premium
products.
```

Using Parts of the Matched Pattern

While using the entire matched pattern in a replacement string is useful, it is often more useful to use only a portion of the matched pattern and to rearrange the parts in the replacement string.

For example, suppose a source file contains C-style declarations of this type:

```
#define Util_Menu 284
#define Tool_Menu 295
```

and you want to convert them so they look like this, Pascal-style:

```
const int Util_Menu = 284;
const int Tool_Menu = 295;
```

The pattern to find the original text is straightforward:

```
#define [ \t]+.+ [ \t]+\d+ [^0-9]*$
```

This pattern matches the word “#define” followed by one or more tabs or spaces, followed by one or more characters of any type, followed by one or more tabs or spaces, followed by one or more digits, followed by zero or more characters that are *not* digits (to allow for comments), followed by the end of the line.

The problem with this pattern is that it matches the entire line. It does not provide a way to remember the individual parts of the found string.

If you use subpatterns to rewrite the above search pattern slightly, you get this:

```
#define [ \t]+(.+) [ \t]+(\d+) [^0-9]*$
```

The first set of parentheses defines a subpattern which remembers the name of the constant. The second set remembers the value of the constant.

The replacement string would look like this:

```
const int \1 = \2;
```

The sequence `\1` is replaced by the name of the constant (the first subpattern from the search pattern), and the sequence `\2` is replaced by the value of the constant (from the second subpattern).

Our example throws out any comment that may follow the C-style constant declaration. As an exercise, try rewriting the search and replace patterns so they preserve the comment, enclosing it in `(*...*)` style Pascal comment markers.

Here are some more examples:

| Data | Search for | Replace | Result |
|-----------------|-----------------------------------|------------------------------------------|---------------------------------------|
| 4+2 | <code>(\d+)\+(\d+)</code> | <code>\2+\1</code> | 2+4 |
| 1234+5829 | <code>(\d+)\+(\d+)</code> | <code>\1+\1</code> | 1234+1234 |
| 2152 B.C. | <code>(\d{4})[\t]B\.C\.</code> | <code>\1 A.D.</code> | 2152 A.D. |
| 1,234.56 | <code>\\$?([0-9,]+)\.(\d+)</code> | <code>\1 dollars and \2 cents</code> | 1,234 dollars and 56 cents |
| \$4,296,459.19 | <code>\\$?([0-9,]+)\.(\d+)</code> | <code>\1 dollars and \2 cents</code> | 4,296,459 dollars and 19 cents |
| \$3,5,6,4.00000 | <code>\\$?([0-9,]+)\.(\d+)</code> | <code>\1 dollars and \2 cents</code> | 3,5,6,4 dollars and 00000 cents |

Case Transformations

Replace patterns can also change the case of the original text when using subpattern replacements. The syntax is similar to Perl's, specifically:

| Modifier | Effect |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\u</code> | Make the next character uppercase |
| <code>\U</code> | Make all following characters uppercase until reaching another case specifier (<code>\u</code> , <code>\L</code> , <code>\l</code>) or <code>\E</code> |
| <code>\l</code> | Make the next character lowercase |
| <code>\L</code> | Make all following characters lowercase until reaching another case specifier (<code>\u</code> , <code>\U</code> , <code>\l</code>) or <code>\E</code> |
| <code>\E</code> | End case transformation opened by <code>\U</code> or <code>\L</code> |

Here are some examples to illustrate how case transformations can be used.

Given some text:

```
mumbo-jumbo
```

and the search pattern:

```
(\w+) (\W) (\w+)
```

the following replace patterns will produce the following output:

```
\U\1\E\2\3    MUMBO-jumbo
\u\1\2\u\3    Mumbo-Jumbo
```

Note that case transformations also affect literal strings in the replace pattern:

```
\U\1\2fred    MUMBO-FRED
\lMUMBLE\2\3  mUMBLE-jumbo
```

Finally, note that `\E` is not necessary to close off a modifier; if another modifier appears before an `\E` is encountered, that modifier will take effect immediately:

```
\Ufred-\uwilma          FRED-Wilma
```

Examples

The example patterns in this section describe some common character classes and shortcuts used for constructing `grep` patterns, and addresses some common tasks that you might find useful in your work.

Matching Identifiers

One of the most common things you will use `grep` patterns for is to find and modify identifiers, such as variables in computer source code or object names in HTML source documents. To match an arbitrary identifier in most programming languages, you might use this search pattern:

```
[a-z][a-zA-Z0-9]*
```

This pattern matches any sequence that begins with a lowercase letter and is followed by zero or more alphanumeric characters. If other characters are allowed in the identifier, add them to the pattern. This pattern allows underscores in only the first character of the identifier:

```
[a-z_][a-zA-Z0-9]*
```

The following pattern allows underscores anywhere *but* the first character, but allows identifiers to begin with an uppercase or lowercase letter:

```
[a-zA-Z][a-zA-Z0-9_]*
```

Matching White Space

Often you will want to match two sequences of data that are separated by tabs or spaces, whether to simply identify them, or to rearrange them.

For example, suppose you have a list of formatted label-data pairs like this:

```
User name:          Bernard Rubble
Occupation:         Actor
Spouse:             Betty
```

You can see that there are tabs or spaces between the labels on the left and the data on the right, but you have no way of knowing how many spaces or tabs there will be on any given line. Here is a character class that means “match one or more white space characters.”

```
[ \t]+
```

So, if you wanted to transform the list above to look like this:

```
User name("Bernard Rubble")
Occupation("Actor")
Spouse("Betty")
```

You would use this search pattern:

```
([a-z ]+):[ \t]+([a-z ]+)
```

and this replacement pattern:

```
\1\("\2"\)
```

Matching Delimited Strings

In some cases, you may want to match all the text that appears between a pair of delimiters. One way to do this is to bracket the search pattern with the delimiters, like this:

```
".*"
```

This works well if you have only one delimited string on the line. But suppose the line looked like this:

```
"apples", "oranges, kiwis, mangos", "penguins"
```

The search string above would match the entire line. (This is another instance of the “longest match” behavior of BBEdit’s grep engine, which was discussed previously.)

Once again, non-greedy quantifiers come to the rescue. The following pattern will match “-delimited strings:

```
".+?"
```

Marking Structured Text

Suppose you are reading a long text document that does not have a table of contents, but you notice that all the sections are numbered like this:

```
3.2.7      Prehistoric Cartoon Communities
5.19.001   Restaurants of the Mesozoic
```

You can use a grep pattern to create marks for these headings, which will appear in the Mark popup menu. Choose Find & Mark All from the Mark popup menu in the navigation bar. Then, decide how many levels you want to mark. In this example, the headings always have at least two digits and at most four.

Use this pattern to find the headings:

```
^(\\d+\\.\\d+\\.?.?\\d*\\.?.?\\d*) [ \t]+ ([a-z ]+)
```

and this pattern to make the file marks:

```
\\1 \\2
```

The ^ before the first search group ensures that BBEdit matches the numeric string at the beginning of a line. The pattern

```
\\.?.?\\d*
```

matches a (possible) decimal point and a digit sequence. The other groups use the white space idiom and the identifier idiom. You can use a similar technique to mark any section that has a section mark that can be described with grep.

Marking a Mail Digest

You can elaborate the structured text technique to create markers for mail digests. Assume that each digest is separated by the following lines:

```
From: Sadie Burke <sadie@burke.com>
Date: Sun, 16 Jul 1995 13:17:45 -0700
Subject: Fishing with the judge
```

Suppose you want the marker text to list the subject and the sender. You would use the following search string:

```
^From: [ \t]+(.*)\r.*\rSubject: [ \t]+(.*)
```

And mark the text with this replacement string:

```
\2 \1
```

Note that for the sequence `\r.*\r` in the middle of the search string, the `\r` before “Subject” is necessary because as previously discussed, the special character `.` does not match carriage returns. (At least, not by default. See “Advanced Topics,” below, for details on how to make dot match *any* character, including carriage returns.)

Rearranging Name Lists

You can use `grep` patterns to transform a list of names in first name first form to last name first order (for a later sorting, for instance). Assume that the names are in the form:

```
Junior X. Potter
Jill Safai
Dylan Schuyler Goode
Walter Wang
```

If you use this search pattern:

```
^(.*) ([^ ]+)$
```

And this replacement string:

```
\2, \1
```

The transformed list becomes:

```
Potter, Junior X.
Safai, Jill
Goode, Dylan Schuyler
Wang, Walter
```

Advanced Grep Topics

BBEdit’s new PCRE-based `grep` engine offers unparalleled syntactical power. The topics below cover areas that show how `grep` can effectively match very complicated patterns of text—matches which were impossible to achieve with older versions of BBEdit. However, with this power comes complexity.

If you are new to `grep`, it is possible that the topics covered in this section will not make much sense to you. That's OK. The best way to learn `grep` is to use it in real life, not by reading example patterns. In many cases, the basic `grep` syntax covered previously in this chapter will be all that you need.

If you are an experienced user of `grep`, however, many of the topics covered below will be of great interest.

Matching Nulls

The `grep` engine used in versions of BBEdit prior to 6.5 was unable to search text that contained null characters (ASCII value zero), but this limitation has since been removed. Here's one way to match a null:

```
\x{0}
```

Backreferences

The following charts explain the rules BBEdit uses for determining backreferences.

In Search Patterns

| Modifier | Effect |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\0</code> | A backslash followed by a zero is an octal character reference. Up to two further octal characters are read. Thus, <code>"\040"</code> will match a space character, and <code>"\07"</code> will match the ASCII BEL (<code>\x07</code>), but <code>"\08"</code> will match an ASCII null followed by the digit 8 (because octal characters only range from 0-7). |
| <code>\1-9</code> | A backslash followed by a single decimal digit from 1 to 9 is always a backreference to the <i>N</i> th captured subpattern. |
| <code>\10-99</code> | A backslash followed by two decimal digits, which taken together form the integer <i>N</i> (ranging from 10 to 99), is a backreference to the <i>N</i> th captured subpattern, <i>if</i> there exist <i>N</i> capturing sets of parentheses in the pattern. If there are fewer than <i>N</i> captured subpatterns, the <code>grep</code> engine will instead look for up to three octal digits following the backslash. Any subsequent digits stand for themselves. So, in a search pattern, <code>"\11"</code> is a backreference if there are 11 or more sets of capturing parentheses in the pattern. If not, it matches a tab. <code>"\011"</code> always matches a tab. <code>"\81"</code> is a backreference if there are 81 or more captured subpatterns, but matches an ASCII null followed by the two characters "8" and "1" otherwise. |

In Character Classes

| Modifier | Effect |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\OCTAL</code> | Inside a character class, a backslash followed by up to three octal digits generates a single byte character reference from the least significant eight bits of the value. Thus, the character class <code>"[\7]"</code> will match a single byte with octal value 7 (equivalent to <code>"\x07"</code>). <code>"[\8]"</code> will match a literal "8" character. |

In Replacement Patterns

| Modifier | Effect |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\WWW+</code> | If more than two decimal digits follow the backslash, only the first two are considered part of the backreference. Thus, <code>"\111"</code> would be interpreted as the 11th backreference, followed by a literal "1". You may use a leading zero; for example, if in your replacement pattern you want the first backreference followed by a literal "1", you can use <code>"\011"</code> . (If you use <code>"\11"</code> , you will get the 11th backreference, even if it is empty.) |
| <code>\WN</code> | If two decimal digits follow the backslash, which taken together represent the value N, and if there is an Nth captured substring, then all three characters are replaced with that substring. If there is not an Nth captured substring, all three characters are discarded—that is, the backreference is replaced with the empty string. |
| <code>\W</code> | If there is only a single digit N following the backslash and there is an Nth captured substring, both characters are replaced with that substring. Otherwise, both characters are discarded—that is, the backreference is replaced with the empty string. In replacement patterns, <code>\0</code> is a backreference to the entire match (exactly equivalent to <code>"&"</code>). |

POSIX-Style Character Classes

BBEdit now provides support for POSIX-style character classes. These classes are used in the form `[:CLASS:]`, and are *only* available inside regular character classes (in other words, inside another set of square brackets).

| Class | Meaning |
|--------------------|-------------------------------------------|
| <code>alnum</code> | letters and digits |
| <code>alpha</code> | letters |
| <code>ascii</code> | character codes 0-127 |
| <code>blank</code> | horizontal whitespace |
| <code>cntrl</code> | control characters |
| <code>digit</code> | decimal digits (same as <code>\d</code>) |
| <code>graph</code> | printing characters, excluding spaces |

| Class | Meaning |
|--------|---------------------------------------|
| lower | lower case letters |
| print | printing characters, including spaces |
| punct | punctuation characters |
| space | white space (same as \s) |
| upper | upper case letters |
| word | “word” characters (same as \w) |
| xdigit | hexadecimal digits |

For example: `[:digit:]+` is the same as: `[d]+`

POSIX-style character class names are case-sensitive.

It is easy to forget that POSIX-style character classes are only available inside regular character classes. The pattern `[:space:]`, without enclosing square brackets, is just a character class consisting of the characters “:”, “a”, “c”, “e”, “p”, and “s”.

The names “ascii” and “word” are Perl extensions; the others are defined by the POSIX standard. Another Perl extension supported by BBEdit is negated POSIX-style character classes, which are indicated by a `^` after the colon. For example, to match any run of non-digit characters:

```
[[:^digit:]]+
```

Non-Capturing Parentheses

As described in the preceding section “Creating Subpatterns”, bare parentheses cluster and capture the subpatterns they contain. The portion of the matching pattern contained within the first pair of parentheses is available in the backreference `\1`, the second in `\2`, and so on.

Opening parentheses are counted from left to right to determine the numbers of the captured subpatterns. For example, if the following grep pattern:

```
((red|white) (king|queen))
```

is matched against the text “red king”, the backreferences will be set as follows:

```
\1 "red king"
\2 "red"
\3 "king"
```

Sometimes, however, parentheses are needed only for clustering, not capturing. BBEdit now supports non-capturing parentheses, using the syntax:

```
(?: PATTERN)
```

That is, if an open parenthesis is followed by “?:”, the subpattern matched by that pair of parentheses is not counted when computing the backreferences. For example, if the text “red king” is matched against the pattern:

```
(?:(red|white) (king|queen))
```

the backreferences will be set as follows:

```
\1 "red"  
\2 "king"
```

Perl-Style Pattern Extensions

BBEdit's grep engine supports several extended sequences, which provide grep patterns with super-powers from another universe. Their syntax is in the form:

```
(?KEY...)
```

in other words, an open parenthesis followed by a question mark, followed by a KEY for the particular grep extension, followed by the rest of the subpattern and a closing parenthesis. This syntax—specifically, an open parenthesis followed by a question mark—was not valid in older versions of BBEdit, thus, none of these extensions will conflict with old patterns.

We have already seen one such extension in the previous section of this document—non-capturing parentheses: (?:...). The remainder are listed in the chart below, and discussed in detail afterward.

| Extension | Meaning |
|------------------|---------------------------------------------------|
| (?:...) | Cluster-only parentheses, no capturing |
| (?#...) | Comment, discard all text between the parentheses |
| (?imsx-imsx) | Enable/disable pattern modifiers |
| (?imsx-imsx:...) | Cluster-only parens with modifiers |
| (?=...) | Positive lookahead assertion |
| (?!...) | Negative lookahead assertion |
| (?<=...) | Positive lookbehind assertion |
| (?<!...) | Negative lookbehind assertion |
| (?()... ...) | Match with if-then-else |
| (?()...) | Match with if-then |
| (?>...) | Match non-backtracking subpattern ("once-only") |
| (?R) | Recursive pattern |

Comments

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

Search for: `foo(?# Hello, this is a comment)bar`

Will match: foobar

Pattern Modifiers

The settings for case sensitivity, multi-line matching, whether the dot character can match returns, and “extended syntax” can be turned on and off within a pattern by including sequences of letters between “(?” and “)”.

| Modifier | Meaning | Default |
|----------------|----------------------------------------------------------------------|-----------------------------------------------------|
| <code>i</code> | case insensitive | according to Case Sensitive checkbox in Find window |
| <code>m</code> | allow <code>^</code> and <code>\$</code> to match at <code>\r</code> | on |
| <code>s</code> | allow <code>.</code> to match <code>\r</code> | off |
| <code>x</code> | ignore most white space and allow inline comments in grep patterns | off |

i — By default, BBEdit obeys the “Case Sensitive” checkbox in the Find window (or the corresponding property of the search options when using the scripting interface). The `(?i)` option overrides this setting.

m — By default, BBEdit’s grep engine will match the `^` and `$` metacharacters after and before returns, respectively. If you turn this option off with `(?-m)`, `^` will only match at the beginning of the document, and `$` will only match at the end of the document. (If that is what you want, however, you should consider using the new `\A`, `\Z`, and `\z` metacharacters instead of `^` and `$`.)

s — By default, the magic dot metacharacter `.` matches any character except return (“`\r`”). If you turn this option on with `(?s)`, however, dot will match *any* character. Thus, the pattern `(?s)+` will match an entire document.

x — When turned on, this option changes the meaning of most whitespace characters (notably, tabs and spaces) and `#`. Literal whitespace characters are ignored, and the `#` character starts a comment that extends until a literal return or the “`\r`” escape sequence is encountered. Ostensibly, this option intends to let you write more “readable” patterns.

Perl programmers should already be familiar with these options, as they correspond directly to the `-imsx` options for Perl’s `m//` and `s///` operators. Unadorned, these options turn their corresponding behavior on; when preceded by a hyphen (`-`), they turn the behavior off. Setting and unsetting options can occur in the same set of parentheses.

| Example | Effect |
|-----------------------|-----------------------------------------|
| <code>(?imsx)</code> | Turn all four options on |
| <code>(?-imsx)</code> | Turn all four options off |
| <code>(?i-msx)</code> | Turn “i” on, turn “m”, “s”, and “x” off |

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern, the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

```
(?i) abc
a (?i) bc
ab (?i) c
abc (?i)
```

In other words, all four of the above patterns will match without regard to case. Such “top level” settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at the top level, the right-most setting is used.

If an option change occurs inside a subpattern, the effect is different. An option change inside a subpattern affects *only* that part of the subpattern that follows it, so, if the “Case Sensitive” checkbox is turned on:

```
Search for: (a (?i) b) c
Will match: abc or aBC
```

and will not match anything else. (But if “Case Sensitive” is turned off, the “(?i)” in the above pattern is superfluous and has no effect.) By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example:

```
Search for: (a (?i) b | c)
```

matches “ab”, “aB”, “c”, and “C”, even though when matching “C”, the first branch is abandoned before the option setting.

These options can also be set using the clustering (non-capturing) parentheses syntax defined earlier, by inserting the option letters between the “?” and “:”. The scope of options set in this manner is limited to the subpattern contained therein. Examples:

```
Search for: (?i:saturday|sunday)
Will match: SATURDAY or Saturday or SUNday (and so on)

Search for: (?i:foo) (?-i:bar)
Will match: foobar or FOObar
Will not match: FOOBAR or fooBAR
```

Positional Assertions

Positional assertions “anchor” a pattern, without actually matching any characters. Simple assertions have already been described: those which are invoked with the escape sequences `\b`, `\B`, `\A`, `\Z`, `\z`, `^` and `$`. For example, the pattern `\bfoo\b` will only match the string “foo” if it has word breaks on both sides, but the `\b`’s do not themselves match any characters; the entire text matched by this pattern are the three characters “f”, “o”, and “o”.

Lookahead and lookbehind assertions work in a similar manner, but allow you to test for arbitrary patterns to anchor next to. If you have ever said to yourself, “I would like to match ‘foo’, but only when it is next to ‘bar’,” lookaround assertions fill that need.

Positive lookahead assertions begin with “(?=”, and negative lookahead assertions begin with “(?!)”. For example:

```
\w+(?=;)
```

will match any word followed by a semicolon, but the semicolon is not included as part of the match.

```
foo(?!bar)
```

matches any occurrence of “foo” that is *not* followed by “bar”. Note that the apparently similar pattern:

```
(?!foo)bar
```

does not find an occurrence of “bar” that is preceded by something other than “foo”; it finds any occurrence of “bar” whatsoever, because the assertion (?!foo) is always true when the next three characters are “bar”. A lookbehind assertion is needed to achieve this effect.

Positive lookbehind assertions start with “(?<=”, and negative lookbehind assertions start with “(?<!”. For example:

```
(?<!foo)bar
```

does find an occurrence of “bar” that is not preceded by “foo”. The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=Martin|Lewis)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is different compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. (Lookbehinds in conjunction with non-backtracking [a.k.a. “once-only”] subpatterns can be particularly useful for matching at the ends of strings; an example is given in the section on once-only subpatterns below.)

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?!999)foo
```

matches “foo” preceded by three digits that are not “999”. Notice that each of the assertions is applied independently at the same point in the subject string. First there is a check that the previous three characters are all digits, and then there is a check that the same three characters are not “999”. This pattern does not match “foo” preceded by six characters, the first of which are digits and the last three of which are not “999”. For example, it does not match “123abcfoo”. A pattern to do that is:

```
(?<=\d{3} . . . ) (?<!999) foo
```

This time the first assertion looks at the preceding six characters, checking that the first three are digits, and then the second assertion checks that the preceding three characters are not “999”. Assertions can be nested in any combination. For example,

```
(?<=(?!foo) bar) baz
```

matches an occurrence of “baz” that is preceded by “bar” which in turn is not preceded by “foo”, while

```
(?<=\d{3} (?!999) . . . ) foo
```

is another pattern which matches “foo” preceded by three digits and any three characters that are not “999”.

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If any kind of assertion contains capturing subpatterns within it, these are counted for the purposes of numbering the capturing subpatterns in the whole pattern. However, substring capturing is carried out only for positive assertions, because it does not make sense for negative assertions.

Conditional Subpatterns

Conditional subpatterns allow you to apply “if-then” or “if-then-else” logic to pattern matching. The “if” portion can either be an integer between 1 and 99, or an assertion.

The forms of syntax for an ordinary conditional subpattern are:

```
if-then:          (? (condition) yes-pattern)
if-then-else:   (? (condition) yes-pattern | no-pattern)
```

and for a named conditional subpattern are:

```
if-then:          (?P<NAME> (condition) yes-pattern)
if-then-else:   (?P<NAME> (condition) yes-pattern | no-pattern)
```

If the condition evaluates as true, the “yes-pattern” portion attempts to match. Otherwise, the “no-pattern” portion does (if there is a “no-pattern”).

If the “condition” text between the parentheses is an integer, it corresponds to the backreferenced subpattern with the same number. (Do not precede the number with a backslash.) If the corresponding backreference has previously matched in the pattern, the condition is satisfied. Here’s an example of how this can be used. Let’s say we want to match the words “red” or “blue”, and refer to whichever word is matched in the replacement pattern. That’s easy:

```
(red|blue)
```

To make it harder, let's say that if (and only if) we match "blue", we want to optionally match a space and the word "car" if they follow directly afterward. In other words, we want to match "red", "blue", or if possible, "blue car", but we do not want to match "red car". We cannot use the pattern:

```
(red|blue) ( car)?
```

because that will match "red car". Nor can we use:

```
(red|blue car|blue)
```

because in our replacement pattern, we want the backreference to only contain "red" or "blue", without the " car". Using a conditional subpattern, however, we can search for:

```
((blue) | (red)) (? (2) car)?
```

Here's how this pattern works. First, we start with "((blue)|(red))". When this subpattern matches "blue", \1 and \2 are set to "blue", and \3 is empty. When it matches "red", \1 and \3 are set to "red", and \2 is empty.

Next comes the conditional subpattern "(?(2) car)". The conditional test is on "2", the second backreferenced subpattern: if \2 is set, which in our case means it has matched the word "blue", then it will try to match " car". If \2 is not set, however, the entire conditional subpattern is skipped. The question mark at the end of the pattern makes this conditional match optional, even if \2 is set to "blue".

Here's an example that uses an assertion for the condition, and the if-then-else form. Let's say we want to match a run of digits of any length, followed by either " is odd" or " is even", depending on whether the matched digits end with an odd or even digit.

```
\d+(?(?<=[13579]) is odd| is even)
```

This pattern starts with "\d+" to match the digits. Next comes a conditional subpattern, with a positive lookbehind assertion as the condition to be satisfied. The lookbehind assertion is true only if the last character matched by \d+ was also in the character class [13579]. If that is true, we next try to match " is odd"; if it is not, we try to match " is even". Thus, this pattern will match "123 is odd", "8 is even", and so on, but will not match "9 is even" or "144 is odd".

Once-Only Subpatterns

With both maximizing (greedy) and minimizing (non-greedy) repetition, failure of what follows normally causes the repeated item to be reevaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it to fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern "\d+foo" when matching against the text "123456bar".

After matching all 6 digits and then failing to match “foo”, the normal action of the grep engine is to try again with only 5 digits matching the `\d+` item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be reevaluated in this way, so the matcher would give up immediately on failing to match “foo” the first time. The notation is another kind of special parenthesis, starting with “(?)”, as in this example:

```
(?>\d+)bar
```

This kind of parentheses “locks up” the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Backtracking past it to previous items, however, works as normal.

In most situations, such as in the example above, the time saved by using once-only subpatterns is insignificant—a few small fractions of a second, at most. With some complicated grep patterns or with humongous lines of text, however, you can save tremendous amounts of time using once-only subpatterns.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both `\d+` and `\d+?` are prepared to adjust the number of digits they match in order to make the rest of the pattern match, `(?>\d+)` can only match an entire sequence of digits.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of a line of text. Consider a simple pattern such as:

```
abcd$
```

when applied to a long line of text which does not match (in other words, a long line of text that does not end with “abcd”). Because matching proceeds from left to right, the grep engine will look for each “a” in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as:

```
^.*abcd$
```

the initial `.*` matches the entire line at first, but when this fails (because there is no following “a”), it backtracks to match all but the last character, then all but the last two characters, and so on. Once again the search for “a” covers the entire string, from right to left, so we are no better off. However, if the pattern is written as:

```
^(?>.*)(?<=abcd)
```

there can be no backtracking for the `.*` item; it can match only the entire line. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the whole match fails immediately. For long strings, this approach makes a significant difference to the processing time.

When a pattern contains an unlimited repeat inside a subpattern that can itself be repeated an unlimited number of times, the use of a once-only subpattern is the only way to avoid some failing matches taking a very long time (literally millions or even billions of years, in some cases!). The pattern:

```
(\D+|<\d+>)*[!?]
```

matches an unlimited number of substrings that either consist of non-digits, or digits enclosed in `<>`, followed by either `!` or `?`. When it matches, it runs quickly. However, if it attempts to match this line of text:

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

it takes a long time before reporting failure. So long, in fact, that it will effectively “freeze” BBEdit. This is not really a crash, per se, but left to run on its own, it might take years before it finally fails. (We are not sure, frankly, because much like determining how many licks it takes to get to the center of a Tootsie Pop, we do not feel like waiting long enough to find out.)

The reason this takes so long to fail is because the string can be divided between the two repeats in a large number of ways, and all have to be tried before the grep engine knows for certain that the pattern will not match. (The example used `[!?`] rather than a single character at the end, because both PCRE and Perl have an optimization that allows for fast failure when a single character is used. They remember the last single character that is required for a match, and fail early if it is not present in the string.) If the pattern is changed to

```
((?>\D+)|<\d+>)*[!?
```

sequences of non-digits cannot be broken, and failure happens quickly.

Recursive Patterns

Consider the problem of matching a string in parentheses, allowing for unlimited nested, balanced parentheses. Without the use of recursion, the best that can be done is to use a pattern that matches up to some fixed depth of nesting. It is not possible to handle an arbitrary nesting depth. Perl 5.6 has provided an experimental facility that allows regular expressions to recurse (among other things). It does this by interpolating Perl code in the expression at run time, and the code can refer to the expression itself. Obviously, BBEdit’s grep engine cannot support the interpolation of Perl code. Instead, the special item `(?R)` is provided for the specific case of recursion. The following recursive pattern solves the parentheses problem:

```
\(((?>[^( )]+)|(?R))*\)
```

First it matches an opening parenthesis. Then it matches any number of substrings which can either be a sequence of non-parentheses, or a recursive match of the pattern itself (that is, a correctly parenthesized substring). Finally there is a closing parenthesis.

This particular example pattern contains nested unlimited repeats, and so the use of a once-only subpattern for matching strings of non-parentheses is important when applying the pattern to strings that do not match. For example, when it tries to match against this line of text:

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa (
```

it yields “no match” quickly. However, if a once-only subpattern is not used, the match runs for a very long time indeed because there are so many different ways the `+` and `*` repeats can carve up the subject, and all have to be tested before failure can be reported.

Browsers are special kinds of windows that let you see a lot of information about files at once. Browsers typically have two panes: one pane lets you select a file, the other displays detailed information about the file (often its contents). If you have performed a Find All search, you have already seen an example of a BBEdit browser.

In this chapter

| | |
|-----------------------------------------------------|-----|
| Browser Overview | 187 |
| <i>List Pane</i> – 187 • <i>Toolbar</i> – 188 | |
| <i>Text View Pane</i> – 188 • <i>Splitter</i> – 188 | |
| Disk Browsers | 189 |
| <i>Disk Browser Controls</i> – 189 | |
| <i>Contextual Menu Commands</i> – 190 | |
| <i>Dragging Items</i> – 190 | |
| <i>Using the List Pane in Disk Browsers</i> – 190 | |
| Search Results Browsers | 191 |
| Error Results Browsers | 192 |

Browser Overview

All BBEdit browsers share the same basic structure and behavior. All browsers have a toolbar, a file list, and a text pane. You can either edit files directly in any browser window, or open them separately.

List Pane

The top pane of a browser lists the items available in the browser. This pane shows different information for different kinds of browsers:

| Browser | File List pane contains |
|------------------------------------|----------------------------------------------------------|
| Disk browser | Files and folders that BBEdit can open |
| Search results | File and line number of each match |
| Error results (or) general results | File, line number, and status message for each condition |

You can open both files and folders from the list pane. When you double-click a folder name, BBEdit replaces the file list pane with the contents of the folder. When you double-click a file name, BBEdit opens the file in an editing window. If the file list pane also included a line number, BBEdit scrolls to that line.

Controls above the list may allow you to determine what kinds of items are displayed in the list. For example, in disk browsers, there is a popup menu that lets you choose to display text files, all files, or other types of files, and another that lets you return the browser to a parent directory of the current folder. In error browsers, checkboxes allow you to hide or show all errors, warnings, or notes.

For results browsers, BBEdit can either show a hierarchical listing (where all the results associated with a particular file are grouped under that file, using disclosure triangles similar to those in the Finder's list views to reveal or hide the results list), or a flat listing showing each individual result on a separate line. You can choose which of these display methods to use by pressing the Make File List Flat/Hierarchical button (next to the Open button.)

To remove items from the display list, select them and press the Delete key, or choose Clear from the Edit menu.

In results browsers, you may Control-click on items in the list to bring up the contextual menu with relevant commands—primarily “Copy”, as well as Subversion commands if the selected item is a file within a Subversion working copy.

Toolbar

The browser toolbar is like the toolbar in editing windows. Some browsers have additional buttons and controls in the status area as well.

These standard items—the pencil icon; the Function, Text Options, Mark, Path popup menus; and the Info buttons—should already be familiar to you, since they appear on BBEdit document windows by default. See “Window Anatomy” in Chapter 4 for an explanation of these standard BBEdit functions.

Text View Pane

When you click on a file name in the list pane, BBEdit displays that file in the text view pane, and you can edit the file just as if it were open in a document window.

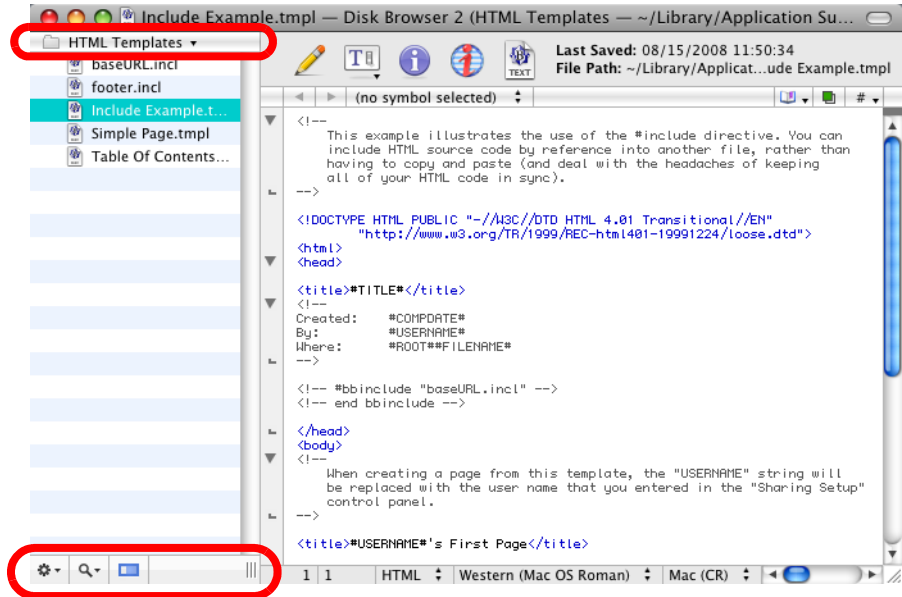
Splitter

You can change the size of the file list pane or the text view pane by dragging the double line that separates them. Double-clicking on the splitter bar will collapse the text view pane completely, and double-clicking on it again (in the bottom of the browser window) will restore the text pane to its previous proportions. You can also choose the Hide Editor or View Editor commands in the View menu to hide or display the text view pane.

Disk Browsers

Use a disk browser to explore the contents of a disk or a folder without opening each file one at a time.

To open a disk browser, pull down the File menu and choose Disk Browser from the New submenu. BBEdit opens a new disk browser that starts in your home directory, but you can navigate to any desired location:



The name and path of the file (if any) and directory currently being viewed are displayed in the title bar of the window. The file list pane displays all the items in the current folder. Click on a file in the file list pane to open it in the text pane, or double-click to open the file into a text window.

Disk Browser Controls

The menus at the top and bottom of the file list pane let you create new files and folders, open existing files and folders, reveal them in the Finder or navigate to them in the Terminal, limit the kinds of files to show in the list pane, and navigate through your disks and folders.

Directory Menu

The Directory popup menu at the top of the file list pane always shows the currently active folder. You can use this menu to “back out” of any folder you are currently in to a higher-level folder (as you can by Command-clicking the name of a folder in the Finder).

Action Menu

The commands on the Action (gear) popup menu at the bottom of the file list pane allow you to open the selected items, reveal them in the Finder, copy their paths, navigate to their location in the Terminal, move them to the Trash, or create a new file or folder.

Filter Menu

The Filter (magnifying glass) popup menu at the bottom of the file list pane lets you specify the kinds of files you want BBEdit to list in the browser.

You can choose whether to display all files or only text files with the “Text Files Only” option, and whether or not to display invisible files and folders with the “Invisible Items” option. You can also select a file filter to further limit what files BBEdit should display. You can define additional file filters in the Filters panel of the Setup window.

Toggle Editor Button

Click this button to collapse or expand the browser’s text view pane. (This button has the same effect as choosing the View/Hide Editor command in the View menu.)

Contextual Menu Commands

If you select one or more items in the file list pane and bring up the contextual menu, BBEdit will offer a variety of commands including those available from the Action menu.

Dragging Items

You can select and drag files and folders from a disk browser’s file list to any location, either within BBEdit or elsewhere, which can accept file or folder drags. For example, you can drag a file from a disk browser to a project window to add it to that project, or to an editing window to insert its contents, or to a folder in the Finder to copy or move it.

Using the List Pane in Disk Browsers

The list pane of a disk browser displays disks, files, and folders. When you are at the computer level, the list shows all mounted volumes.

When you click a folder or disk in the list pane, BBEdit displays the names of all the files it can open in the text pane, subject to the criteria specified by the Show and Filter menus.

When you click a file name in the list pane, BBEdit displays that file in the text pane.

To open a folder or disk and display its contents in the file list pane, you can either double-click it, or Select it and press Command-Down Arrow.

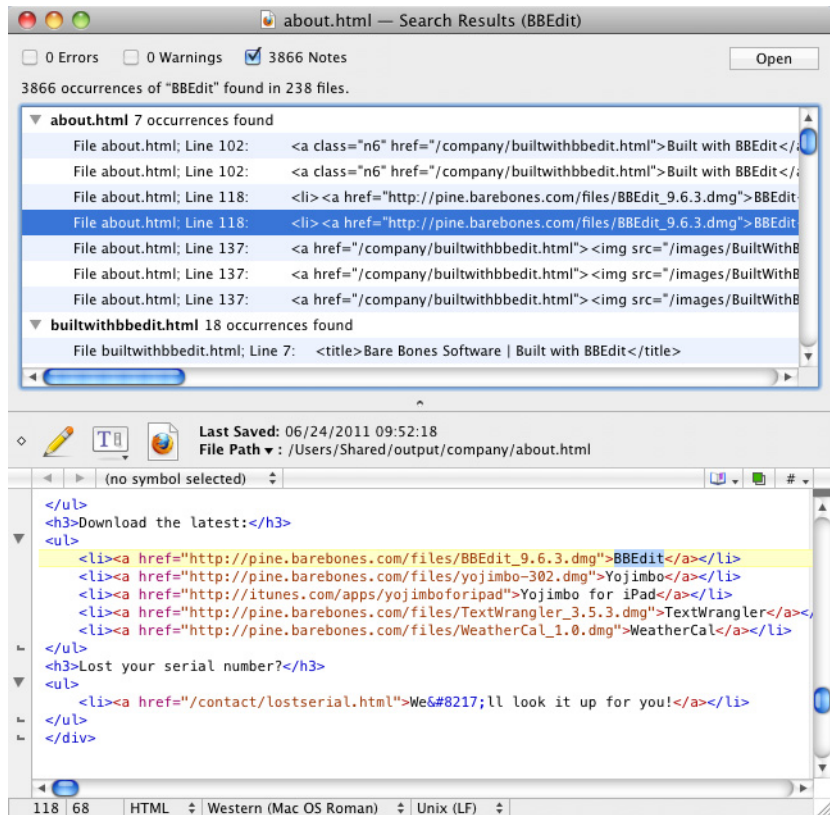
To go up one level to the enclosing folder or disk, you can either choose the enclosing folder from the directory popup menu, or press Command-Up Arrow

You can also use Quick Look to examine any non-text file by selecting it and pressing the spacebar.

Note When the list pane has input focus, the browser window’s AppleScript “selection” property will return a list of the files currently selected. See “Getting and Setting Properties” on page 289 for further details.

Search Results Browsers

If you selected the Batch Find option when performing a multi-file search, BBEdition displays every occurrence of the search string in the searched files in a search results browser.



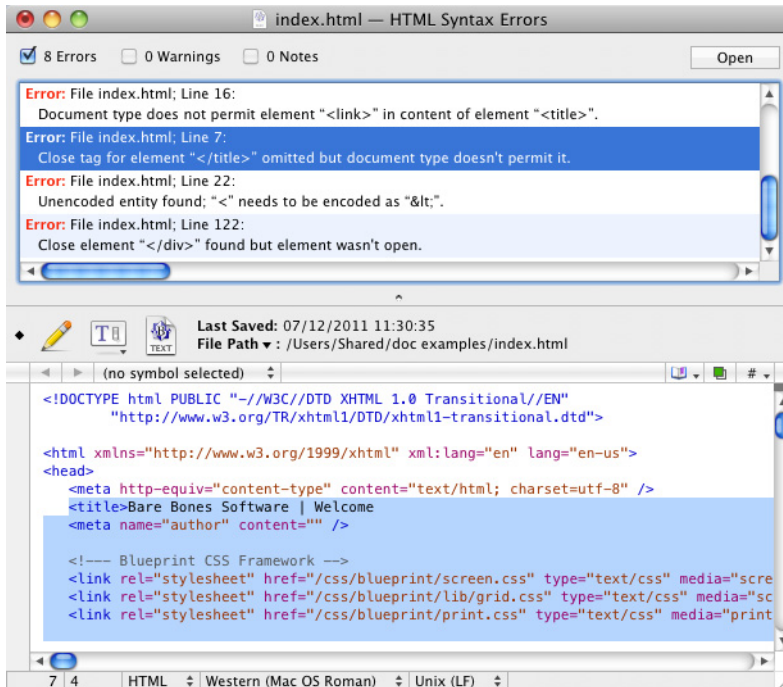
The items at the top of the window tell you how many matches BBEdition found in the set of files you specified, as well as whether any error conditions or warnings were generated during the search. The list pane lists each line that contains the matched text. Every match is identified by file and line number. To choose whether to display the search errors, warnings, and results, use the checkboxes at the top of the browser.

To open the file which contains a particular match, just click on that match in the results list. After you have opened a file, you can use the Search menu commands to continue searching it. (See Chapter 7 for more information on searching.)

The Open button opens the selected items using BBEdition. To open the selected items using the Finder, hold down the Option key while clicking the Open button.

Error Results Browsers

When you use the HTML syntax checker, link checker, or update tool, BBEdit will open an error results browser to display any errors generated by the command. BBEdit will also open an error results browser to list errors generated by Perl scripts.



Each entry in the list pane corresponds to an error, warning, or note. You can use the checkboxes for each type of item to suppress or display the associated results as desired.

If you click on a entry in the file list, BBEdit will open the corresponding file in the text display pane and select the section of text related to the error.

You can use the Preferences command to customize much of BBEdit's behavior. You can decide which windows are open when you launch BBEdit, set the default options for windows, set the default options for searches, and so on. This chapter describes BBEdit's extensive preference options.

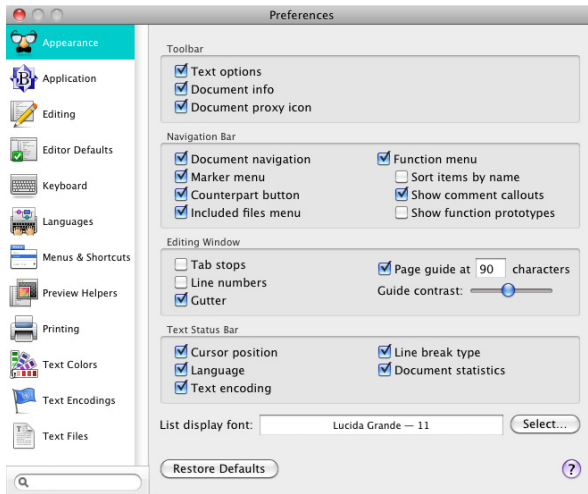
In this chapter

| | |
|------------------------------------------------|------------|
| The Preferences Window | 193 |
| <i>Searching the Preferences</i> – 195 | |
| <i>Application Preferences</i> – 197 | |
| <i>Editing Preferences</i> – 200 | |
| <i>Editor Defaults Preferences</i> – 200 | |
| <i>Keyboard Preferences</i> – 202 | |
| <i>Languages Preferences</i> – 204 | |
| <i>Menus & Shortcuts Preferences</i> – 205 | |
| <i>Preview Helpers Preferences</i> – 206 | |
| <i>Printing Preferences</i> – 207 | |
| <i>Text Colors Preferences</i> – 208 | |
| <i>Text Encodings Preferences</i> – 209 | |
| <i>Text Files Preferences</i> – 210 | |
| <i>Expert preferences settings</i> – 211 | |
| The Setup Window | 212 |
| <i>Bookmarks</i> – 212 | |
| <i>Filters</i> – 212 | |
| <i>Patterns</i> – 212 | |
| <i>Sites</i> – 212 | |

The Preferences Window

The Preferences window provides control over many aspects of BBEdit's behavior. You can decide which actions BBEdit should perform when you launch it, set default options for editing behavior, examine and set or modify keyboard shortcuts, create and apply text color schemes, and so on.

To open the Preferences window, choose the Preferences command from the BBEdit menu.



To select a preference panel, click its name in the list at the left side of the window. The text area at the top of the Preferences window gives you a brief description of the options provided by the currently displayed preference panel.

BBEdit's Preferences window is non-modal: you can leave it open and change preference settings while you work, or close it at any time by clicking its close button or by choosing Close Window from the File menu. Any changes you make to preference options take effect immediately unless otherwise indicated.

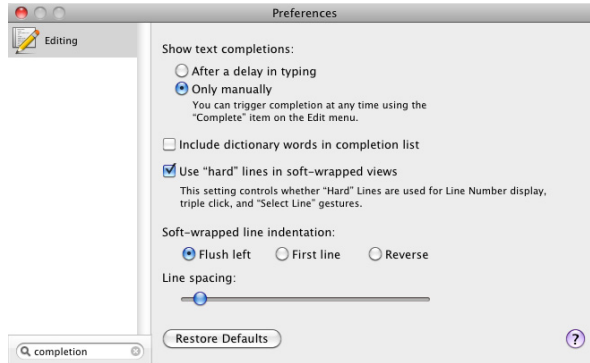
IMPORTANT

BBEdit employs the standard system preferences mechanism to store your preference settings. Accordingly, you can modify preference options directly by issuing “defaults write” commands. However, if you choose to modify your preferences by means of “defaults write” commands other than those documented in this manual or the “Expert Preferences” page of BBEdit’s built-in Help book, without explicit advice from Bare Bones Software technical support, you take responsibility for any adverse effects.

If you discard your BBEdit preferences file, you will have to re-activate the application with your product serial number, in addition to re-selecting any customized preference options you may have chosen.

Searching the Preferences

You can perform keyword searches to quickly locate preference options in the Preferences window. To do this, just click in the search field below the list of preference panels, and type a word or partial word into the field. As you type, BBEdit will search for instances of the current term and display all the panels which contain it. You can then select any of the listed panels to view and change the options within it. For example, here is the Preferences window with an active search for the term “completion”.



Restore Defaults

Each of BBEdit’s preference panels contains this button, which you can click to reset all preference options within the current panel to their factory default settings.

Appearance Preferences

The Appearance preferences let you choose which control and display elements appear in text windows and in other windows which include text panes.

Toolbar

When any of the listed options are on, BBEdit displays the toolbar (see page 73). You can also show or hide the toolbar independently in any editing window.

Text options

When this option is on, BBEdit displays the Text Options popup in the toolbar (see page 73).

Document info

When this option is on, BBEdit displays information about the current document’s date and path in the toolbar (see page 73).

Document proxy icon

When this option is on, BBEdit displays the document proxy icon in the toolbar (see page 73). This icon serves as a proxy for the document file; you can click it to reveal the current file in the Finder, or drag it anywhere the original file can be dragged.

Navigation Bar

When any of the listed options are on, BBEdit displays the navigation bar (see page 75). You can also show or hide the navigation bar independently for each text window. This option is on by default.

Document Navigation

When this option is on, BBEdit displays the Previous and Next buttons and the Document popup menu in the navigation bar (see page 76).

Marker menu

When this option is on, BBEdit displays the Marker popup menu in the navigation bar (see page 77).

Counterpart button

When this option is on, BBEdit displays the Counterpart button in the navigation bar (see page 78).

Included files menu

When this option is on, BBEdit displays the Included Files popup menu in the navigation bar (see page 78).

Function menu

When this option is on, BBEdit displays the Function popup menu in the navigation bar (see page 76). The related options below control how items appear in the menu.

Sort items by name

If this option is on, BBEdit sorts the items in the Function popup menu by name. Otherwise, items appear in the same order in the menu as they appear in the file. This option is off by default.

Show comment callouts

When this option is on, BBEdit will suppress callouts embedded in comments from appearing in the Function popup. This option is on by default.

Show function prototypes

When this option is on, BBEdit displays the names of function prototypes as well as function definitions in the Function popup menu. Otherwise, the menu does not include entries for function prototypes. This option is on by default.

Editing Window

These options control additional elements which BBEdit can display in editing windows.

Tab stops

If this option is on, BBEdit displays tab stops as vertical grid lines within the content area of text windows, using the tab width set in the Editor Defaults panel.

Line numbers

If this option is on, BBEdit displays line numbers along the left edge of the window.

Gutter

When this option is on, BBEdit displays the gutter (see page 81). You can show or hide the gutter independently for each text window. This option is on by default.

Page Guide at N characters

When this option is on, BBEdit displays the page guide at the specified character width. The page guide is a visible boundary indicator, whose color and contrast you can adjust (see page 208). This option is on by default.

Guide Contrast

You can use this sliding control to adjust the contrast level of the page guide display region. (See “Tab stops” on page 196.)

Text Status Bar

When any of the listed options are on, BBEdit displays the status bar (see page 80). You can show or hide the status bar independently for each text window.

Cursor position

When this option is on, BBEdit displays the current location (line and column) of the insertion point, or the endpoint of the current selection range in the status bar (see page 80).

Language

When this option is on, BBEdit displays the Language popup menu in the status bar (see page 80).

Text encoding

When this option is on, BBEdit displays the Text Encoding popup menu in the status bar (see page 81).

Line break type

When this option is on, BBEdit displays the Line Break Type popup menu in the status bar (see page 81).

Document statistics

When this option is on, BBEdit displays an item in the status bar which shows the number of characters, words, and lines in the document (and, if there's a selection, the number of characters, words, and lines in the selection range).

List Display Font

This option controls the font and size used to display text in browser list panes, including disk browser, search results browsers, etc. To change this option, click Set to bring up the standard Font panel, and choose the desired font and size. The default setting is 11 point Lucida Grande.

Application Preferences

The Application preferences control how BBEdit checks for updates, when open files are verified, what action BBEdit performs at startup, and various other global settings.

Open documents into the front window...

This option controls whether BBEdit should attempt to open newly created or opened documents into the frontmost window (if possible), or whether each document should open directly into a separate text window.

This option is active by default, and while it is, BBEdit will handle documents in the following manner.

When you open an existing document, BBEdit will look for a project document which contains the document's file. If there is one, BBEdit will open the document that project's window (and bring the project window to the front). Otherwise, BBEdit will open the document into the frontmost editing window (and bring that window to the front if it is not already there.)

When you create a new document (via the File menu), BBEdit will never open that document in a project document's window: it will either use the frontmost editing window (if one is available), or make a new editing window if necessary. If you wish to explicitly create a new document explicitly within a project, you may use the "New Document..." within the project's action menu (or contextual menu) to do so.

Automatically refresh documents as they change on disk

This option controls whether BBEdit checks if documents (files) have changed on disk while they're open. If an open document has changed on disk, and there are no unsaved changes, BBEdit will automatically reload the document. If a document has changed on disk and also has unsaved changes, BBEdit will ask whether you want to reload the document from disk or keep the unsaved changes. This option is on by default.

The effects of the Revert command (from the File menu), and of a file Reload (which occurs when a document is reloaded by a refresh action) are both undoable.

Remember the N most recently used items

This text field lets you choose how many files appear on the Open Recent sub-menu of the File menu, and how many folders appear on the folder search popup menu in the Find Differences folder lists.

Always Show Full Paths in "Open Recent" Menu

Check this option to have BBEdit always display full paths in the Open Recent menu. If this option is off, BBEdit will only display path info when it's needed to distinguish between files with the same name.

When BBEdit becomes active

This preference controls what BBEdit does when you launch it, or activate it when there are no open windows (e.g. by clicking its Dock icon while the application is already running). To override any of these actions when launching BBEdit, hold down the following modifiers.

| Modifier(s) | Function |
|-------------|-------------------------------------------------------------------------------------------|
| Option | Suppress startup items only |
| Shift | Do not attempt to reopen documents, and suppress all external services and startup items. |

Do Nothing

Choose this option to prevent BBEdit from opening a new text editing window.

New text document

Choose this option to have BBEdit open a new, empty text editing window.

Reopen documents that were open at last quit

When this option is on, BBEdit will remember what documents (as well as disk browsers and FTP/SFTP browsers) were open when you choose the “Quit” command, and will attempt to reopen those documents the next time you launch it. This option is on by default.

Restore unsaved changes

When this option is on, BBEdit will preserve the contents of any unsaved document contents when you quit (including untitled documents) and restore those documents the next time you launch it. If you prefer the traditional Quit behavior, turn this option off.

Include documents on servers

When this option is on, BBEdit will attempt to reopen documents from remote servers when you launch it.

Automatically check for updates

This option controls whether BBEdit automatically looks to see if a newer version is available. Regardless of the setting of the checkbox, you can manually check for an update at any time by clicking the Check Now button.

The version checking mechanism used by BBEdit protects your privacy. It works by requesting information about the currently available version from Bare Bones Software’s web server. The server will log the date, time and originating address of the request, and which versions of the OS and BBEdit you are using. This information is used to guide the future development of BBEdit; it is not personalized and will not be disclosed. Click the Privacy button to see the actual request which BBEdit sends.

This option is not present in copies of BBEdit obtained via the Mac App Store.

Editing Preferences

The Editing preferences control the behavior of various general editing behaviors.

Show text completions

This option lets you choose when BBEdit will display text completions: automatically after a short delay while typing, or manually upon typing a trigger key (F5 by default) or choosing the Complete command in the Edit menu.

Include dictionary words in completion list

When this option is on, BBEdit will include dictionary words in the text completion list. This option is on by default.

Use “hard” lines in soft-wrapped views

When this option is on, the line number bar, cursor position display, and Go To Line commands in editing views will use line and character position numbers that correspond to the “hard” line breaks actually present in the document, rather than the soft-wrapped line breaks.

Additionally, when this option is on, line selection commands and gestures, including the Select Line command, triple-clicking, and click selection in the left margin, will treat only “hard” line breaks as line boundaries.

Soft-wrapped line indentation

This option lets you specify how BBEdit should indent soft wrapped text: flush with the left edge of the window, at the same indent level as the first line of the paragraph, or indented one level deeper than the first line of the paragraph.

Line spacing

This control allows you to adjust the amount of space between lines of text in editing views. The default value is consistent with previous versions of BBEdit.

Editor Defaults Preferences

The Editor Defaults preferences control the behavior of newly created document windows and documents without saved state information. Many of the options in this panel parallel options provided in the Text Options sheet and in the Text Options popup in the toolbar. The difference is that the options in the Text Options sheet and the Text Options popup control only the behavior of the active window, while the Editor Defaults preferences control the behavior of all new windows.

Auto-indent

When this option is selected, pressing the Return key in new windows automatically inserts spaces or tabs to indent the new line to the same level as the previous line.

Tip To temporarily invert the sense of the Auto Indent option while typing, hold down the Option key as you press the Return key.

Balance while typing

When this option is selected, BBEdit flashes the matching open parenthesis, brace, bracket, or curly quote when you type a closing one. This option is useful when you are editing source files, to ensure that all delimiters are balanced.

Use typographer's quotes

When this option is on, BBEdit will automatically substitute curly (or typographer's) quotes (“ ” ‘ ’) for straight quotes (" ") in any new documents you create.

Tip To type a straight quote when this option is selected (or to type a curly quote when the option is deselected), hold down the Control key as you type a single or double quote.

Note You should avoid using typographer's quotes when creating or editing any plain-text documents such as email message content or source code.

Auto-expand tabs

When this option is on, BBEdit inserts an appropriate number of spaces instead of a tab character every time you press the Tab key.

Show invisible characters

This option shows or hides non-printing characters in the window. Select this option when you want to see line breaks, tabs, and gremlins (invisible characters). BBEdit uses these symbols to represent non-printing characters:

| Symbol | Meaning |
|--------|-------------------------------|
| Δ | tab |
| ◇ | space |
| • | non-breaking space |
| ↵ | line break |
| ¶ | page break |
| ¿ | other non-printing characters |

Show Spaces

When this option is on (and Show Invisibles is also active), BBEdit will display placeholder characters for spaces. Turn this option off to suppress the display of spaces (reducing visual clutter when you are displaying invisible characters).

Note Non-breaking spaces (typed by pressing Option-space) will not be displayed with a placeholder.

Check spelling as you type

When this option is on, BBEdit will automatically check spelling as you type, and underline any potentially misspelled words. Turn this option off to prevent BBEdit from automatically checking spelling.

You can turn on automatic spell checking for the active document only by choosing Check Spelling as You Type from the Text menu. (See “Check Spelling As You Type” on page 107.)

Default font

This option controls the standard font and font size which BBEdit uses to display the contents of text windows. To change this option, click Set to bring up the standard Font panel, and choose the desired font and size. The default font is “Consolas for BBEdit” at 12 points.

(The “Consolas for BBEdit” font is available for use only with BBEdit, under license from Ascender Corp.)

Note You can also adjust the default tab width on a per-language basis. To do so, select a language entry in the Languages preference panel, click “Options” to bring up the language options sheet, and enter the desired tab width in the Editing section of this sheet.

Tab Width

This option controls the default number of spaces that BBEdit uses to represent the width of a tab character.

Soft Wrap Text

When this option is selected, BBEdit soft-wraps the text in the file to the right margin that you choose: the Page Guide, the window width, or a specified number of characters, as selected by the options below the checkbox.

Keyboard Preferences

The Keyboard preferences control BBEdit’s response to the use of various special keys, including the ability to recognize Emacs key bindings

Use Tab key to navigate Placeholders

When this option is on, BBEdit will jump to the next placeholder in the document (if any) when you press the Tab key, or the previous placeholder if you press Shift-Tab. This behavior is equivalent to the Go to Previous/Next Placeholder commands in the Search menu (see page 154). For additional details, see “Selection and Insertion Placeholders” on page 263. This option is on by default.

“Home” and “End” Keys

Choose “Scroll to Beginning and End of Document” to have the Home and End keys perform these respective actions. This is the default setting, which reflects the standard key motion behavior in Macintosh applications.

Choose “Move Cursor to Beginning and End of Current Line” to have the Home and End keys perform these respective actions instead. This option may be useful for those accustomed to Windows editing key behavior.

Enter key generates Return

When this option is on, BBEdit will generate a carriage return when you press the Enter key.

When this option is off, pressing the Enter key will bring the current insertion point (or selection range) into view.

Note Pressing the Enter key in a Unix worksheet will always execute the current line (or the selected lines).

Allow Tab key to indent text blocks

When this option is on, you can press the Tab key to invoke the Shift Right command, or Shift-Tab to invoke the Shift Left command; this may be useful for those accustomed to Windows editing key behavior. When this option is off, pressing Tab will insert a tab character in the normal manner. This option is off by default.

Enable Shift-Delete for forward delete

When this option is on, holding down the Shift key with the Delete key makes the Delete key work the same way as the Forward Delete key on extended keyboards.

Option-¥ on Japanese keyboards

This option controls whether typing Option-yen on a Japanese keyboard generates a yen symbol “¥” or a backslash “\”.

Emulate Emacs key bindings

If turned on, this option allows you to use the basic Emacs navigation keystrokes to move around in editing views. It is not a full Emacs emulation mode; rather, it is more of a comfort blanket for individuals with Emacs key bindings hard-wired into their muscle memory. See Appendix B, “Editing Shortcuts,” for a list of the Emacs commands BBEdit supports.

Display status window

When both this option and “Emulate Emacs key bindings” are on, BBEdit will display a small palette which shows Emacs shortcuts as you type them.

Languages Preferences

The Languages preferences allow you to configure how BBEdit maps file names to language types (e.g. “.html” to HTML), and allows you to apply customized behavior and display parameters to any installed language.

Installed Languages

Click the “Installed Languages” button at the bottom of this panel to see a complete list of installed languages, together with the language module version number (if applicable) and filename extension(s) associated with each language. (This list includes both languages intrinsically supported by BBEdit, and those added via installed language modules.)

By default, BBEdit will apply your active preference settings within each language.

If you wish to modify how BBEdit treats documents having a particular language, e.g. to have BBEdit use a specific tab width or a custom color scheme, you may add a custom language preference.

To create such a preference, click the plus (+) button below the list of Custom Language Preferences, and select the desired language from the resulting popup. When you do so, BBEdit will display a language options sheet which contains the following sections:

- **General:** In this section, you can view or change the comment-start and comment-end strings used by the Un/Comment command on the Text menu for the selected language, or to view or change the Reference URL Template used by the Find in Reference command.
- **Editor:** In this section, you can view or change the default display and editing options used for documents in the selected language. (These options parallel the options provided by the Text Options command.)
- **Display:** In this section, you can view or change the default items which appear on the navigation bar and status bar for documents in the selected language. You can also choose any available color scheme to use for syntax coloring of documents in the selected language.

To remove an existing language preference, select the desired entry in the list of Custom Language Preferences, and click the minus (-) button below the list. Once you have removed the entry, BBEdit will again apply its active global preferences settings to all documents with that language.

Custom Extension Mappings

BBEdit includes a set of default file extension mappings which cover the most common usages for each supported language, while each language module ordinarily contains extension mappings for the language it supports.

You may add (or remove) additional extension mapping via the Custom Extension Mappings list. To add a mapping, click the plus (+) button below the list, click in the Suffix column and type the desired filename extension, then select the associated language via the adjacent popup. (You can also edit existing mappings in the same manner.)

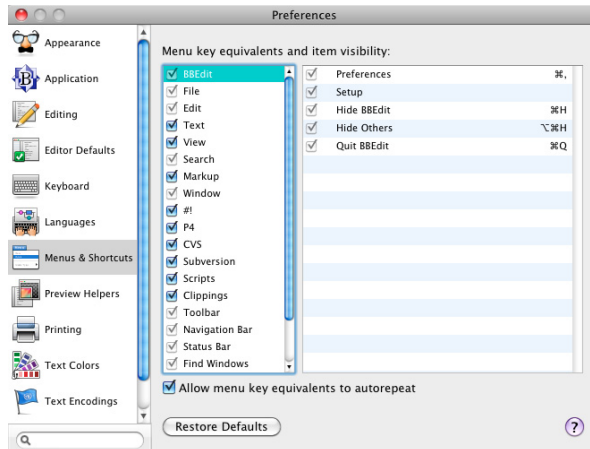
Note You can use wildcards in the suffix to indicate single characters (?), any number of characters (*), or a single digit (#). For example, "page.#html" could map to a different language from ".html".

Change clipping Set to match document's language

When this option is on, each time you bring a new editing window to the front, BBEdit automatically selects the first clipping set whose name maps to the same language as that window. For example, if the front window is an HTML document, BBEdit selects the first clipping set whose name ends in .html or .htm, or any of the other file suffixes mapped to HTML in the Languages preference panel. If there is no match, the active clipping set will not change.

Menus & Shortcuts Preferences

The Menus & Shortcuts preferences allow you to show or hide whole menus or individual commands. You can also assign key equivalents to commands and various window elements, as well as to clippings and scripts.



Menu Key Equivalents and Item Visibility

This section of the preference panel displays a hierarchical list of each menu and menu command available within BBEdit.

You can hide any menu or command which is not necessary for BBEdit to function, by turning off the checkbox next to that item's name. (The checkbox is disabled for necessary items, such as the File menu and the Quit command.)

You can assign or change the keyboard shortcut (key equivalent) for any menu command, as well as items on the Text Options, Markers, and Line Breaks toolbar popup menus, by double-clicking on the right-hand portion of that command's list item and typing the desired key equivalent in the Set Key sheet.

To clear the key equivalent from a menu command, double-click on the right-hand portion of that command's list item and press the Delete key.

Click Restore Defaults to restore all key equivalents to their factory default values (as listed in Appendix A).

Available Key Combinations

All menu key combinations must include either the Command key or the Control key (or both), except function keys, which may be used unmodified. The Help, Home, End, Page Up and Page Down keys can be used in menu key combinations as well. The Help key can be assigned without modifiers; the others must be used in combination with at least either the Command or Control key.

Note The system may preempt certain key combinations, such as Command-Tab.

Allow menu key equivalents to autorepeat

Turn this option on to enable autorepeat when typing key equivalents. (This option is off by default since according to the Macintosh Human Interface Guidelines, menu commands should not autorepeat.)

Preview Helpers Preferences

The Preview Helpers preference panel lists all web browsers on your machine which are available to preview HTML documents.

Web Browsers Available for Previewing

This list displays all the web browsers known to BBEdit. Browsers are listed by name and version number, in the same form as they appear in the Preview With submenu of the Markup menu.

The browser list includes each individual browser application that is available on your Mac. For example, if you have Firefox 5.0 and Firefox 3.6.18 on your hard disk, both applications will be listed and available for previewing.

You can use the plus (+) and minus (-) buttons to respectively to add a browser to the list or to remove an entry.

If you click the “Restore Defaults” button, BBEdit will ask the system for a list of all installed applications which claim to support HTTP URLs, and will add these applications to the list if they are not already present. If using this button does not add a browser which you know is available, you can add it directly with the Add button. (Sometimes, the system may not properly inform BBEdit of every browser application present.)

Previewing in Windows browsers through VMWare

BBEdit now recognizes the application stubs created by VMWare Fusion (version 2.0 or later), so that you can preview documents in Internet Explorer or other Windows-based browsers while running VMWare Fusion. Any documents you wish to preview must be readable by the running Windows virtual machine via either a shared folder or VMWare Fusion's “Mirrored Folder” option.

If the browser list does not include these VMWare-hosted applications, you can add them either manually or by using the Find All button as described above.

Printing Preferences

The Printing preferences control BBEdit's default document printing behavior.

Print using document's font

When this option is on, BBEdit uses the document's display font and tab settings when printing.

Printing font

This option specifies the default font BBEdit uses for printing when "Print using document's font" is not active. Click Set to bring up the standard Font panel, where you can choose a font and font size. The current printing font options appear in the display box.

Frame printing area

When this option is on, BBEdit draws a box along the edges of the printed text.

Print page headers

When this option is on, BBEdit prints the page number, the name of the file, the time and date printed in a header at the top of each page.

Print full pathname

When this option is on, BBEdit prints the full pathname of the file being printed in the header.

Time stamp

This option let you choose whether the date that appears in the printed page header is the date that the file was last modified or the date that the file was printed.

Print line numbers

When this option is on, BBEdit prints line numbers along the left edge of the paper.

1-inch Gutter

When this option is on, BBEdit leaves a one-inch margin along the left edge of the paper. Use this option if you usually store printed pages in three-ring binders.

Print color syntax

If this checkbox is on, BBEdit prints all colorized text within the document in color. You should generally use this option only on color printers, as colorized text may come out in difficult-to-read dithered shades of gray on black-and-white printers.

Text Colors Preferences

The Text Colors preferences let you adjust the default colors that BBEdit applies to syntax elements, as well as the foreground and background text colors and highlight colors.

You may also create and load custom color schemes within this panel. To save a color scheme, click the “Save Scheme...” button and name the theme. To load a saved color scheme, choose it in the Color Scheme popup menu.

The format for color scheme files is the same as that used by the BBColors tool:

`http://www.daringfireball.net/projects/bbcolors/`

and BBEdit will import any existing BBColors files which you have created. (However, the ‘bbcolors’ command-line tool has no effect in BBEdit 10.)

You can further associate a saved color scheme with any language via the Custom Language Preferences list in the Languages preference panel. (See “Languages Preferences” on page 204.)

How to Change an Element’s Color

The color bars show the colors that BBEdit uses to display different interface and language elements. To change the color for any element, click the adjacent color box to open the system color picker which you can use to select a new color. To restore all colors and options to their default settings, click the Restore Defaults button.

General

The three initial options control the foreground (text) and background (window) colors and the color of the underline used by the spelling checker to mark questioned words.

Spaces

This option controls the color BBEdit uses to display spaces when the Show Invisibles and Show Spaces display options are active.

Other invisibles

This option controls the color BBEdit uses to display invisible characters other than spaces when the Show Invisibles display options is active.

Use Custom Highlight Color

Turn this option on to have BBEdit use custom highlight colors. You can choose the primary and secondary highlight colors.

Highlight Insertion Point

When this option is on, BBEdit highlights the line currently containing the insertion point using the indicated color. You can choose the line highlight color.

Source Code

These options control the colors that BBEdit uses to display the corresponding language elements.

- Keywords are those terms defined in a language’s specification

- Predefined names are words which are not language keywords, but which are predefined by a language's reference implementation, or which are part of a language's standard library/framework support, or which have other special meaning to developers writing code in that language.
- Comments are all text set off by a language's designated comment marker(s).
- String and numeric constants are as defined by a language's specification.
- ctags symbols are any words or elements identified in an associated ctags file.

Markup

These options control the colors that BBEdit uses to display the corresponding types of HTML and XML tags and any attribute names and values within such tags.

Text Encodings Preferences

The top of the Text Encodings preference panel contains an alphabetical list of every character set encoding available in the system, and allows you to choose which of these encodings BBEdit includes in its menus. These menu are:

- The Read As popup menu in the Open dialog
- The Encoding popup menu in the Options dialog within the Save dialog
- The Encoding popup in the status bar
- The character set popup menus in various HTML tools dialogs (including the New HTML Document dialog)
- The encoding selection popup menus in this preference panel

To include an encoding for display, select it and click Enable. To remove an encoding from display, select it and click Disable. To include all encodings or remove all but the required the encodings, click the Enable All or Disable All buttons respectively.

(All available Unicode encodings are permanently enabled and cannot be turned off.)

Tip To keep the length of the encoding menus manageable, you should add only those encodings which you use frequently.)

Default text encoding for new documents

BBEdit uses the encoding specified by this option for new documents which do not contain an intrinsic encoding specification.

If file's encoding can't be guessed, try

If BBEdit cannot determine a file's proper encoding by examination, it will try opening the file using the encoding(s) contained in this list, in the order they appear.

Text Files Preferences

The Text Files preferences control how BBEdit opens and saves files, including whether to make backups.

Line breaks

This option controls what kind of line breaks BBEdit writes when creating a new file. You can choose:

- Unix line breaks (ASCII 10) for general use. This is the default option.
- Classic Mac line breaks (ASCII 13) if you will be using the file with Classic Macintosh applications.
- Windows line breaks (ASCII 13/10) if the file will reside on a Windows server or if you are sending it to someone who uses a Windows system

Ensure file ends with line break

When this option is on, BBEdit will add a line break at the end of the file if there is not already one present.

Strip trailing whitespace

When this option is on, BBEdit will trim all trailing non-vertical whitespace from the document file before writing it out.

Backups

These options control whether BBEdit should make backup copies of edited files, and the manner in which it does so.

Make backup before saving

Turn this option on to have BBEdit automatically make a backup copy of each file that you save. BBEdit creates a single backup file for each file that you save in the same folder as that file. This option is global and backups can no longer be made on a per-file basis. However, you can exclude individual files from being backed up by adding an Emacs variable to them (see “Emacs Local Variables” on page 44).

When this option is on, and you close a document with unsaved changes and elect to discard those changes (“Don't Save”), BBEdit will automatically save a snapshot of the document's contents in the same directory as the document, and the snapshot file's name will follow the Emacs convention “#foo.txt#” (or if the “Preserve file name extension” (see below) is on, the snapshot's name will be “#foo#.txt”).

Keep historical backups

When this option is on, BBEdit will preserve backups in the folder “~/Documents/BBEdit Backups/” and the “Preserve File Name” option (see below) will automatically be turned on and locked.

Within the backup folder will be one folder for each day's backup files. The format of the dated folder name is static and non-localized: YYYY-MM-DD. Inside of each day's backup folder will be all of the backup files made on that day, each named using a timestamped format.

You may change the location of the backup folder by placing a folder alias named “BBEdit Backups” in your “Documents” folder (~/Documents/) and BBEdit will follow the alias.

Alternatively, if you use Dropbox, you may just move the “BBEdit Backups” folder into your Dropbox folder (~/Dropbox/BBEdit Backups/) and BBEdit will use it in that location.

Preserve file name extension

By default, the backup files which BBEdit creates are named in accordance with current system conventions (which themselves follow the old Emacs convention): the backup file takes the name of the original with a tilde appended; for example, “foo.html~” is the backup of “foo.html”.

If you want backup files to have the same filename extension as the originals, turn on this option to have BBEdit place the tilde after the “base” name of the file; for example, “foo~.html”.

Controlling Backups with Emacs Variables

You may also use an Emacs variable to control whether or not a given file is backed up. There are two ways to do this:

Absolute: If the variable line/block contains a “make-backup-files” variable, that variable's value will override the global “Make Backup Before Saving” preference.

```
-*- make-backup-files: 1 -*- --> always back up this file  
-*- make-backup-files: 0 -*- --> never back up this file
```

If the first letter of the variable's value is “y”, “t”, or “1”, the value is “yes”, otherwise it's “no”. These are all synonymous:

```
make-backup-files: yes  
make-backup-files: y  
make-backup-files: true  
make-backup-files: t  
make-backup-files: 1
```

Inhibit: If the variable's line/block contains a “backup-inhibited” variable, and its value is true (see above), then the file will never be backed up, even if “Make backup before saving” is turned on in the global preferences.

Expert preferences settings

In addition to the preference settings which can be made through the Preferences window, BBEdit supports a number of expert preferences which you can adjust by issuing an appropriate “defaults write” command.

The “Expert Preferences” page within BBEdit's built-in Help book (choose “BBEdit Help” from the “Help” menu) contains a complete, current listing of these options.

The Setup Window

The Setup window allows you to manage several types of configuration info which BBEdit uses, including FTP/SFTP bookmarks, file filters, grep search patterns, and website configurations. (In older versions, most of this information was managed through the Preferences window.)

Bookmarks

This panel lists any bookmarks you have created for FTP and SFTP servers. You may click the plus (+) button to create a new bookmark, double-click any bookmark item to edit its stored options (or rename it), or select a bookmark and click the minus (-) button to remove it.

Filters

The Filters panel lists all the file filters you have defined for use with multi-file searches, Find Differences, and disk browsers. You may click the plus (+) button to create a new filter, double-click any filter item to edit its stored options (or rename it), or select a filter and click the minus (-) button to remove it. (For more information on using file filters in searches, see Chapter 7.)

Patterns

This list displays all the grep patterns (regular expressions) you have stored via the Grep pattern popup in the Find and Multi-File Search windows. These patterns are also available in most commands which allow you to specify grep patterns, such as the Process Lines commands in the Text menu.

You may click the plus (+) button to create a new pattern, double-click any pattern item to edit its stored options (or rename it), or select a pattern and click the minus (-) button to remove it.

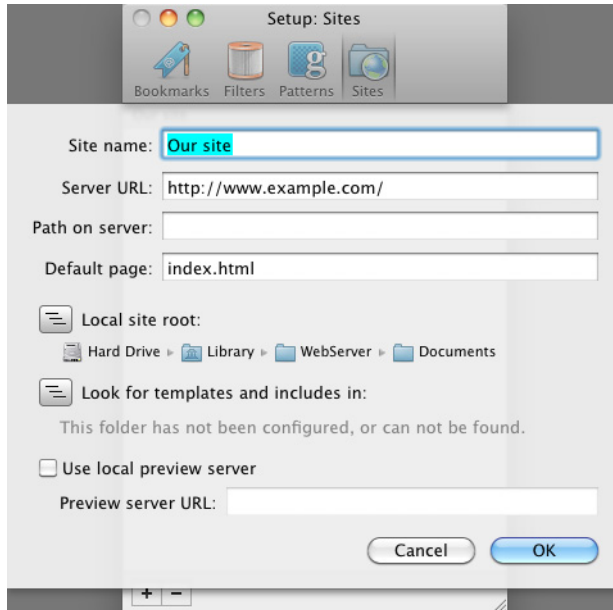
Sites

This list allows you to define web site configurations which associate information about a site with a local working folder containing a copy of the site's content. (In older versions of BBEdit, these configurations were managed via the Sites panel of the Setup window.)

All of BBEdit's commands which generate or operate on links, such as Edit Markup and Check Links, will recognize and take account of these site configurations to enable accurate construction and editing of links in the files you edit. (Please refer to Chapter 11 for further details on using these commands.)

You may define as many separate site configurations as you require. Click the plus (+) button to create a new site, double-click any existing site to edit its stored options (or rename it), or select a site and click the minus (-) button to remove it.

The site configuration sheet is shown below.



Site name

Enter the name by which this web site configuration should be listed in the preference panel. BBEdit also uses this name in the web site popup menu of the New HTML Document dialog (see “Creating a New Document” in Chapter 11) and in all other HTML tools dialogs which allow selection of a web site configuration.

Server URL

Enter the URL of your web server here, such as “http://www.example.com/” in the figure. BBEdit uses this information to determine which links are on (local to) your server.

Path on server

Enter the server path of your site’s main page here. For example, if your web site is at “http://www.example.com/foo/bar/”, you would enter “http://www.example.com/” for the Web Server Name (as noted above) and “foo/bar/” for the Site Path on Server.

Default page

Specify the default name used by your server for the document that is sent to a web browser when a browser accesses a directory without specifying a file name. Examples include “index.html”, “default.html”, “index.shtml”, and “index.”.

Local Site Root

Click the button to the left of this option and use the standard folder navigation dialog to select the root folder containing your local copy of the web site’s content. To open the current root folder in the Finder, double-click on the graphical path below this option.

Look for templates and include files in

Click the button to the left of this option and use the standard folder navigation dialog to select the local folder that contains your HTML document templates and include files. To open the current templates folder in the Finder, double-click on the graphical path below this option.

Use local preview server

If you have a web server running on your Mac, you can preview HTML pages through it by activating this option, and entering the base URL for your preview server. (To start the built-in web server, turn on the “Personal Web Sharing” option in the Sharing panel of the System Preferences.)

Preview server URL

When you are configuring a new web site, if the local site root folder is located within the “Sites” folder of your home folder (~Sites/), BBEdit will create an appropriate local http URL and enter it in this field. Otherwise, you can specify the local http URL for your site root folder.

Note If your web content folders are not located within ~/Sites/, or if you want to use virtual domains, need to enable PHP, etc., you must modify your machine’s Apache config file accordingly, since BBEdit’s site configurations cannot directly enable such an arrangement.

This chapter describes the use of BBEdition's HTML Tools, a powerful suite of utilities for creating and maintaining HTML documents and entire web sites.

In this chapter

| | |
|---------------------------------------------------------------------------------------------------|-----|
| Introduction to the HTML Tools | 215 |
| <i>Recommended Books</i> – 216 • <i>Recommended Online Resources</i> – 216 | |
| <i>What You Need</i> – 216 | |
| Configuring the HTML Tools | 217 |
| Using the HTML Tools | 217 |
| <i>Creating a New Document</i> – 218 • <i>File Addressing</i> – 220 | |
| <i>Checking Syntax</i> – 220 • <i>Format Customization</i> – 222 | |
| <i>Previewing Pages</i> – 222 | |
| HTML Tool Descriptions..... | 224 |
| <i>Edit Markup</i> – 224 • <i>Close Current Tag</i> – 226 | |
| <i>Balance Tags</i> – 226 • <i>Document Type</i> – 226 • <i>Character Set</i> – 226 | |
| <i>CSS submenu</i> – 226 • <i>Body Properties</i> – 233 • <i>Head Elements</i> – 233 | |
| <i>Block Elements</i> – 234 • <i>Lists</i> – 235 • <i>Tables</i> – 236 • <i>Forms</i> – 237 | |
| <i>Inline Elements</i> – 239 • <i>Phrase Elements</i> – 242 | |
| <i>Font Style Elements</i> – 243 • <i>Frames</i> – 243 • <i>Check</i> – 244 • <i>Update</i> – 245 | |
| <i>Includes</i> – 245 • <i>Utilities</i> – 246 • <i>Tidy</i> – 247 • <i>Preview</i> – 249 | |
| The HTML Tools Palette | 250 |
| <i>HTML Tools Palette Tips</i> – 251 • <i>HTML Tools Palette</i> – 251 | |
| <i>Other Palettes</i> – 252 | |
| HTML Translation..... | 253 |
| <i>Convert Paragraphs</i> – 253 • <i>HTML Entities</i> – 253 | |
| <i>Remove Tags</i> – 253 | |
| Templates..... | 254 |
| <i>Template Setup</i> – 254 • <i>Using a Template</i> – 254 | |

Introduction to the HTML Tools

IMPORTANT

Please be sure to read both this introduction and the next section, “Configuring the HTML Tools,” before attempting to create Web pages using these tools.

Already the most powerful set of utilities ever created for web developers, BBEdition's built-in HTML commands are more powerful than ever. These commands streamline the process of creating HTML documents, help you check for common usage errors, and speed up development time, without sacrificing flexibility or forcing you to work within the limits of visual editing tools.

BBEdit's HTML and CSS editing features and this chapter are written with the assumption that you already understand HTML. If you do not, we suggest one or more of the references listed below. None are published by or otherwise affiliated with Bare Bones Software, Inc., but other BBEdition users have found them useful for HTML usage and design issues.

Recommended Books

HTML for the World Wide Web with XHTML and CSS: Visual QuickStart Guide (6th Edition), Elizabeth Castro. Peachpit Press, 2006. ISBN: 0-32143-084-0

Cascading Style Sheets: The Definitive Guide (2nd Edition), Eric A. Meyer. O'Reilly and Associates, 2004. ISBN: 0-596-00525-3

Recommended Online Resources

HTML Help by The Web Design Group

<http://www.htmlhelp.com/>

The Bare Bones Guide to HTML by Kevin Werbach (no relation to Bare Bones Software)

<http://werbach.com/barebones/>

The W3 Consortium site

<http://www.w3.org/>

evolt.org — Browser Archive

<http://browsers.evolt.org/>

WebMonkey by Wired.com

<http://www.webmonkey.com/>

What You Need

Before you start, make sure you have the following available:

- A modern Web browser for previewing your pages. Safari is the obvious choice since it's supplied with the system, but you may want to try Firefox or Google Chrome as well. (These are the most widely used browsers but they often do not display pages the same way.)
- A general-purpose FTP client such as Fetch, Interarchy, or Transmit. While BBEdition does have built-in FTP tools, such dedicated applications are, naturally, more powerful, and of course also allow you to upload things other than text files. You will find them useful in creating and managing your web site.
- Access to a web server, either your own or someone else's, where you will publish your pages on the web. (Your Internet service provider can help you find the answers to questions about using their server facilities, or obtaining your own domain name, setting up your own dedicated server, and so on.)

You will also want to be familiar with BBEdition's basic capabilities. The other chapters in this manual will help you learn more about editing and searching text using BBEdition.

Configuring the HTML Tools

Before using the HTML Tools, you should prepare your source documents and your copy of BBEdit.

First, create a folder to contain working (draft) copies of your HTML documents. Put the HTML documents you are working on in this folder (either copying them from your web site or moving them from elsewhere on your hard drive); then upload them to your Web server when you are satisfied with them.

Next, launch BBEdit and select the Sites panel of its Setup window, then create a configuration as detailed on page 212. For now you need only set up a configuration for one web site, but you can create more site configurations later.

Using the HTML Tools

There are three ways to use BBEdit's HTML Tools commands: via the HTML Tools floating palette (see next page), via the Markup menu, and directly via the Edit Markup command and markup panel. These methods are functionally equivalent in most respects.

Most beginning users find it easiest to use the HTML Tools through the palette. There are three basic types of buttons on the HTML Tools palette:

- Those you simply click to perform an action or bring up a settings dialog before performing an action—for example, New Document, Close Current Tag, or Preview in BBEdit.
- Those that provide popup menus containing related options—for example, Heading, List, Table, and Form.
- Those which bring up BBEdit's markup panel to fill in attributes and values—for example, Anchor, Image, or Div.

The second means of using the HTML Tools is from BBEdit's Markup menu. This allows you to make your own choice between the drag and drop convenience of palettes, and the less screen-intensive menus; either way, you will still be able to access all of BBEdit's capabilities. Most common tags, as well as many utility functions, are available through items in the Markup menu or one of its submenus. Key equivalents (if assigned) are displayed next to the menu item. (You can change or set key equivalents for menu commands in the Menus & Shortcuts preference panel.)

Most of BBEdit's HTML Tools commands apply to the frontmost document—either at the current insertion point, or on the current selection range, as appropriate. Some utility functions, however, can operate on many documents. The Tool Descriptions section provides more details on what each command does.

The third way to use BBEdit's HTML Tools is by choosing the Edit Markup command to bring up BBEdit's markup panel, in which you can directly add or edit tags and their corresponding attributes and attribute values.

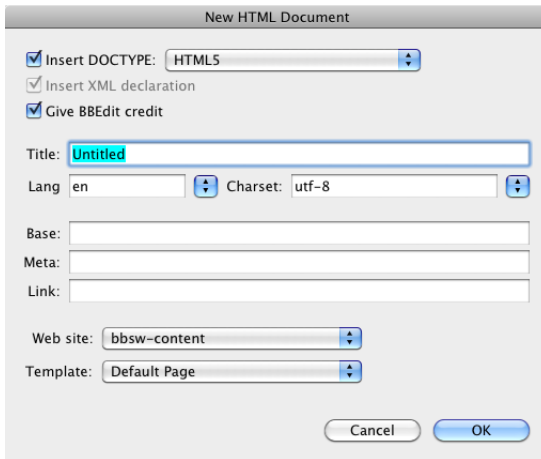
Creating a New Document

You can create an HTML document simply by taking any text file and adding HTML markup to it, but there's a better way. BBEdit includes a New Document command to create the basic skeleton of an HTML document for you.

To create a new HTML document, you can do either of the following:

- Chose New from the File menu and then choose HTML Document from the New submenu.
- Click the New Document button in the HTML Tools palette.

In either case, the following dialog appears:



In many cases, you can simply specify a title for the document and click OK, ignoring the other options. However, we suggest that you fill out this dialog as completely as possible. The function of each field is described below.

Insert DOCTYPE

Choose the type of this HTML document from the popup menu to have BBEdit insert an SGML prolog containing the desired document type. This information is largely ignored by browsers; however, HTML syntax checkers (such as the one built into BBEdit) use it to determine which constructs are legal according to the HTML standard you select. Available DOCTYPEs include:

- HTML5
- HTML 4.01 (Transitional, Frameset, and Strict versions)
- XHTML 1.0 (Transitional, Frameset, and Strict versions)
- XHTML 1.1

Insert XML declaration

Choose this checkbox to have BBEdit insert an XML declaration. If the DOCTYPE selected in the popup menu below is not an XML-based type (that is, is not an XHTML version), this checkbox will be disabled.

Give BBEdit credit

This option generates a `<META NAME="generator" CONTENT="BBEdit 10">` tag in the document, indicating that you used BBEdit to create it.

Title

Enter the HTML title for the document (which can be different from the file name) here. This text will appear in the title bar of a browser's window when this document is opened.

Lang

This option indicates the language this document is written in. This information can be used by search engines and translation software to help Web users find pages in their own language.

Charset

This option indicates the character set used by the document. If you do not specify a character set, the character set chosen in the user's browser will be used.

Note You can choose which character sets appear in this popup menu by using the Text Encodings preference panel.

Base

Enter the URL for this document's BASE tag. The BASE tag indicates the actual location of the document on a server, and all relative URLs specified in the document will be resolved by the browser relative to this location. No BASE tag is created if you leave this field blank.

Meta

Enter the META tag to be included at the top of the document here, if any. (META tags can be used for "client-pull" techniques, for indicating search keywords, and for a wide variety of other purposes.)

Link

If you want to use a LINK tag to specify a relationship between this document and other documents, an email address, style sheet, or other information about the document, enter the desired information in this field.

Note If you use a template to create the HTML document, the template must include the `#META#` and `#LINK#` placeholders to indicate the location at which this information should be inserted into the generated document.

Web Site

This popup menu displays the web sites projects defined in the Sites panel of BBEdit's Setup window, allowing you to switch between them for the purpose of choosing a template to base the new document on.

Template

This popup menu displays the templates specified in the “Templates & Includes” folder associated with the selected web site in the Sites configuration panel. (The names of template files must end with “.tmpl” to be displayed in this menu.) Selecting a template other than Default will use the specified template to create a new document, potentially ignoring some or all of the settings specified in this dialog.

File Addressing

Many HTML tags require you to specify the pathname or URL of a file or folder, such as to identify a base address, style sheet, or hypertext link. When you edit such a tag in BBEdit’s markup panel, you can type or paste the path directly or click the File button to the left of the attribute value field to bring up an Open sheet in which you can select the desired file.



In general, URLs may be constructed in any of the following three ways:

- *Full addressing* specifies the complete URL, including the scheme (“http:”), the server’s domain name, and the complete directory path leading to the file within that server.
- *Root addressing* specifies just the file’s location within its host server.
- *Relative addressing* specifies the file’s location relative to that of the HTML document referring to it.

For example, if the web site resides on a server named “www.example.com” in directory “foo/bar,” and you are creating a document in that directory named “index.html” with a link to file “target.html” in subdirectory “flapdoodle”, the full address would be

```
http://www.example.com/foo/bar/flapdoodle/target.html
```

the root address would be

```
/foo/bar/flapdoodle/target.html
```

and the relative address would be

```
flapdoodle/target.html
```

Checking Syntax

You can use BBEdit’s Check Syntax command (see page 244) to validate your HTML documents to the specification defined in their <!DOCTYPE> prolog. BBEdit will apply HTML5 rules when checking any document that does not contain a <!DOCTYPE> specification,

Note that an HTML document may display the way you expect it to in a browser and still contain invalid HTML. Browsers are designed to be lenient in the markup they accept, so you can get away with a certain amount of “sloppy” markup. However, producing well-formed (syntactically correct) HTML documents is the best way to assure that your document will display in some reasonable fashion in a wide variety of Web browsers, even those you have not tested the page in.

Syntax Checking Partial Documents

BBEdit can check the syntax of either complete or partial HTML documents. You may find the ability to check partial documents useful if you are preparing template sections for inclusion into other documents, whether this is done locally or via a server-side mechanism.

In order to check the syntax of a partial HTML document, the document must contain a balanced portion of the content tree. For example, you can check a partial document which contains a set of paragraphs or a table; you cannot check a partial document which contains an unclosed <BODY> or <DIV>.

Additionally, for partial documents which do not contain a DOCTYPE, you can specify one by means of a “#bbpragma doctype=” comment, which specifies what the root or parent element of the partial page’s content is. For example, if your partial document consists of <BODY> content:

```
<!-- #bbpragma doctype="-//W3C//DTD XHTML 1.0 Transitional//EN"
root_element="body" encoding="utf-8" -->
```

In this comment, you must use either the public identifier text for the DOCTYPE you wish to check against (see above) or the “display name” for the document type used in the New HTML Document dialog.

Finally, you may also specify an “encoding=” attribute to declare the encoding of the partial document.

Ignoring Sections of Documents

You can mark sections of HTML documents to have BBEEdit ignore these sections when performing a syntax check. This can be useful for purposes such as checking documents which must from necessity incorporate non-standard markup to support old browsers, or which contain customized server constructs. To mark a section, enclose it with “#bbpragma ignore_errors=” directives, as follows:

```
<!-- #bbpragma ignore_errors="on" -->
ignored markup
<!-- #bbpragma ignore_errors="off" -->
```

Note that when you check a document containing ignored sections, BBEEdit’s syntax parser still runs through the markup contained in these sections; it simply does not report errors encountered there. You should thus be mindful of the following conditions:

- The presence of fragmentary tags or similarly malformed content in an ignored section can cause a syntax check to fail.

- An error may still be generated for an unclosed element which resides within an ignored section, if its lack of closure results in an error cascade which continues beyond that section.
- If you terminate a document inside of an ignored section, an error will be generated.

Format Customization

The “Pretty Print” option of BBEdit’s Format command is implemented internally using a Dreamweaver-style source format profile. Advanced users may override the factory format profile by placing an appropriately constructed file at:

```
~/Library/Application Support/BBEdit/SourceFormat.profile
```

Previewing Pages

IMPORTANT

BBEdit’s Preview commands allow you to view your pages in one or more web browsers. You can display an automatically-updated page preview directly within BBEdit by choosing the Preview in BBEdit command, use the Preview in <Default Browser> command to use the current default browser, or choose a specific browser from the Preview In submenu. You can also preview the page in all running browsers or in a text-only format.

In addition to static previews, BBEdit also supports live local previewing of web pages through the web server built into Mac OS X. This capability enables you to easily preview pages which are built using server-side technologies, for example, DHTML or PHP.

To enable live previewing for a defined web site project, you must turn on the “Use local preview server” option and enter an appropriate “Path on server” in that site’s configuration (see page 214). You must also have turned on the “Personal Web Sharing” option in the Sharing panel of the System Preferences to start your Mac’s built-in web server.

Once you have done so, whenever you preview a file from that web site project using any of the Preview In commands, BBEdit will have the selected browser load that file’s corresponding page through the built-in web server.

Printing Previewed Pages

When a BBEdit preview window is frontmost, you can use the Page Setup and Print commands to print a copy of the displayed page.

Note

Due to limitations of the WebKit rendering engine which BBEdit employs, the format of the printed output may not exactly match the screen rendition.

Previewing from a File Vault-protected Home Folder

If your home folder is protected by FileVault, you will have to change the permissions on your home folder and your Sites folder (~/Sites/), before you can preview documents via “Personal Web Sharing”. Your home folder should have permissions of at least (drwx--x--x), and your Sites directory should have permissions of at least (drwxr-xr-x).

To set these permissions using the Terminal application, you can issue the following commands:

```
~ username$ cd ~
~ username$ cd ..
~ username$ chmod 711 username
~ username$ cd ~
~ username$ chmod 755 Sites/
```

where the command text is shown in bold, and **username** is the short form of your user account name. You can also use any file utility which is capable of setting folder permissions to make these changes.

Using Preview Templates

You can now apply templates and customized CSS to pages displayed by the Preview in BBEdit command. In order to do this:

- Place a fully structured HTML document in “~/Library/Application Support/BBEdit/Preview Templates/”. This document may contain anything you like but should define the basic structure and appearance of your desired page. Within the document, place this single placeholder: `#DOCUMENT_CONTENT#`.
- Make a new text document and add some content to it; you may add tagged content, however, this document should not have a complete HTML tag structure.
- Choose Preview in BBEdit to preview the document. BBEdit’s Preview window will display a row of items below the toolbar, including “Template:” and “CSS:”. In the Template popup menu, you can choose the template that you saved in the first step. When you do so, BBEdit will replace the `#DOCUMENT_CONTENT#` placeholder in that template with the contents of the document that you previewed.

Thus, you can use this technique to preview a fragmentary document without having to replicate the chrome defined in the template into that document.

Further, if you place a valid CSS document in “~/Library/Application Support/BBEdit/Preview CSS”, that document will be available in the Preview window’s “CSS:” popup menu and choosing it will apply that CSS to the Preview window’s contents.

Previewing Code and Text

BBEdit 10.1 supports previewing documents which are not HTML (or HTML generators) via the Preview in BBEdit command. For example, if you preview a C++ document, you will see an HTML rendering of that document as BBEdit displays it in the editing window. (The HTML displayed in the Preview window is the same HTML markup that you can generate by using the Copy as Styled HTML or Save as Styled HTML commands.) This is useful in situations where you want to typeset your code, and wish to experiment with different page templates and styles.

When previewing documents in a particular language, you can specify a default CSS file for BBEdit to use by placing an appropriately named CSS file in the “Preview CSS” folder of BBEdit’s application support folder. The name of that file should follow the pattern “DefaultCSS_*.css”, in which the “*” is replaced by the name of the language. For example, to designate a default CSS file for previewing Markdown, you should name that file “DefaultCSS_Markdown.html”.

HTML Tool Descriptions

This section describes all of the HTML Tools commands as they appear in the hierarchical Markup menu. For a description of the tools as they appear on the palette, see the HTML Tool Palette Index, which appears after the tool descriptions.

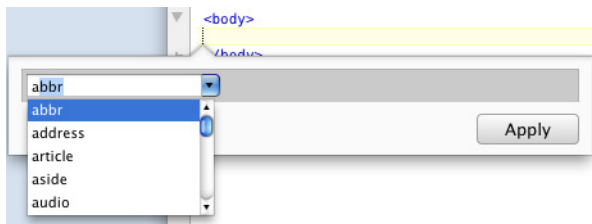
Note Tools that create tags insert the tag at the insertion point unless otherwise specified. Such tools also place an end tag automatically where appropriate.

You should already be familiar with HTML before using BBEdit’s HTML tools. BBEdit’s markup panel will help you associate correct attributes with each tag, and provide shortcuts to help you enter information; however, it does not (and cannot) know what intent or the final results of your markup will be. There is no substitute for knowing HTML.

Edit Markup

BBEdit 10 presents a new, context-sensitive markup panel for creating and editing HTML tags. When you choose the Edit Markup command from the menu, or invoke it by pressing Command-Control-M, BBEdit will open the markup panel, in which you can select any tag that is valid in the current HTML context of the insertion point and then add attributes and attribute values to that tag.

For example, if the insertion point is positioned directly inside the document’s body section (delineated by the `<body>` and `</body>` tags), the markup panel will open in tag selection mode, and you can either use type-ahead or the arrow keys to select among any available tag:



If you have selected a tag which does not require attributes or you don’t wish to add attributes, you may click Apply or press the Return key to enter that tag into the document. Choosing, for example, ‘audio’ will insert an `<audio>` tag at the insertion point, while choosing ‘article’ will insert an `<article></article>` tag pair (with any additionally chosen attributes) and leave the insertion point positioned between the two tags for easy content entry. You can close the markup panel without taking any action by typing the Esc key.

If you select a tag which has required or optional attributes, just press the Tab key to move the insertion point through the markup panel's fields, where you can choose or type the desired attributes and attribute values. To pre-insert all recommended attributes for a tag, you may either click the Fill (gear) control in the upper right corner of the markup panel or type Command-Control-M while the panel is open.

For example, here is the markup panel showing the DIV tag selected with pre-inserted attributes:

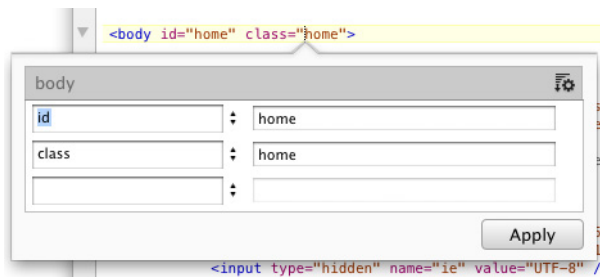


BBEdit's Tag Maker command also works within CSS. With the insertion point in a CSS selector or declaration, Tag Maker will display a list of valid CSS 2.1 properties. Option-click while clicking the Insert button and Tag Maker will open a follow-up dialog for editing the property (for classes that BBEdit offers editing dialogs). Invoking Tag Maker when the insertion point is not within a CSS selector will create a new skeleton rule set.

Tag Maker is also available through contextual menus. Simply Control-click at any point in your markup, and BBEdit will present all valid tags or attributes for the context of the insertion point within the Insert Tag submenu of the contextual menu.

Editing Existing Tags

You may also edit existing tags by placing the insertion point within them and choosing Edit Markup in the Markup menu or the HTML Tools palette (or by typing Command-Option-M). When you do this, BBEdit will bring up the markup panel pre-filled with all existing attributes and attribute values of that tag. For example, here is the markup panel invoked against an existing BODY tag:



Edit Markup also works with CSS. Choose Edit Markup while the insertion point is within a selector's property or value, and BBEdit will display a context-appropriate dialog for editing many common properties.

Close Current Tag

The Close Current Tag command inserts a closing tag to match the nearest opening tag preceding it. If the closing tag is placed on a new line, it will use the same indent level as the opening tag. For instance, if the insertion point is preceded by a `<P>` (Paragraph) tag plus some text content, Close Current Tag will insert a matching `</P>` tag to close the paragraph.

Note If you frequently work with HTML documents, you may want to assign a key equivalent to this command in the Menus & Shortcuts preference panel.

Balance Tags

When Balance Tags is chosen, BBEdit expands the selection to encompass the content of next outermost set of enclosing tags. The easiest way to understand how this works is to see it in action. Place the insertion point in an HTML document's `<TITLE>` element and choose Balance Tags. The title will be selected, since it lies between the tags `<TITLE>` and `</TITLE>`. Choose Balance Tags again, and BBEdit selects everything between `<HEAD>` and `</HEAD>`, the next set of enclosing tags outside the `<TITLE>` element. Choosing the command once more will select everything between `<HTML>` and `</HTML>`.

Use this command to quickly select an element for editing or just to check to see whether all your nested elements are formed correctly. If BBEdit sounds the system alert beep when you expect it to select text, it cannot find a matching set of tags around the selected text.

Document Type

The Document Type submenu allows you to select the desired document type (DTD) for the current document. (If the document already contains a DOCTYPE declaration, that option will be checked.) When you select a document type, BBEdit will insert the corresponding declaration into the document.

Character Set

The Character Set submenu allows you to select the encoding (character set) of the current document. When you select an encoding, BBEdit will insert the appropriate encoding declaration into the document.

Note You can specify which encodings appear in the menu via the Text Encodings preference panel.

CSS submenu

This submenu allows you to create, edit, and format Cascading Style Sheet markup. BBEdit has built-in support for CSS. When you are editing stand-alone CSS files or HTML files with embedded CSS, syntax coloring is available, and the Function menu lists CSS selectors, as well as CSS files referenced by `@import` directives and `<link>` tags. Choose an external stylesheet from the Function menu and BBEdit will open that stylesheet for editing.

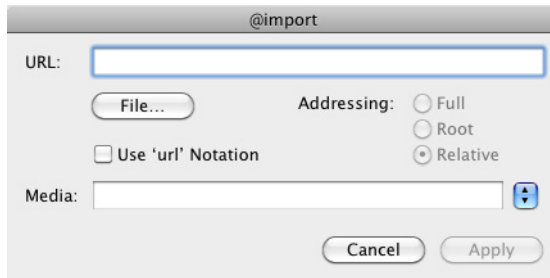
IMPORTANT BBEdit supports all CSS 2.1 properties. Although there are no explicit dialog editors for many such properties, you can create or modify them using the Tag Maker command, or by using Insert Property from the contextual menu.

The CSS function parser supports the following syntax for laying a mark in the function menu:

```
/* bbmark string to appear in the menu */
```

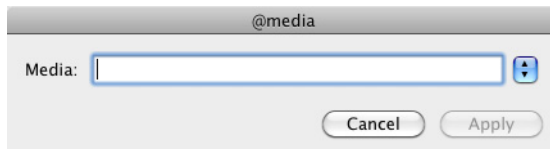
@import

The `@import` directive instructs a web browser to load an external style sheet. This dialog box allows you to select a file (or drag and drop one from the Finder) and choose whether to use the optional “url” notation for specifying the location of the style sheet. (Remember that `@import` must come before other CSS rules inside a `<style>` tag or in a stand-alone CSS document; otherwise it will not work.)



@media

The `@media` directive allows you to control which stylesheet should be used for different output media, e.g. screen versus printer. You can use this dialog to add media rules, or given an existing media rule, you can use the CSS editing dialogs or Edit Markup to edit the rulesets within.



For example:

```
@media print {
    body { font-size: 10pt; }
}

@media screen {
    body { font-size: 13px; }
}

@media screen, print {
    body { line-height: 1.2; }
}
```

Format

The CSS Format command will reformat your CSS markup for easier reading.

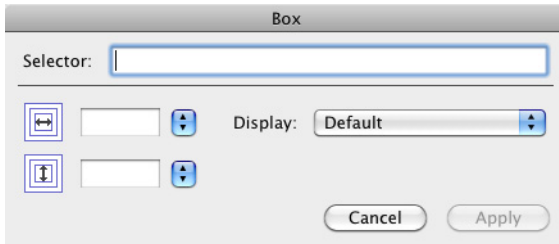
In stand-alone CSS files, if there is a selection range, only the selected text is formatted. If there is no selection range, the whole file will be formatted.

In HTML files with embedded CSS, if there is a selection range, only the selected text will be formatted. If there is no selection range, BBEdit will format all CSS in the `<style></style>` tag pair that encloses the insertion point. If the insertion point is outside a `<style></style>` tag pair, or if the selection range spans a `<style></style>` tag pair, the formatter will simply beep.

When formatting CSS embedded into HTML, BBEdit will indent the CSS based on the indent level of the opening `<style>` tag, plus one additional tab stop for better readability. BBEdit's CSS markup tools (listed below) use the same rules for formatting as does the Format command.

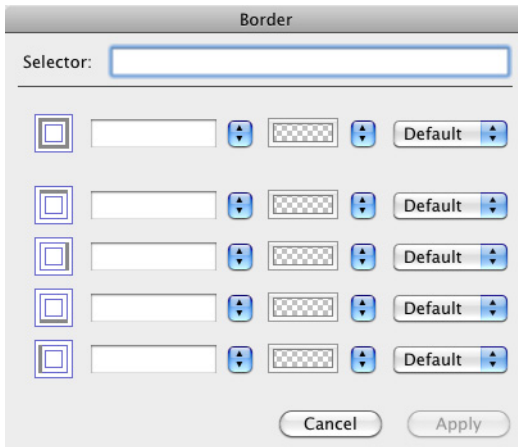
Box

The Box dialog allows you to specify a selector's width, height, and display properties.



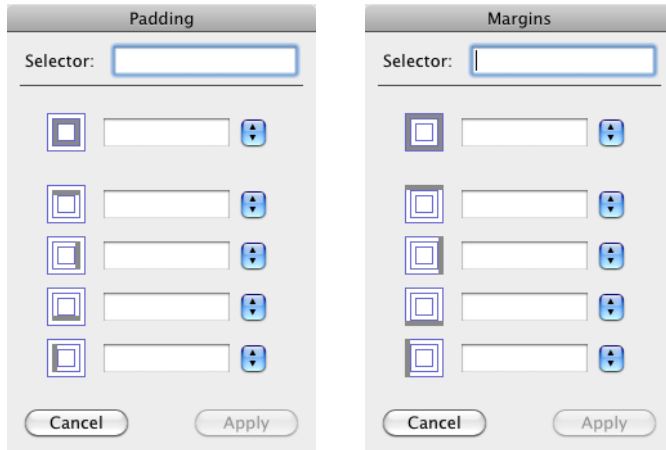
Border

The Border dialog allows you to edit the border-width, border-color, and border-style properties for a selector. The first row lets you specify values that apply to all four sides. The color buttons let you select colors using the system color picker; the popup menus next to them let you select colors by name. The icons on the left side of the dialog represent (from top to bottom), the entire border, top, right, bottom, and left.



Padding/Margins

These similar dialogs allow you to edit the padding and margin properties. In both cases, the icons on the left in the dialogs represent the entire box, top, right, bottom, and left, respectively.



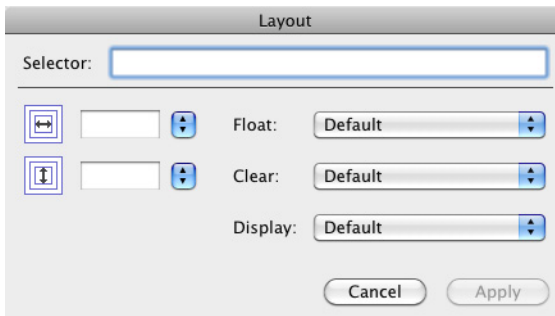
When you are working with the individual margin fields in these dialogs as opposed to the overall value, they behave the same way CSS value replication does:

- If right is missing, it takes on the value of top
- If bottom is missing, it takes on the value of top
- If left is missing, it takes on the value of right

so an empty field has special meaning - it means “replicate the related value”. If you want to specify a value for any given side, you must enter it explicitly.

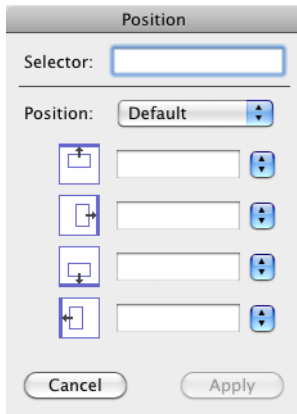
Layout

The Layout dialog allows you to edit the page layout properties of a selector.



Position

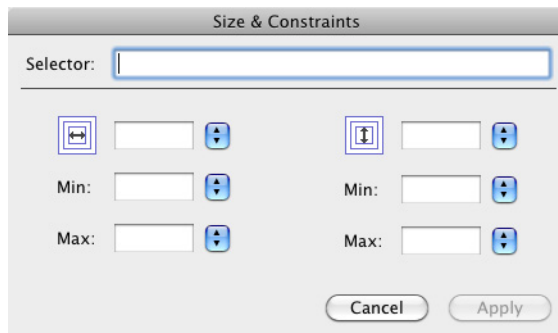
The Position dialog allows you to edit the positioning properties of a selector.



The Position dialog box has a title bar labeled "Position". It contains a "Selector:" text input field. Below it is a "Position:" dropdown menu currently set to "Default". There are four rows of controls, each consisting of a small icon (representing top, right, bottom, and left alignment) and a text input field with a vertical spinner button to its right. At the bottom are "Cancel" and "Apply" buttons.

Size & Constraints

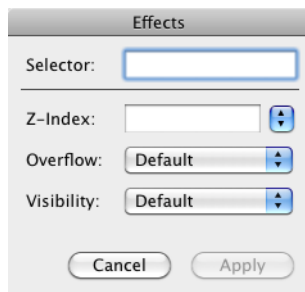
The Size & Constraints dialog allows you to edit the size and constraint properties of a selector.



The Size & Constraints dialog box has a title bar labeled "Size & Constraints". It contains a "Selector:" text input field. Below it are two columns of controls. The left column has a width icon, a "Min:" label, and a "Max:" label, each with a text input field and a vertical spinner button. The right column has a height icon, a "Min:" label, and a "Max:" label, each with a text input field and a vertical spinner button. At the bottom are "Cancel" and "Apply" buttons.

Effects

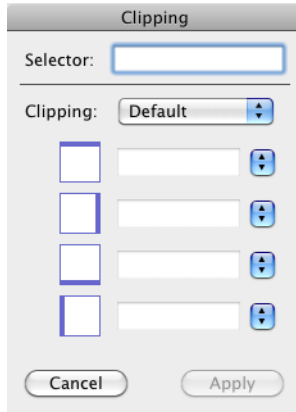
The Effects dialog allows you to edit the z-index, overflow, and visibility properties of a selector.



The Effects dialog box has a title bar labeled "Effects". It contains a "Selector:" text input field. Below it are three rows of controls: "Z-Index:" with a text input field and a vertical spinner button; "Overflow:" with a dropdown menu set to "Default"; and "Visibility:" with a dropdown menu set to "Default". At the bottom are "Cancel" and "Apply" buttons.

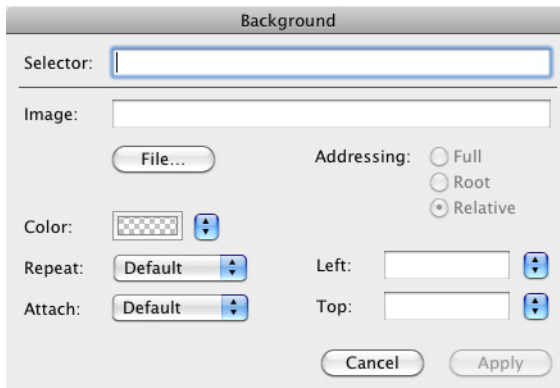
Clipping

The Clipping dialog allows you to edit the clipping properties of a selector.



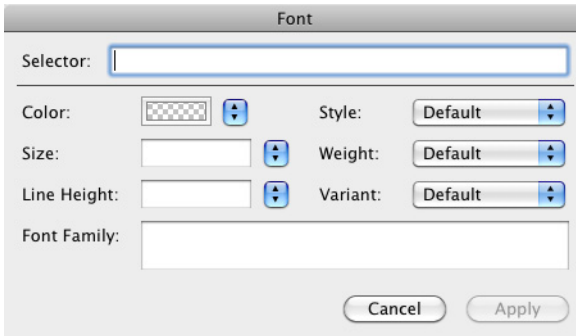
Background

The Background dialog allows you to edit background-image, background-color, background-repeat, background-position, and background-attachment properties. The Image field allows you to select an image file by clicking the File button, or by using drag and drop from the Finder.



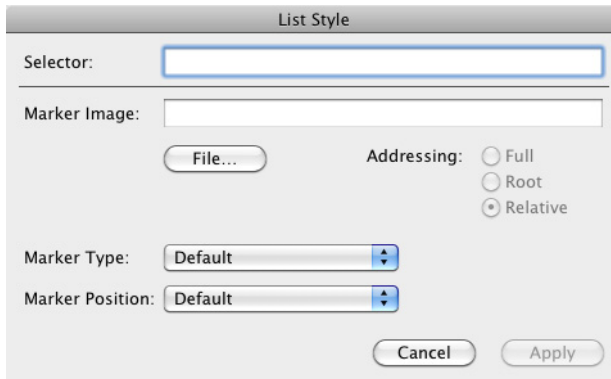
Font

The Font dialog allows you to edit values for the following font properties: color, font-size, line-height, font-family, font-style, font-weight, and font-variant. Note that BBEdit will parse the “font:” shortcut property, but never generates it; instead, BBEdit generates exploded values for font-style, font-variant, font-family, and font-weight.



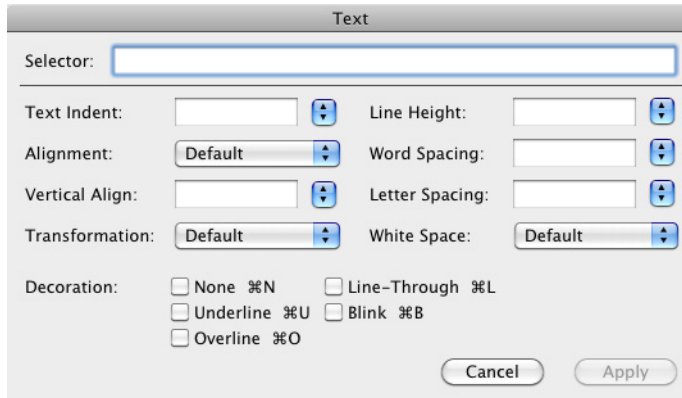
List

The List dialog allows you to edit the following list properties: marker image and addressing format, marker type, and marker position.



Text

The Text dialog allows you to edit the following text properties: text-decoration, text-indent, text-align, vertical-align, line-height, text-transform, word-spacing, letter-spacing, and white-space.



Body Properties

This command selects the BODY tag in the current document, and brings up the markup to edit it.

Head Elements

The commands in this submenu bring up the markup panel at an appropriate point within the current document's HEAD tag.

Base

The BASE tag determines the default location of documents referenced in the current document. The recommended attribute for this tag is 'href' whose value is the document or folder that all relative links within the document should be considered relative to. You can enter the Href by typing it into the supplied field or by clicking the Folder button to select a folder on your hard disk.

Link

The LINK tag tells the browser about a document related to the current document. The most common use for this tag is to point to an externally defined CSS stylesheet document. You can choose to indicate whether this link defines a REL (forward relation) or REV (reverse relation), the type of the relation (for example, a stylesheet), the media, the URL of the referenced file (including full, root, or relative addressing), and optional Type, Hreflang, Target, and Charset attributes.

Meta

META tags are used to define browser-specific or optional information that is not a part of the HTML specification. You can choose to create either a NAME or an HTTP-EQUIV variation of the META tag. The latter is frequently used for “client-pull” applications in browsers that support it, but more generically it makes most browsers behave as if the specified line was received as part of the HTTP protocol header. After choosing the type of META tag and the desired value of the NAME or HTTP-EQUIV field, enter a value for the tag’s CONTENT attribute and optionally its SCHEME attribute.

Script

This command begins a section of client-side script code (by default, JavaScript, although some browsers support other scripting languages). You can choose to execute a script contained in an external file by entering a URL in the Source field (click File to choose the file using an Open dialog). You can also enter values for the TYPE of script, the script LANGUAGE it is written in, and the character set or CHARSET of the script.

Noscript

This command begins a section of HTML to be displayed only if the web browser does not support client-side scripting; often used to provide alternate content following a <SCRIPT> block. A matching </NOSCRIPT> tag is also inserted.

Style

This command begins a stylesheet declaration.

Block Elements

This submenu lets you add HTML elements that behave as paragraphs or other types of text blocks. Since many block elements can be containers for other elements, most have an explicit or implicit ending tag (for example, </P> to close a paragraph) as well as an opening tag (for example, <P>). If text is selected when one of these commands is chosen, these opening and ending tags are placed before and after the selection.

Paragraph

This command begins a new paragraph element. You can specify alignment, ID, a class (for obtaining formatting cues from a stylesheet), and inline CSS style information.

Div

This command begins a new division. A division is a generic block of text containing one or more paragraphs (or other block elements) that all have some type of structural attribute in common. Use <DIV> when no predefined block type is appropriate. You can specify alignment, ID, a class (for obtaining formatting cues from a stylesheet), and inline CSS style information.

Horizontal Rule

This command inserts a <HR> tag. You can specify the alignment, the thickness (size) of the rule, its width, and whether it contains a three-dimensional “shade.”

Heading

This command inserts a heading of any level and allows you to specify the alignment of the heading.

H1 through H6

These commands insert a heading of the specified level. (The alignment attribute used will match the last one chosen when using the Heading tool.)

Address

This command inserts an <ADDRESS> block. The formatting of this element is browser-dependent but it is usually used to indicate that a block of text is a postal address.

Blockquote

This command inserts a block quote—that is, several lines of text that have been quoted from another document. (Most browsers display this as indentation, leading many authors to use this tag to indent a section of text, although stylesheets are a more correct and flexible way to accomplish this.) You may optionally indicate the document being quoted, if it is available on the Web, using the Cite field.

Center

This command inserts a block centering tag. This tag, while permitted in current HTML specifications, is deprecated since it includes no information about the content being centered. It is generally considered better form to use stylesheets or P or DIV tag ALIGN attributes instead.

Deleted Text

This command inserts a block formatted to indicate that the enclosed text has been deleted (usually with a horizontal line through it—that is, “struck out”). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Inserted Text

This command inserts a block formatted to indicate that the enclosed text has been inserted (usually by underlining the text). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Noscript

This command begins a section of HTML to be displayed only if the Web browser does not support client-side scripting; it is often used to provide alternate content following a <SCRIPT> block. A matching </NOSCRIPT> tag is also inserted.

Preformatted

This command begins a section to be reproduced with line breaks as specified in the HTML document. (Normally, browsers convert carriage returns to white space for display, breaking lines only at <P> or
 tags.) Most browsers use a monospaced font for this type of block.

Lists

These commands add numbered or bulleted lists to your HTML documents. If text is selected, the selected text is converted to a list, with each line (terminated by a carriage return) becoming a list item.

List

This command inserts a list. You can choose the type of list (unordered, ordered, definition, menu, or directory) and also the type of marker for an unordered (“bulleted”) list. You can also suggest a compact display format for the list. When converting existing text to a list, you can choose to ignore blank lines in the text being converted, to mark up only list items (and not insert the list header), and whether to indent the list items. When converting text to a definition list, DT (term) and DD (definition) tags are applied to alternating lines in the selection.

Unordered/Ordered/Definition/Menu/Directory

These commands convert the selected text to the indicated type of list, or insert a new list (as with the List command) using the options set in the last List dialog displayed.

List Items

This command converts selected text to list items (one line becomes one item), or inserts an tag if no text is selected.

Tables

The commands on this submenu all have to do with building HTML tables. HTML tables are frequently used for layout purposes as well as for the display of tabular data, although strictly speaking their use for layout should be avoided as much as possible.

Table

This command inserts <TABLE> and </TABLE> tags around the selected text. You can also specify border, width, spacing, padding, frame, ruling, alignment, and background color.

Row

This command inserts <TR> and </TR> tags around the selected text (if any). You can specify the desired horizontal and vertical cell alignment and a row background color. If horizontal alignment to a specific character is specified, you can also indicate the character that determines alignment and the character offset to the first alignment character in the line.

TD, TH

These commands insert a table data cell element or a heading cell element, respectively. (Both have the same options, though most browsers render TH elements differently from TD elements.) You can specify the width and height of the cell, the number of rows or columns it should span, its vertical and horizontal alignment (including alignment to a character and the offset to the first such character), whether the text in the cell should be permitted to wrap, the background color of the cell, and the scope of the header information in this cell, if any. You can also specify the axes, an abbreviated version of the cell’s content, and which header cells contain information about the current cell. Many of the less familiar and infrequently used attributes have use in certain applications such as speech accessibility. To provide maximum accessibility for tabular data, we suggest you consult the appropriate HTML version specification.

Caption

This tag specifies a caption for a table. You may also optionally specify the caption's vertical alignment.

Colgroup, Col

These tags are used to define column and column groups. Browsers that understand HTML 4 tables can, for example, be told to format a number of columns the same way, or to place rules between column groups, using this construction. The contents of a column group may be one or more <COL> elements (or none at all, if the SPAN attribute is used). You can specify the span of the column or group, its desired width, and its vertical and horizontal alignment. Cells within this column group may inherit some or all of these attributes depending on the attributes of the individual <TD> or <TH> elements.

THead, TFoot, TBody

These tags define an optional table section element, which can be used independently of the <TH> tag. (The latter indicates that particular cells should be displayed in a heading "style", which is usually displayed by browsers as boldface.) <TH> may be used anywhere in a table that a "heading look" is desired. In contrast, these three related tags define the logical divisions of a table. Browsers might hold the table's header or footer fixed on the screen while scrolling a lengthy body up and down, for instance. All three tags allow you to select vertical and horizontal alignment, which may be inherited by cells inside the element depending on the attributes of <TR>, <TD>, and <TH> tags.

Convert to Table

This command provides a quick way to convert tab-delimited or comma-delimited lines of text to tables. You must specify the delimiter to be used (either tabs or commas), and you can optionally have the entire first row of the table or the first cell of each row converted to <TH> rather than <TD> elements. If One Cell Per Line is marked, each cell will be placed on its own line in the resulting HTML; otherwise cells will be placed on a single line.

Forms

This submenu contains commands that help you build HTML forms, which are used for accepting user data for processing by a client-side script or a server-side CGI program (or other server-side technology, such as Active Server Pages).

Form

This tag defines a form. The Method can be either GET (encoding the form data in the URL) or POST (sending the form data separately after the HTTP transaction header). The Action should be the URL of the CGI program (or other server-side script, such as ASP). Enctype and Accept-Charset define the encoding type and character set for the transaction (usually, you will not need to use these fields). Use the On Submit and On Reset fields to enter the names of JavaScript handlers to be used for the Submit and Reset buttons, respectively. The Target field sets the frame to be used for the page returned by the CGI.

Button

This tag creates a form button. Choose a type (Submit, Reset, or Button), specify a name and value for the form element, and set optional attributes such as Disabled, Tab Index (the order in which the button will be reached by the Tab key), and Access Key (the key the user can press to activate the button in the browser). (The latter two options are HTML 4 features and may not work on all popular browsers.) You can also enter the names of JavaScript onFocus and onBlur handlers for the button.

Field Set, Legend

In HTML 4, you can group your form's fields and other controls into sets of related fields by using the FIELDSET container. Within the FIELDSET container, the LEGEND tag is used to define a title for the field set. Browsers differ in how they represent field sets visually, but some browsers may draw a rectangle around the related controls as in dialog boxes. In this case the Align attribute of the LEGEND tag can be used to set the alignment of the legend relative to the visual representation of the field set. (Browsers that do not support these tags will ignore them, and the contents of the LEGEND container will be displayed as any other text.)

Input

This tag defines an input field, which can be a text or password input, various types of buttons, and even files, images or hidden fields. Specify the name and the default value of these fields, and, if applicable, their size, maximum length, tab index, access key, and disabled or read-only attributes. (Disabled, Read Only, Tab Index, and Access Key are HTML 4 features and may not be supported by all popular browsers.) You may also specify handlers for the JavaScript onFocus, onBlur, onSelect, and onChange handlers.

Label

HTML 4 allows you to specify that text next to a control is a Label, and in browsers that understand the tag, clicking the label associated with a button activates the corresponding control. BBEdit lets you create a <LABEL> tag, specifying the name of the control it should be associated with, an optional keyboard equivalent to activate the control, and onFocus and onBlur JavaScript handlers.

Select

This tag defines a scrolling list or popup menu. Enter the name of the control, the number of items to display (leave the size blank for a popup menu rather than a scrolling list), and whether the list allows multiple items to be selected. Optionally mark the control as disabled and specify onFocus, onBlur, and onChange handlers.

Option Group

Using the <OPTIONGROUP> tag, you can create submenus in popup menus in browsers that support them. All <OPTION> tags within an <OPTIONGROUP> container are displayed as items cascading from the specified submenu label. (In browsers that do not understand <OPTIONGROUP>, users will see a simple straight list of all defined options.)

Option

This tag defines an option in a popup menu or a scrolling list. Enter the desired label and value for the option, and mark the Selected checkbox to make the option the default or initial choice.

Text Area

This tag defines a scrolling text area field for entering large amounts of data. You can specify the name of the file, its size in rows and columns, and optional HTML 4-only attributes such as Disabled, Read Only, Access Key, and Tab Index. You can also specify script handlers for onFocus, onBlur, onSelect, and onChange events.

Inline Elements

Inline elements are HTML elements that can appear as part of a paragraph, such as anchors, images, applets, client-side scripts, image maps, and more.

Anchor

This command inserts an HTML anchor (<A>) tag. Anchors can either be hyperlinks or be used as the target of hyperlinks to provided multiple targets on a single page. The anchor must have an associated URL in the HREF field to be a link; it must have a name in the Name field to be a target. The Target field is used to specify which frame the linked page should appear in.

Image

This command inserts an tag to display an image. As with the anchor tag, you can select the Source by choosing a file or typing a URL.

After choosing an image file, you can specify alternate text (which will appear in browsers that do not support images or for users who are surfing with image-loading turned off), enter the Size of the image, select the amount of horizontal and vertical Space for wrapping around the image, and choose the thickness of the border and the image's alignment. (Image height and width should be specified whenever possible to speed layout of the page in the browser; BBEdit will enter these values for you automatically when you choose an image file.)

If you drag and drop an image file into an HTML document, BBEdit will automatically generate and insert an image tag at the drop point with the image's actual dimensions pre-filled.

Applet

This command inserts the <APPLET> tag for specifying a Java applet. You will need to specify the location the folder that contains your main Java class file (the codebase) as well as the name of the main class file. If the file is in a .ZIP or .JAR archive, you can specify its name here as well. If you will control the applet via a client-side script, enter a name for it. You should always enter the desired size for the applet's display area. You can also specify alignment and white space around the applet, along with ALT text to be displayed if the applet cannot be used.

Object

The <OBJECT> tag is a generic tag for including almost any type of data in a page, including images and Java applets. (It can also be used to insert ActiveX controls and data intended to be used by plug-ins.) However, this tag may not be supported in all popular browsers. For this reason we suggest using and <APPLET> for those types of objects and use <OBJECT> only for embedding other types of data, such as that used by plug-ins. For an example of this, see the Web Design Group's HTML Help reference page:

<http://www.htmlhelp.com/reference/html40/special/object.html>

The <OBJECT> tag, like the and <APPLET> tags, allows you to reserve screen space in the browser window, recommend an amount of white space between the object and surrounding text, align the object, set its border, specify alt text to be displayed if the object cannot be displayed, and so forth. You will also need to specify at least the codebase and class ID of the object for ActiveX controls, and fill in the Data field for embedded objects such as Shockwave animations which will be handled by plug-ins. The Standby field can be used to tell browsers a text message to be displayed while the object is loading. For more information on the <OBJECT> tag, consult the HTML 4 specification.

Param

To pass parameters to a Java applet, ActiveX control, or plug-in, the <PARAM> tag can be used between the <OBJECT> and </OBJECT> (or <APPLET> and </APPLET>) tags. Each parameter to be passed to the object requires a separate <PARAM> tag. You must specify the name and value of each parameter; the actual parameter names and values required will vary depending on the object being embedded.

Script

This tag begins a section of client-side script code (by default, JavaScript, although some browsers support other scripting languages). You can choose to execute a script contained in an external file by entering a URL in the Source field (click File to choose the file using an Open dialog). You can also enter values for the TYPE of script, the script LANGUAGE it is written in, and the character set or CHARSET of the script. Mark the DEFER checkbox to add a DEFER attribute.

Note You can choose which character sets appear in the Charset popup menu by using the Text Encodings preference panel.

Map

This tag embeds a client-side image map in the document. You must enter a name by which the map can be referenced in the Use Map attribute of the Image tag. Individual clickable areas within the image map are provided by the <AREA> tags inserted between the <MAP> and </MAP> tags.

Area

This tag defines a clickable area within a client-side image map. Each clickable area requires a separate <AREA> tag. You will need to specify the document to be loaded when the area is clicked (or mark the No HREF checkbox to cause clicks in the area to be ignored), along with its Target frame if the page is being used in a frameset. You can choose the desired map shape (rectangular, circular, polygonal, or the default URL) using the

Shape popup menu and enter the desired coordinates of the shape in a comma-separated list in the Coords field. (For rectangles this is in the order left, top, right, bottom; for circles it is in the order X, Y, radius. For polygons this should be a comma-separated list of coordinates in X, Y form.) You can also set the tab index of the field for keyboard control on browsers that support it. You may also wish to specify JavaScript onFocus and onBlur handlers.

Break

This command enters a line break tag,
, into the document. If multiple lines are selected, a line break tag will be inserted after each.

Font

This tag selects the font, size, and/or color for the selected text. This tag is deprecated and should generally not be used; stylesheets are a more flexible and more content-oriented way of achieving this end.

Base Font

This tag selects the default font, size, and/or color for the text in this document. Like , this tag is deprecated; it is considered better form to use stylesheets.

Bidirectional Override

This command inserts a <BDO> tag to note that the enclosed text is in a language that should be rendered in a different direction (either left-to-right or right-to-left) than the default text order for the document's primary language. You can specify the desired text order and the language, so that savvy browsers can switch fonts or script systems to display the text correctly.

Quotation

This command marks the selected text as a quotation. Use this only for short quotes within a paragraph; use <BLOCKQUOTE> for quotations consisting of a paragraph or more of text.

Span

This command marks the selection as belonging to a certain class of information—such as a book title—usually so that its text style can be retrieved from a stylesheet. (In contrast with <DIV>, which marks paragraph-level classes, marks character-level classes.) You will be prompted for an ID for this span, a class name (which should correspond to a stylesheet entry), and inline style information. All are optional.

The Span command can now create nested span elements. This means that in order to edit an existing span element (since they can be nested), you must place the insertion point within the open tag of the desired instance.

Subscript

This command marks the selected characters as a subscript (lowered below the baseline).

Superscript

This command marks the selected characters as a superscript (raised above the baseline).

Phrase Elements

Phrase elements are HTML tags that mark sentences or phrases within a block element (such as a paragraph) with certain content-related styles, such as emphasis, strong emphasis, citation, and so on. Indirectly this determines the displayed format of the enclosed text (although exactly what “emphasis” and so on mean is left up to the browser or the stylesheet).

Abbreviation

The enclosed text is an abbreviation.

Acronym

The enclosed text is an acronym.

Citation

The enclosed text is a citation of another document.

Computer Code

The enclosed text is computer source code.

Deleted Text

This command inserts a block formatted to indicate that the enclosed text has been deleted (usually with a horizontal line through it—that is, “struck out”). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Defined Term

The enclosed text is term defined in a clippings.

Emphasis

The text should be displayed with visual emphasis (most browsers interpret this as italic text).

Inserted Text

This command inserts a block formatted to indicate that the enclosed text has been inserted (usually by underlining the text). You may optionally specify a citation (indicating a reference to another file) and a date and time.

Input Text (Kbd)

The enclosed text is text to be entered on a computer keyboard (used in instructions).

Sample Output

The enclosed text is sample output from a computer program (used in instructions).

Strong Emphasis

The text should be displayed with strong emphasis (most browsers interpret this as boldface).

Variable

The text is a placeholder in an instruction or tutorial, and should be replaced with an actual value of the appropriate type before actually performing the indicated operation.

Font Style Elements

Like Phrase Elements, Font Style Elements mark relatively short pieces of text within a block element. However, they are concerned more with the appearance of the text than its structural function in the document.

Big

This displays the enclosed text in a larger font than usual.

Small

This displays the enclosed text in a smaller font than usual.

Bold

This displays the enclosed text in boldface type.

Italic

This displays the enclosed text in italic type.

Strike-Through

This displays the enclosed text in a strike-through style.

Teletype Text

This displays the enclosed text in a monospaced font, as on a computer terminal or teletype.

Underline

This displays the enclosed text in an underlined style.

Frames

The commands in the Frames submenu help you design documents that use frames. The first document loaded by the browser contains at least one `<FRAMESET>` tag and one or more `<FRAME>` tags, which specify the number and sizes of the desired browser window subdivisions and indicate the URLs of the files to be loaded into each.

Frame Set

This defines a frame set, a series of one or more frames. You indicate whether the frame set divides the browser window vertically (ROWS) or horizontally (COLS), and then indicate the size of each frame in a comma-separated list, using `*` to tell the browser to use whatever space is left over from the other specified frames.

Frame sets can be nested. For example, if you want to create a framed Web page with three rows, with the middle row divided into two independent columns, you would first define a frame set consisting of three rows. Instead of defining the second row with a `<FRAME>` tag, however, you would open another `<FRAMESET>` tag there, this time to specify the two columns for the middle frame (which would then be specified by `<FRAME>` tags).

Frame

This defines a frame in a frame set document. You will need to specify the URL of the file to be displayed in this frame (either using the button, or by drag and drop). If the frame will be targeted by links in another frame, you will also need to give the frame a name. You can optionally specify a long description for the frame, choose whether the frame can be scrolled, and indicate whether the user should be able to resize the frame. You can also set margins and borders for the frame. (Borders are the visible lines between frames. Margins determine how far each frame's content appears from its border or from the window edge.)

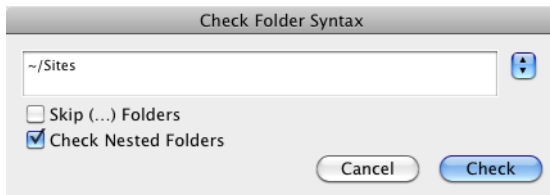
No Frames

HTML markup included between the `<NOFRAMES>` and `</NOFRAMES>` tags is displayed by browsers that do not support frames. This is where you should include a link to a non-frame (or text-only) version of the site. Although many current browsers support frames, some users do not like the feature and intentionally disable it in their browser.

Check

The Check submenu contains utilities for finding errors in your HTML markup and document links. You can run syntax or link checks on the current document, a specified folder, or the current site (as defined in the Sites panel of the Setup window). You can also perform a Balance Tags operation on the current document.

When you are checking a folder or a site using these tools, a dialog like the one below appears.



The popup menu to the right of the path box includes all of the web sites that you have specified in the Sites panel of the Setup window; choosing any of these selects its designated site root folder for checking. The Other entry on the popup menu displays an Open dialog, allowing you to navigate to and choose any other desired folder. (You can also drag a folder from the Finder directly into the path box.) The Skip (...) Folders checkbox specifies that subfolders whose names are enclosed in parentheses should not be checked; Check Nested Folders indicates that nested folders should be included in the check.

If a Check Syntax or Check Links operation generates any errors or warnings, BBEdit will display an error results browser listing. For more details on the error results browser format, please refer to Chapter 9.

Syntax

This command invokes BBEdit's syntax checker, which validates your HTML document to the specification defined in the `<!DOCTYPE>` prolog at the top of the document. Errors are displayed in an error results browser. Scroll through the list at the top of the window to see the errors that have been found; click to see the text that caused the error in the lower part of the window. Double-click an error message to open the file for editing.

Links

This command causes BBEdit to scan your document, or a folder of documents, looking for links and object references (such as images and Java applets) that cannot be resolved. Note that BBEdit only looks at pages on your site as defined in the Sites panel of the Setup window, not at any links that go offsite.

Update

The Update submenu contains commands for updating IMG tags, includes, and placeholders in the current document, the selected folder, or the current site (as selected in the Sites panel of the Setup window). BBEdit displays a results browser after the operation so you can see what was changed.

For more details on the results browser format that BBEdit displays, please refer to Chapter 9.

Choose the Document, Folder, or Site command from the Update submenu to update all includes and placeholders in the selected set of file(s). (Appendix C contains information regarding the use of placeholders.)

Choose the Document Images, Folder Images, or Site Images command from the Update submenu to update the HEIGHT and WIDTH attributes of image tags (and optionally to insert empty ALT attributes when missing) in the selected set of file(s).

When you are updating a folder or a site using either of these sets of tools, a dialog like the one below appears.



The popup menu to the right of the path box includes all of the web sites projects that you have added in the Sites panel of the Setup window; choosing any of these selects its designated site root folder for updating. The Other entry on the popup menu displays an Open dialog, allowing you to navigate to and choose any other desired folder. (You can also drag a folder from the Finder directly into the path box.) The Skip (...) Folders checkbox specifies that subfolders whose names are enclosed in parentheses should not be updated; Check Nested Folders indicates that nested folders should be included in the update.

Includes

The Includes submenu contains commands for inserting one-time include directives, “persistent” include directives, and placeholders. See Appendix C for more information.

Choosing Include or Persistent Include prompts you to choose a file using an Open dialog and then inserts the appropriate markup. Choosing Placeholders displays a scrolling list of available placeholders; selecting one and clicking Insert places it into the document.

Utilities

The Utilities submenu contains commands for automatically editing the current HTML document for ease of editing and for consistency.

Format

This command formats the current HTML document for easier editing. The appearance of the document in a browser is generally not affected (except in the case of Document Skeleton). You can choose from among the following:

- Pretty Print: A balanced format suitable for general use
- Strict Hierarchical: All nested HTML structures are indented according to their depth

Note This command may add white space, which can affect display in a browser

- Plain: Places each tag on a separate line with no indenting
- Compact: Absolute minimum white space
- Gentle Compact: A slightly more human-readable version of Compact
- Document Skeleton: A hierarchical view with all non-tag content removed
- Don't Reorganize: Allows normalizing of case, quote, and entity settings, as well as encoding entities within attributes, without otherwise changing the existing structure

You can also have the Format command operate on the whole document or only the selection, normalize the tags to uppercase or lowercase, normalize quote marks around attributes, and encode special characters, or entities, found in attributes.

If you choose the “Format...” command (with the ellipsis), BBEdit displays a dialog allowing you to choose the formatting options. If you choose the Format command (without the ellipsis), BBEdit uses the previous options.

Optimize

This command reformats the document to use the absolute minimum of characters while remaining syntactically valid. You will have difficulty editing your document in this format (in fact, if you do not have Soft Wrap turned on in the text options, you might think most of your document has vanished, because the command strips out all line breaks), but rest assured that your document will appear the same in your browser as it always has. Use one of the Format commands discussed above to put your page back into an editable format if you need to make changes. This command also applies the various Cleaner tools automatically.

Translate Text to HTML

This command allows you to translate plain text to HTML. The resulting sheet presents flexible options for converting paragraphs and translating extended characters to HTML entities. You can also choose to convert only the contents of the current selection or create a new document containing the results. (See “HTML Translation” later in this chapter for more information.)

Translate HTML to Text

This command allows you to translate HTML to plain text. The resulting sheet presents options for removing tags, converting tag-delimited paragraphs, and translating HTML entities to extended characters. You can also choose to convert only the contents of the current selection or create a new document containing the results. (See “HTML Translation” later in this chapter for more information.)

Remove Comments or Markup

This command removes all HTML comments or HTML tags, respectively, from the selection. Note that removing comments will not remove comment markers around client-side scripts like JavaScript, where they are required for proper functioning of the page on older browsers, but will remove the comment markers used by placeholders and indexes, making these items difficult to update in the future.

Comment, Uncomment

Comment adds comment markers (`<!-- -->`) around the selected text, regardless of whether it already contains tags or not, causing the browser to ignore that text. (It also converts any previously existing comment markers in the selection range to use `~` in place of `--`, to prevent older browsers from misinterpreting the comment.) Uncomment removes all comment tags from the selection and converts any modified comments back to real HTML comments.

Raise Tag Case/Lower Tag Case

These commands convert all HTML tags in the document to either upper case or lower case.

Tidy

The Tidy commands provide various options for reformatting and cleaning up your HTML documents. To perform these commands, BBEdit makes use of code from the HTML Tidy Library project.

<http://tidy.sourceforge.net/>

Clean Document

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit presents a sheet in which you may select various options for modifying the frontmost document’s tags. The available options are:

- **Remove Bogus Markup:** This option controls whether BBEdit should strip out surplus presentational tags and attributes, replacing them by style rules and structural markup as appropriate. (This option works well on the HTML saved by Microsoft Office products.)
- **Insert Blank Line before `
`:** This option controls whether BBEdit should output a line break before each `
` element.
- **Discard Empty Paragraphs:** This option controls whether BBEdit should discard empty paragraphs. If you do not set this option, BBEdit will replace each empty paragraph with a pair of `
` elements as HTML4 precludes empty paragraphs.

- **Discard Comments & Optional Tags:** This option controls whether BBEdit should remove all comments and optional tags.
- **Enclose Block Text:** This option controls whether BBEdit should insert a `<P>` element to enclose any text it finds in any element that allows mixed content for HTML transitional but not HTML strict.
- **Enclose Text:** This option controls whether BBEdit should enclose any text it finds in the body element within a `<P>` element. This is useful when you want to take existing HTML and use it with a style sheet.
- **Escape `<CDATA>`:** This option controls whether BBEdit should convert `<![CDATA[]]>` sections to normal text.
- **Give Tidy Credit:** This option controls whether BBEdit should add a meta element to the document head to indicate that the document has been tidied. (BBEdit will not add such a meta element if one is already present.)

Reflow Document

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit presents a sheet in which you may select various options for reformatting the current document's tag structure. The available options are:

- **Input is XML:** This option controls whether BBEdit should use the Tidy XML parser rather than the error-correcting HTML parser.
- **Indent Block-Level Tags:** This option controls whether BBEdit should indent block-level tags.
- **Indent Attributes:** This option controls whether BBEdit should begin each attribute on a new line.
- **Wrap ASP:** This option controls whether BBEdit should wrap text contained within ASP pseudo elements, which look like: `<% ... %>`.
- **Wrap Attributes:** This option controls whether BBEdit should wrap attribute values, for easier editing.
- **Wrap JSTE:** This option controls whether BBEdit should wrap text contained within JSTE pseudo elements, which look like: `<# ... #>`.
- **Wrap PHP:** This option controls whether BBEdit should line wrap text contained within PHP pseudo elements, which look like: `<?php ... ?>`.
- **Wrap Script Literals:** This option controls whether BBEdit should wrap string literals that appear in script attributes. BBEdit will wrap long script string literals by inserting a backslash character before the line break.
- **Wrap Sections:** This option controls whether BBEdit should line wrap text contained within `<![...]>` section tags.
- **Give Tidy Credit:** This option controls whether BBEdit should add a meta element to the document head to indicate that the document has been tidied. (BBEdit will not add such a meta element if one is already present.)

Convert to XHTML

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit converts the contents of the frontmost document to XHTML.

If no DOCTYPE or namespace was present, BBEdit will set them as appropriate. If a DOCTYPE or namespace was given, it will be checked for consistency with the content of the document, and in the event of an inconsistency, the corrected values will appear in the output. Entities will be written using the named form, and the original case of tags and attributes will be preserved.

Convert to XML

You can apply this command to the active document by choosing it from the Markup menu, or invoke it as a text factory action or within an AppleScript to process multiple documents. When you choose this command from the Tidy submenu, BBEdit converts the contents of the frontmost document to well-formed XML. Any entities not defined in XML 1.0 will be written as numeric entities to allow them to be parsed by a XML parser, and the original case of tags and attributes will be preserved.

Check Accessibility

This option instructs BBEdit to check the frontmost document's compliance to various WCAG accessibility guidelines, at various levels of strictness.

Preview

The Preview commands provide various options for previewing your HTML documents in a web browser.

Preview in BBEdit

Choosing this command will open a live content preview window within BBEdit which is linked to the document that was frontmost when you chose the command. You can go back from the preview window to its corresponding source document by clicking on the document icon button in the preview window, or by choosing the Show Document command from the Markup menu.

The preview window uses the same content rendering engine as Safari, and is updated whenever you modify the document. Closing the document will also close the preview window. (You can of course have multiple preview windows open on multiple documents.)

The preview window will not automatically display changes made in any related files, such as images or linked CSS files. However, you can use the Refresh BBEdit Preview command (see below) to update the preview window's display of both the source document and all related files.

If you choose this command when the current document is a Markdown source file, BBEdit will run that file through the Markdown script and generate a preview window which reflects how that file would be rendered in a web browser. The contents of the preview window will update as you edit the file.

Refresh BBEEdit Preview

This command works in conjunction with the Preview in BBEEdit command. How it behaves depends on the situation in which it's invoked:

- If the front window is a BBEEdit Preview window, its associated HTML file will be reloaded, together with any related files which were changed behind the preview window's back (e.g. images, linked CSS files).
- If the front window is a text document, and there exists a preview window previously created by a "Preview in BBEEdit" command on that document, then the associated preview window will be reloaded.
- If the front window is a text document, and any preview windows are open, the frontmost preview window will be reloaded, even if it does not necessarily belong to the front document.

BBEEdit Preview windows also contain a "Reload" button, which has the same effect as this command. Finally, whenever you save a CSS file, BBEEdit will automatically refresh all open BBEEdit Preview windows.

Preview in <Default Browser>

This command will display the name of the current default web browser, and choosing it will cause BBEEdit to display the frontmost document in that browser.

You can choose a browser from the Preview With submenu, or use Preview by itself to use the last chosen browser. You can also preview the page in all running browsers or in a text-only format.

Preview With

The Preview With command provides a submenu listing all installed web browsers and versions. The default browser, i.e. the one which BBEEdit will use unless you choose otherwise, has a checkmark listed next to its name, while the names of any browsers that are currently running will be underlined. You may add browsers which are not listed by using the Preview Helpers preference panel (see page 206).

You can preview the frontmost document in any available browser by choosing it from the menu. Making such a choice will also cause BBEEdit to use that browser as the default until another is selected.

Alternatively, you can choose the Preview as Text item to generate a text-only rendering within BBEEdit, or the All Running Browsers item to preview the current document in all running browsers.

The HTML Tools Palette

The main HTML Tools Palette is the place from which you will probably access the HTML Tools most frequently. You can invoke the HTML Tools palette at any time by selecting it from the Palettes submenu in the Window menu. BBEEdit remembers which palettes you had open when you quit, so if you open the HTML Tools palette, it will remain open until you close it again, even on subsequent uses of BBEEdit.

HTML Tools Palette Tips

A list of all the tools that are available on the HTML Tools palette appears below. In most cases, their behavior corresponds obviously with the tool descriptions in the previous section. In the few cases where there are significant differences, these are noted.

Some palette buttons are actually popup menus (indicated by a small downward-facing triangle on the right side), and clicking such a button will bring up a popup menu of options you can choose.

HTML Tools Palette

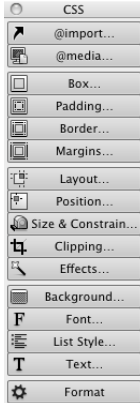
| Tool | Menu-Based Equivalent |
|--------------------|--------------------------------------------------|
| New Document | File > New > HTML Document |
| Edit Markup | Markup > Edit Markup |
| Close Current Tag | Markup > Close Current Tag |
| Balance Tags | Markup > Balance Tags |
| Doc Type | Markup > Document Type |
| Character Set | Markup > Character Set |
| Body Properties | Markup > Head Elements > Body Properties |
| Anchor | Markup > Inline Elements > Anchor |
| Image | Markup > Inline Elements > Image |
| Break | Markup > Inline Elements > Break |
| Paragraph | Markup > Block Elements > Paragraph |
| Div | Markup > Block Elements > Div |
| Heading | Markup > Block Elements submenu |
| List | Markup > Lists > List, Markup > Lists submenu |
| Table | Markup > Tables > Table, Markup > Tables submenu |
| Form | Markup > Forms submenu |
| Check Syntax | Markup > Check > Document Syntax |
| Check Links | Markup > Check > Document Links |
| Update | Markup > Update submenu |
| Preview in BBEEdit | Markup > Preview in BBEEdit |
| Preview | Markup > Preview in submenu |

Other Palettes

In addition to the main HTML Tools palette, BBEdit includes three other palettes that may be useful to HTML authors. These are the CSS palette, the Entities palette, and the Utilities palette, and you can display (or close) each of these palettes via the Palettes submenu in the Window menu.

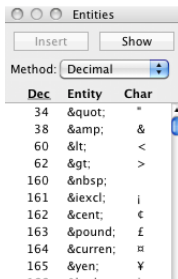
CSS palette

Click any of the buttons in this palette to invoke the corresponding CSS editing dialog.



HTML Entities palette

In HTML, extended characters must be encoded as entities, since different computers define the extended ASCII characters differently. The HTML Entities palette lists these entities.



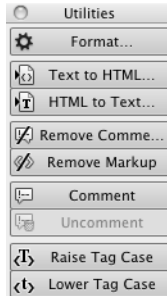
Entities can be inserted by name (“©” is the copyright symbol, ©) or number (“&169;” for ©) by choosing the desired method from the small popup menu at the top of the HTML Entities palette. (We suggest inserting entities by name, since they are more readable, unless browser compatibility requires use of the decimal versions.)

Double-click an entity name to insert it into the active document, or click once to select the desired entity and then click Insert.

The list of entities presented in the HTML Entities palette is now sortable by decimal value, name (case-insensitive, so “é” and “É” are grouped together), or character (sorted by the character position after all diacriticals have been stripped, so that all “as are grouped, and so on). Click on a column label to set the sort order accordingly. (The default is to sort by decimal value.)

Utilities palette

Click any of the buttons in this palette to invoke the corresponding command.



HTML Translation

BBEdit provides the Translate Text to HTML and Translate HTML to Text commands to help you quickly transform existing content. Here is more information about how the conversion options provided by these commands behave.

Convert Paragraphs

When converting text to HTML, BBEdit finds paragraphs in the same way as the Paragraph command, and then adds opening and closing paragraph tags around them.

When converting HTML to text, BBEdit will (if necessary) place hard line breaks around each paragraph in the resulting text.

HTML Entities

When this option is set, the Translate Text to HTML command will convert all extended characters in the current document into HTML entities, using either names or the code (in decimal or hexadecimal). You can specify whether the tool should ignore < and >. This is useful when translating text already marked up as HTML. You can also specify that all Unicode text should be converted to entities.

Remove Tags

When this option is set, the Translate HTML to Text command will remove all HTML tags and comments.

Templates

In addition to providing many facilities for creation and markup of individual documents, the HTML Tools also incorporate a Template facility, which can be used to quickly create (or revise) a set of HTML documents that share a common format, structure, or content. You can design a skeleton document, make a template from it, and then use that template over and over again to produce new pages ready to fill with content, or to insert into existing text documents to provide an uniform structure or appearance. Templates may also employ placeholders and include files (see Appendix C), adding even more power to this useful function.

Template Setup

A folder named HTML Templates, which contains some sample templates, is provided as part of the standard BBEdit installation. We suggest, in most cases, that you continue to use this as your templates folder. If you would prefer, however, you can set up your own template folder wherever you like. (See “Look for templates and include files in” on page 214.) If you plan to maintain multiple sets of templates for different projects, you may find this option very useful.

Using a Template

A template is a simple text file that contains boilerplate text or HTML content that will form the foundation for the document you are creating. Template files must have the file name suffix “.tmpl” in order to be recognized.

When creating a template file, you can convert or reuse an existing document, or you can write one from scratch. Simply rename the file by adding the suffix “.tmpl” to it, and then move or copy it into your active HTML Templates folder.

Templates are always invoked using the New Document tool. The Template option appears as a popup menu at the bottom left of the New Document dialog, and all template files in the Templates folder appear in this menu. (The “Default” setting is not a template per se, but rather a directive to create a blank HTML document framework containing whatever Title, Base, Meta, Link, SGML Prologue (and so on) values you specify. It is always available, regardless of the contents in your Templates folder.)

Once you have specified the appropriate settings and chosen Create, BBEdit will open a new Untitled window containing the full text of the selected template file. Note that the template file itself is never changed by this action; rather, its contents are simply copied into the new document.

Note Templates can make full use of placeholders and include files, which are fully documented in Appendix C.

Using Clippings

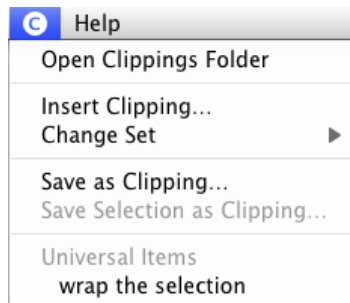
This chapter describes BBEdit's powerful Clippings command. Clippings provide you an easy way to store and enter any sort of frequently used text, including program code, HTML markup, or just about anything else. Clippings can be language-sensitive, and their optional ability to run scripts and insert the results, further extends their flexibility and usefulness.

In this chapter

| | |
|---------------------------------------------|-----|
| The Clippings Menu..... | 255 |
| The Clippings Palette..... | 256 |
| Managing Clipping Sets..... | 256 |
| Manually Sorting Clipping Sets..... | 257 |
| Creating and Editing Clippings..... | 257 |
| Inserting Clippings..... | 259 |
| Assigning Key Equivalents to Clippings..... | 260 |
| Clipping Substitution Placeholders..... | 261 |
| Using Scripts in Clippings..... | 266 |

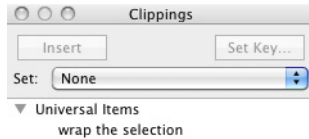
The Clippings Menu

The Clippings menu contains commands which you can use to insert and manage text clippings. The Clippings menu also presents the contents of the currently active clipping set, plus the contents of the Universal Items clipping set. You can choose any available clipping to insert its contents into the active document, or use the Insert Clipping command. (See “Inserting Clippings” on page 259.)



The Clippings Palette

Choosing the Clippings command from the Palettes submenu of the Window menu opens the Clippings palette, shown below. This window lists the contents of the active clipping set and the contents of the Universal Items clipping set. Names that are too long to fit within the width of the window are truncated with ellipses (...).



“Hovering” the mouse over such a truncated name displays a tool tip showing the full name. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Names that fit entirely within the window without truncation do not display a tool tip.

Managing Clipping Sets

Although BBEdit does not include any clipping sets by default, you can find a variety of customer-contributed clipping sets on our website:

http://www.barebones.com/support/bbedit/clippings_library.html

or create your own.

Installing New Clipping Sets

To install a new clipping set, just copy its folder into the “Clippings” folder within your local BBEdit application support folder.

The Clippings folder can contain aliases to other folders, either at its top level or nested. This allows you to place clipping sets in other locations (for example, if you are updating clippings from source control systems that make it inconvenient to use the main Clippings folder), or to combine clipping sets by creating a new top-level folder and then placing aliases inside it to other clipping sets.

Language Sensitivity of Clipping Sets

If you have selected the option to Change Clipping Set to Match Document's Language in the Languages preference panel, BBEdit will select a clipping set to use with the current document according to the following rules:

- If there is a clipping set whose name ends in the same suffix as the document, that set will be selected.
- If there is a clipping set with the same language mapping as the document (as determined from the mappings in the Languages preference panel), that set will be selected.
- If there are multiple clipping sets with the same language mapping (for example, "HTML Clippings.html" and "HTML Clippings.shtml"), the clipping set that appears first in the Set popup menu will be selected. (You can control the order in which clipping sets appear as described in the next section, "Manually Sorting clippings.")

These selection rules are intended to provide maximum flexibility while automatically doing the right thing as often as possible.

Manually Sorting Clipping Sets

By default, the Set popup menu displays clipping sets in alphabetical order. However, you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name: for example "(00)Web template" would sort before "(01)HTML Template". The first three characters of such names are not displayed in the menu. You can also insert a divider by including an empty folder whose name ends with the string "- ***". (You can use anything you want for the rest of the name, to make it appear where you want it in the menu.)

Creating and Editing Clippings

You can create a clipping by typing or pasting any desired text, or text and substitution placeholders, into a BBEdit document window and then choosing Save As Clipping from the Clippings menu. BBEdit will display a sheet in which you can name your clipping, and assign it to any existing or new clipping set.

The screenshot shows a dialog box for saving a clipping. It features a text input field for the name, currently containing "insert date & time". Below this are two radio buttons: "Save in set:" (which is selected) and "Create new set:". The "Save in set:" option is connected to a dropdown menu that currently displays "Universal Items". The "Create new set:" option is connected to an empty text input field. At the bottom right of the dialog are two buttons: "Cancel" and "Save".

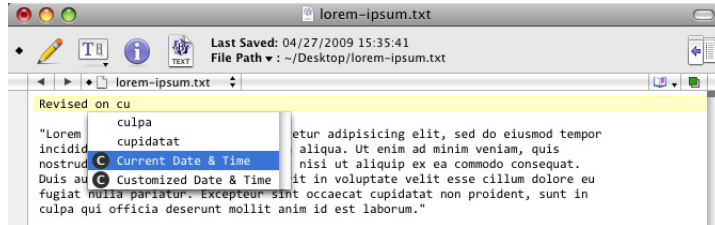
You can also create a clipping from the current selection by choosing Save Selection as Clipping from the Clippings menu, and using the clipping creation sheet as described above. Using this command does not affect the name or location of the document from which you create the clipping.

If you wish to further organize clippings within a set, choose Open Clippings Folder from the Clippings menu. You can create multiple levels of subfolders inside the Clippings folder, to better organize different types of content. The first level of such subfolders appear in the Set popup menu of the Clippings palette, allowing you to reveal only the group of clippings you wish to work with at a given time. (Any clippings not placed in a subfolder are always shown in the Clippings palette, as are the contents of the Universal Items set.)

You can edit a clipping by holding down the Option key and choosing the desired item from the Clippings menu. You can also edit a clipping by selecting it in the Clippings palette, then holding down the Option key so that the Insert button changes to Edit, and clicking Edit, or by Option-double-clicking the clipping directly.

Inserting Clippings

The quickest way to insert clippings is to use BBEdit's text completion feature. Just type a clipping item's name (or the beginning of a name), then either pause or invoke the Complete command by pressing F5 (or choosing it in the Edit menu) and BBEdit will display any matching clippings, as well as other available completions, in the completion popup.

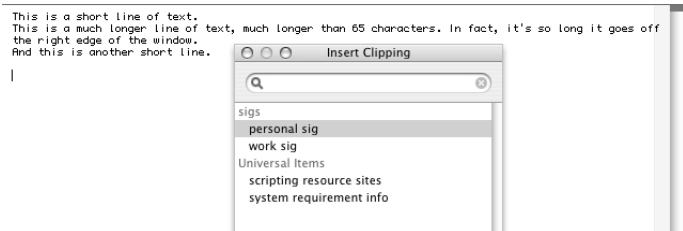


You can select a clipping in the popup by using the up- and down-arrow keys and type Return or Tab to insert that clipping, replacing the word or partial word.

You can also select and insert a clipping by choosing the Insert Clipping command from the Clippings menu.

When there is a word or partial word immediately to the left of the insertion point and you choose this command, BBEdit attempts to match that word to a clipping name in the current and universal sets

- If there is a single match, BBEdit replaces the word with the contents of the matched clipping.
- If there are multiple matches, BBEdit brings up the Insert Clipping popup (below) and lists all the matching clippings.



You can continue typing to further narrow the list and select a clipping, or use the up- and down-arrow keys to select a clipping and type Return to insert that clipping. You can also insert any listed clipping by double-clicking its name in the panel.

- If there is no match, BBEdit brings up the Insert Clipping panel with focus in the filter field, but does not filter the clipping list.

You can type in the filter field to narrow the list and select a clipping, or use the up- and down-arrow keys to select a clipping, and then type Return to insert the selected clipping. You can also insert any listed clipping by double-clicking it. BBEdit replaces the word with the contents of the clipping and closes the panel.

When there is no word or partial word to the left of the insertion point and you choose this command, BBEdit brings up the Insert Clipping panel with focus in the filter field and does not filter the clipping list. You can type in the filter field to narrow the list and select a clipping, or use the up- and down-arrow keys to select a clipping, then type Return to insert the selected clipping. You can also insert any listed clipping by double-clicking it. BBEdit inserts the contents of the clipping at the insertion point.

Note: If there is a word or partial word before the insertion point, BBEdit looks for a clipping of the form ‘clipping name **begins with** partial word’. However, when there is no word or partial word, BBEdit filters items in the Insert Clipping panel based on whether their names **contain** the currently-entered string. This makes it easier to filter clipping sets which contain many items having common prefixes (e.g. “apache_*” and “mysql_*” in the PHP clipping set).

You can use the Clippings menu to insert any listed clipping at the insertion point, or in place of the current selection, by choosing its name in the menu.

You can use the Clippings palette to insert any listed clipping by double-clicking its name in the window. Alternatively, you can click a clipping’s name to select it and then click the Insert button, or drag the clipping directly to the desired location in a document window.

When you insert a clipping, BBEdit always reads the clipping file from disk—if a clipping’s file is open and has unsaved changes, those changes will not be used.

Assigning Key Equivalents to Clippings

The Set Key button in the Clippings window lets you assign key equivalents for easy access to frequently used clippings. To assign a key to a clipping:

- 1 **Select the clipping in the Clippings window.**
- 2 **Click the Set Key button to display the Set Key sheet.**
- 3 **Type the key equivalent.**

You can use any combination of the Command, Shift, Option, and Control keys in the key equivalent, provided that it must include at least the Command or Control key to be valid. You can also use function keys, with or without additional modifiers.

- 4 **Click Set.**

Note If you try to assign a key equivalent that is already used elsewhere, BBEdit warns you that there is a conflict and asks you whether you want to reassign that key equivalent to the new item.

To remove a key equivalent from a clipping:

- 1 **Select the clipping in the Clippings window.**
- 2 **Click the Set Key button to display the Set Key sheet.**

Clipping Substitution Placeholders

When you insert a clipping containing a placeholder into an editing window, BBEdit replaces the placeholder with appropriate substitution text. This is similar to the operation of BBEdit's HTML Templates and Update features. The following table shows the placeholders you can use in a clipping:

| Placeholder | Replaced by... |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #BASENAME# | The name of the file stripped of its rightmost period-delimited portion. For example, if the file is named "test.html", the base name is "test", while if the file is named "test.foo.html", the base name is "test.foo". |
| #CLIPBOARD# | Contents of the current clipboard |
| #DATE# | Current date, formatted according to your Format settings in the International panel of the System Preferences |
| #DATETIMEXXX# | Inserts a localized, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below) |
| #DATETIME_GMT XXX# | Inserts the universal, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below) |
| #FILE# | File name of the document into which the item is inserted |
| #FILE_EXTENSIO N# | The filename extension for the file (determined as the rightmost period-delimited portion of the filename, without the period). For example, whether the file is named "test.html" or "test.foo.html", the filename extension is "html". |
| #FUNCTION# | If the item is being inserted into a source file, the name of the current function |
| #GMTIME YYY# | The current GMT time formatted according to the parameters YYY (see "Time Formats" below) |
| #INDENT# | When used in a clipping with multiple lines, causes every line after the first to be indented to the same whitespace level as the line in which the item was inserted (see the supplied WML clippings for examples) |
| #INLINE# | Strips all trailing vertical white space from the item before insertion (see also the Text Files preference panel) |

| Placeholder | Replaced by... |
|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #INSERTION# | Marks the place where BBEdit will place the insertion point after inserting the item; if multiple #INSERTION# placeholders are used, the second and subsequent occurrences are replaced with a placeholder "<##>", which can be used with Go to Next/Previous Placeholder in the Search menu |
| #LOCALTIME YYY# | The current local time formatted according to the parameters YYY (see "Time Formats" below) |
| #NAME# | The long name of the active user account. (There is no intrinsic placeholder for the short name, but you can use #inline##system whoami# to obtain it.) |
| #PLACEHOLDERS TART#label#PLAC EHOLDEREND# | Inserts a placeholder "hop" point which you can go to by using Next/Previous Placeholder. |
| #SCRIPT filename# | Result of running the specified AppleScript |
| #SELECT# | Selected text |
| #SELECTIONORIN SERTION# | If there was a selection when the clipping was expanded, it will be put at this position; otherwise, the insertion point will remain here. |
| #SELECTIONORPL ACEHOLDER label# | <p>If there is an active selection, the placeholder will be replaced with the selected text.</p> <p>If there is no selection, a placeholder named with the specified text ("label") will be inserted into the document.</p> <p>This placeholder is particularly useful when building clippings for insertion via both BBEdit's auto-completion mechanism and the clippings palette (or direct key equivalent).</p> |
| #SELSTART# and #SELEND# | Mark a range within the inserted material to be selected after the insertion. You can use multiple pairs of these placeholders within a single clipping. |
| #SYSTEM shell_script# | Given the full path to a shell command or script, BBEdit will run that command or script and insert the result. |
| #TIME# | Current time, formatted according to your Format settings in the International panel of the System Preferences |
| #UUID# | A 128-bit UUID (universally unique identifier), formed by combining a value unique to the computer on which it was generated (usually the Ethernet hardware address) with a value representing the number of 100-nanosecond intervals since October 15, 1582 |

Placeholders are not case-sensitive. If you want to include a literal placeholder in a clipping, escape the first # with a backslash, as in \#DATE#.

Selection and Insertion Placeholders

You can use multiple #SELSTART#/#SELEND# pairs together with any number of #INSERTION# placeholders.

Example:

Suppose you have defined the following clipping which contains an insertion placeholder:

```
typedef struct #SELECT#  
{  
#INSERTION#  
} #SELECT#, *#SELECT#Ptr, **#SELECT#Handle;
```

If the selected text in your editing window is “MyStruct” and you insert this clipping, BBEdit will insert the following in the editing window:

```
typedef struct MyStruct  
{  
|  
} MyStruct, * MyStruct Ptr, ** MyStruct Handle;
```

(where the vertical bar marks the position of the insertion point).

Example:

Suppose you have defined the following clipping which contains multiple pairs of selection placeholders:

```
MyFancyFunction(#selstart#arg1#selend#,  
#selstart#arg2#selend#, #selstart#arg3#selend#);
```

When you insert this clipping, BBEdit will place the following text in the editing window:

```
MyFancyFunction(arg1, <#arg2#>, <#arg3#>);
```

and the string “arg1” will be selected. You can then use the Go To Next/ Previous Placeholder commands from the Search menu to hop to the other arguments and enter the desired values.

Jump Placeholder Format

When you apply a clippings item that contains multiple #INSERTION# cookies, the second and subsequent cookies are replaced with special jump placeholder strings. These strings have the form “<#...#>” where the content “...” between the two # signs is either alphanumeric text, or empty.

You can also directly create and insert jump placeholders at any desired points within a document.

Older versions of BBEdit generated temporary placeholders of the form “#•#” for clippings containing multiple instances of #INSERTION#. If you have any existing clippings which directly employ the old placeholder format, you will need to modify them to use the supported placeholder format.

In addition to jump placeholders, you can also insert “optional” placeholders of the form <#?#>. When the “Go to Next Placeholder” command would select such a placeholder, BBEdit will place the insertion point at the specified position and remove the optional placeholder.

Optional-Argument Placeholder Format

Optional-argument placeholders have the form “<#* ... #>” where the content “...” between the two # signs is either alphanumeric text, or empty. These placeholders may be used to represent optional arguments generated during completion. You can select such placeholders in the usual fashion, but additionally, if you delete a selected optional-argument placeholder with the Backspace key, BBEdit will also delete any leading whitespace back to a preceding comma (if there is one).

Date Formats

The #DATETIME XXX# and #DATETIME XXX# placeholders allow you to insert the corresponding date and time values with flexible formatting.

In order to use these placeholders, you must substitute XXX with an ICU date/time format string. ICU is the mechanism used by Mac OS X for date formatting. For full details, please refer to the section “Date/Time Format Syntax” in the ICU documentation:

<http://icu.sourceforge.net/userguide/formatDateTime.html>

Examples:

`#DATETIME EEE, MMM d, yy 'at' h:mm a#`

produces:

Tue, Jul 3, 09 at 5:48 PM

`#DATETIME_GMT EEE, MMM d, yy 'at' h:mm a#`

produces:

Tue, Jul 3, 09 at 9:49 PM

`#DATETIME EEEE 'at' h 'o'clock' a#`

produces:

Tuesday at 5 o'clock PM

Time Formats

The `#GMTIME YYYY#` and `#LOCALTIME YYYY#` placeholders offer you the option to insert the specified time value with flexible formatting.

In order to use these placeholders, you must substitute YYYY with a time format using the same expansion options offered by the ‘strftime’ routine (see ‘man strftime’ for further details).

Examples:

`#LOCALTIME %r %z on %A#` produces: 06:50:13 PM -0400 on Monday

`#GMTIME %r %z#` produces: 10:50:13 PM +0000

Using Scripts in Clippings

The `#script filename#` placeholder is a powerful option which allows you to insert variable or conditional content from a clipping, by invoking any compiled AppleScript or Unix shell script.

The script itself can either be located in the same folder as the clipping that invokes it (in which case you need only specify its name, such as “MyDateScript”) or you can supply a full pathname to a script on any mounted volume in either POSIX or classic Mac format. An instance of a placeholder referencing the former would be

```
#script /Users/me/example/folder/Script.scpt#
```

and an instance referencing the latter would be:

```
#script Hard Drive:Users:me:example:folder:Script.scpt#
```

The script must return a text string (or a value that can be coerced to a string). This result string can itself contain additional clippings placeholders, which will be interpreted before the item is inserted in the current document.

WARNING

Note that this makes it possible for one script to invoke another. You must take care to not create a script execution loop, which could hang the application!

Scripting BBEdit

BBEdit offers access to nearly all of its features and commands via AppleScript. This chapter provides a brief overview of AppleScript, discusses BBEdit’s scripting model, and explains how you can use scripts within BBEdit.

An excellent way to learn how to script BBEdit is to look at the scripts others have written for it, or to turn on recording in your script editor while you perform actions in BBEdit. A number of example scripts are included in the standard distribution package. The BBEdit Talk discussion group is also a good resource for learning more about scripting.

<http://groups.google.com/group/bbedit/>

IMPORTANT

Regardless of whether you are new to scripting BBEdit or are familiar with scripting previous versions, we strongly recommend that you carefully review the sections “BBEdit and AppleScript” and “Working with Scripts” in this chapter.

In this chapter

| | |
|---------------------------------------------------------------------------------|-----|
| AppleScript Overview | 269 |
| <i>About AppleScript</i> – 270 | |
| <i>Scriptable Applications and Apple Events</i> – 270 | |
| <i>Reading an AppleScript Dictionary</i> – 271 | |
| <i>Recordable Applications</i> – 276 • <i>Saving Scripts</i> – 277 | |
| <i>Using Scripts with Applications</i> – 277 • <i>Scripting Resources</i> – 278 | |
| Using AppleScripts in BBEdit..... | 279 |
| <i>Recording Actions within BBEdit</i> – 279 • <i>The Scripts Menu</i> – 280 | |
| <i>The Scripts Palette</i> – 281 • <i>Organizing Scripts</i> – 281 | |
| <i>Attaching Scripts to Menu Items</i> – 282 | |
| <i>Attaching Scripts to Events</i> – 283 | |
| BBEdit’s Scripting Model | 287 |
| <i>Script Compatibility</i> – 287 • <i>Getting and Setting Properties</i> – 289 | |
| <i>Performing Actions</i> – 290 • <i>Common AppleScript Pitfalls</i> – 295 | |

AppleScript Overview

If you are familiar with AppleScript, you should have little difficulty scripting BBEdit. It has a robust and highly flexible object model. If you do not know much about scripting, though, read on for an introduction to the necessary concepts.

About AppleScript

AppleScript is an English-like language which you can use to write scripts that automate the actions of applications, and exchange data between applications. Although AppleScripts can manipulate applications' user interfaces by taking advantage of the system's GUI Scripting capability, this is not their primary function. Rather, scripts talk directly to an application's internals, bypassing its user interface and interacting directly with its data and capabilities.

If you want to insert some text into a document, emulating a user typing into an editing window is not the most efficient way of accomplishing this. With AppleScript, you just tell the application to insert the text directly. If you want the application to save the frontmost document, you need not mime choosing Save from the File menu, but rather just tell the application to save its frontmost document.

Note AppleScript is actually a specific language which resides atop the general Open Scripting Architecture (OSA) provided by Mac OS X. Although AppleScript is by far the most common OSA language, there are others, including a JavaScript variant. All OSA languages are capable of accomplishing similar things, although the actual commands used differ from one language to the next. In this chapter, we will focus exclusively on AppleScript, since it is the standard scripting language, but you should bear in mind that there are other options.

Scriptable Applications and Apple Events

Since AppleScripts must have direct access to an application's internal data structures, any application that will be used in an AppleScript must be designed to allow this access. We say such applications are *scriptable*. BBEdit is scriptable, as are many, many other programs. However, it is important to note that not *every* application is scriptable, and AppleScripts are not the best solution for automating applications that are not.

What goes on in an application that is scriptable? The foundation of AppleScript is something called the *Apple Event*. Macintosh applications are designed around an event loop; they go around in circles waiting for you, the esteemed user, to do something (choose a menu command, press some keys, and so on). These actions are passed to the application by the operating system in the form of an *event*. The application decodes the event to figure out what you did, and then performs an appropriate operation. After an event has been handled, the application goes back to waiting for another one. (At this point, the Mac OS may decide to give some time to another application on your computer.)

Apple Events are special events that applications send to each other, enabling a feature called *inter-application communication* (IAC). (It's a mouthful, but it just means applications can talk to each other.) Apple Events are also the way AppleScripts tell applications what to do, and which data to retrieve. So to be scriptable, an application must first support Apple Events.

Apple Events in their naked form are raw and cryptic things—bits of hieroglyphics only a programmer could love. So a scriptable application also has a *scripting dictionary*. The scripting dictionary tells any application that lets you write AppleScripts, such as the standard Script Editor, the English-like equivalent for each Apple Event and each event's parameters.

It is important to note that because Apple Events were originally designed to allow applications to communicate with each other, AppleScripts automatically inherit the ability to talk to more than one application. It is common in the publishing industry, for instance, to write scripts that obtain product information from a FileMaker Pro database and insert it into an InDesign file. This integration is one of the Macintosh's primary strengths.

You use AppleScript's *tell* verb to indicate which application you are talking to. If you are only sending one command, you can write it on one line, like this:

```
tell application "BBEdit" to count text documents
```

If you are sending several commands to the same application, it is more convenient to write it this way:

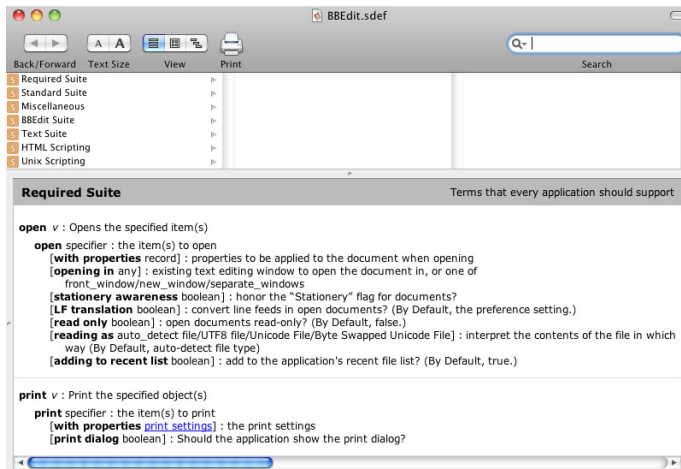
```
tell application "BBEdit"
    count text documents
    repeat with x from 1 to the result
        save text document x
    end repeat
end tell
```

The Script Editor automatically indents the lines inside the *tell* block for you so you can more easily follow the organization of the script.

Reading an AppleScript Dictionary

To display an application's AppleScript dictionary, you can simply drag that application onto the Script Editor icon, or use the Script Editor's Open Dictionary command. As we noted earlier, all scriptable applications include a dictionary that tells AppleScript how to convert English-like commands into the Apple Events actually expected by the application. The Script Editor uses this same information to display a sort of "vocabulary guide" that helps you write your scripts.

We will naturally use BBEdit's dictionary, shown below, to illustrate how to read a dictionary.



(You will probably want to make the window bigger if you have room on your screen.)

Down the left side is a list of every event and object supported by the application. An event is a verb—it tells the application what to do. A class is a noun: a piece of data, or a structured collection of data, inside the program. In BBEdit, for instance, classes are things like files, windows, the clipboard, browsers, and so on.

Suites

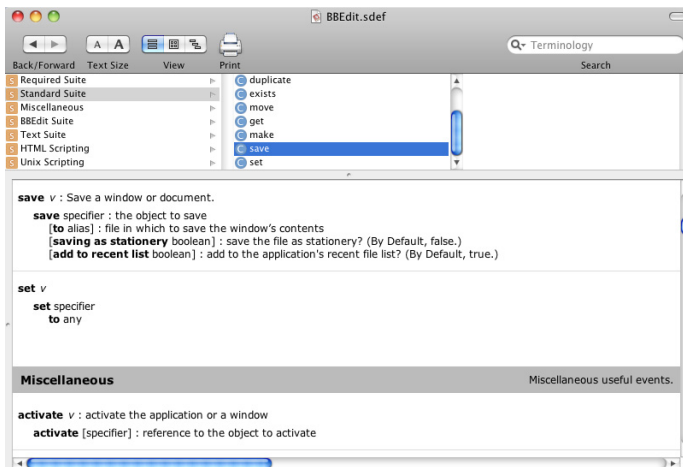
The first thing you will notice is that the events and classes are divided into *suites*. A suite is just a collection of related events and classes. Apple, for instance, has decreed that all applications should support particular events, which together are called the Required Suite. Another Apple-defined suite is the Standard Suite: if an application offers certain functions which Apple considers to be common, it should use these standard terms, so that scripters do not need to learn a new term for each application they work with. After that, it is a free-for-all—each developer is free to organize their events and classes however they think best.

In addition to the Required and Standard suites, BBEdit has a Miscellaneous suite, a BBEdit Suite, a Text suite, and an HTML Scripting suite. Additionally, if you have any scriptable plug-ins installed (as many of the supplied ones are), you will see additional suite entries for each such plug-in.

Within each suite, events—verbs—are displayed in normal text, while classes—nouns—are italicized. Most commands sent to BBEdit will start with one of the verbs. (In some cases, *get* might be implied.)

Events

Let's look more closely at one of the events—*Save* is a good one to start with. It is shown below.



The right side of the window shows the syntax of the selected event, as well as a brief description of its function. The boldface words are keywords; they must be included exactly as shown or the script will not compile. The normal text tells you what kind of information goes after each keyword. For example, after *save* you must give a reference; the italicized comment next to that line indicates that it is a reference to the window to be saved. In other words, some window object, which in BBEdit would be *window 1* for the frontmost window, or *window "Text File"* if you want to specify a window by name. (we will show you how to figure all that out in a moment—you have to look at the window class’s dictionary entry.)

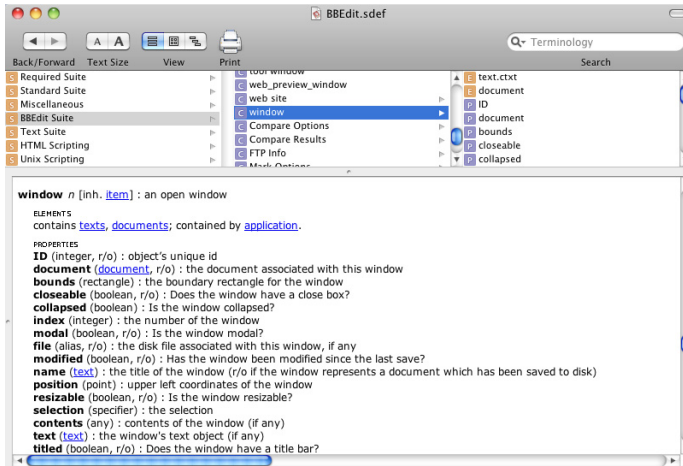
Anything in square brackets is optional. Most of the rest of the *save* event is optional, in fact. The basic event just saves the frontmost window to the same file from which it was opened. However, you can also optionally include the word *to* followed by a file reference. (You specify a file simply by using the word *file* followed by the path name of the file, as in *file "Hard Disk:Users:BBSW:Documents:My file"*.) If you specify a file to save the window to, the text will be saved into that file instead of the file it came from—like using Save As instead of Save.

The last three optional parts of the *save* event are denoted as boolean. That means they take either a true or a false value. In AppleScript, there are a couple of different ways to specify boolean values. You can write *saving as stationery true* to tell BBEdit to save the file as a stationery document. Or you can write *with saving as stationery*. You will notice that the last two parameters default to true if you do not specify them as false. To do that, you would use *add to recent list false* or *without add to recent list*. Whichever way you write it, you will notice that when you compile the script, AppleScript rewrites it using “with” or “without”. Since that is the syntax AppleScript seems to like best, that is probably the one you should get used to thinking in.

Let’s take a look at another one: the prosaic *get*. Select *get* from BBEdit’s dictionary listing and take a quick look at its class definition. You use *get* to retrieve information from an application. You must specify a reference to the object you want to retrieve, and you can specify a *coercion*—a condition that tells AppleScript to treat one type of data as if it were another—by adding the *as* clause. However, after that is the *Result:* line, which we have not seen before. This line tells you what type of value the command returns. (This value is placed in the AppleScript system variable called *the result*.) *Get* can retrieve any kind of object, so it can return anything, as indicated here. Other events might return a specific type of result, or none at all. (*Save* did not have a *Result:* line in its dictionary entry, which means it does not return a result.)

Classes and the Class Hierarchy

Let's look now at a typical class definition: *window* will do nicely. It is in the BBEdit Suite, toward the bottom.



All windows in BBEdit belong to this class. A class defines a particular kind of object; a particular example of an object belonging to the class is said to be an instance of that class, or just an object of that class. So here we are looking at the class itself; each individual window object has all these properties.

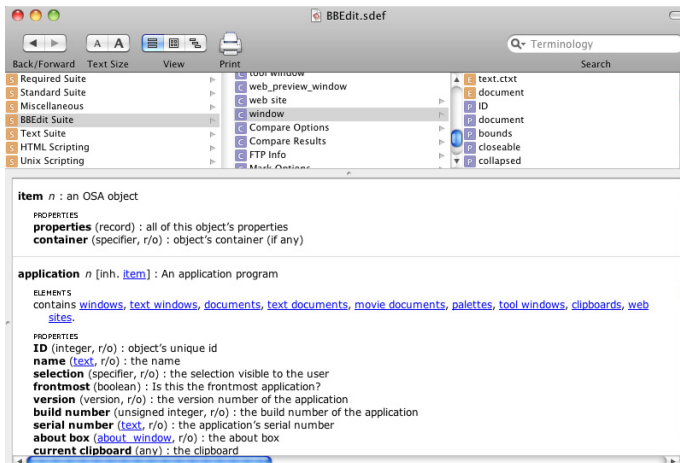
After a tag line that tells you about the class (“an open window”) comes the plural form. AppleScript lets you refer to windows either singly or as a group, so it needs to know what the plural of every term is. For example, try this little script:

```
tell application "BBEdit" to count windows
```

The result of this script is the total number of window objects currently displayed by BBEdit.

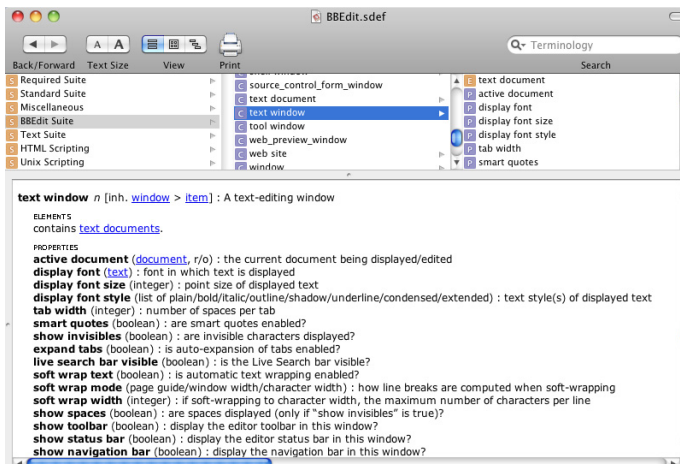
After the plural form comes a list of properties. Some objects do not have properties—for example, a string—but many applications do. An object's properties are merely a collection of data that describes that particular object. For example, as you look down the list of window properties, you will see that every window has a name, every window has a position, every window has bounds (the area of the screen it covers), and so on.

The first item on the list, though, is *<inheritance> item*. This tells you that a window is a kind of item, and that it therefore has all the properties of an item. Take a quick look at *item*'s class definition, shown below.



You will see three properties: *properties*, *ID*, and *container*. The first entry *properties* is a record containing all the object's properties. In other words, because a window is an item, it has, in addition to all its listed properties, another property which returns all the other properties as a record—a single piece of data that can be stored in a variable. Every class in BBEdit is part of a hierarchy with the *item* class at the top, so every object in BBEdit “inherits” the *properties* property. This catch-all property can be handy for making exact duplicates of objects, among other uses.

You may realize that BBEdit has several kinds of windows; you can see their classes listed in the dictionary: clipboard window, differences window, disk browser window, project window, text window, tool window, and the like. Let's look at *text window*:



You can see that a text window inherits all the properties of the *window* class. And, since the *window* class inherits all the properties of the *item* class, this means that the *text window* class also has the *properties* property defined by the *item* class.

To make explicit what you might have already gathered, classes in AppleScript form a hierarchy. That is, classes can be based on other classes. Such a class is called a *subclass*, and the class on which a subclass is based is referred to as its *parent class*. (In AppleScript, classes can only have one parent. Multiple inheritance is a feature found in more complex languages.)

The idea of a class hierarchy makes it easier for us to add new features to BBEdit, since when we want to create a new kind of window, half the work is already done. However, when scripting, you may need to flip back and forth between two or more class definitions to find all the properties of the object you are working with. (This is, technically speaking, a limitation of Apple's Script Editor. There is no reason the inherited properties could not automatically be included in a subclass listing by a smarter editor, for example, Script Debugger, which does this.)

Now that we have the class hierarchy under control, let's look at the properties themselves more closely. We will stick with the *text window* class at this point.

Properties of an object are referred to using the preposition *of*. For example, the following line of script returns the font of the frontmost text window.

```
tell application "BBEdit" to get display font of
text window 1
```

Note In this specific example, you can just write *get display font of window 1*. AppleScript will figure out that window 1 is more specifically a text window, and therefore has a *display font* property, even though the generic *window* class does not have any such property. All the properties of the object are available even if you did not use its specific class name. However, in most cases, you should specify exactly the object you want; this distinction is especially important when dealing with text documents (content) versus text windows (display elements).

You can set the properties using the *set* event, like so:

```
tell application "BBEdit" to set display font of text
window 1 to "Geneva"
```

Let's go back to the *window* class for a moment. Most of the properties of this class are marked with the abbreviation *[r/o]*. That stands for Read-Only. In other words, you can only *get* these properties, not *set* them.

Recordable Applications

Once an application accepts Apple Events, it actually makes a good deal of sense for an application to be designed in two parts: the user interface that you see, and the "engine" that does all the work. (An application designed this way is sometimes said to be *factored*.) The user interface then communicates with the engine via Apple Events.

The design of the Apple Event system makes it possible to "record" events into a script. This feature not only lets you automate frequently performed tasks with little hassle, it also can be an enormous aid in writing larger and more complicated scripts, because the application tells you what events and objects to use for the kind of task you record.

Because of the important recording functionality they enable, applications that have been factored and use Apple Events to let the two halves communicate are said to be *recordable*. It is important to note that not all scriptable applications are recordable.

Saving Scripts

Any AppleScript can be saved in what's called a *compiled script file*. A compiled script file contains the actual Apple Events; by generating these events when you save the file, the operating system does not have to convert your English-like commands into events each time you run the script, which means it loads faster. When double-clicked in the Finder, a compiled script file automatically opens in the Script Editor, where it can be run. A script can also be saved as a stand-alone application, or *applet*, in which case double-clicking the script's Finder icon automatically runs the script. Both types of files can be saved with or without the English-like *source code*; if you save it without the source code, other users you give the script to will not be able to make any changes to it (of course, you should also keep a copy of the script *with* the source for yourself).

Using Scripts with Applications

Although you can place a script applet in the global Scripts menu, or in any folder, and use it any time you need it, many applications (including BBEdit) provide a special menu that lets you launch compiled scripts intended specifically for use with that one application. Since you do not have to save them as applets, they take up less disk space and launch more quickly. They also show up only in the application you use them with, rather than cluttering your global Scripts menu.

Some applications go even further, allowing you to define scripts to be run when certain things happen in the program. For example, an application might let you define a script to be executed when the user chooses *any* menu item. The script might then perform some pre-processing, and then exit by telling the application whether to continue with the menu command or to cancel it. As a simple example, a script might check to see what printer is selected when the user chooses the Print command. If it is the expensive color dye-sublimation printer, on which printing a page costs several dollars, the script could remind the user of that fact and confirm their intention (through an alert) before continuing with the print operation.

An application that supports such a feature (or any method of integrating user-written scripts seamlessly into its user interface) is said to be *attachable*, because the scripts become “attached” to the features of the program. (BBEdit is now attachable; more details about using this feature are provided later in this chapter.)

Scripting Resources

Covering all the details you might need to write your own AppleScripts is not something we can reasonably do in this manual. AppleScript, despite its deceptively simple English-like syntax, is a sophisticated object-oriented language with many subtleties. For this reason, we suggest you consult supplemental documentation and resources if you are a beginning scripter.

A good place to start is with someone else's script: find a script that does *almost* what you want it to and repurpose it. Even if you cannot find a script that does anything close to what you want, reading others' scripts is a good way to learn how AppleScript “thinks” and how BBEdit's particular AppleScript implementation behaves.

In addition to the basic AppleScript documentation included with the system, you may find the following resources useful in your quest to understand scripting.

Books

AppleScript: The Definitive Guide (Second Edition), Matt Neuberg. O'Reilly and Associates, 2006. ISBN: 0-596-10211-9

Discussion Groups

AppleScript Users

<http://www.lists.apple.com/applescript-users.html>

Official mailing list run by Apple for AppleScript users.

BBEdit Talk

<http://groups.google.com/group/bbedit>

The BBEdit Talk discussion group is an excellent place to ask BBEdit-specific scripting questions.

Mac Scripting

<http://listserv.dartmouth.edu/scripts/wa.exe?A0=MACSCRIPT>

Unofficial list covers AppleScript and other Macintosh scripting languages, with occasional forays into peripheral topics.

Web Sites

AppleScript: The Language of Automation

<http://www.macosxautomation.com/applescript/>

An excellent starting point.

AppleScript — Apple Developer Connection

<http://developer.apple.com/AppleScript/>

Detailed information for developers and advanced users.

MacScripter.Net

<http://macscripter.net/>

A good selection of AppleScript-related news and topics, including the “AppleScript FAQ” and discussion forums.

ScriptWeb

<http://www.scriptweb.com/>

This site covers all scripting languages, not just AppleScript. Also, it has an extensive directory of scripting additions.

Software

Script Debugger

<http://www.latenightsw.com/>

Despite its name, Script Debugger is more than a debugger; it is actually an enhanced replacement for Apple's Script Editor, featuring variable monitoring, step/trace debugging, an object browser for an application's objects, and much more.

Using AppleScripts in BBEdit

BBEdit has been scriptable for years, and we have continually worked to refine its level of scripting support. In addition to providing extensive script access to its commands and data, BBEdit is both attachable *and* recordable.

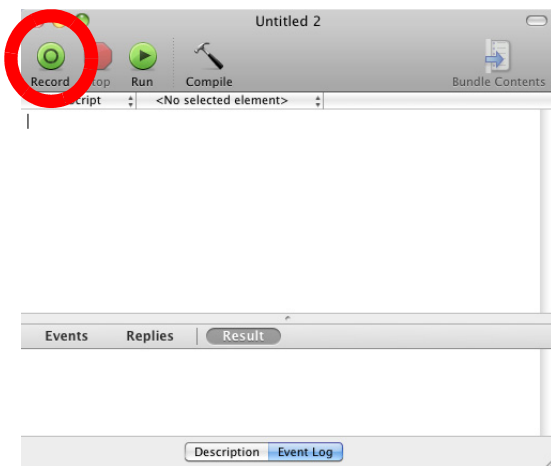
This section describes how you can create and employ AppleScripts within BBEdit via recording and BBEdit's various scripting facilities, while the following section covers BBEdit's scripting commands and other issues related to preparing scripts for use.

Recording Actions within BBEdit

Any language is easier to read than to write, easier to understand than to speak. AppleScript is no different. That's because, even though all the commands it uses are English words arranged in ways that more or less make grammatical sense, you still have to know (or find out from the application's dictionary) exactly which words to use, and what order they should go in. But it is easy to get started making scripts by recording them.

First, launch both BBEdit and the Script Editor.

When you launch the Script Editor, a new, blank script window appears. Click the Record button, circled in the illustration below.



Now switch to BBEdit and perform your task. Remember that the Script Editor is recording *everything* you do in every recordable application you are running, not just BBEdit. If you do something in the Finder, for instance, that will get recorded too. Since almost everything you do is recorded, remember that if you make an error, and then Undo it, your recorded script will faithfully make the same mistake and undo it when you run it later. It will be possible to fix minor errors later, but things always go more smoothly if you do not make any mistakes, so take your time and try to do it right the first time.

Now switch back to the Script Editor and click the Stop button. After a brief pause, your script is compiled and ready for use. Try clicking the Run button to see it work. (It might not work correctly. If you recorded a search and replace operation changing every “cat” to “dog”, you already changed the document while recording the script, and of course the script will not do anything when you run it.)

Finally, save the script in the BBEdit Scripts folder so that it shows up in BBEdit’s script menu. Choose Save As from the File menu, and then use the Script Editor’s Save dialog to put the script in your BBEdit Scripts folder. Now try selecting it from the script menu in BBEdit.



The Scripts Menu

The Scripts menu (left) in BBEdit’s menu bar contains several commands. It also lists all AppleScripts (as well as Automator actions, text factories, and Unix scripts) present in the Scripts folder within BBEdit’s application support folder, providing a quick way to access frequently used scripts. You can place scripts within subfolders (up to 4 levels deep) of the Scripts folder to organize them.

Note AppleScripts written for use in as BBEdit filters or scripts should be saved as compiled (data fork) script files, not script applications.

In addition to the list of available scripts, the Scripts menu provides the following commands.

Open Script Editor

Choose this item to switch to the system's default AppleScript editor. If the script editor is not running, BBEdit launches it.

Open Scripting Dictionary

Choose this item to switch to your preferred AppleScript editor and open BBEdit's scripting dictionary for viewing. If the script editor is not running, BBEdit launches it.

Open Scripts Folder

Choose this item to open the Scripts folder which is located within BBEdit's application support folder. (See "Scripts" on page 32.)

Start Recording

Select this item to record all available actions that you perform within BBEdit (or any other recordable applications which you switch to). When this command is active, the menu item will change to Stop Recording, and a tape icon will flash over the Apple menu. When you choose Stop Recording, BBEdit will display a Save dialog which allows you to save a script file containing the recorded actions.

Running and Editing Scripts

Choose the item corresponding to any script to run that script. Hold down the Option key when choosing a script item to have BBEdit open the script for editing in your preferred script editor, or hold down the Shift key when choosing a script item to have BBEdit reveal the script file in the Finder. If you choose a folder node rather than a script item, BBEdit will open the corresponding folder in the Finder.

The Scripts Palette

The Scripts command, located in the Palettes submenu of the Window menu, opens a palette listing all available scripts. Names that are too long to fit within the width of the window are truncated with ellipses (...).

"Hovering" the mouse over such a truncated name displays a tool tip showing the full name. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Names that fit entirely within the window without truncation do not display a tool tip.

Organizing Scripts

Items in the Scripts menu or Scripts window are displayed in alphabetical order by default, but you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example "(00)Save All" would sort before "(01)Close All.") For names of this form, the first three characters are not displayed in the window. You can also insert a divider by including an empty folder whose name ends with the string "- ***". (The folder can be named anything, so it sorts where you want it.) These conventions are the same as those used by the utilities FinderPop and OtherMenu.

Attaching Scripts to Menu Items

BBEdit lets you attach scripts to menu items. By this, we mean that you can write scripts that BBEdition automatically calls before or after performing a menu command. For example, if you want BBEdition's Open from FTP/SFTP Server command to launch your favorite FTP client, you can simply attach a script to that menu item. Scripts can return a value that tells BBEdition whether to continue with the command that was selected, or to cancel the operation (in which case only the script is executed).

Scripts attached to BBEdition menu items must be stored in the Menu Scripts folder of BBEdition's application support folder. These files should be compiled scripts, not script applications. Scripts are named to indicate which menu item they go with: first the name of the menu (or the submenu) upon which the item is immediately located, then a bullet "•" (Option-8) character, then the name of the menu item. For example, to attach a script to the Open from FTP/SFTP Server menu item, you would name it "File•Open from FTP/SFTP Server", while to attach a script to the New Document menu item, you would name it "New•Text Document".

Some of BBEdition's menus have icons rather than names. BBEdition uses the following names for its icon menus: "#!" [the 'Shebang' menu], "CVS", "Compiler", and "Scripts". Furthermore, the New With Stationery submenu is named "Stationery" for purposes of attachability.

When you choose a menu command which has an attached script, BBEdition will pass the menu name and command (item) name to the script's MenuSelect handler, if it has one. If the script contains no MenuSelect handler, BBEdition executes the script's run handler.

The script's MenuSelect handler can tell BBEdition to skip performing the chosen command by returning "true", or have it continue on and perform the command by returning "false". If MenuSelect returns "false", BBEdition will call the script's PostMenuSelect handler, if it has one, after it performs the menu command.

Here is a simple example, which adds a confirmation dialog to the Save command (addressed as “File•Save”). Note that we test the menu and item names to make sure the script is attached to the Save command—if it is attached to some other command, it does nothing.

```
on menuselect(menuName, itemName)
    if menuName = "File" and itemName = "Save" then
        set weHandledCommand to true
        display dialog "Are you sure you want to save?" ~
            buttons {"No", "Save"} default button 2
        if button returned of the result is "Save" then
            -- the application should do its work
            set weHandledCommand to false
        else
            -- we handled the command, app does no work,
            -- postmenuselect doesn't get called
            display dialog "The document was not saved." ~
                buttons {"OK"} default button 1
        end if
        return weHandledCommand
    end if
end menuselect

on postmenuselect(menuName, itemName)
    -- this is called after the application has processed
    -- the command
    display dialog "The document was saved." ~
        buttons {"OK"} default button 1
end postmenuselect
```

Attaching Scripts to Events

IMPORTANT

BBEdit now offers enhanced script attachability: in addition to adding scripts to menu commands, you can now attach scripts to certain application and document events.

To access these events, your attachment scripts must contain function names which correspond to the names of the events' attachment points. Except when otherwise noted, all of the following considerations apply:

- Every function takes a single argument which is a reference to the object in question: the application for application entry points, or the document being opened/closed/saved/etc for document entry points.
- Any function associated with an attachment point whose name contains ‘should’ must return a Boolean result: ‘true’ or ‘false’. If it returns ‘true’, the operation will continue. If it returns ‘false’ or throws an error (see below) then the operation will be cancelled. So, for example, ‘applicationShouldQuit’ returning ‘true’ will allow the application to quit; returning ‘false’ will not.
- If an attachment script causes a scripting error and does not handle it within the script itself, BBEEdit will report the error. In the case of functions which are used to allow a ‘should’ action, this will prevent the action from occurring.

Here are the available attachment points:

Application attachment points

- *applicationDidFinishLaunching*: called when the application has completed startup.
- *applicationShouldQuit*: called when you choose the Quit (or the application receives a ‘quit’ event for any other reason).
- *applicationDidQuit*: called when the application has finished shutting down and is about to exit.

Document attachment points

- *documentDidOpen*: called when a document has been opened and is ready for use. (Since BBEdit supports multiple types of documents, your script should allow for the argument to be a document of any type.)
- *documentShouldClose*: called when the application is preparing to close a document.
- *documentDidClose*: called when the application has closed a document.
- *documentShouldSave*: called when the application is trying to determine whether a given document should be saved.
- *documentWillSave*: called when the application is about to begin saving a document. (note that this will only be called after a successful return from a ‘documentShouldSave’.
- *documentDidSave*: called after a document has been saved successfully.
- *documentWillUnlock*: called when BBEdit is going to make a document writeable. (For example, when you click the pencil to unlock a document)
- *documentDidUnlock*: called when BBEdit has successfully made a document writeable.
- *documentWillLock*: called when BBEdit is going to make a document read-only.
- *documentDidLock*: called when BBEdit has successfully made a document read-only.

Using Attachment Scripts

Scripts attached to events must be stored in the “Attachment Scripts” folder of BBEdit’s application support folder (see page 30).

You can write one script to handle each attachment point, or one script to handle the attachment points for an entire class of objects, or one script to handle all of the attachment points for the entire application.

You can also mix and match scripts to meet specialized needs: for instance, by using one script to implement a particular attachment point for documents, and a second script to handle the remaining attachment points.

BBEdit associates scripts to attachment points by means of the script’s file name. There are three ways to specify a script’s role:

1 **<ObjectClass>.<entryPoint>**

2 <ObjectClass>

3 <ApplicationName>

The first form is the most specific: the ‘ObjectClass’ may be either “Application” or “Document”, while the ‘entryPoint’ is one of the attachment points described above appropriate to that object class.

For example, a script which implemented only the *documentDidSave* attachment point should have the name “Document.documentDidSave.sctpt” and contain a subroutine named ‘documentDidSave’, thus:

```
on documentDidSave
    -- do something useful and appropriate
end documentDidSave
```

Note Adding the filename suffix ‘.sctpt’ is not mandatory, but you should follow the current system conventions suggested when creating scripts with the AppleScript Editor (or any other script editor such as Script Debugger).

The second form allows you to implement all of the attachment points for a single object class in a single script file, if desired.

For example, you could create a script named “Application.sctpt” containing subroutines for as many of the application attachment points as you wish:

```
on applicationDidFinishLaunching
    -- do something relevant
end applicationDidFinishLaunching
on applicationShouldQuit
    -- hello world
    return (current date as string contains "day")
end applicationShouldQuit
```

Likewise, to implement all of the attachment points for the Document class, you could create a script named “Document.sctpt”, and put subroutines in it for the document attachment points:

```
on documentDidSave
    -- do something relevant
end documentDidSave
...
on documentWillClose
    ...
end documentWillClose
```

The third form allows you to write a single all-encompassing script which contains subroutines for all of the attachment points in the application. To do this, name the script “BBEdit.scpt” and include whatever subroutines you wish to implement. For example:

```
on applicationShouldQuit
    -- hello world
    return (current date as string contains "day")
end applicationShouldQuit

on documentWillClose
    ...
end documentWillClose
```

When figuring out which script to run, BBEdit will first look for a script whose name exactly matches the attachment point, e.g. “Document.documentShouldSave.scpt”. If there is no such script, BBEdit will then look for a script whose name matches the object class at the attachment point, e.g. “Document.scpt”. Finally, if there are no scripts with either an exact or a class match, BBEdit will look for an application-wide script: “BBEdit.scpt”.

Note You do **not** have to implement attachment subroutines for all attachment points, or for all classes—only the ones you need. If there is no attachment script or subroutine, BBEdit proceeds normally.

Filtering Text with AppleScripts

The Text Filters folder in BBEdit’s application support folder contains executable items, such as compiled AppleScripts, Automator workflows, and Unix filters, which you may apply to the active document via the Apply Text Filter command in the Text menu.

When you apply such an item, BBEdit will pass either the selected text (or the contents of the active document, if there is no selection) as a reference to a ‘RunFromBBEdit’ entry point within your AppleScript, and your script should return a string which BBEdit will use to replace the selected text (or the contents of the document). If your script does not contain a ‘RunFromBBEdit’ entry point, BBEdit will call its run handler, again passing a reference to the current selection range.

BBEdit's Scripting Model

This section provides a high-level overview of BBEdition's scripting model that will, where appropriate, contrast the current scripting framework against older versions of BBEdition, and suggest how you can modify your existing scripts for compatibility.

IMPORTANT

Because BBEdition's scripting dictionary changes whenever we add features, it should be considered the definitive reference in any situation where it and this document differ. We have found Script Debugger from Late Night Software to be an excellent tool for browsing and navigating BBEdition's scripting dictionary, as well as for preparing and testing scripts.

<http://www.latenightsw.com/>

Script Compatibility

BBEdit's scripting model has changed significantly from that used in versions prior to 6.0. Consequently, scripts prepared for such versions may need to be revised in order to work properly. Further, since BBEdition now allows multiple documents to be open within a single text window, you may need to revise other existing scripts.

Distinguishing Between Script Elements

Because different applications handle different types of data, you should be aware that the actual data, or the interface items, referred to by a particular name may not be consistent from application to application. The following sections describe how several common elements are handled in BBEdition.

Applying Commands to Text

Since BBEdition supports opening multiple documents within a single text window, all scripting commands which operate on text must specifically target the text contents of a window, or a document within that window, rather than the window itself.

For example, you may use:

```
count lines of text of window 1
```

or:

```
count lines of active document of window 1
```

but not:

```
count lines of window 1
```

Documents vs. Windows

In old versions of BBEdition, the object classes *document* and *window* could be used interchangeably, and generally had the same properties listed in the scripting dictionary. This is no longer the case.

The class *window* now corresponds to a window (of any type—text or otherwise) on screen, and thus the properties of the *window* class now refer strictly to properties of a window on screen. If a document is associated with a window, the document is accessed as the *document* property of the window:

```
document of text window 1
```

The class *document* refers to a document, and as with a window, the document's properties pertain strictly to the condition of a document (that is, something that can be saved to disk and opened later). Note that this does not mean a document must be saved to a file, only that it could be.

As a rule, documents and windows are associated with each other, but it is important to remember that there is not a one-to-one correspondence between windows and documents. For example, the About box is a window which has no document associated with it. Furthermore, in current versions of the application, there is no such thing as a document with no associated window.

Here is a general overview of the object classes used in BBEdit:

Classes of Windows

- *window*: the basic window class contains properties that can be fetched and set for any window on screen: position, size, and so forth.
- *palette*: the palette class refers to windows that float above all others on the screen; the HTML tools palette, scripts list, and so on.
- *text window*: the text window class provides properties which are specific to text-editing windows as on-screen entities. These properties pertain mostly to the display of text in the window: *show_invisibles*, *auto_indent*, and so on. In addition to the text-editing-specific properties, the basic window properties are also accessible.
- *project window*: provides a way to reference windows corresponding to open projects. A group window does not present any properties beyond the basic *window* class, but provides a way to differentiate project windows from other types of window.
- *disk browser window*: provides a way to reference windows corresponding to open disk browsers. A disk browser window does not present any properties beyond the basic *window* class, but provides a way to differentiate disk browser windows from other types of window.
- *results browser*: provides a way to reference results generated by a batch operation. A results browser does not present any properties beyond the basic *window* class, but provides a way to differentiate results windows from other types of window.
- *search results browser*: a subclass of results browser, referring specifically to the results of a single-file Find All command or a multi-file search.

Classes of Document

As with windows, there are various classes of document:

- *document*: the basic document class contains properties that apply to any sort of document: whether it has unsaved changes, the alias to the file on disk, and so on.
- *text document*: text documents contain information specific to text files opened for editing in BBEdit.

- *group document*: refers to a document corresponding to an open project. A project document does not present any properties beyond the basic *document* class, but provides a way to differentiate project documents from other types of document.
- *picture document*: refers to a document corresponding to an open picture file. A picture document does not present any properties beyond the basic *document* class, but provides a way to differentiate picture documents from other types of document.
- *movie document*: refers to a document corresponding to an open QuickTime movie file. A movie document does not present any properties beyond the basic “document” class, but provides a way to differentiate movie documents from other types of document.
- *QuickTime document*: refers to a document corresponding to an imported Quicktime image file. A QuickTime document does not present any properties beyond the basic “document” class, but provides a way to differentiate QuickTime documents from other types of documents.

“Lines” and “Display_lines”

The “line” element refers to a “hard” line, that is, a stream of characters that begins at the start of file or after a line break, and which ends at the end of file or immediately before a line break. This is consistent with the previous semantics of “line” in hard-wrapped documents, and these semantics now apply in soft-wrapped documents as well.

The “display_line” element refers to a line of text as displayed on screen (bounded by soft and/or hard line breaks).

The “startLine” and “endLine” properties of a text object now always refer to the “hard” start and end of lines. In other words, if a text object crosses multiple soft-wrapped lines, the startLine and endLine properties will be the same.

Both “startDisplayLine” and “endDisplayLine” properties are now part of the text object class. These serve the same purpose as the startLine and endLine semantics for soft-wrapped views in older versions of BBEdit.

Getting and Setting Properties

One significant improvement in BBEdit’s new scripting framework is the ability to get and set multiple properties of an object with a single scripting command. Every object has a property called *properties*. This property returns a record which contains all of the properties which can be fetched for that object. For example, the script command

```
properties of text window 1
```

will return a result like this one:

```
{id:55632400, container:application "BBEdit", bounds:{31, 44, 543, 964}, closeable:true, collapsed:false, index:1, modal:false, file:alias "Hard Disk:Users:Shared:doc_examples:index copy.html", modified:false, name:"index copy.html", position:{31, 44}, resizable:true, selection:"", contents:"..."}
```

Conversely, to set one or more properties at once is very easy:

```
set properties of text window 1 to { show invisibles: true, show
spaces : true, soft wrap text : true }
```

Only the properties specified will be changed. The rest will not be modified.

It is important to note that when setting properties in this fashion, you can only set modifiable properties. If you attempt to set any read-only properties, a scripting error will result:

```
set properties of text window 1 to { show invisibles: true,
modal: false, expand tabs: true }
```

The above script command will turn on Show Invisibles and then report a scripting error, since *modal* is a read-only property.

Performing Actions

The following sections provide basic information on how to perform various common actions via AppleScript.

Scripting Searches

The ability to script searches presents you with a very powerful tool, since you can prepare a script which instructs BBEdit to perform a whole series of search or search and replace operations.

Consider the scripting command below:

```
tell application "BBEdit"
find "BBEdit(.*?)$" searching in document of text window 1 -
options { search mode: Grep } with selecting match
end tell
```

In previous versions, the *find* command always operated on the front window. Now, you must explicitly specify the text to be searched, either by specifying an explicit tell target, or by supplying a *searching in* parameter. So the following scripts are equivalent:

```
tell application "BBEdit"
find "BBEdit" searching in document of text window 1
end tell
```

and

```
tell application "BBEdit"
tell document of text window 1
find "BBEdit"
end tell
end tell
```

Note that either the tell-target or the *searching in* parameter must resolve to something that contains text. As a shortcut, you can specify a window, and if the window contains text, the search can proceed. You can also specify a text object:

```
find "Search Text" searching in (lines 3 thru 5 of document of
text window 2)
```

Also unlike previous versions of BBEdit, the defaults for parameters not specified in the *find* command are no longer controlled by the user interface (that is, the Find window).

When performing a *find*, BBEdit will return a record describing the results of the search. This record contains a Boolean which indicates whether the search was successful, a reference to the text matched by the search, and the text string matched by the search. Given the first example above, the results might look like this (after reformatting for clarity):

```
{found:true,  
found object:characters 55 thru 60 of text window 1 of  
application "BBEdit",  
found text:"BBEdit"}
```

Scripting Single Replaces

To do a single find and replace via AppleScript, you can write:

```
tell application "BBEdit"  
  
set result to (find "BBEdit" searching in text window 1-  
with selecting match)  
  
if (found of result) then  
    set text of (found object of result) to "Replacement"  
end if  
  
end tell
```

When performing a *grep* search, you cannot just replace the matched pattern with a replacement string; the *grep* subsystem needs to compute the substitutions. The *grep substitution* event is provided for this purpose; given a preceding successful *grep* search, it will return the appropriate replacement string. So if you perform a *grep* search, the script would look like:

```
tell application "BBEdit"  
  
set result to find "BBEdit(.+)$" searching in text window 1 -  
options {search mode:grep}  
  
if (found of result) then  
    set text of (found object of result) to -  
    grep substitution of "\\1"  
end if  
  
end tell
```

Note that when using a backslash “\” character in AppleScript, it needs to be “escaped” by means of another backslash; thus, in the above example, “\\1” used in the script, will become the *grep* replacement string “\1” when passed to BBEdit.

Scripting Multi-File Searches

In BBEdit, a multi-file search is a simple extension of the *find* scripting command. To search a single file or folder for all occurrences matching the search parameters, specify the file or folder as the *searching in* parameter of the search.

For example, to find all occurrences of “index.html” in a web site, one might use the following scripting command:

```
find "index.html" searching in (alias "Files:WebSite:")
```

Likewise, to find JavaScript line comments:

```
find "//.+ $" searching in (alias "Files:WebSite:") ~
  options {search mode: Grep}
```

To search in a single file:

```
find "crash" searching in (alias "Files:WebSite:index.html")
```

Scripting the Clipboard

BBEdit has multiple clipboards. These are fully accessible via the scripting interface. Due to operating system constraints, most clipboard operations require BBEdit to be frontmost.

Here are some examples:

```
count clipboard
```

- Returns the number of clipboards supported by the application

```
clipboard 1
```

- Returns {index:1, contents:"Files:WebSite:", length:14, is multibyte:false, display font:"ProFont", display font size:9, style:{plain}}

```
clipboard 1 as text
```

- Returns "Files:WebSite:"

```
clipboard 1 as reference
```

- Returns clipboard 1 of application "BBEdit"

```
current clipboard
```

- Returns the current clipboard as a record (you can coerce it to reference or text or get individual properties)

To set the text in a given clipboard to literal text:

```
set contents of clipboard 3 to "foobar"
```

To set the text in a clipboard to text represented by an object specifier:

```
set contents of clipboard 3 to selection of window 2
```

To copy the contents of one clipboard to another:

```
set contents of clipboard 5 to clipboard 3
```

or, to set the current clipboard to the contents of a different clipboard, (thus making it exportable to the system clipboard):

```
set current clipboard to clipboard 3 as text
```

or finally, with even less typing involved:

```
set current clipboard to clipboard 5
```

To make any clipboard the current clipboard, select it:

```
select clipboard 5
```

Scripting Text Factories

You can apply a text factory to a file via the AppleScript interface. The minimum invocation is:

```
apply text factory <file reference> to <reference>
```

The "to" parameter can be a single reference or a list of references, as for the multi-file "find" or "replace" events.

Optional parameters include "filter", "saving", "recursion", "text files only", "search invisible folders", all with the same meanings as in the multi-file "replace" event.

Setting Text Encodings

When specifying the encoding to use for opening or saving a file, you may either use the encoding's internet name, or its exact display name (as shown in the Read As popup menu).

For example:

```
open {file "Hard Disk:Users:Shared:example.txt"} reading as  
"Western (ISO Latin 1)"
```

```
open {file "Hard Disk:Users:Shared:example.txt"} reading as  
"iso-8859-1"
```

Arranging Documents and Windows

BBEdit provides considerable control for handling windows and documents both directly and via AppleScript.

Opening Documents

The "open" command supports additional options, which allow you to override your window handling preferences on a case by case basis:

```
open aFileList opening in <value>
```

As in previous releases, <value> may be a reference to an existing text window. However, you may instead specify "front_window", "new_window", or "separate_windows", which have the following effect:

- front_window: All files in aFileList are opened in the frontmost text window. (If there is no text window open, BBEdit will create a new one.)
- new_window: All files in aFileList are opened into a new text window.
- separate_windows: Each file in aFileList is opened into its own text window.

Moving Documents

The “move” command can be used to move text documents between text windows. For example:

```
tell application "TextWrangler"
  if (count of text windows) > 0 then
    select text window 1
    repeat while (count of text windows) > 1
      set ct to count documents of text window 2
      repeat with i from 1 to ct
        move document 1 of text window 2 to text window 1
      end repeat
    end repeat
  else
    beep
  end if
end tell
```

Referencing Documents

Previously, documents were indexed inside of multi-document windows by their display order in the file list. This meant that “document 1” of the application might not be the active document, which in turn required scripts to make special provisions to deal with the presence of multiple documents in a single window.

In order to handle this, BBEdit 8.0 introduced the “active document” property, which you could always use to specify the currently active document of a given text window. For example:

```
active document of text window 1 of application "BBEdit"
```

Although BBEdit still supports the “active document” property, this is no longer necessary. Instead, if a text window is frontmost:

```
document 1 of application "BBEdit"
```

```
document 1 of text window 1 of application "BBEdit"
```

```
active document of text window 1 of application "BBEdit"
```

now all refer to the same document. The side effect of this change is that if you wish to access documents within a text window by index, that index is:

- a) not related to the visual ordering of documents in the file list, and,
- b) documents’ indexes may change over time

This situation is effectively no different than handling documents which are contained in individual text windows, i.e. the index will change over time when you select different windows. If your script needs to keep a permanent references to a particular document, you should refer to that document by its id rather than its index.

Common AppleScript Pitfalls

Here are some things to watch out for when scripting BBEdit with AppleScript.

The Escape Issue

AppleScript uses the backslash character as an escape character. You can use `\r` to indicate a carriage return or `\t` to indicate a tab character. More importantly, you can use `\"` or `'` to include a quote mark or apostrophe in a string that is delimited by quotes or apostrophes. If you want to specify a real backslash, you must write `\\`.

That's not all that confusing until you start writing AppleScripts that call on BBEdit's powerful grep searching capability. BBEdit *also* uses the backslash as an escape character. If you want to search for an actual backslash in a document, you have to tell BBEdit to search for `\\`. However, if you do that in AppleScript, you must keep in mind that AppleScript will first interpret the backslashes before passing them to BBEdit. To pass one backslash to BBEdit from AppleScript, you must write two in AppleScript.

So to tell BBEdit to search for a single literal backslash from an AppleScript, you must write no fewer than *four* backslashes in the script. Each pair of backslashes is interpreted as a single backslash by AppleScript, which then passes two backslashes to BBEdit. And BBEdit interprets those two backslashes as a single one for search purposes. (This proliferation of backslashes can make your scripts look a bit like a blown-over picket fence.)

The Every Item Issue

When writing a script that loops through every item of a BBEdit object (for example, every line of a document), do not do it like this:

```
repeat with i in every line of text document 1
    -- do stuff here...
end repeat
```

This forces BBEdit to evaluate “every line of document 1” every time through the loop, which will slow your script significantly. Instead, write

```
set theLines to every line of text document 1
repeat with i in theLines
    -- do stuff here...
end repeat
```


Working with Development Tools

This chapter describes how to set up BBEdit to work with development environments. BBEdit offers a arsenal of capabilities in support of development tasks, beginning with syntax coloring and function browsing support for numerous languages, as well as support for Exuberant Ctags. BBEdit also offers direct integration with the system-supplied Perl, Python, and Ruby environments, as well as shell scripts and other Unix scripting tools, and with the CVS and Perforce source control systems. Additionally, you can invoke BBEdit from the command line via optional tools, or employ shell worksheet windows to store and execute frequently used commands.

In this chapter

| | |
|--------------------------------------------------------------------------------|-----|
| Configuring BBEdit for Development Environments | 298 |
| <i>Syntax Coloring</i> – 298 • <i>Exuberant Ctags</i> – 298 | |
| <i>Switching Between Counterpart Files</i> – 300 | |
| BBEdit and the Unix Command Line | 300 |
| <i>Shell Worksheets</i> – 300 | |
| <i>Installing the Command Line Tools</i> – 302 | |
| <i>The “bbedit” Command Line Tool</i> – 302 | |
| <i>The “bbdiff” Command Line Tool</i> – 303 | |
| <i>The “bbfind” Command Line Tool</i> – 304 | |
| Unix Scripting: Perl, Python, Ruby, Shells, and more! | 305 |
| <i>Using Unix Scripts</i> – 305 • <i>Language Resources</i> – 305 | |
| <i>Line Endings, Permissions and Unix Scripts</i> – 306 | |
| <i>Configuring Perl</i> – 307 • <i>Configuring Python</i> – 307 | |
| <i>Configuring Ruby</i> – 307 • <i>Shebang Menu</i> – 307 | |
| <i>Filters and Scripts</i> – 309 • <i>Filters</i> – 310 • <i>Scripts</i> – 310 | |
| <i>Additional Notes</i> – 310 | |
| Working with CVS..... | 311 |
| <i>Quick Start for SSH-accessed Repositories</i> – 311 | |
| <i>Other Configurations</i> – 311 • <i>Performing Commit Operations</i> – 312 | |
| <i>CVS Menu Commands:</i> – 312 | |
| Working with Perforce..... | 315 |
| <i>Perforce Menu Commands:</i> – 315 | |
| Working with Subversion..... | 317 |
| <i>Configuring Subversion</i> – 317 • <i>Command Line Integration</i> – 317 | |
| <i>Subversion Commands:</i> – 317 | |

Configuring BBEdit for Development Environments

By default, BBEdit will display separate menus for CVS, Perforce, and Subversion if these tools are installed on your Mac in their standard locations. You can enable or disable display of any tool's menu by checking or unchecking its entry in the Menus & Shortcuts preference panel.

Note If your cvs, p4, and/or svn clients are installed in a non-standard location, you can configure BBEdit to use them as described in "Expert preferences settings" on page 211.

Syntax Coloring

Syntax coloring is the practice of drawing keywords and other language elements in colors which differ from the standard text color to add emphasis and improve the readability of your code. BBEdit offers built-in syntax coloring support for a wide range of programming languages and other types of structured content. You can adjust BBEdit's default text colors or define color schemes in the Text Colors preference panel, or assign a color scheme to a specific language in the Languages preference panel.

Exuberant Ctags

In addition to its native function browsing capability, BBEdit supports the use of Exuberant Ctags for navigating source code files. Complete information about Exuberant Ctags is available on the project web site.

<http://ctags.sourceforge.net/>

Source for the implementation used by BBEdit is available within the application package, or from our web site.

<http://www.barebones.com/support/develop/ctags.shtml>

Using ctags

BBEdit allows you to generate and use tags files as text completion sources, and will recognize any tags files associated with your documents.

You may place tags files generated via 'bbedit --maketags' in the Completion Sources folder of BBEdit's application support folder (see page 30) for use as text completion sources.

If a tags file is found in the same directory as the front document, or in any parent directory up the chain from the front document, you can employ the ctag information by selecting a word, and either choosing Find Definition from the Search menu, or by Control-clicking and accessing the Definitions submenu of the contextual menu.

If you choose Find Definition, BBEdit will use any available ctags information (either from completion sources or associated with the current document) to find definitions of the selected word, and open a sheet from which you can choose the desired definitions to view. Select a definition to open it, or use the Show All button to open a search results browser showing all of the available definitions.

If you use the contextual menu, the Definitions submenu will contain a list of the available definitions. Select a definition to open it, or choose Show All to open a search results browser showing all of the available definitions.

Tag File Generation and Updating

Because individual workflows and setups vary widely, BBEdit does not attempt to generate or update ctags information; it only uses existing tags files.

- You can generate tags files for whatever fields you wish. BBEdit requires the signature (`--fields=+S`) to build the definitions menu. For our own source code, we include class members (`+a`) and function implementations (`+m`) in addition to the required fields.
- BBEdit looks for tags files starting in the same folder as the current document, and crawls upwards from there.

In order to keep your tags up to date, we recommend incorporating a suitable script into your build system, or employing some other mechanism such as a cron task. Xcode's documentation contains information on creating build phases that run shell scripts and similar solutions should be possible for other IDEs.

For example, we use the below Python script to update the tags for our source code whenever a build is started in Xcode. Since ctags is efficient, this script only takes a few seconds to run, and will update the tags even if one or more files does not compile.

```
#!/usr/bin/python
import os
import sys

TAGS_TEMP_FILE = '/tmp/tags'

projectDir = os.environ['SRCROOT']
projectName = os.environ['PROJECT_NAME']

ctagsExecutablePath = "/Applications/BBEdit.app/Contents/
Helpers/ctags"

baseArgs = '--excmd=number --tag-relative=no --
fields=+a+m+n+S -f /tmp/tags -R'
appendArg = '--append'

os.chdir('/')

sourceDir = os.path.join(projectDir, "ApplicationSource")
tagsFile = os.path.join(projectDir, 'tags')

# create the project's tags in '/tmp'
if os.access(sourceDir, os.F_OK):
    buildTagsCommand = '''%s' %s '%s' ''' %
(ctagsExecutablePath, baseArgs, sourceDir)
    output = os.popen(buildTagsCommand).read()

# move it where it goes
os.rename(TAGS_TEMP_FILE, tagsFile)
```

Tag Files as Completion Sources

You can now add tags files to specific locations to make symbols available as completion data sources when editing in desired languages. In particular:

- When you build a (coded) language module, if you place a file named “tags” in the language module’s “Resources” directory, BBEdit will use those tags as completion sources.
- You can generate a tags file (using `exuberant ctags` or “`bbedit --maketags`”) and place the resulting file in `Application Support/BBEdit/Completion Data/<language name>/`, where “<language name>” is the name of the language as it appears in the list of installed languages (or on the Languages popup menu).

So, for example, if you were to generate a tags file for the 10.6 SDK so that you could add completion data when editing Objective-C files, the file would go in `Application Support/BBEdit/Completion Data/Objective-C/`.

Tags files can be given any appropriate name, so you can have multiple tags files for a single language, and they will all be examined when generating completions.

Switching Between Counterpart Files

When editing any source file which has a counterpart (header), you can press the Counterpart button in the navigation bar or type `Control-Option-up` arrow to switch to its counterpart file, or vice versa. (BBEdit uses the suffix mapping options in the Languages preference panel to determine whether a particular file is a source or header file.)

BBEdit and the Unix Command Line

This section describes BBEdit’s facilities for interacting with the Unix command line: shell worksheets for issuing commands *to* the Unix shell and the “`bbedit`”, “`bbdiff`”, and “`bbfind`” command line tools for invoking BBEdit *from* the command line.

Shell Worksheets

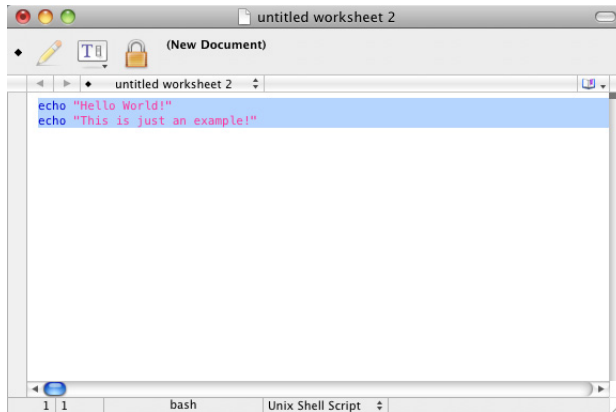
BBEdit allows you to store and execute Unix command lines by means of a “shell worksheet.” Choose Shell Worksheet in the New submenu of the File menu to open a new worksheet using your default Unix shell (generally ‘`bash`’).

Shell worksheets are stored in a private document format which is not text-based. This format allows BBEdit to store auxiliary data in the worksheet file’s data fork, thus ensuring that worksheets can safely be stored in version control systems that are not resource fork aware.

Using Worksheets

You can type, delete, and edit text in a worksheet window just as in an ordinary BBEdit document window. To invoke a Unix command, type the command, then press the Enter key or `Command-Return`, or click in the status area at the bottom-left of the worksheet window. (Keep in mind that Enter and Return are different keys; pressing Return by itself inserts a carriage return instead of executing a command.) You can execute more than one

command at a time by selecting multiple lines and pressing Enter or Command-Return. The output will appear in the worksheet window below the line or lines containing the commands executed. Unlike a terminal, this does not have to be at the end of the document: you can type commands anywhere in the worksheet window, or place the insertion point back on a previously executed command to run it again.



Note If the selection range is non-empty, only the exact text selected will be executed; if there is just an insertion point, the entire line containing it will be executed (even if it is not at the end of the line).

The status area at the bottom-left of the worksheet window shows the name of the Unix process currently executing (or the name of the shell itself when no process is running). This can be useful for seeing what is going on when a process hangs or takes a long time to complete. You can kill the currently running process by typing Control-C or Command-Period in the worksheet window. Also, clicking in the status area sends the currently selected text (or the line containing the insertion point) as a command to the Unix shell.

Keep in mind that shell worksheets are not terminal windows. If you have ever used MPW, you will probably feel right at home using shell worksheets. If you are only familiar with terminal emulators, however, you will find that shell worksheets work quite differently. Command line editing gestures do not work, nor will any Unix commands that expect to be dealing with terminals. (For example, try running “emacs” in a shell worksheet.)

When you drag files or folders into a worksheet window, the behavior is different than when dragging these items into an ordinary document window. An unmodified drag of a single file or folder will insert the POSIX-style path of that item at the drop location. Additionally, rather than selecting the inserted text as in a normal editing window, the insertion point will be left at the end of the current line, so you can easily continue entering additional information or execute the line as a command.

Dragging multiple files and folders will now produce a set of paths for those items, with spaces for separators rather than carriage returns. This makes it easier to add arguments to a line for immediate execution as part of a command.

If you hold down the Command key while dragging, it will cause the file’s contents to be inserted (or a folder listing, if the item you are dragging is a folder).

The default working directory for new worksheet windows is the user's home directory. This directory is also used as the search directory for any Open Selection or Open File by Name operations executed from within the worksheet. New shell windows are colored using the "Unix Shell Script" language.

New shell worksheets initially run in normal user mode as the currently logged-in user. This is indicated by the locked padlock icon on the rightmost button at the top of the worksheet window. You can switch to superuser mode by clicking the padlock button; you will be prompted to enter your administrator password. This is equivalent to executing the command

```
sudo -s
```

in a terminal window; the worksheet's status area will indicate this with a "sudo:" prefix to the current state, and the padlock icon will change to unlocked. Clicking the button again "locks" the padlock and returns to normal user mode.

WARNING

If you are not familiar with Unix command line tools, we strongly urge you to obtain and read an introductory guide to using a Unix shell. Command line tools can be very useful, but if used incorrectly, they can render files, or even your entire system, unusable.

Default Worksheet Stationery

When creating a new worksheet window, BBEdit will look for a worksheet stationery file named "Default Worksheet Stationery". This file is located in the Stationery folder of BBEdit's application support folder. (See Chapter 2 for more information regarding BBEdit's application support folder.) If the default worksheet stationery exists, you will see the contents of this file in every new worksheet window you create.

BBEdit ships with a default worksheet stationery file that provides a small tutorial on using worksheet windows. When you grow tired of seeing this tutorial in every new worksheet, you can either remove the "Default Worksheet Stationery" file from the Stationery folder, or replace it with one of your own.

Installing the Command Line Tools

The first time you run BBEdit after installation, it will offer to install the "bbedit", "bbdiff", and "bbfind" command line tools for you. If you choose not to do so, you can choose "Install Command Line Tools" from the BBEdit (application) menu at any time to install (or re-install) the current version of each command line tool.

If older versions of the tools are installed, choosing this command will update them; it will not overwrite existing versions of the tools with older versions.

The "bbedit" Command Line Tool

You can use the "bbedit" command line tool to open files into BBEdit via the Unix command line.

To open a file into BBEdit from the command line, type

```
bbedit filename
```

where *filename* is the name of the file to be opened. You may also specify a complete FTP or SFTP URL to a remote file or folder to have BBEdit open the file, or an FTP/SFTP browser to the folder.

To launch BBEdit without opening a file (or to activate the application if it is already running), type

```
bbedit -l
```

You can also pipe STDIN to the “bbedit” tool, and it will open in a new untitled window in BBEdit: for example,

```
ls -la | bbedit
```

If you just type

```
bbedit
```

with no parameters, the tool will accept STDIN from the terminal; type Control-D (end-of-file) to terminate and send it to BBEdit.

The complete command line syntax for the “bbedit” tool is

```
bbedit [ -bcChlpsuvVw --resume ] [ -e <encoding_name> ]  
      [ -t <string> ] [ +<n> ] [ file (or) <S/FTP URL> ... ]
```

See the “bbedit” tool’s man page (“man bbedit”) for a complete description of the available switches and options.

The “bdiff” Command Line Tool

You can use the “bdiff” command line tool to apply BBEdit’s Find Differences command to a pair of files or folders specified on the Unix command line.

To invoke the Find Differences command from the command line, type

```
bddiff oldfile newfile
```

or

```
bddiff oldfolder newfolder
```

where *oldfile* and *newfile* are the names of the files, or *oldfolder* and *newfolder* are the names of the folders, to be compared. You can also specify options for how the Find Differences command will be applied, which correspond to those available in the dialog.

The complete command line syntax for the “bdiff” tool is

```
bddiff [ --<options> ] [ OLDFILE NEWFILE | OLDFOLDER NEWFOLDER  
 ]
```

See the “bdiff” tool’s man page (“man bdiff”) for a complete description of the available switches and options.

Invoking “bdiff” as an External Helper

When using “bdiff” as an external diff helper for any other program, e.g. Perforce or Subversion, you should invoke it with the --wait option.

The “bbfind” Command Line Tool

You can use the “bbfind” command line tool to access BBEdit’s powerful multi-file search from the Unix command line.

To perform a multi-file search from the command line, type

```
bbfind search-string search-path
```

where *search-string* is your search string (or pattern) and *search-path* is a list of path(s) to search. You can also specify options which control how the search should be performed; these options correspond to those available in the Multi-File Search window.

If no search paths are specified on the command line, “bbfind” will attempt to read them from standard input. This makes it easy to process the output of other tools such as “find”. For example:

```
`find . -name "*.py" -print | bbfind blah`
```

takes the paths printed by “find” and searches those files.

By default, “bbfind” expects that input will be separated by Unix newlines (\n). If instead, the input is being generated programmatically and contains “NUL”-separated paths, you can specify the “-0” option. Again using “find” as an example input source:

```
`find . -name "*.py" -print0 | bbfind blah -0`
```

The complete command line syntax for the “bbfind” tool is

```
bbfind search-string [-cEghInRSvVwZ0 --<long_form_switches>  
 search-path ]
```

See the “bbfind” tool’s man page (“man bbdiff”) for a complete description of the available switches and options.

Unix Scripting: Perl, Python, Ruby, Shells, and more!

BBEdit provides robust integration with numerous Unix scripting environments, including Perl, Python, Ruby, and shell scripts.

Using Unix Scripts

BBEdit works directly with the native Perl, Python, and Ruby environments provided by Mac OS X, and supports similar integration with shell scripts and any other Unix scripting language.

BBEdit's Unix scripting features are accessed via the Shebang menu: “#!”. (Why “Shebang”? Because executable Unix scripts traditionally start with the two-character sequence “#!”. Some people pronounce these two characters “hash-bang,” others say “sharp-bang,” but the most common pronunciation is simply “shebang.”)

The “shebang line” is the first line of the script, and includes a Unix-style path to the interpreter for the language—for example, “#!/usr/bin/perl”, or “#!/usr/local/bin/python”.

While BBEEdit does not entirely depend upon the accuracy of the shebang line (if your script file has an accurate language mapping), it is always a good practice, and sometimes necessary, to specify the full path to the executable in the shebang line.

Language Resources

Perl is an acronym for Practical Extraction and Report Language (or alternatively, Pathologically Eclectic Rubbish Lister) and was developed by Larry Wall. If you are interested in learning Perl, the quintessential Perl references are:

Learning Perl (4th Edition), by Randal L. Schwartz & Tom Phoenix. O'Reilly and Associates, 2005. ISBN: 0-596-10105-8

Programming Perl (3rd Edition), by Larry Wall, Tom Christiansen, Jon Orwant. O'Reilly and Associates, 2000. ISBN: 0-596-00027-8

The following are excellent Internet resources for the Macintosh implementation of Perl, and Perl in general:

Perl.com from O'Reilly and Associates
<http://www.perl.com/>

Perl Mailing Lists
<http://lists.cpan.org/>

Python is a portable, interpreted, object-oriented programming language, originally developed by Guido van Rossum. If you are interested in learning Python, consider the following books:

Learning Python (2nd Edition), by Mark Lutz & David Ascher. O'Reilly and Associates, 2003.
ISBN: 0-596-00281-5

Programming Python (2nd Edition), by Mark Lutz. O'Reilly and Associates, 2001.
ISBN: 0-596-00085-5

Internet resources for Python:

Python home page

<http://www.python.org>

Python Cookbook

<http://aspn.activestate.com/ASPN/Cookbook/Python>

Ruby is an interpreted scripting language with an emphasis on object-oriented programming, which has fast become a favorite of Web developers. Ruby was created by Yukihiro Matsumoto. If you are interested in learning Ruby, consider the following books:

Programming Ruby: The Pragmatic Programmer's Guide (2nd Edition), by Dave Thomas, with Chad Fowler and Andy Hunt. Pragmatic Bookshelf, 2004.

ISBN: 0-9745140-5-5

Ruby Cookbook, by Lucas Carlson & Leonard Richardson. O'Reilly and Associates, 2006.

ISBN: 0-596-52369-6

Internet resources for Ruby:

Ruby home page

<http://www.ruby-lang.org/>

RubyGarden Wiki

<http://wiki.rubygarden.org/Ruby>

Setting Environment Variables for GUI Apps

In Mac OS X, the system maintains separate environments for CLI and GUI applications. Thus, any changes you make within either environment will not affect the other.

Apple Technical Q&A 1067 describes the procedure necessary to set environment variables for use within BBEdit and other GUI applications.

<http://developer.apple.com/qa/qa2001/qa1067.html>

Note Please note you must log out and back into your account before the system will register environment changes made via this mechanism.

Line Endings, Permissions and Unix Scripts

To execute scripts, the script interpreter for any given language requires source code to be encoded with native line endings, i.e. Unix line breaks for Perl and most other shell scripting languages. BBEdit will warn you if you attempt to run a script which does not have Unix line endings.

Additionally, to execute scripts anywhere outside of BBEdit (e.g. in the Terminal), the system requires that the script file have 'execute' permissions set. Thus, when you first save any script file which contains a shebang (!) line, BBEdit will automatically set execute permissions for your login account (a+x, as modified by the umask) on that file.

Configuring Perl

BBEdit can make full use of the system's default Perl install with no need for further configuration. However, if you wish to install and work with multiple versions of Perl, you will need to specify the appropriate version in your scripts' shebang lines.

Search Paths

By default, Perl looks for modules in its standard library path and in the current directory. You may also use modules from other locations by specifying their paths in the PERL5LIB environment variable.

To do so, you must first create an “environment.plist” file as described under “Setting Environment Variables for GUI Apps” on page 306, then add a key-value pair consisting of the variable and the desired paths as follows:

```
<key>PERL5LIB</key>  
<string>/Users/example/Sandbox/myPerlMods/:/Users/Shared/  
moreMods/</string>
```

(You may specify multiple directories by separating their paths with colons per the standard convention.)

Affrus

BBEdit also supports the use of Affrus for debugging Perl scripts. Affrus is an integrated Perl development environment from Late Night Software.

<http://www.latenightsw.com/>

If Affrus is installed on your Mac, BBEdit will use it by default rather than the command line Perl debugger. In this case, BBEdit will pass complete environment parameters and control to Affrus, but will not attempt to retrieve output. You can prevent (or enable) BBEdit's use of Affrus by issuing the following Terminal command.

```
defaults write com.barebones.bbedit UseAffrusForPerlDebugging -  
bool NO
```

Configuring Python

BBEdit expects to find Python in /usr/bin, /usr/local/bin, or /sw/bin. If you have installed Python elsewhere, you must create a symbolic link in /usr/local/bin pointing to your copy of Python in order to use pydoc and the Python debugger.

Configuring Ruby

BBEdit can make full use of the system's default Ruby install with no need for further configuration. However, if you wish to install and work with multiple versions of Ruby, you will need to specify the appropriate version in your scripts' shebang lines.

Shebang Menu

The commands in this menu allow you to run Unix scripts directly within BBEdit.

Check Syntax

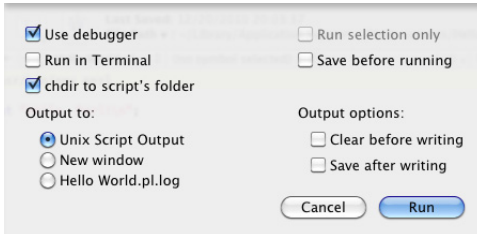
Checks the syntax for the frontmost window. Errors are displayed in a standard BBEdit error browser (see Chapter 9, “Browsers,” for more details on working with error browsers). This command is only available for Perl and Python scripts.

Run

Runs the script in the frontmost window by default. Any output from the script is displayed in a new BBEdit window. The output window is titled “Unix Script Output”, and the file is created in the Unix Support folder in BBEdit’s application support folder. By default, errors for Perl and Python scripts are displayed in an error browser; errors for other languages are displayed as text in the output window.

Run...

Displays the Run a Script sheet, which allows you to set options before running the script in the frontmost window.



Selection Only: Check this box to execute only the selected text in the frontmost document window. This box is disabled if there’s no active selection.

Save Before Running: Check this box to save the source file before running the script.

Output to: Choose to display output in a new window, to direct it to the Unix Output file, or to write it to a file in BBEdit’s Logs folder (~/Library/Logs/BBEdit/).

Use Debugger: Check this box to run Perl, Python, or Ruby scripts in the interpreter’s debugger.

Run in Terminal: This command runs the script in a new Terminal window.

Chdir to Script’s Folder: Check this box to set the working directory to the folder that contains the script before running it.

Output Options: Mark these checkboxes to clear the output file before writing and to save it after writing, respectively.

Run in Terminal

This command will run the script in a new Terminal window, regardless of the settings in the Run a Script dialog.

Run in Debugger

Runs the script in the interpreter’s debugger, regardless of whether the Use Debugger option is set for the Run command; also, any output options set in the Run command will be ignored. The Run in Debugger command is only available for Perl and Python.

Run File

Runs a script from an arbitrary file rather than from a BBEdit window. The Run a Script File dialog appears. You can select a file by clicking the File button or by dragging a file to the path box at the top of the dialog from the Finder. The options are the same as the ones described above for the Run a Script dialog.

Show POD/Show Module Documentation

When the frontmost document is a Perl file and you invoke the Show POD command, BBEdit will process the document contents using by the command line pod2text tool and display the result in a new text window.

Note POD stands for Plain Old Documentation, and is the standard Perl documentation format.

When the frontmost document is a Python file, the name of this command will change to Show Module Documentation, and if you invoke it, BBEdit will display the module documentation.

Filters and Scripts

Before you begin using Unix filters and scripts with BBEdit, you should locate and familiarize yourself with the Text Filters and Scripts folders, which resides within BBEdit's application support folder. (See Chapter 2 for details.).

The contents of the Text Filters and Scripts subfolders are presented respectively in the Apply Text Filters submenu and the Scripts menu, as well as the Text Filters and Scripts floating palettes.

Document State

For convenience, BBEdit sets some runtime environment variables to provide information about the front document's state right before a Unix filter or script is run:

| Variable | Description |
|----------------------|--------------------------------------------------------------------------|
| BB_DOC_LANGUAGE | Name of the document's current language (not set if language is "none") |
| BB_DOC_MODE | Emacs mode of the document's current language |
| BB_DOC_NAME | name of the document |
| BB_DOC_PATH | path of the document (not set if the document is unsaved) |
| BB_DOC_SELEND | (zero-based) end of the selection range (not set if not text document) |
| BB_DOC_SELEND_COLUMN | (one-based) de-tabbed column number of BB_DOC_SELEND |
| BB_DOC_SELEND_LINE | (one-based) line number of BB_DOC_SELEND |
| BB_DOC_SELSTART | (zero-based) start of the selection range (not set if not text document) |

| Variable | Description |
|------------------------|--------------------------------------------------------|
| BB_DOC_SELSTART_COLUMN | (one-based) de-tabbed column number of BB_DOC_SELSTART |
| BB_DOC_SELSTART_LINE | (one-based) line number of BB_DOC_SELSTART |

Note Selection ranges and other offsets are expressed in **characters**, not bytes.

Filters

Filters operate on the selected text of the frontmost document. BBEdit will pass either the selected text (if any) or the contents of the entire document as input to the filter on STDIN and any output generated by the filter overwrites the selection.

Note This method represents a change from versions 10.0.1 and prior, in which BBEdit wrote a temporary file and passed it on `argv[0]`. Thus, if you have any existing Unix filters (in the "Text Filters" folder), you **will** need to modify those filters in order for them to continue working.

There are two ways to run filters: through the Apply Text Filters submenu in the Text menu or via the Text Filters palette. To open the Text Filters palette, select it from the Palettes submenu in the Window menu. You can run a filter by selecting it from the list and clicking the Run button, or you can simply double-click the filter name in the list.

Hold down the Option key while double-clicking a filter or selecting it from the menu to open the file for editing instead of running it. You can also hold down the Shift key while selecting a filter item from the Apply Text Scripts submenu to reveal the file in the Finder, or you can select a folder node from the menu to open that folder in the Finder.

Scripts

Scripts do not operate on the text of the frontmost window, but rather run directly. You can run scripts from the Scripts menu or the Scripts palette. Hold down the Command key while selecting or double-clicking a script to open the Run a Script options dialog; hold down the Option key while selecting or double-clicking a script to open the script's file for editing instead of running it; hold down the Shift key while selecting or double-clicking a script to reveal its file in the Finder (or while selecting a folder node, to reveal that node in the Finder).

Additional Notes

In addition to the features detailed above, BBEdit offers some additional options which it may help you to be aware of.

Setting Menu Keys for Scripts

The Filters and Scripts palettes both have a "Set Key" button at the top. Select a filter or script in the list and click this button to set a keyboard shortcut for the selected item. You may also assign key equivalents to scripts or filters within the Menus & Shortcuts preference panel.

Manually Sorting the Filter and Script Lists

By default, items in the Apply Text Filters submenu and the Scripts menu display in alphabetical order. However, you can force items to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example “00)Foo” would sort before “01)Bar.”) For such files, the first three characters are not displayed in BBEdit. You can also insert a divider by including an empty folder whose name ends with the string “-***”. (The folder can be named anything, so it sorts where you want it.)

Canceling Filter or Script Execution

You can press the Cancel button in the progress dialog or type Command-. (Command-period) to cancel a task directly from within BBEdit. Since BBEdit must kill the spawned Unix process with a SIGINT, any unflushed data in open filehandles (including STDOUT and STDERR) will be lost unless the script takes measures to prevent this.

Working with CVS

BBEdit offers integrated support for CVS (the Concurrent Versions System source control package).

CVS Information

<http://www.cvshome.org/>

Quick Start for SSH-accessed Repositories

If you already have a CVS working copy from a repository that you access via SSH, BBEdit should now work “out of the box”. You need only open a file from that working copy, or create a project which includes it.

Other Configurations

If you use any other means than the above to access a CVS server, you will need to configure your local CVS setup and repository access from the command line, and perform an initial checkout.

How this configuration must be done varies from user to user and from organization to organization. The below article on Apple’s web site provides a general overview of using CVS; for more details, you should consult your local CVS guru as necessary.

Mac OS X: Version Control with CVS

<http://developer.apple.com/internet/opensource/cvsoverview.html>

Once you have configured CVS and checked out a working copy, you may apply the commands in BBEdit’s CVS menu to files and folders within that working copy.

Performing Commit Operations

IMPORTANT

By default, all CVS commit operations open a CVSEditor window to permit entry of checkin comments, in place of the text fields previously presented in the command dialogs. This window allows you to enter detailed checkin messages, with a better text composition environment than the optional dialog.

CVS Menu Commands:

BBEdit's CVS menu contains the following commands.

Update to Head

This command replaces a file with the latest revision of that file from the CVS repository. To update the file in the front-most window, choose Update File from the CVS menu. To update any file on disk, hold down the Shift key while choosing Update File from the CVS menu, and choose a file from the Open dialog that appears.

Update to Revision...

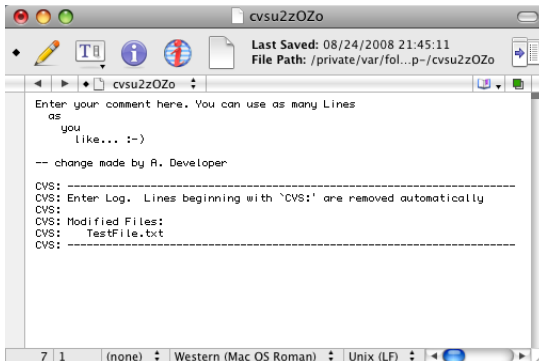
Choosing this command will bring up a dialog listing all available revisions for the current file. Select any of these revisions to perform an update against the current file.

Note

To preserve the previous functionality of this command, if your file has local modifications, BBEdit will ask whether you want to merge those mods with the selected revision (as "cvs update" would normally do), or discard them and fetch the selected revision as-is (which BBEdit formerly did).

Commit...

Choosing this command will bring up a CVSEditor window (see below) to permit you to enter checkin comments, and then commits the current document into the repository. If the document has unsaved changes, it will be saved first. You cannot commit a file without having first added it to the repository using the Add command.



To enter a comment, type whatever you wish above the lines which are prefixed with "CVS:". Then, perform a Save (Command-S) and Close the window (Command-W); once you have done so, BBEdit will commit the document, and display the CVS log.

Note

Although BBEdit lets you commit a file without a comment, it's considered good form to supply a comment that summarizes the changes you made to the file.

Commit Parent Folder...

When you choose this command, BBEdit will prompt you for a comment using the Commit Folder dialog, and then commit all of the files contained in the parent folder of the current document, just as if you had selected that folder and chosen the Commit Folder command. If you do not need to enter a comment, hold down the Shift key as you choose this command, and BBEdit will commit the folder's files without displaying a dialog box.

Update Working Copy...

When you choose this command, and select a folder in the Update Working Copy dialog, BBEdit will update all of the files contained in the folder to the latest available revisions. You can select any recently accessed folder from the popup menu, or choose the Other entry to bring up an Open dialog, allowing you to navigate to and choose any desired folder to update. You can also drag a folder from the Finder directly into the path box.

Commit Working Copy...

When you choose this command, BBEdit will bring up the Commit Folder dialog, which allows you to batch commit all the files within a specific folder that have uncommitted changes.

You can select any recently accessed folder from the popup menu, choose the Other entry to bring up an Open dialog, in which you can select any desired folder, or drag a folder from the Finder directly into the path box.

Show Working Copy Status...

When you choose this command and select a folder in the Show Working Copy Status dialog, BBEdit will check whether any files contained within that folder have uncommitted changes. If there are any such files, BBEdit will display a results browser containing a list of them.

You can select any recently accessed folder from the popup menu, or choose the Other entry to bring up an Open dialog, allowing you to navigate to and choose any desired folder to commit. You can also drag a folder from the Finder directly into the path box.

Compare Revisions...

When working with a file that is in a CVS repository, you can use this command to access a dialog listing all available revisions of the file. When you choose a revision from the list, BBEdit will check that revision out into a temporary file, and automatically perform a Find Differences between it and the current file.

Compare Arbitrary Revisions...

When working with a file that is in a CVS repository, hold down the Shift key and Compare Revisions becomes Compare Arbitrary Revisions. You can use this command to bring up a dialog which contains two lists, each of which shows all available revisions of the file. Choose a revision from each list to have BBEdit check them out into temporary files, and automatically perform a Find Differences between them.

Compare with Base

When working with a file that is in a CVS repository, choosing this command will check out the base revision of this file as a temporary file, and compare that to the current document.

Compare with Head

When working with a file that is in a CVS repository, choosing this command will check out the head revision of this file into a temporary file, and compare that to the current document.

Show Annotation

When working with a file that is in a CVS repository, choosing this command will perform a ‘cvs annotate’ on the file.

Show Revision History

When working with a file that is in a CVS repository, choosing this command will show the file’s revision history in a new display window.

Show CVS Status

When working with a file that is in a CVS repository, choosing this command will bring up the CVS Info dialog, which displays the local path to the file, and its current status and revision information.

Add

When you choose this command, BBEdit will add the current file to the appropriate CVS repository if it is not already present. Note that after using this command, you must also perform a Commit in order to store the file’s contents to the repository.

Remove

When working with a file that is in a CVS repository, you can choose this command to remove the file from the repository. Note that after using this command, you must also perform a Commit Parent Folder (or a Commit Folder on the repository as a whole) before the file will actually be removed.

Go to Previous/Next Conflict

When you perform a CVS update on one or more files, if there are conflicts between your revisions and those in the repository, CVS will add markers in the affected file(s) at the appropriate points. If you update a single file and one or more conflicts result, BBEdit will automatically jump to the first conflict and you can use these commands to move to other instances (if any exist). If you update multiple files and conflicts result, you can choose either of these commands to move between instances of conflicts in a file. If there are no conflicts in the current file, BBEdit will beep when you select either command.

Export Project...

This command creates a copy of a project at the selected location. When you choose this command, BBEdit displays a dialog that lets you choose a CVS project and then displays a Save dialog that lets you choose the location in which to create a copy of it.

Open Log File

Opens the “CVS.log” file, or brings it to the front if it is already open. (This file is stored in the “BBEdit” folder within the “Logs” folder of your user account: ~/Library/Logs/BBEdit/.)

Working with Perforce

BBEdit features integrated support for the Perforce source control system.

<http://www.perforce.com/>

In order to enable BBEdition's Perforce integration, you must have the "p4" command line tool installed. Once you have configured Perforce, you can work with files and folders in client spaces using the commands on BBEdition's P4 menu. You can apply many commands to the current document. You can also select one or more files from a results browser window, such as a search results browser, or the results browser for the Show Opened command, and apply commands.

If you have an open document which BBEdition does not recognize as being associated with any client space, BBEdition will look for \$P4CONFIG in its environment. If BBEdition finds such a value, and finds a config file in an appropriate directory, then it will honor that configuration.

Note In any such case, you must first open a document associated with the configuration, so that BBEdition knows where to set the working directory, before any commands on the Perforce menu, even non file-specific commands, will become enabled.

Perforce Menu Commands:

BBEdition's Perforce menu contains the following commands.

Edit

Performs a 'p4 edit' on the frontmost document, or the selected files.

Revert

Performs a 'p4 revert' on the frontmost document, or the selected files.

Revert & Sync To Head

Performs a 'p4 revert' followed by a 'p4 sync' on the frontmost document. or the selected files.

Sync To Head

Performs a 'p4 sync', passing the file path of the frontmost document, or of the selected files in a results browser, to the command.

Sync To Revision...

Displays a revision list for the frontmost document, and then syncs the file to the chosen revision

Sync All

Performs a 'p4 sync' with no additional arguments.

Submit...

Opens a change list editor window in BBEdition, and then submits the changes described within.

Show Opened

Displays a results browser showing all the files opened or added to the repository. You can select files in this browser, and apply many other commands to them.

Compare Revisions...

Displays a revision list for the frontmost document, and then compares the contents of that document to the revision you chose from the list.

Compare Arbitrary Revisions...

Displays two revision lists, and then compares the two revisions you choose to each other without changing the version you have checked out.

Compare with Base

Compares the contents of the frontmost document to the revision it is derived from.

Compare with Head

Compares the frontmost document to the head revision.

Show Annotation

Opens a window containing the results of a 'p4 annotate'.

Show Revision History

Shows the revision history for the current document. (Equivalent to a 'p4 filelog'.)

Add

Adds the frontmost document to the P4 repository. As with the command line tool, you must then perform a Submit to register this change with the server.

Delete

Deletes the frontmost document from the P4 repository. As with the command line tool, you must then perform a Submit to register this change with the server.

Revert & Delete

Performs a 'p4 revert' on the chosen document file prior to performing a deletion, allowing modified files to be discarded.

Go to Previous/Next Conflict

Searches the current document for the previous or next marker which indicates a sync conflict.

Open Log File

Opens the log file which contains the output from BBEdit's Perforce commands (~/.Library/Logs/BBEdit/Perforce.log).

Working with Subversion

BBEdit now features integrated support for the Subversion source control system.

<http://subversion.tigris.org/>

Configuring Subversion

In order to enable BBEdition's Subversion integration, you must have Subversion 1.1 or later installed and available in the PATH for GUI applications (as controlled by ~/.MacOSX/environment.plist). You must also have a Subversion working copy which has been checked out via the command line.

Command Line Integration

You can use the “bbdiff” command line tool as an external diff tool for Subversion. You can do so either directly, e.g.:

```
svn diff --diff-cmd bbdiff --extensions "--resume --wait" -rPREV <FILE>
```

or by specifying the tool as the default “diff-cmd” in your Subversion config (~/.subversion/config) as appropriate.

Subversion Commands:

You can use most Subversion commands on either the active document, or the selection in a results browser or disk browser. Exceptions are as follows.

These commands are always available:

- Update Working Copy...
- Commit Working Copy...
- Show Working Copy Status...
- Open Log File

These commands require either an open document, or that you have selected a single item in a results browser or disk browser:

- Compare Revisions (all variants)
- View Annotation
- View Revision History
- Go to Previous Conflict
- Go to Next Conflict

Revert

Discards local changes to the active document's file, reverting it back to BASE. Has no effect if there are no local modifications to the file.

Update to Head

Updates the active document's file to the HEAD revision, merging local changes with the head revision, and marking conflicts as necessary. The first conflict in the file, if present, will be selected after the command successfully runs.

Revert & Update to Head

Discard local modifications before updating to HEAD.

Commit...

Commits the local file to the repository. BBEdit will display a Subversion commit window into which you can enter your commit message. This window will be prefilled with form text below the insertion point, consisting of a special separator line and a listing of the files which are about to be committed and their status. This separator line, and all lines below it, will be removed and will not appear as part of the commit message.

Update Working Copy...

Updates all the files contained in a working copy (folder tree) to the HEAD revision. BBEdit will display a dialog in which you can select the desired working copy, by either dragging and dropping a folder to the folder well, or by using the popup menu on the right which allows you to select any defined Subversion config, or any recently-accessed folder which is under Subversion's control.

Commit Working Copy...

Commits all changed files in a working copy to the Subversion repository. Uses a working copy selection dialog like that for the Update Working Copy command (above).

Show Working Copy Status...

Shows pending changes/status of the selected working copy. If you choose to show updates, BBEdit will query the server and also display files for which there are newer revisions on the server. Uses a working copy selection dialog like that for the Update Working Copy command (above).

Compare Revisions...

Displays a dialog which allows you to select any previous/future revision of the current file in the repository and perform a Find Differences between that revision and the local revision.

Compare Arbitrary Revisions...

Displays a dialog which allows you to choose any two revisions of the current file from the repository and perform a Find Differences on those revisions.

Compare with Base/Compare with Previous/Compare with Head

These are convenience commands which allow you to compare the local revision with BASE, PREV or HEAD revisions respectively.

Resolved

Marks the current file as resolved if it was previously marked as in conflict with updates received from the repository.

Show Annotation

Displays the content of the file line by line with author and revision information.

Show Revision History...

Brings up a sheet on which you can display options, and then displays the revision history of the local file. You can hold down the Option key when selecting this command to use the last settings.

Add

Schedules the current file, or all files selected in a browser, for addition to the repository. (You must perform a Commit to add the file(s) to the repository.) This command will not be available if the selected files are already part of the repository.

Delete

Schedules the current file, or all files selected in a browser, for deletion from the repository. (The local file(s) will be deleted immediately, but you must perform a Commit to permanently delete them from the repository.)

Revert & Delete

Reverts local changes to the current file, or all files selected in a browser, then schedules the file(s) for deletion from the repository. (The local file(s) will be deleted immediately, but you must perform a Commit to permanently delete them from the repository.)

Go to Previous /Next Conflict

Selects the previous or next conflict in the file, if any.

Open Log File

Opens BBEEdit's Subversion log file (~/.Library/Logs/BBEdit/Subversion.log).

Language Modules and Packages

Language modules are special files that you can install to add support for syntax coloring, and optionally, function browsing, for programming languages beyond those built in. Many people have prepared language modules for BBEdit, and these modules are available from various web sites (including our own).

Packages are collections of related supporting items, such as filters, scripts, and language modules, which you can add to BBEdit to extend it.

This chapter describes the basic procedures for installing and using language modules and packages, and provides references to information about producing such items.

In this chapter

| | |
|--------------------------------------------|-----|
| Codeless Language Modules | 322 |
| <i>Installing Language Modules</i> – 322 | |
| <i>Overriding Existing Modules</i> – 322 | |
| <i>Codeless Language Modules</i> – 322 | |
| <i>Code-based Language Modules</i> – 322 | |
| <i>Language Module Compatibility</i> – 322 | |
| Plug-In Obsolescence | 323 |
| Plug-In Obsolescence | 323 |

Language Modules

Language modules are add-on items which provide syntax coloring and function browsing for programming languages that BBEdit does not natively support.

There are two types of language modules: coded, and codeless. Coded language modules must be prepared according to the requirements of BBEdit's language module interface. (See Appendix D.) Codeless language modules are text documents prepared in a specific plist format. (See below.)

After you install a language module and relaunch BBEdit, syntax coloring and function browsing will be available for the language(s) supported by that module. To verify that a language module is active, or to modify or add file suffix mappings for the language(s) it provides, use the Languages preference panel (see page 204).

Installing Language Modules

To install a language module, move or copy the module file into the Language Modules folder of your BBEdit application support folder. If no such folder exists, you may create it.

After installing a new language module, you must quit and relaunch BBEdit in order to use it.

To remove an installed language module, you must remove the item's file from the Language Modules subfolder of your BBEdit application support folder, then quit and relaunch BBEdit.

Overriding Existing Modules

Language modules can override existing language definitions, including the built-in definitions. If there is more than one module present which supports a given language, BBEdit will use the module with the most recent modification date.

Codeless Language Modules

A codeless language module is a specially-formatted text file which allows you to describe the properties of a source code language via a set of basic parameters. BBEdit will then use these parameters to perform syntax coloring and function navigation for the specified language.

Codeless language modules are written as “property lists” (or “plists”), which is an XML format that Mac OS X uses for many purposes. You can create or edit codeless language module files with BBEdit itself, with the Mac OS X Property List Editor (located in /Developer/Applications/Utilities/if you have installed the Apple Developer Tools package), or with a third-party editor such as PlistEdit Pro.

<http://www.fatcatsoftware.com/plisteditpro/>

You can find complete specifications for creating codeless language modules in Appendix D (see page 359), or in the Developer Information section of our web site.

<http://www.barebones.com/support/develop/>

Code-based Language Modules

BBEdit also supports producing code-based language modules to handle more complex languages or document formats. You can find complete specifications for creating code-based language modules in the Developer Information section of our web site.

<http://www.barebones.com/support/develop/>

Language Module Compatibility

IMPORTANT

You will **not** be able to use any third-party language modules which do not support Unicode text, or which were built in CFM format. If BBEdit encounters such a module, it will not load that module, and will log a message to the system console.

Contact the developers of such a module, or visit the Bare Bones Software web site (see above) for more information on the availability of updated modules.

Packages

BBEdit now supports installing “packages” of items to extend its functionality. A package is simply a pre-defined group of the sort of items you can individually place into BBEdit’s application support folder (and subfolders) to extend BBEdit; however, the package format makes it easier to handle and install sets of related items.

Each package is a folder whose name must end in “.bbpackage”, and the items within this folder must conform exactly to the following requirements.

The package folder must contain the following item:

- Contents [folder]

The “Contents” folder may contain these two items (which are currently not required, and are reserved for future use):

- Resources [folder]
- Info.plist [file]

The “Contents” folder may also contain any or all of the following subfolders:

- Clippings
- Language Modules
- Scripts
- Text Filters

The items contained within each subfolder will behave as though they were present within the subfolder of the same name inside BBEdit’s application support folder. (These subfolders are all optional, though obviously a package which contains none of them will be of no benefit.)

In order to use a populated package, you should place it within the “Packages” subfolder of BBEdit’s application support folder (~/.Library/Application Support/BBEdit/Packages/).

Plug-In Obsolescence

IMPORTANT

BBEdit 10.0 no longer supports BBXT-based code plug-ins, and will not load any such items present in the “Plug-Ins” application support folder.

If you used any third-party commercial plug-in, please contact its developer for information on alternative solutions.

A

This appendix provides a quick reference for key assignments and a comprehensive list of the commands that are available from BBEdit's user interface.

In this appendix

| | |
|-----------------------------------------|-----|
| Keyboard Shortcuts for Commands..... | 325 |
| Assigning Keys to Menu Commands..... | 326 |
| <i>Available Key Combinations</i> – | 326 |
| Listing by Menu and Command Name | 327 |
| Listing by Default Key Equivalent | 338 |

Keyboard Shortcuts for Commands

Many of BBEdit's commands have pre-defined keyboard shortcuts. BBEdit also lets you reassign the shortcuts for any menu command, clippings entry, plug-in, or script to suit your own way of working.

To change the keyboard shortcut for any menu command, you can use the Menus & Shortcuts preference panel. (See “Assigning Keys to Menu Commands” on the following page.)

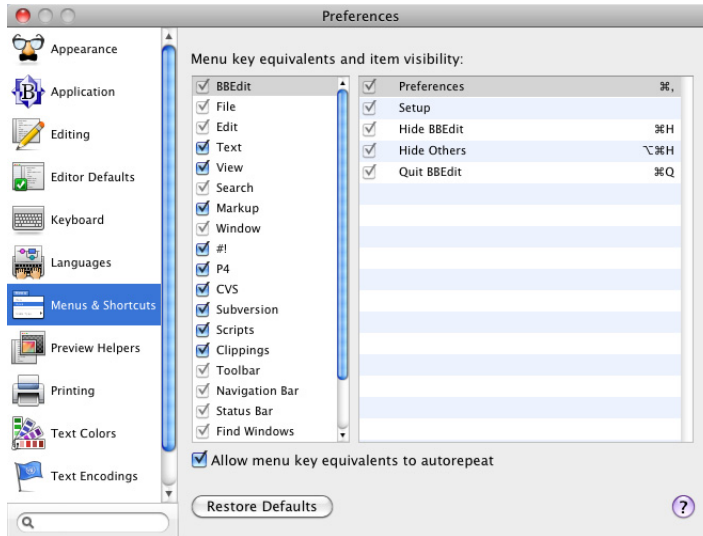
Many other BBEdit features can have keyboard shortcuts assigned as well. Here's how to set them:

| Feature | Set Keys in... |
|----------------|-----------------------------------------------------------------|
| Menu commands | Menus & Shortcuts preference panel |
| Clippings | Clippings palette |
| Text filters | Menus & Shortcuts preference panel, or the Text Filters palette |
| Scripts | Menus & Shortcuts preference panel, or the Scripts palette |
| Stationery | Menus & Shortcuts preference panel, or the Stationery palette |

To display any of BBEdit's floating palette windows, use the Palettes submenu in the Window menu.

Assigning Keys to Menu Commands

You can assign your own keyboard shortcuts (key equivalents) to any of BBEdit's menu commands, as well as items on the Text Options, Markers, and Line Breaks toolbar popup menus, by choosing Preferences from the BBEdit menu to bring up the Preferences window, then selecting the Menus & Shortcuts preference panel.



To set the key equivalent for a menu command, locate and select the entry for the command under the appropriate menu section, then double-click on the right-hand part of the line containing that command, and type the desired keystroke.

To remove an existing key equivalent from a command, double-click on the existing key combination and press the Delete key.

Click the Restore Defaults button to restore all key equivalents to their default values (as listed in this Appendix).

Available Key Combinations

All menu key combinations must include either the Command key or the Control key (or both), except function keys, which may be used unmodified. The Help, Home, End, Page Up and Page Down keys can be used in menu key combinations as well. The Help key can be assigned without modifiers; the others must be used in combination with at least either the Command or Control key.

Listing by Menu and Command Name

BEdit Menu

| | |
|-------------------------------|---------------------------------------|
| About BEdit | |
| License | |
| Preferences | Cmd-, |
| Setup | |
| Check for Updates... | |
| Install Command Line Tools... | <i>(not in Mac App Store version)</i> |
| Services | <i>(submenu)</i> |
| Hide BEdit | Cmd-H |
| Hide Others/Show All | Cmd-Opt-H |
| Quit BEdit | Cmd-Q |

File

| | |
|------------------------------|------------------------|
| New | <i>(see next page)</i> |
| New With Stationery | <i>(submenu)</i> |
| Open... | Cmd-O |
| Open from FTP/SFTP Server... | Cmd-Ctl-O |
| Open Selection | Cmd-D |
| Reveal Selection | Cmd-Opt-D |
| Open Counterpart | Cmd-Opt-uparrow |
| Open Recent | <i>(submenu)</i> |
| Reopen Using Encoding | <i>(submenu)</i> |
| Close Window | Cmd-Shift-W |
| Close All Windows | Cmd-Opt-W |
| Close Document | Cmd-W |
| Close All Documents | Cmd-Opt-Shift-W |
| Close & Delete | |
| Save | Cmd-S |
| Save All | Cmd-Opt-S |
| Save As... | Cmd-Shift-S |
| Save a Copy... | |
| Save as Styled Text... | |
| Save as Styled HTML... | |
| Save to FTP/SFTP Server... | Cmd-Ctl-S |
| Save a Copy to FTP Server... | Cmd-Opt-Shift-S |
| Revert | |
| Reload from Disk | |
| Save Project | |
| Export | |
| Hex Dump File | |
| Hex Dump Front Document.. | |

| | |
|-----------------|-----------------|
| Page Setup... | |
| Print... | Cmd-P |
| Print All | Cmd-Opt-P |
| Print One Copy | Cmd-Shift-Opt-P |
| Print Selection | |

Edit

| | |
|--------------------------|------------------------|
| Undo | Cmd-Z |
| Redo | Cmd-Shift-Z |
| Clear Undo History | Cmd-Ctl-Z |
| Cut | Cmd-X |
| Cut & Append | Cmd-Shift-X |
| Copy | Cmd-C |
| Copy & Append | Cmd-Shift-C |
| Copy as Styled Text | |
| Copy as Styled HTML | |
| Paste | Cmd-V |
| Paste Previous Clipboard | Cmd-Shift-V |
| Paste Column | Cmd-Ctl-V |
| Clear | |
| Select All | Cmd-A |
| Select None | Cmd-Shift-A |
| Select Line | Cmd-L |
| Select Paragraph | Cmd-Opt-L |
| Complete | F5 |
| Insert | <i>(see next page)</i> |
| Copy Path | <i>(submenu)</i> |
| Show Clipboard | |
| Previous Clipboard | Ctl-[|
| Next Clipboard | Ctl-] |
| Text Options... | Cmd-Opt-, |
| Document Options... | Cmd-Ctl-, |
| Printing Options... | Cmd-Shift-, |
| Normalize Options... | |
| Special Characters... | |

File --> New

Text Document
(with selection)
(with Clipboard)
HTML Document...
Text Window
Project...
Disk Browser
FTP/SFTP Worksheet
Shell Worksheet
Text Factory

Edit --> Insert

File Contents...
File/Folder Paths...
Folder Listing...
Page Break
Short Time Stamp
Full Time Stamp
Emacs Variable Block...

Edit --> Copy Path

Copy Path
Copy Full Path
Copy URL
Copy Name

Text

Apply Text Filter (submenu)
Apply Text Filter [last filter]
Exchange Characters
Exchange Words (Opt)
Change Case...
Change Case (submenu)
Shift Left Cmd-[
Shift Left One Space Cmd-Shift-[
Shift Right Cmd-]
Shift Right One Space Cmd-Shift-]
Un/Comment Selection
Hard Wrap... Cmd-\
Hard Wrap Cmd-Opt-\
Add Line Breaks
Remove Line Breaks
Convert to ASCII
Educate Quotes
Straighten Quotes
Add/Remove Line Numbers...
Add/Remove Line Numbers (Opt)
Prefix/Suffix Lines...
Prefix/Suffix Lines (Opt)
Sort Lines...
Sort Lines (Opt)
Process Duplicate Lines...
Process Duplicate Lines (Opt)
Process Lines Containing...
Process Lines Containing (Opt)
Rewrap Quoted Text... Cmd-'
Rewrap Quoted Text Cmd-Opt-'
Increase Quote Level
Decrease Quote Level
Strip Quotes
Zap Gremlins...
Zap Gremlins (Opt)
Entab...
Entab (Opt)
Detab...
Detab (Opt)
Normalize Line Endings
Find Next Misspelled Word Cmd-;
Find All Misspelled Words Cmd-Opt-;
Clear Spelling Errors

Check Spelling as You
Type
Show/Hide Spelling Panel Cmd-Shift-;

View

Text Display (submenu)
Show/Hide Toolbar
Show/Hide Navigation Bar
Show/Hide Editor
Show/Hide Files Cmd-0
Show/Hide Open Documents
Show/Hide Recent Documents
Show/Hide Worksheet & Scratchpad
Balance Cmd-B
Balance & Fold Cmd-Shift-B
Fold Selection
Unfold Selection
Collapse Enclosing Folds
Collapse All Folds
Expand All Folds
Previous Document Cmd-Opt-[
Next Document Cmd-Opt-]
Move to New Window Cmd-Opt-O
Open in Additional Window
Reveal in Finder
Reveal in Project List
Go Here in Terminal
Go Here in Disk Browser-

View -> Text Display

Show/Hide Fonts Cmd-T
Soft Wrap Text
Show/Hide Page Guide
Show/Hide Tab Stops
Show/Hide Line Numbers
Show/Hide Gutter
Show/Hide Invisibles
Show/Hide Spaces

Search

| | |
|----------------------------------|------------------|
| Find... | Cmd-F |
| Multi-File Search | Cmd-Shift-F |
| Find in Disk Browser or Project | |
| Live Search | Cmd-Opt-F |
| Find Next | Cmd-G |
| Find Previous | Cmd-Shift-G |
| Find All | Cmd-Opt-G |
| Find Selected Text | (Cmd-H or none) |
| Find Previous Selected Text | Cmd-Shift-H |
| Use Selection for Find | Cmd-E |
| Use Selection for Find (grep) | Cmd-Shift-E |
| Use Selection for Replace | Cmd-Opt-E |
| Use Selection for Replace (grep) | Cmd-Opt-Shift-E |
| Replace | Cmd-= |
| Replace All | Cmd-Opt-= |
| Replace to End | Cmd-Shift-= |
| Replace & Find Next | |
| Go to Line... | Cmd-J |
| Go to Line | Cmd-Opt-J |
| Go to Center Line | Cmd-Shift-J |
| Go to Previous Error | Cmd-Opt-uparrow |
| Go to Next Error | Cmd-Opt-dnarrow |
| Go to Previous Placeholder | Ctl-Shift-` |
| Go to Next Placeholder | Ctl-` |
| Go to Function Start | |
| Go to Function End | |
| Go to Previous Function | |
| Go to Next Function | |
| Jump Back | |
| Jump Forward | |
| Set Jump Mark | |
| Find Differences... | |
| Compare Two Front Documents | |
| Compare Against Disk File | |
| Apply to New | Cmd-left arrow |
| Apply to Old | Cmd-right arrow |
| Compare Again | |
| Find Definition | Cmd-hyphen |
| Find in Reference | Cmd-Shift-hyphen |

Markup

See "Markup" on page 333.

Window

| | |
|---------------------------------|----------------|
| Minimize Window | |
| Minimize All Windows | (Opt) |
| Bring All to Front | |
| Palettes | (see below) |
| Workspace | (see below) |
| Show Scratchpad | |
| Show Unix Worksheet | |
| Save Default [type of] Window | |
| Arrange | |
| Tile Two Front Windows | (Opt) |
| Get Info | |
| Reveal in Finder | |
| Cycle Through Windows | Cmd-^ |
| Cycle Through Windows Backwards | Cmd-Shift-^ |
| Exchange With Next | |
| Synchro Scrolling | (Open windows) |

Window -> Palettes

| |
|--------------------|
| Hide/Show Palettes |
| ASCII Table |
| Clippings |
| Colors |
| Filters |
| Scripts |
| Stationery |
| Windows |
| HTML Markup Tools |
| CSS |
| Entities |
| Utilities |

Window -> Workspace

| |
|---------------------|
| Save Workspace... |
| Delete Workspace... |

Shebang (#!)

Check Syntax
Check Selection Syntax (Opt)
Run
Run...
Run in Terminal
Run in Debugger
Run File...
Show POD/Show Module
Documentation

P4

Edit
Revert
Revert & Sync to Head
Sync to Revision...
Sync All
Submit...
Show Opened
Compare Revisions...
Compare Arbitrary
Revisions...
Compare with Base...
Compare with Head
Show Annotation
Show Revision History...
Add
Revert & Delete
Open Log File
Go to Previous Conflict
Go to Next Conflict
Open Log File

CVS

Update to Head
Update to Head... (Shift)
Update to Revision...
Update Folder...
Commit...
Commit Parent Folder...
Commit Parent Folder No (Shift-Opt)
Comment
Commit File... (Shift)
Commit Without (Shift-Opt)
Comment
Update Working Copy...
Commit Working Copy...
Show Working Copy (Shift)
Status...
Compare Revisions...
Compare Arbitrary
Revisions
Compare with Base
Compare with Head
Show Annotation
Show Annotation... (Shift)
Show Revision History
Show Revision History... (Shift)
Show CVS Status
Add
Add... (Shift)
Remove
Remove... (Shift)
Go to Previous Conflict
Go to Next Conflict
Export Project...
Open Log File

Subversion

Revert
Update to Head
Revert & Update to Head (Shift)
Update to Revision...
Commit...
Update Working Copy...
Commit Working Copy...
Show Working Copy
Status...
Compare Revisions...
Compare Arbitrary
Revisions...
Compare with Base
Compare with Previous
Compare with Head
Resolved
Show Annotation
Show Revision History...
Show Revision History (Opt)
Add
Delete
Revert & Delete (Shift)
Go to Previous Conflict
Go to Next Conflict
Open Log File

Scripts

Open Script Editor
Open Scripting Dictionary
Open Scripts Folder
Start/Stop Recording
*(Installed scripts &
factories)*

Clippings

Open Clippings Folder
Insert Clipping...
Change Set (submenu)
Save as Clipping...
Save Selection as
Clipping...
(Available clippings)

Help

Search
BBEdit Help
User Manual
Tutorial
Service and Support

Toolbar (*in-window*)**Text Options** (popup menu)

Soft Wrap Text
 Show/Hide Page Guide
 Show/Hide Tab Stops
 Show/Hide Line Numbers
 Show/Hide Gutter
 Show/Hide Invisibles
 Show/Hide Spaces
 Use Typographer's Quotes
 Auto-Expand Tabs

Navigation Bar (*in-window*)

Open Files Menu Ctl-Opt-F
Open Function Menu Ctl-Opt-N
Open Includes Menu Ctl-Opt-I
Open Marker Menu Ctl-Opt-M
Markers (popup menu)

Set Marker... (Opt)
 Set Marker (Opt)
 Clear Markers... (Opt)
 Clear All Markers (Opt)
 Find & Mark All... (Opt)
 Find & Mark All (Opt)

Status Bar (*in-window*)

Open Language Menu
Open Text Encodings Menu
Open Breaks Menu Ctl-Opt-B
Line Breaks (popup menu)

Macintosh
 Unix
 DOS

Miscellaneous Commands

Zoom Window Cmd-/
 Zoom All Windows Cmd-Opt-/
 Zoom Window Full Screen Cmd-Opt-Ctl-/
 Zoom All Windows Full Screen
 Open URL (Cmd-click in URL)

Markup

Edit Markup... Cmd-Ctl-M
 Close Current Tag
 Balance Tags Cmd-Opt-B
 Document Type...
 Character Set...
 CSS (*see below*)
 Body Properties...
 Head Elements (*see below*)
 Block Elements (*see next column*)
 Lists (*see next column*)
 Tables (*see page 335*)
 Forms (*see page 335*)
 Inline Elements (*see page 335*)
 Phrase Elements (*see page 336*)
 Font Style Elements (*see page 336*)
 Frames (*see page 336*)
 Check (*see page 336*)
 Update (*see page 337*)
 Includes (*see page 337*)
 Utilities (*see page 337*)
 Tidy (*see page 337*)
 Preview in BBEdit Cmd-Ctl-P
 Refresh BBEdit Preview
 Show Document
 Preview in <Default Browser>
 Preview in (*see page 337*)

Markup --> CSS

@import...
 @media...
 Box...
 Padding...
 Border...
 Margins...
 Layout...
 Position...
 Size & Constraints...
 Clipping...
 Effects...
 Background...
 Font...
 List Style...
 Text...
 Format

Markup --> Head Elements

Base...
Link...
Meta...
Script...
Noscript
Style...

Markup --> Block Elements

Paragraph...
Paragraph (Opt)
Div...
Horizontal Rule...
Horizontal Rule (Opt)
Heading... Cmd-Ctl-H
H1
H2
H3
H4
H5
H6
Address
Blockquote...
Center
Deleted Text...
Inserted Text...
Noscript
Preformatted

Markup --> Lists

List... Cmd-Ctl-L
Unordered
Ordered
Definition
Menu
Directory
List Item
List Items

Markup --> Tables

| | |
|---------------------|-----------|
| Table... | Cmd-Ctl-T |
| Row... | |
| Row | (Opt) |
| TD... | |
| TD | (Opt) |
| TH... | |
| TH | (Opt) |
| Caption | |
| Colgroup... | |
| Col... | |
| THead... | |
| TFoot... | |
| TBody... | |
| Convert to Table... | |

Markup --> Forms

| | |
|-----------------|--|
| Form... | |
| Button... | |
| Field Set | |
| Legend... | |
| Input... | |
| Label... | |
| Select... | |
| Option Group... | |
| Option... | |
| Text Area... | |

Markup --> Inline

| | |
|---------------------------|-----------|
| Anchor... | Cmd-Ctl-A |
| Image... | Cmd-Ctl-I |
| Applet... | |
| Object... | |
| Param... | |
| Script... | |
| Map... | |
| Area... | |
| Break... | Cmd-Ctl-B |
| Break | |
| Font... | |
| Base Font... | |
| Bidirectional Override... | |
| Quotation | |
| Span... | |
| Subscript | |
| Superscript | |

Markup --> Phrase Elements

Abbreviation
Acronym
Citation
Computer Code
Defined Term
Deleted Text...
Emphasis
Inserted Text...
Input Text (Kbd)
Sample Output
Strong Emphasis
Variable

Markup --> Font Style Elements

Big
Small
Bold
Italic
Strike-Through
Teletype Text
Underline

Markup --> Frames

Frame Set...
Frame...
No Frames

Markup --> Check

| | |
|------------------|-----------|
| Document Syntax | Cmd-Ctl-Y |
| Document Links | Cmd-Ctl-K |
| Folder Syntax... | |
| Folder Links... | |
| Site Syntax... | |
| Site Syntax | (Opt) |
| Site Links... | |
| Site Links | (Opt) |

Markup --> Update

| | |
|--------------------|-----------|
| Document | Cmd-Ctl-U |
| Folder... | |
| Site... | |
| Site | (Opt) |
| Document Images... | |
| Document Images | (Opt) |
| Folder Images... | |
| Site Images... | |
| Site Images | (Opt) |

Markup --> Includes

Persistent Include...
Include...
Placeholders...

Markup --> Utilities

Format... Cmd-Opt-Shift-F
Format
Optimize
Translate Text to HTML
Translate HTML to Text
Remove Comments
Remove Markup
Comment
Uncomment
Raise Tag Case
Lower Tag Case

Markup --> Tidy

Clean Document...
Reflow Document...
Convert to XHTML
Convert to XML
Check Accessibility

Markup --> Preview In

New Text Window
in All Running Browsers
(Installed browsers)

Listing by Default Key Equivalent

| Key | Command |
|------------|--------------------------------------------------------------|
| Cmd-0 | View: Show/Hide Files |
| Cmd-A | Edit: Select All |
| Cmd-B | Text: Balance |
| Cmd-C | Edit: Copy |
| Cmd-D | File: Open Selection <i>or</i> File: Open File by Name |
| Cmd-E | Search: Use Selection for Find |
| Cmd-F | Search: Find... |
| Cmd-G | Search: Find Again |
| Cmd-H | Search: Find Selection <i>or</i> BBEdit: Hide BBEEdit |
| Cmd-J | Search: Go to Line... |
| Cmd-L | Edit: Select Line |
| Cmd-N | File: New: Text Document |
| Cmd-O | File: Open... |
| Cmd-P | File: Print... |
| Cmd-Q | BBEdit: Quit BBEEdit |
| Cmd-S | File: Save |
| Cmd-T | View: Text Display: Show/Hide Fonts |
| Cmd-U | Compiler: Bring Up To Date |
| Cmd-V | Edit: Paste |
| Cmd-W | File: Close Document/Close Window |
| Cmd-X | Edit: Cut |
| Cmd-Z | Edit: Undo |
| Cmd-, | BBEdit: Preferences |
| Cmd-` | Window: Cycle Through Windows |
| Cmd-- | Search: Find Definition |
| Cmd-; | Text: Find Next Misspelled Word |
| Cmd-^ | Text: Rewrap Quoted Text... |
| Cmd-[| Text: Shift Left |

| Key | Command |
|---------------------|------------------------------------|
| Cmd-] | Text: Shift Right |
| Cmd-/ | Zoom Window |
| Cmd-= | Search: Replace |
| Cmd-\ | Text: Hard Wrap... |
| Cmd-left arrow | Search: Apply to New |
| Cmd-right arrow | Search: Apply to Old |
| Cmd-Ctrl-A | Markup: Inline: Anchor... |
| Cmd-Ctrl-B | Markup: Inline: Break... |
| Cmd-Ctrl-H | Markup: Block Elements: Heading... |
| Cmd-Ctrl-I | Markup: Inline: Image... |
| Cmd-Ctrl-K | Markup: Check: Document Links |
| Cmd-Ctrl-L | Markup: Lists: List... |
| Cmd-Ctrl-M | Markup: Edit Markup... |
| Cmd-Ctrl-N | File: New: HTML Document... |
| Cmd-Ctrl-O | File: Open from FTP/SFTP Server... |
| Cmd-Ctrl-P | Markup: Preview in BBEdit |
| Cmd-Ctrl-S | File: Save to FTP/SFTP Server... |
| Cmd-Ctrl-T | Markup: Tables: Table... |
| Cmd-Ctrl-U | Markup: Update: Document |
| Cmd-Ctrl-V | Edit: Paste Column |
| Cmd-Ctrl-X | Markup: Misc: Index Site... |
| Cmd-Ctrl-Y | Markup: Check: Document Syntax |
| Cmd-Ctrl-Z | Edit: Clear Undo History |
| Cmd-Ctrl-, | Edit: Document Options |
| Cmd-Ctrl-/ | Zoom Window Full Screen |
| Cmd-Ctrl-down arrow | Search: Go to Previous Error |
| Cmd-Ctrl-up arrow | Search: Go to Next Error |
| Cmd-Opt-B | Markup: Check: Balance Tags |
| Cmd-Opt-D | File: Reveal Selection |

| Key | Command |
|------------------|------------------------------------------|
| Cmd-Opt-E | Search: Use Selection for Replace (grep) |
| Cmd-Opt-F | Search: Live Search |
| Cmd-Opt-G | Search: Find All |
| Cmd-Opt-H | BEdit: Hide Others |
| Cmd-Opt-J | Search: Go to Line |
| Cmd-Opt-L | Edit: Select Paragraph |
| Cmd-Opt-M | Markup: Edit Markup... |
| Cmd-Opt-N | File: New: Disk Browser |
| Cmd-Opt-O | View: Move to New Window |
| Cmd-Opt-P | File: Print All |
| Cmd-Opt-R | Compiler: Run |
| Cmd-Opt-S | File: Save All |
| Cmd-Opt-W | File: Close All Windows |
| Cmd-Opt-, | Edit: Text Options |
| Cmd-Opt-; | Edit: Find All Misspelled Words |
| Cmd-Opt-' | Text: Rewrap Quoted Text |
| Cmd-Opt-[| View: Previous Document |
| Cmd-Opt-] | View: Next Document |
| Cmd-Opt= | Search: Replace All |
| Cmd-Opt-/ | Zoom All Windows |
| Cmd-Opt-\ | Text: Hard Wrap |
| Cmd-Opt-up arrow | File: Open Counterpart |
| | |
| | |
| Cmd-Opt-Shift-E | Search: Enter Replace Pattern |
| Cmd-Opt-Shift-F | Markup: Utilities: Format... |
| Cmd-Opt-Shift-I | Compiler: Set Breakpoint & Go |
| Cmd-Opt-Shift-S | File: Save a Copy to FTP Server.. |
| Cmd-Opt-Shift-W | File: Close All Documents |
| | |
| | |

| Key | Command |
|-------------|----------------------------------------|
| Cmd-Shift-A | Edit: Select None |
| Cmd-Shift-B | View: Balance & Fold |
| Cmd-Shift-C | Edit: Copy & Append |
| Cmd-Shift-E | Search: Use Selection for Find (grep) |
| Cmd-Shift-F | Search: Multi-File Search |
| Cmd-Shift-G | Search: Find Previous |
| Cmd-Shift-I | Compiler: Set Breakpoint |
| Cmd-Shift-J | Search: Go to Center Line |
| Cmd-Shift-N | File: New: Text Window |
| Cmd-Shift-P | File: Page Setup |
| Cmd-Shift-S | File: Save As... |
| Cmd-Shift-V | Edit: Paste Previous Clipboard |
| Cmd-Shift-W | File: Close Window <i>{special}</i> |
| Cmd-Shift-X | Edit: Cut & Append |
| Cmd-Shift-Z | Edit: Redo |
| Cmd-Shift-, | Edit: Printing Options |
| Cmd-Shift-` | Misc.: Cycle Through Windows Backwards |
| Cmd-Shift-- | Search: Find in Reference |
| Cmd-Shift-; | Text: Show Spelling Panel |
| Cmd-Shift-[| Text: Shift Left One Space |
| Cmd-Shift-] | Text: Shift Right One Space |

| Key | Command |
|-----------------|------------------------------------------|
| Ctl-` | Search: Go to Next Placeholder |
| Ctl-[| Edit: Previous Clipboard |
| Ctl-] | Edit: Next Clipboard |
| Ctl-Tab | Switch to Header/Source File |
| Ctl-Opt-F | Navigation Bar: Open Files Menu |
| Ctl-Opt-I | Navigation Bar: Open Includes Menu |
| Ctl-Opt-M | Navigation Bar: Open Marker Menu |
| Ctl-Opt-N | Navigation Bar: Open Function Menu |
| Cmd-Shift-Opt-E | Search: Use Selection for Replace (grep) |
| Cmd-Shift-Opt-P | File: Print One Copy |

Editing Shortcuts

In BBEdit you can perform many editing functions (including word selection or deletion) directly from the keyboard. Chapter 4 contains complete details on BBEdit's text editing features. This appendix provides a quick reference to available keyboard and mouse shortcuts for word selection and deletion.

In this appendix

| | |
|-----------------------------|-----|
| Mouse Commands | 343 |
| Arrow and Delete Keys | 344 |
| Emacs Key Bindings | 345 |

Using universal-argument – 346

Mouse Commands

| | No Modifier | Shift |
|--------------|----------------------|--------------------------|
| Click | move insertion point | extend selection |
| Double-click | select word | extend selection to word |
| Triple-click | select line | –none– |

Triple-clicking is the same as clicking in a line and then choosing the Select Line command from the Edit menu.

Holding down the Command or Option keys as you click or double-click triggers special actions:

| | Option | Command | Command/ Option |
|--------------|--------|----------|-----------------------------------------|
| Click | –none– | Open URL | –none– |
| Double-click | –none– | –none– | find next instance of the selected text |

Arrow and Delete Keys

You can use the arrow keys to move the insertion point right, left, up, and down. You can augment these with the Command and Option keys to move by word, line, or screens, or with the Shift key to create or extend selections. For example, pressing Shift-Option-Right Arrow selects the word to the right of the insertion point.

You can hold down the Control key while using the arrow keys to scroll through editing windows without moving the position of the insertion point.

| Key | Modifier | Action |
|--------------------|----------|--------------------------------------------------------------------------------------------------------|
| (left/right) Arrow | | Move 1 character left/right |
| (left/right) Arrow | Option | Move 1 word left/right |
| (left/right) Arrow | Command | Move to beginning/end of line |
| (left/right) Arrow | Control | Jump to the previous/next character transition from lower case to upper case OR the next word boundary |
| (up/down) Arrow | | Move up/down 1 line in file |
| (up/down) Arrow | Command | Move to top/bottom of file |
| (up/down) Arrow | Option | Move to previous/next screen page |
| (up/down) Arrow | Control | Scroll view up/down |
| [any of the above] | Shift | Make or extend a selection range |
| Delete | | Deletes selection range, or character preceding (to the left of) the insertion point. |
| Delete | Command | Deletes all characters backwards to beginning of line |
| Delete | Option | Deletes all characters back to beginning of word |
| Delete | Shift | (same as Forward Delete) |
| Forward Delete | | Deletes selection range, or character after (to the right of) the insertion point |
| Forward Delete | Command | Deletes all characters forward to end of the current line |
| Forward Delete | Option | Deletes all characters forward to end of word |
| Forward Delete | Shift | (same as Forward Delete alone) |

Note The meaning of the Command and Option modifiers listed above may be exchanged, depending on which settings you have selected for Exchange Command and Option Key Behavior in the Keyboard preference panel .

Emacs Key Bindings

The Keyboard preference panel contains an option labelled Use Emacs Key Bindings. When this option is on, BBEdit will accept the following Emacs-style keyboard navigation commands. The Escape key is used in lieu of the Emacs “Meta” key; to type these key equivalents, press and release the Escape key followed by the specified letter key—for example, to type “Esc-V” press and release the Escape key and then type the letter V.

| Key Sequence | Action |
|--------------|----------------------------------------------------------------------------|
| Ctl-A | beginning-of-line (Move insertion point to start of current line) |
| Ctl-B | backward-char (Move insertion point backward 1 place) |
| Ctl-D | delete-char (Delete forward 1 character) |
| Ctl-E | end-of-line (Move insertion point to end of current line) |
| Ctl-F | forward-char (Move insertion point forward 1 place) |
| Ctl-G | keyboard-quit (cancel pending arguments) |
| Ctl-K | kill-line (Delete to end of current line) |
| Ctl-L | recenter (Scrolls the current view so the selection is centered on screen) |
| Ctl-N | next-line (Move insertion point down one line) |
| Ctl-O | open-line (Inserts line break without moving insertion point) |
| Ctl-P | previous-line (Move insertion point up one line) |
| Ctl-R | isearch-backward (Live Search backward) |
| Ctl-S | isearch-forward (Live Search forward) |
| Ctl-T | transpose-chars (Exchange Characters) |
| Ctl-U | universal-argument (<i>See note below</i>) |
| Ctl-V | scroll-up (Page down) |
| Ctl-W | kill-region (Cut) |
| Ctl-Y | yank (Paste) |
| Ctl-_ | undo (Undo) |

| Key Sequence | Action |
|--------------|-----------------------------------------------------------------|
| Ctl-X Ctl-C | save-buffers-kill-emacs (Quit) |
| Ctl-X Ctl-F | find-file (Open file) |
| Ctl-X Ctl-S | save-buffer (Save current document) |
| Ctl-X Ctl-W | write-file (Save As) |
| Esc-< | beginning-of-buffer (Move insertion point to start of document) |
| Esc-> | end-of-buffer (Move insertion point to end of document) |
| Esc-Q | fill-paragraph (Hard Wrap with current settings) |
| Esc-V | scroll-down (Page up) |
| Esc-W | copy-region-as-kill (Copy) |
| Esc-Y | yank-pop (Paste Previous Clipboard) |

Using universal-argument

The universal-argument command (Ctl-U) does not work quite the same way as it does in Emacs. In BBEdit, it is a simple repeat-count. For example, if you type Ctl-U, then a 3, and then Ctl-N, the insertion point will move down three lines. There is no visual feedback as you type the number, and no way to backspace or otherwise edit the number. If you make a mistake, the best you can do is type Ctl-G (keyboard-quit) and start over.

C

Placeholders and Include Files

This appendix lists the placeholder tokens used by BBEdit templates and include files, and describes the use and capabilities of include files.

In this appendix

| | |
|--------------------------------------------------------------------------|-----|
| Placeholders | 347 |
| <i>Date Formats – 350 • Using the #RELATIVE# Placeholder – 351</i> | |
| Include Files..... | 352 |
| <i>Simple Includes – 352 • Persistent Includes – 353</i> | |
| <i>Include Files with Variables – 353 • Including AppleScripts – 354</i> | |
| <i>Including Unix Scripts – 355 • Other Include Notes – 357</i> | |

Placeholders

Placeholders are processed under the following circumstances:

- When a new HTML document is created from a template, the placeholders in the template are replaced with their current values. (The new document receives the substituted text; the original template file is not modified.)
- When the Update Document command (part of the HTML Tools) is invoked, any placeholders in the documents being updated are replaced with their current values. (Since the placeholders are replaced, subsequent updates *do not* update the substituted text.) Although this command is part of the HTML Tools, it can be used in any document whenever you want to use placeholders.
- When a file is included in another file using the `#bbinclude` directive (or a related directive), any placeholders in the included file are replaced with their current values before the text is included. (The include file itself is not changed, only the included text is substituted.) All of the above methods of invoking placeholders can also invoke included files, which can have placeholders of their own.

Note The placeholders described in this chapter are only available for use with the HTML Tools' Update command. They cannot be used with BBEdit's Clippings command, nor can Clippings placeholders be used in include or template files.

BBEdit supports the following placeholders. Placeholders are not case-sensitive.

| Placeholder | Replaced By... |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #ABBREVDATE# | Abbreviated date—e.g., Sun, Aug 16, 2009 |
| #BASE# | The BASE tag as entered using the New HTML Document command |
| #BASENAME# | The name of the file stripped of its rightmost period-delimited portion. For example, if the file is named "test.html", the base name is "test", while if the file is named "test.foo.html", the base name is "test.foo". |
| #BASE_URL# | The value of the BASE URL specified in an HTML document's header (useful if you want to refer to the document's location on the server) |
| #CHARSET# | The character set specified in the New HTML Document command |
| #COMPDATE# | Compact Date format—e.g., 16-Aug-09 |
| #CREATIONDATE# | The creation date of the current file—e.g., 16-Aug-09 |
| #CREATIONTIME# | The creation time of the current file, formatted according to your Format settings in the International panel of the System Preferences |
| #DATETIME XXX# | Inserts a localized, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below) |
| #DATETIME_GMT XXX# | Inserts the universal, region-aware date whose format is specified by the ICU format string XXX (see "Date Formats" below) |
| #DIRPATH# | The path on the server as specified in the Sites panel of the Setup window. Strips any leading slash from the path string |
| #DOCSIZE# | The size of the current document plus included images in bytes |
| #DOCTITLE# | The title of the current document as extracted from the <TITLE> tag |
| #DONT_UPDATE# | Marks a document so that the HTML Update tool will ignore it during processing |
| #FILE_EXTENSION# | The filename extension for the file (determined as the rightmost period-delimited portion of the filename, without the period). For example, whether the file is named "test.html" or "test.foo.html", the filename extension is "html". |
| #FILENAME# | The file name of the current file |
| #GENERATOR# | Generator name used for "Give BBEdition Credit" in New HTML Document function (e.g., "BBEdit 10") |

| Placeholder | Replaced By... |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| #GMTIME YYYY# | The current GMT time formatted according to the parameters YYYY (see "Time Formats" below) |
| #LANGUAGE# | The language specified in the New HTML Document command, in format (space)lang="en" |
| #LINK# | The LINK tag as entered using the New HTML Document command |
| #LOCALPATH# | The full local path to the current file |
| #LOCALTIME YYYY# | The current local time formatted according to the parameters YYYY (see "Time Formats" below) |
| #LONGDATE# | Long Date format—e.g., Sunday, August 19, 2009 |
| #MACHINE# | The machine name as specified in the Sharing section of the System Preferences. |
| #META# | Any META tag entered using the New HTML Document command |
| #MODIFIEDDATE# | Modification date of the current file—e.g., 16-Aug-09 |
| #MODIFIEDTIME# | Modification time of the current file, in the format specified in the International section of the System Preferences |
| #MONTHDAYNUM# | Numeric value of the day of the month |
| #MONTHNUM# | Numeric value of the current month |
| #PATH# | Path to access your documents from the Web server, as specified in your HTML Web Site preferences |
| #PREFIX# | As #DIRPATH# but does not strip the leading slash of the path |
| #REAL_URL# | The real URL for the current document in its current location |
| #RELATIVE# | The relative path from the current file back up to the Local Server Root (inserts a path of the form "../.." to tell the browser to "back up" to the site's root directory) |
| #ROOT# | Path to the Local Site Root, as specified in your HTML Web Site preferences |
| #ROOTPATH# | The file's path relative to the Local Server Root specified in the HTML Web Site preferences to the current file |
| #SERVER# | URL of your Web server, as specified in your HTML Web Site preferences. |
| #SHORTDATE# | Short Date. Day, month, year—e.g., 08/16/09 |

| Placeholder | Replaced By... |
|-----------------|------------------------------------------------------------------------------------------------------|
| #SHORTUSERNAME# | Returns the login (short) name instead of the full user name |
| #TIME# | Current time, according to your Format settings in the International panel of the System Preferences |
| #TITLE# | Title of the current document as entered using the New HTML Document command |
| #USERNAME# | The owner name (for the currently logged in user) |
| #YEARNUM# | The current year—e.g., 2009 |

Date Formats

The #DATETIME XXX# and #DATETIME XXX# placeholders allow you to insert the corresponding date and time values with flexible formatting.

In order to use these placeholders, you must substitute XXX with an ICU date/time format string. ICU is the mechanism used by Mac OS X for date formatting. For full details, please refer to the section “Date/Time Format Syntax” in the ICU documentation:

<http://icu.sourceforge.net/userguide/formatDateTime.html>

Examples:

```
#DATETIME EEE, MMM d, yy 'at' h:mm a#
```

produces:

Tue, Jul 3, 09 at 5:48 PM

```
#DATETIME_GMT EEE, MMM d, yy 'at' h:mm a#
```

produces:

Tue, Jul 3, 09 at 9:49 PM

```
#DATETIME EEEE 'at' h 'o'clock' a#
```

produces:

Tuesday at 5 o'clock PM

Time Formats

The #GMTIME YYY# and #LOCALTIME YYY# placeholders offer you the option to insert the specified time value with flexible formatting.

In order to use these placeholders, you must substitute YYY with a time format using the same expansion options offered by the ‘strftime’ routine (see ‘man strftime’ for further details).

Examples:

```
#LOCALTIME %r %z on %A#
```

produces:

```
06:50:13 PM -0400 on Monday
```

```
#GMTIME %r %z#
```

produces:

```
10:50:13 PM +0000
```

Using the #RELATIVE# Placeholder

When dealing with large web sites that have multiple content folders, it is often useful to specify relative rather than absolute paths for linking documents. The #RELATIVE# placeholder allows you to easily generate relative references in templates and include files by providing a virtual path that uses the “..” construction to “back up” the hierarchy to the root directory of the site.

To use this placeholder, write your links as if they were all relative to the top of your web site, including #RELATIVE# as the first “directory” in the path. For example, consider that you have the following file structure, where each page includes a file which references the separate GIF image.

```
My_Web_Site:
  Folder1:
    File1.html
  Folder2:
    File2.html
    File3.html
  Folder3:
    Folder4:
      Folder5:
        File4.html
  Graphics:
    Buttons:
      my_footer_button.gif
```

If you write a relative link as follows:

```

```

and then run the Update command, the following links will be generated.

In File1.html,

```
../Graphics/Buttons/my_footer_button.gif
```

In File2.html,

```
../Graphics/Buttons/my_footer_button.gif
```

In File3.html,

```
../Graphics/Buttons/my_footer_button.gif
```

In File4.html,

```
../../../../Graphics/Buttons/my_footer_button.gif
```

Include Files

An include file, or just an “include,” is a special form of placeholder whose substitution happens to be the contents of another file. If you have used C or certain other programming languages, you may already be familiar with the concept. Using includes, you can reuse standard bits of text content or HTML markup in several templates or clippings entries without having to revise all of those individual files whenever you revise the included text.

Include File Locations

BBEdit looks for include files first in the same directory as the document containing the directive, then in the same directory as the document into which the processed document is being inserted, and finally in the HTML Templates folder specified by the configuration for the current site in the Sites panel of the Setup window.

Simple Includes

A simple include takes the following form:

```
#bbinclude "filename"
```

where filename is the full name of the file whose contents you wish to include. When such an include is present in a template or clippings entry, it is replaced with the contents of the specified file when the template is used to build a new document, or when the clippings entry is inserted. (The original template or clippings file is not changed.)

Imagine that you have ten different templates, each of which contains your name, address, phone number, email address, and a copyright statement with the current year in them. Rather than pasting this info into all ten templates, you can create a file named “address.html”, put it in your Templates folder, and include this statement:

```
#bbinclude "address.html"
```

in each of the templates, at the appropriate point. Later, when the new year arrives, or you move, you only have to update one file, not all ten templates. (You could use the #YEARNUM# placeholder for the year and only need to update the include file when you move!)

Headers and footers are probably the most common uses for include files, but any template or clippings entry may use as many include statements as you wish. Included files themselves may also use #bbinclude directives, up to 16 levels deep.

Persistent Includes

Simple includes are appropriate for use situations where you want the inclusion to happen only once. Once the file has been included, however, it cannot be changed in any automated fashion. Since the `#bbinclude` directive is replaced by the included text, the Update tool cannot tell the included text is any different from any other text.

Includes become even more powerful, however, when you can update existing files to incorporate revised include text at a later date. For example, suppose you generate several dozen HTML documents using a template that uses an `#bbinclude` directive to insert a standard footer containing your email address. Later, you change your email address. After you change it in the footer document, only *new* HTML files you create from the template will include your new address. What you would really like to be able to do is update all the files you have *already* created to include the revised footer.

Since this capability is needed primarily in web site maintenance, BBEdit lets you embed the include directive in an HTML comment. An “end `bbinclude`” comment is also required. The included text is inserted *between* the two comment markers, but the comments themselves remain in place. The comments are not shown in the browser. This is known as a *persistent* include.

A persistent include looks like this:

```
<!-- #bbinclude "filename" -->
<!-- end bbinclude -->
```

The first time a persistent include is processed, it is handled much like a simple include. However, since the include directives remain in place, and because they mark the beginning and end of the inserted text, the Update tool can “rip out” the obsolete included text and replace it with the updated file. Using persistent includes and the Update Folder or Update Site commands, you can easily make these sorts of changes to entire sites in moments.

IMPORTANT

Any changes you have made to the included text after its initial inclusion will be discarded when the persistent include is updated, even if you have not changed the include file.

Inline versus Block Includes

By default, BBEdit places included content in the document as a block, to ensure that it does not occupy the same line as the include directives. However, if you wish to override this behavior and have BBEdit place the included content inline, you may do so by adding the special option `#bbincludeoptions#="inline=true"` to the include directive.

```
<!-- #bbinclude "filename" #bbincludeoptions#="inline=true" -->
<!-- end bbinclude -->
```

Include Files with Variables

Include files can be extended even further through the use of variables, which provide a means of inserting arbitrary text when the included file is processed, so that not all instances of the included file are exactly the same. Variables are essentially placeholders that you make up yourself. Some possible uses are to insert names, taglines, alt strings for images, or file names (for files other than the current document) into documents.

Note A variable name consists of a string of alphanumeric characters, enclosed in number signs (the `#` character). Spaces are not allowed in variable names, but underscores may be used to represent word breaks.

Variables can be placed anywhere in an include file, just like placeholders. When you include that file in a document, you specify the variable names and values with it. Consider an include file named "footer.html", which contains the following

```
<HR>
<IMG SRC="#MY_GRAPHIC#" ALT="#MY_ALT_DESC#">
<H1>#MY_TITLE#</H1>
<BIG>This document copyright 1998-2009 by Sid Zookim.</BIG>
```

In your document, the Include reference would look like this:

```
...
<BODY>
...
<!-- #bbinclude "footer.html"
#MY_GRAPHIC#="test1.gif"
#MY_ALT_DESC#"a test image"
#MY_TITLE#"A Test Title"
-->
<!-- end bbinclude -->
...
</BODY>
...
```

Note that the values of placeholders are specified *inside* the HTML comment of a persistent include, using a #PLACEHOLDER#="Value" syntax. The quote marks around the value are mandatory; if you need to include a quote mark in the actual value, escape it with a backslash.

Including AppleScripts

BBEdit allows included files to be compiled AppleScript scripts. The script should contain an "on include" handler which is passed two parameters: a reference to the file from which the script is being called, and a record containing one field for each variable passed in the #bbinclude directive. Scripts can of course also retrieve information from BBEdition, other scriptable applications, or the system. The handler's return value is inserted into the file that included it.

Given the HTML document below:

```

<html>
<head>
  <title>Include Test</title>
  <meta name="generator" content="BBEdit 10">
</head>
<body>

<!-- #bbinclude "foo.script" #x#="3" #author#="JEK"-->

<!-- end bbinclude -->

</body>
</html>

```

The following script inserts three lines: the first containing the file's path, the second containing the parameter "x" passed to it in the #bbinclude directive, and the third containing the parameter "author."

```

on include(f, vars)
  set s to f as text
  set s to "File Path: " & s & return & return as text
  set s to s & "x: " & x of vars & return & return as text
  set s to s & "Name: " & author of vars & return as text
  return s
end include

```

The resulting document might look like this:

```

<html>
<head>
  <title>Include Test</title>
  <meta name="generator" content="BBEdit 10">
</head>
<body>

<!-- #bbinclude "foo.script" #x#="3" #author#="JEK"-->
File Path:  Boot:Desktop Folder:incl_test.html

x:  3

Name:  JEK
<!-- end bbinclude -->

</body>
</html>

```

Including Unix Scripts

BBEdit lets you include scripts written in Perl, Python, Ruby, or any other Unix scripting language. The complete path name of the file being processed is passed to the script as its first argument, and any variables in the include statement are passed as additional arguments. (For Perl, all these can be retrieved by your script via @ARGV.)

Any text sent to STDOUT by the script will be taken as the value of the `#bbinclude` operation and inserted into the HTML file. If an error occurs while running the script, the `STDERR` output, if any, will be inserted into the file along with the `STDOUT`, and a single line indicating the error will be added to the error browser.

For example, enter this directive in your HTML file:

```
<!-- #bbinclude "foo.pl" #length#="2" #width#="3" -->
<!-- end bbinclude -->
```

Then use this source code for “foo.pl”, and save it in the same folder as the HTML file, or in the “Templates and Includes” folder specified in this web site’s configuration (as defined in the Sites panel of the Setup window):

```
#!/usr/bin/perl -w
my $file = shift @ARGV;
my %args = @ARGV;
my $area = $args{"length"} * $args{"width"};
print "Filename: $file\n";
print "Area: $area\n";
```

When you run the Update command, BBEdit will place the file name in the script’s variable `$file` and the “length” and “width” variables in the associative array (hash) `%args`.

After the update, the BBEdit file will look like this:

```
<!-- #bbinclude "foo.pl" #length#="2" #width#="3" -->
Filename: Mac HD:Desktop Folder:sample.html
Area: 6
<!-- end bbinclude -->
```

In addition, BBEdit will pass information about the current HTML Tools settings to the script in the following environment variables:

```
BBEditServerURL
BBEditServerPath
BBEditDefaultFileName
BBEditTemplateDirectory
BBEditRootDirectory
BBEditLowercaseTags
BBEditLowercaseAttributes
BBEditAlwaysQuoteAttributes
```

To access these in your Perl code, use the `%ENV` environment variable hash. For example, this line of Perl will print the web server name specified by this web site’s configuration (in the Sites panel of the Setup window):

```
print $ENV{BBEditServerURL};
```

Here's an example Python include script.

```
#!/usr/local/bin/python
import os
import string
import sys

print "Hello Python World!"
print "======"
print "File being updated: ", sys.argv[1]
print

userVariables = {}
for i in range(2, len(sys.argv), 2):
    userVariables[sys.argv[i]] = sys.argv[i+1];

print
print "Dumping the user variables passed to the script"
print "======"
print
keys = userVariables.keys();
keys.sort()
for k in keys:
    print "%-30s %s" % (k, userVariables[k])

print
print "Dumping the environment variables set by BBEdit"
print "======"
print
for k, v in os.environ.items():
    if (string.find(k, 'BBEdit') == 0):
        print "%-30s %s" % (k, os.environ[k])
```

Other Include Notes

IMPORTANT

Some old versions of BBEdit supported the use of “#include” as an alternative to “#binclude”. However, this syntax made it difficult to mix BBEdit includes and Microsoft Active Server Page (ASP) directives, so it is no longer supported. If you have existing documents which use this syntax, simply change “#include” and “end include” to “#binclude” and “end binclude” to continue using them.

Codeless Language Modules

This appendix lists the syntax elements available for use in codeless language modules. For further details and example modules, visit the Developer and Plug-In Library sections of our web site.

<http://www.barebones.com/support/develop/>

http://www.barebones.com/support/bbedit/plugin_library.html

In this appendix

| | |
|---------------------------------------------------------|-----|
| Creating a Module | 359 |
| <i>Required Elements</i> – 359 | |
| <i>Installing Codeless Language Modules</i> – 360 | |
| <i>Function Scanning with Regular Expressions</i> – 360 | |
| <i>Spell Checking Code Runs</i> – 361 | |
| <i>Starting from a Template</i> – 361 | |
| Language Keys and Properties..... | 363 |

Creating a Module

Codeless language modules are written as “property lists” (or “plist”), which is an XML format that Mac OS X uses for many purposes.

<http://developer.apple.com/documentation/Cocoa/Conceptual/PropertyLists/Articles/XMLPListsConcept.html>

You can create or edit codeless language module files with BBEdit itself, with the Mac OS X Property List Editor, or with a third-party editor such as PlistEdit Pro.

<http://www.fatcatsoftware.com/plisteditpro>

Note The Property List Editor labels boolean properties as “yes” or “no”. However, the actual plisttext file must contain values of either “true” or “false” (written as “<true/>” and “<false/>”).

Required Elements

At a minimum, your codeless language module file must include the appropriate XML header declaration, as well as key/value specifications for each of “BBEditDocumentType”, “BBLMLanguageCode”, and “BBLMLanguageSuffix” in order for BBEdit to load it. The module may then specify any other parameters you desire, including whether to color syntax elements, color a set of keywords, honor case sensitivity, and more. You should save the file with a “.plist” filename extension. If a module fails to load, BBEdit will write some diagnostic information to the system console.

Installing Codeless Language Modules

To install a language module, move or copy the module file into the Language Modules folder of your BBEdit application support folder (~/.Library/Application Support/BBEdit/Language Modules/). If no such folder exists, you may create one.

After installing a new language module, you will need to quit and relaunch BBEdit in order to use it.

Function Scanning with Regular Expressions

Codeless language modules for use with BBEdit 8.5 and later may specify the “Function Pattern” key as a PCRE-compatible regular expression (i.e. a grep pattern) instead of a string.

This expression should return the named subpatterns “function_name”, e.g. (?P<function_name>...), and “function” to identify the function's name (which will be displayed in the function popup menu) and the function as a whole. You can omit the “function” subpattern in order to allow the entire pattern to match against functions, but you should not omit the “function_name” subpattern as if this is not present, BBEdit will not display matches in the function popup.

Since the pattern BBEdit uses internally when searching is a compound of your string, comment, skip, and function patterns, you must use named backreferences rather than positional backreferences in any of these patterns.

Escaping Patterns

Since your patterns are stored as content within the codeless language module's XML plist file, you must entity-encode any unsafe characters which your pattern contains, including “<”, “>”, “&”, etc. (An easy way to accomplish this is to apply BBEdit's Translate command to the pattern, translating “Text to HTML” with the “Ignore < and >” option turned off.)

Nested Functions

A function pattern cannot be used to identify nested functions.

Performance Considerations for Function Scanning

Some expressions can take an extremely long time to locate particular strings. In order to prevent this kind of behavior from locking BBEdit up, any search that takes more than 1.5 seconds will be aborted. This can lead to incomplete function lists and syntax coloring. If you are developing a codeless language module, you can instruct BBEdit to report this condition and certain other grep-related errors in the console log by entering the following Terminal command:

```
defaults write com.barebones.bbedit debugCodelessGrepPats -bool TRUE
```

BBEdit will report some errors immediately upon loading your codeless language module; other errors, such as the search time cap, may not be reported until the corresponding pattern is used.

Spell Checking Code Runs

You can now specify whether “code” runs (i.e. any portions of a file which are not a comment, string, or keyword) can be checked for spelling by adding the following key/value pair to your codeless language module.

```
<key>BBLMCanSpellCheckCodeRuns</key> <true/>
```

Note You should probably enable spell checking if your module is for a “markup” language as opposed to a “scripting” language (e.g. anything that’s not explicitly in a comment is probably text content, and not code).

Starting from a Template

The easiest way to begin creating a codeless language module is to start from a template, or an existing module. The template provided on the following page contains all required key/value pairs, plus a selection of additional parameters which you can fill out or remove as desired.

CodelessLanguageModuleTemplate.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd"><plist
version="1.0">
<dict>
  <key>BBEditDocumentType</key>
  <string>CodelessLanguageModule</string>
  <key>BBLMColorsSyntax</key> <true/>
  <key>BBLMIsCaseSensitive</key> <true/>
  <key>BBLMKeywordList</key>
  <array>
    <string></string>
  </array>
  <key>BBLMLanguageCode</key>
  <string>????</string>
  <key>BBLMLanguageDisplayName</key>
  <string></string>
  <key>BBLMScansFunctions</key> <true/>
  <key>BBLMSupportsTextCompletion</key> <true/>
  <key>BBLMSuffixMap</key>
  <array>
    <dict><key>BBLMLanguageSuffix</key>
      <string></string></dict>
  </array>
  <key>BBLMCanSpellCheckCodeRuns</key><true/>
  <key>Language Features</key>
  <dict>
    <key>Close Block Comments</key>
    <string></string>
    <key>Close Parameter Lists</key>
    <string></string>
    <key>Close Statement Blocks</key>
    <string></string>
    <key>Close Strings 1</key>
    <string></string>
    <key>Close Strings 2</key>
    <string></string>
    <key>End-of-line Ends Strings 1</key> <true/>
    <key>End-of-line Ends Strings 2</key> <true/>
    <key>Escape Char in Strings 1</key>
    <string></string>
    <key>Escape Char in Strings 2</key>
    <string></string>
    <key>Identifier and Keyword Characters</key>
    <string></string>
    <key>Open Block Comments</key>
    <string></string>
    <key>Open Line Comments</key>
    <string></string>
    <key>Open Parameter Lists</key>
    <string></string>
    <key>Open Statement Blocks</key>
    <string></string>
    <key>Open Strings 1</key>
    <string></string>
    <key>Open Strings 2</key>
    <string></string>
    <key>Prefix for Functions</key>
    <string></string>
    <key>Prefix for Procedures</key>
    <string></string>
    <key>Terminator for Prototypes 1</key>
    <string></string>
    <key>Terminator for Prototypes 2</key>
    <string></string>
  </dict>
</dict>
</plist>
```

Language Keys and Properties

| Key | Value Type |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| BEditDocumentType | String ("CodelessLanguageModule") |
| This key/value pair must be present in the property list for the rest of the plist to be examined and loaded, since the file containing the plist need not have any specific file-type or filename-extension. | |
| BBLMLanguageDisplayName | String |
| This is the name displayed for the language module in popup menus and preference panels. Be descriptive, but terse. | |
| BBLMLanguageCode | String |
| This string value should be a unique four-character code for the language that the module supports. Note that the value must be unique with respect to BEdit's built-in languages and with respect to any installed language modules. | |
| Unfortunately, there is no easy way to identify these potential conflicts beforehand, but just keep this all in mind if the contents of a file ends up looking as though it is being treated as some other language than intended. | |
| BBLMColorsSyntax | Boolean |
| This must have the value 'true' for strings and comments to be colored specially by the language module. Keywords will also be colored if the value is 'true', but only if a list of keywords is also supplied in a BBLMKeywordList array (see below). | |
| BBLMScansFunctions | Boolean |
| This must have the value 'true' for the text to be scanned to locate "function definitions" and for a popup menu of function names to be built that allows for quick navigation to those functions. This requires the 'Identifier and Keyword Characters' string described below to be properly specified. | |
| BBLMSupportsTextCompletion | Boolean |
| If this has the value 'true', BEdit will present completions taken from the contents of any text runs (strings or comments) within the current document. | |
| BBLMFileNamesToMatch | Array of Strings |
| BEdit will use the specified strings to test filenames in an exact (but case insensitive) manner to determine whether they should map to the language module. | |

| Key | Value Type |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| BBLMIsCaseSensitive | Boolean |
| <p>If this has the value 'false', letters in keywords and other strings are matched against the text without regard to whether they are both upper or lower case. The value 'true' means that an 'x', for example, will only match another 'x' and not an 'X'.</p> | |
| BBLMKeywordList | Array of String |
| <p>Whenever a string is found to match one of the strings in this array, it is specially colored. For this to happen, the 'Identifier and Keyword Characters' string described below must be properly specified also.</p> | |
| BBLMPredefinedNameList | Array of Dictionaries |
| <p>Whenever a string is found to match one of the strings in this array, it is colored as a "predefined name". For this to happen, the 'Identifier and Keyword Characters' string described below must also be properly specified.</p> | |
| BBLMSuffixMap | Array of Strings |
| <p>Each dictionary entry in this array should contain some or all of the following key/value pairs.</p> | |
| BBLMLanguageSuffix | String |
| <p>Files with names that end with this string value are considered to be files of this module's language. The first character in the suffix string is usually a '.' (dot, period, full stop, whatever). This string must be present and non-empty or the entire dictionary entry will be ignored.</p> <p>Bear in mind that if a suffix is given that overlaps with the suffix map of another language module or BBEdit's built-in languages, confusion may result. Fortunately, all of the suffix mappings can be seen in BBEdit's 'Languages' preference panel.</p> | |
| BBLMCanResolveIncludeFiles | Boolean |
| <p>If this key is present and its value is 'true', BBEdit will send <code>kBBLMResolveIncludeFileMessage</code> for every include chosen off the includes menu. The param block will include a <code>CFStringRef</code> with the name, a <code>CFURLRef</code> to the document on disk (which may be <code>NULL</code>) and a place for you to put a <code>CFURLRef</code> when returning.</p> <p>If the module returns <code>NULL</code> and <code>noErr</code>, then BBEdit will assume that the module declined to do anything with the string and will look for the file as usual.</p> <p>If the module returns a non-<code>NULL</code> URL, BBEdit will resolve it, so the module can make a <code>file://</code>, <code>http://</code>, <code>FTP</code> or <code>SFTP</code> URL and the right thing will happen. If the module returns something other than <code>noErr</code>, BBEdit will not attempt anything else with the include and will report the error.</p> | |

| Key | Value Type |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| BBLMReferenceSearchURLTemplate | String |
| Language modules can now specify a default value for the "Reference URL Template" language-specific preference by including a suitable URL string with this key: http://www.example.com/foobar.cgi?__SYMBOLNAME__ | |
| BBLMIsSourceKind | Boolean |
| If this key is present and its value is 'true', files with this suffix are considered 'source' files. | |
| BBLMIsHeaderKind | Boolean |
| If this key is present and its value is 'true', files with this suffix are considered 'header' files. If both the BBLMIsSourceKind and BBLMIsHeaderKind keys are present and have the value 'true', BBLMIsSourceKind takes precedence, but there should really be only one or the other or neither. If the module's language has a concept of source versus header files and the appropriate values are specified (for example, files with names ending with ".h" are considered header files for C++, whereas files with names ending with ".cp" are considered source files), users will be able to jump between source and header files that share a common prefix (e.g. "foobar.h" and "foobar.cp") using command-tab. | |
| BBLMCanSpellCheckCodeRuns | Boolean |
| If this key is present and its value is 'true', BBEdit will check spelling within "code" runs. | |
| Language Features | Dictionary |
| This dictionary is a container for the following collection of key/value pairs that define the language elements that the module supports. | |

| Key | Value Type |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Identifier and Keyword Characters | String |
| <p>Most languages have keywords and identify other language elements with names that are words made up of letters, digits, and possibly other special characters. The function scanner looks for complete and unbroken sequences of such characters and then tries to decide whether the 'word' is a keyword or some other identifier. This string should contain all of the characters that can be in such a word.</p> <p>Thus, a typical value for this string might be: <code>"0123456789ABCDEFGH IJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz"</code></p> <p>which is the set of characters used in many languages for keywords and other identifiers. Note that the character need not be in any particular order.</p> | |
| Identifier and Keyword Character Class | String |
| <p>If this string is present, it will be used instead of the "Identifier and Keyword Characters" string.</p> <p>This string should be in the form of a grep character class. Any character that is permissible between square brackets ("[" and "]") in a grep character class is permissible here (but do not include the square brackets themselves).</p> <p>Also, this string is not restricted to ASCII characters; it may include any valid UTF-16 characters. You may use grep's <code>\x{...}</code> notation for hexadecimal character codes or other standard character escapes, such as <code>\r</code>, <code>\t</code>, etc., to include characters which are difficult to enter or don't display well (or at all).</p> | |

| Key | Value Type |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Function Pattern | String |
| <p>This key allows you to specify a PCRE-compatible regular expression to identify functions and function names.</p> <p>Your pattern should return the named subpatterns "function_name", e.g. (?P<function_name>...), and "function" to identify the function's name (which will be displayed in the function popup menu) and the function as a whole.</p> <p>You can omit the "function" subpattern in order to allow the entire pattern to match against functions, but you should not omit the <function_name> subpattern as if this is not present, BBEdit will not display matches in the function popup.</p> <p>Since the pattern BBEdit uses internally when searching is a compound of your string, comment, skip, and function patterns, you must use named backreferences rather than positional backreferences within this pattern.</p> <p>If this string is present, the following Language Features will be ignored:</p> <ul style="list-style-type: none"> Prefix for Functions Prefix for Procedures Open Parameter Lists Close Parameter Lists Terminator for Prototypes 1 Terminator for Prototypes 2 Open Statement Blocks Close Statement Blocks Open Block Comments Close Block Comments Open Line Comments Open Strings 1 Close Strings 1 Escape Char in Strings 1 End-of-line Ends Strings 1 Open Strings 2 Close Strings 2 Escape Char in Strings 2 End-of-line Ends Strings 2 | |

| Key | Value Type |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Skip Pattern | String |
| <p>If the Function Pattern string is present in the Language Features dictionary, the presence of this string affects the way BBEdit uses the Function Pattern to scan for function definitions. (Note that you must use named backreferences rather than positional backreferences within this pattern.)</p> <p>When this string is not present, BBEdit uses the Function Pattern in the same way it would as the Search pattern in a Find command. BBEdit will attempt to match the pattern and, if that fails to match, will advance the starting point of the search by one character and try again.</p> <p>When this string is present, after a failed match against the Function Pattern, BBEdit attempts to match the Skip Pattern. If that succeeds, BBEdit will advance the starting point for the next attempt to match the Function Pattern past the text matched by the Skip Pattern. If no match for the Skip Pattern is found, then BBEdit will advance the starting point of the search by one character and apply the Function Pattern again.</p> <p>This can be useful in cases where, for example, strings and comments can contain text that appears to be a function definition but you do not wish them to be placed in the function popup menu. You can define a Skip Pattern to ensure that strings and comments are not included in the search for function definitions.</p> <p>In fact, if you supply a Comment Pattern and/or a String Pattern, you can "call" those patterns as named subpatterns within your Skip Pattern (and even your Function Pattern). The syntax for calling these subpatterns is (?P>comment) and (?P>string) respectively.</p> | |
| Prefix for Functions | String |
| Prefix for Procedures | String |
| <p>In some languages, function definitions begin with a specific keyword. For example, Pascal has functions that return values begin with the keyword 'function' and functions that return no values begin with the keyword 'procedure'. Other languages, such as C and C++, have functions begin simply with their names and other attributes, followed by a list of parameters, followed by the statement block that comprises the function body.</p> <p>If one or both of these prefix strings is present in the plist and non-empty, the function scanner will look for Pascal-style function definitions, otherwise it will look for C-style function definitions.</p> | |

| Key | Value Type |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Open Parameter Lists | String |
| Close Parameter Lists | String |
| <p>A function's list of parameters is almost always enclosed by matching left and right parentheses (probably due to the tradition in mathematics), though there are exceptions. Note that in C-style function definitions empty parameter lists must be designated at a minimum by "()" (or whatever pair of delimiters applies), whereas in Pascal-style function definitions even the delimiters may be omitted.</p> | |
| Terminator for Prototypes 1 | String |
| Terminator for Prototypes 2 | String |
| <p>Some languages allow for a function definition to appear without a body so that other functions that reference it know its "interface" without needing to know its "implementation". Sometimes a keyword is used as a substitute for the body -- in Pascal the keywords 'forward' or 'external' are used. In C-style languages, the function definition is usually just closed off with a semicolon after the parameter list. In either case, if one of the specified strings is encountered before the string value of 'Open Statement Blocks' described below, the function definition is considered to be a bodiless prototype and doesn't appear in the function popup menu.</p> | |
| Open Statement Blocks | String |
| Close Statement Blocks | String |
| <p>Function bodies are usually "statement blocks" that begin and end with *something*. In Pascal, it is literally the keywords 'begin' and 'end'. In C and C-style languages it is usually the characters '{' and '}'. In both cases, such statement block can usually be nested inside one another and the function scanner takes this into account.</p> <p>Note that some languages, such as VBScript, overload the keyword 'END' with another keyword, such as 'SUB', separating the two with one or more spaces. Visually, this is nice because it lets a human reader know what the 'END' ends, but it presents a problem for the function scanner, which is not prepared at this time to treat sequences of keywords as having special meaning. In theory, it would be possible to get by with specifying just 'END' or, more likely, just 'SUB' for the value of 'Close Statement Blocks', but in practice it's hard to say.</p> | |

| Key | Value Type |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Comment Pattern | String |
| String Pattern | String |
| <p>Either pattern may be in the form of any PCRE-compatible regular expression (grep pattern). You must use named backreferences rather than positional backreferences within these patterns.</p> <p>BBEdit will color text that matches the Comment Pattern as comments, and text that matches the String Pattern as strings. All other text will be colored with the default text color except for recognizable keywords.</p> <p>If either (or both) of these strings are present, the following Language Features will be ignored:</p> <ul style="list-style-type: none"> Open Block Comments / Close Block Comments Open Line Comments Open Strings 1 / Close Strings 1 Escape Char in Strings 1 End-of-line Ends Strings 1 Open Strings 2 / Close Strings 2 Escape Char in Strings 2 End-of-line Ends Strings 2 | |
| Open Block Comments | String |
| Close Block Comments | String |
| <p>Block comments are multi-line comments that begin and end with special delimiters, such as <code>/*</code> and <code>*/</code> in C, and <code>{</code> and <code>}</code> in Pascal, where everything in between is ignored entirely.</p> | |
| Open Line Comments | String |
| <p>Line comments begin with a special delimiters, such as <code>//</code> in C, and continue until the end of the line they begin on.</p> | |

| Key | Value Type |
|-----------------------------------|----------------|
| Open Strings 1 | String |
| Close Strings 1 | String |
| Escape Char in Strings 1 | String |
| End-of-line Ends Strings 1 | Boolean |
| Open Strings 2 | String |
| Close Strings 2 | String |
| Escape Char in Strings 2 | String |
| End-of-line Ends Strings 2 | Boolean |

Most languages allow for strings to be enclosed by either matching single-quote characters or matching double-quote characters. Thus, two different sets of key/value pairs can be used for this or for any other kinds of strings that may be needed.

The “escape” values, if matched within a string, cause the character following not to be taken to have any special meaning and passed over. If the 'End-of-line Ends Strings' value is true, the “escape” can be used to pass over the end of the line and allow the string to continue on the following line. In any case, the “escape” may also be used to skip over the leading (or only) character in the 'Close Strings' value and allow that value to be included in the string.

Index

Symbols

“Home” and “End” Keys 203
#bbpragma 221

A

A (anchor) tag 239
ABBR tag 242
accessibility guidelines 249
active windows 70
ActiveX controls 240
ADDRESS tag 235
alternation 168
Anchor command 239
AppleScript 34, 40

- attaching scripts to menu items 282
- in HTML documents 354
- pitfalls 295
- reading dictionary 289
- recording 282

APPLET tag 239
application launch

- overriding default action 199

application launch behavior 199
Application Preferences 197
Apply to New command 155
Apply to Old command 155
AREA tag 240
Arrange command 135
arranging windows 135
arrow keys 344
ASCII table 132
attaching scripts to menu items 282
automatic spell checking 202
Automator 126

- available actions 127

B

B (bold) tag 243
backups 66
Balance Tags command 245
BASE tag 219, 233
BASEFONT tag 241
#basename# placeholder 261
bbdiff 104
bbdiff tool 303
BBEdit Scripts folder 32
BBEdit Startup Items folder 31
bbedit tool 302

BBEdit-Talk mailing list 158
bbfind tool 304
BDO tag 241
bi-directional override 241
BIG tag 243
binary plist files 50
Block Elements submenu 234
BLOCKQUOTE tag 235
BODY tag 233
BOM. see byte-order mark 46, 55
Bonjour 57
bookmarks 57
BR tag 241
broken links 245
browser plug-ins 240
browsers 187

- differences 104
- disk browser 189
- file list panel 190
- search results 143, 191
- setting the list display font 197
- splitter 188
- status bar 188, 189
- text panel 188

BUTTON tag 238
byte-order mark 46, 55
byte-swapped. see Little-Endian 55
bz2-compressed files 50

C

C programming language 119
camel case. see CamelCase 87
CamelCase 87

- keyboard navigation of 87

Cancel button 22
capitalize

- lines 113
- sentences 113
- words 113

CAPTION tag 237
Cascading Style Sheets 226
case sensitivity 140
case transformations 172
CENTER tag 235
changing case 112
character classes 162
character offset specification 50
character set encoding 43, 46, 54
check spelling as you type 202

- Check Spelling command 107
- Check submenu 244
- checking links 245
- checking spelling
 - user dictionary 109
- CITE tag 242
- Classic Mac line breaks 43
- Clean Document 247
- Clear command 22, 70
- Clear key 70
- clearing a marker 106
- client-pull 234
- client-side image maps 240, 241
- client-side scripts 234, 235, 240
- Clipboard 71
 - clipboard 71
 - #clipboard# placeholder 261
 - clipboards, multiple 71
 - Clipping popup 259
 - clipping sets
 - sorting 257
 - clippings 255
 - creating 257
 - editing 258
 - inserting 259
 - manually sorting 257
 - organizing 258
 - substitutions 261
 - Clippings menu 255
 - Clippings palette 256
 - Close Current Tag command 226
 - CODE tag 242
 - COL tag 237
 - COLGROUP tag 237
 - colored text 96
 - columns
 - see rectangular selection 87
 - Command key 22
 - command keys
 - assigning to menu items 325
 - in dialogs 22
 - in menus 21
 - listing by default key 338
 - listing by menu 327
 - shortcuts 343
 - Command-Period 22
 - comments
 - removing 247
 - Compare Again command 155
 - Compare Against Disk File 104
 - Compare Two Front Documents 103
 - comparing files 102
 - multiple files 105
 - complex patterns 166
 - concatenate 101
 - context-sensitive HTML 224, 225
 - contextual menu, in disk browsers 65, 190
 - control characters 119
 - Convert to ASCII 114
 - Convert to Client Side Map command 241
 - Convert to Table command 237
 - Convert to XHTML 249
 - Convert to XML 249
 - Copy & Append command 71
 - Copy as Styled HTML 45
 - Copy as Styled Text 45
 - Copy command 22, 71
 - Copy Path 65
 - Counterpart button 78
 - counterparts
 - overriding defaults 53
 - creating documents 41
 - from templates 254
 - HTML documents 41, 218
 - with clipboard 41
 - with selection 41
 - CSS 226
 - @import 227
 - format 227
 - cursor movement 86
 - using arrow keys 87
 - custom markup 254
 - Cut & Append command 71
 - cut and paste 71
 - Cut command 22, 71

D

 - #date# placeholder 261
 - #datetime xxx# placeholder 261
 - #datetime_gmt xxx# placeholder 261
 - DD tag 236
 - default window position
 - setting 134
 - defined term 242
 - definition list 236
 - DEL tag 235, 242
 - Delete key 70, 91, 344
 - deleted text 235
 - deleting text 70
 - Detab command 120
 - development environments 297
 - configuring BBEdit for use with 298
 - DFN tag 242
 - dialog keyboard shortcuts 22
 - dictionary, AppleScript 289
 - Differences command 104
 - directory list (HTML) 236
 - disclosure triangles 82
 - Disk Browser 41
 - disk browsers 34, 40, 41, 46, 189

- file list panel 190
- status bar 189
- DIV (division) tag 234
- DOCTYPE 218
- document proxy icon 74, 85
- documents
 - comparing 102
 - creating 41
 - editing text 70
 - inserting text 101
 - modification indicator 74
 - opening 373
 - saving 41, 42
 - window anatomy 73
 - window handling 373
- documents drawer. see file list 79
- DOS line breaks
 - see Windows line breaks 43
- double-clicking 46
- drag-and-drop
 - in document windows 72
 - to BBEdit application icon 46
 - to Windows floating window 46
- drawer. see file list 79
- DT tag 236
- dynamic menus 21

E

- Edit Tag command 225
- editing clippings. see clippings, editing 258
- editing text 70
 - shortcuts 343
- Editor Defaults 196
- EM tag 242
- Emacs Key Bindings 203, 345
- Emacs variables 44
 - x-counterpart 53
- encoding 43, 46, 54
- End key 92
- Entab command 120
- Enter key 22
- escape codes 159
- Escape key 22
- Exchange with Next command 135
- expanding tabs 94
- extending the selection 87, 91
- exuberant ctags 298

F

- F keys 92
- Favorites 36
- FIELDSET tag 238
- file filters 147
- File Group

- see Projects 41
- file groups 41
 - see Projects 62
- file list 79
- file list panel 190
- #file# placeholder 261
- file transfer format, FTP/SFTP 59
- #file_extension# placeholder 261
- Filters 310
- filters, file 147
- Find & Mark All command 107
- Find & Replace All Matches 150
- Find Again command 139, 152
- Find All 139, 143
- Find command 137, 140, 151
- Find dialog
 - see Find window 138
- Find Differences command 155
- Find in Next File command 153
- Find in Reference command 155
- Find Selection command 152
- Find window 138
- finding text
 - see searching
- floating windows
 - ASCII table 132
 - HTML Entities 252
 - HTML Tools 217, 250
 - window list 133
- fold indicator 82, 85
- folder, listing contents of 102
- FONT tag 241
- foreign text 111
- FORM tag 237
- Format command 246
- Forms submenu 237
- Forward Delete key 91, 92
- Frame Printing Area 67
- FRAME tag 244
- Frames submenu 243
- FRAMESET tag 243
- freezing line endings 98
- FTP
 - alternate ports 59
- FTP Browsers 61
- function keys 92
- function navigation. see function popup 76
- #function# placeholder 261
- function popup 76

G

- Go To Center Line command 153
- Go To Line command 92, 153
- Go To Previous Error command 153
- gremlins 119

- grep 140
 - alternation 168
 - backreferences 176
 - character classes 162
 - comments 179
 - complex patterns 166
 - conditional subpatterns 183
 - entire matched pattern 170
 - escape codes 159, 163
 - examples 173
 - excluding characters 162
 - longest match issue 168
 - lookahead assertions 182
 - lookbehind assertions 182
 - marking a mail digest 175
 - marking structured text 174
 - matching delimited strings 174
 - matching nulls 176
 - matching white space 173
 - matching words and identifiers 173
 - named backreferences 167
 - named subpattern 166
 - non-capturing parentheses 178
 - non-printing characters 163
 - non-repeating subpatterns 184
 - numbered backreferences 167
 - once-only subpatterns 184
 - pattern modifiers 180
 - positional assertions 181
 - POSIX character classes 177
 - quantifiers 165
 - ranges 162
 - rearranging name lists 175
 - recursive patterns 186
 - repetition 165
 - replacement patterns 170
 - replacing with subpatterns 171
 - setting markers with 107
 - subpatterns 166, 170
 - wildcards 160
- Grep Patterns.xml 33
- gutter 81
- gzip-compressed files 50

H

- Hard Wrap command 97, 99
- hard wrapping 97, 99, 113
- Head Elements submenu 233
- headers 67
- heading tags 234
- hex escapes 141, 163
- hexadecimal 119
- hidden files
 - on FTP servers 58
- Highlight Insertion Point 208

- highlighting of text 70
- hollow diamond 74
- Home key 92
- HR tag 234
- HTML
 - books on 216
 - CSS 226
 - document title 219
 - Web sites about 216
- HTML document, creating 41
- HTML Entities palette 252
- HTML Templates folder 254
- HTML Tools 215
 - Block Elements 234
 - check accessibility 249
 - checking HTML 244
 - convert to XHTML 249
 - convert to XML 249
 - custom markup 254
 - Edit Tag 225
 - entities 252
 - forms 237
 - frames 243
 - Head Elements 233
 - include files 352
 - inline elements 239
 - lists 235
 - Markup menu 217
 - miscellaneous 247
 - new document 218
 - optimizing documents 246
 - palette 217, 250
 - phrase elements 242
 - preferences 217
 - reformatting documents 246
 - scripts 354
 - tables 236
 - Tag Maker 224
 - templates 254
 - tool descriptions 224
 - translation 246, 253
 - updating documents 245
 - utilities 246
 - variables 353
- human interface 21

I

- I (italic) tag 243
- image maps 240, 241
- IMG tag 239
- include files 352
 - variables 353
 - see also templates
- incremental search
 - see Live Search 150

- #indent# placeholder 261
- indenting 113
- Inline Elements submenu 239
- #inline# placeholder 261
- INPUT tag 238
- input, Unix filter 310
- INS tag 235, 242
- inserted text 235
- inserting files 101, 102
- inserting folder listings 102
- inserting page breaks 102
- inserting text 101
- #insertion# placeholder 262
- insertion point 70
- Install Command Line Tools 61, 302
- installing BBEdit 25
- international text 43, 46, 54, 111
- invisible characters 95
- invisible files 50
 - on FTP servers 58
- Invisible Folders 123

J

- Java applets 239, 240
- JavaScript 234, 235, 240
- Jump Back 154
- Jump Forward 154
- jump placeholders 154, 264

K

- KBD tag 242
- Key Equivalent 338
- keyboard shortcuts 326, 343
 - in dialogs 22

L

- LABEL tag 238
- language, source code 95
- launching BBEdit 40
- LEGEND tag 238
- LI tag 236
- line breaks 43, 113
- line breaks, default 43
- line ending format 43
- line endings 43
- line number specification 50
- line numbers
 - show on printout 67
- link checker 245
- LINK tag 219, 233
- list display font 197
- list items (HTML) 236
- Lists submenu 235
- Little-Endian 55

- Live Search 150
- longest match issue 168
- lower case 113

M

- Mac line breaks
 - see Classic Mac 43
- Macintosh Drag and Drop 72
 - see also drag-and-drop
- MAP tag 240
- Mark pop-up menu 106
- Markdown 249
- Marker popup menu 77
- markers
 - clearing 106
 - setting 106
- Markup menu 217
- menu list (HTML) 236
- menus 21
- Menus & Shortcuts preference panel 22
- Menu preference panel 325
- META tags 219, 234
- Misc submenu 247
- monospaced font 235, 243
- mouse shortcuts 343
- moving text 71
- moving the cursor 86
 - using the arrow keys 87
- multi-byte text 43, 46, 54, 111
- multi-file comparisons 105
- multi-file search 141
- Multi-File Search command 137
- multiple clipboards 71
- multiple Undo 72

N

- #name# placeholder 261, 262, 349
- named subpattern 166
- navigation
 - functions 76
 - with jump marks 154
- navigation bar 75
- nested folds 82
- New Window with Selection 42
- NOFRAMES tag 244
- Non-Greedy Quantifiers 169
- non-printing characters 95, 140
- NOSCRIPT tag 234, 235
- numeric keypad 91

O

- OBJECT tag 240
- OL tag 236
- Open command 46

- options 48
- Open dialog 50
- Open File by Name command 53
- Open from FTP/SFTP Server 50
 - Show Files Starting with "." 58
- Open Hidden
 - see Show Hidden Items 50
- Open Recent command 46, 53
- Open Recent menu 198
- Open Selection 50
- Open Selection command 46, 50
- Opening 46
 - binary plists 50
 - bz2-compressed files 50
 - gzip-compressed files 50
- Opening Existing Documents 46
- optimizing HTML 246
- OPTION tag 238
- Option-¥ on Japanese Keyboards 203
- OPTIONGROUP tag 238
- ordered lists 236
- outdenting 113

P

- P (paragraph) tag 234
- page breaks 102
- Page Down key 92
- Page Guide 202
- page guide 95
- Page Up key 92
- paragraph (definition) 70
- Paragraph Fill option 100
- PARAM tag 240
- Paste command 22, 71
- Paste Previous Clipboard 346
- Paste Previous Clipboard command 72
- pattern matching
 - see grep
- pencil icon 74
- Perl 305
- Perl scripts 305
- Perl/Unix Filters palette 325
- persistent includes 353
- Phrase Elements submenu 242
- placeholder strings 154, 264
- placeholders 347
 - #RELATIVE# 264, 350
 - AppleScript 266
 - in glossaries 261
- POSIX-Style Character Classes 177
- PRE tag 235
- Preferences 193
 - Application 197
 - Function Popup 300
 - Printing 67

- Prefix/Suffix Lines plug-in 115
- preformatted text 235
- Previewing in Windows browsers 206
- previewing Markdown content 249
- Print Color Syntax 67
- Print Line Numbers 67
- Print One Copy command 66
- Print Selection 67
- printing 66
- Process Lines Containing plug-in 117
- Project, creating 41
- pull-down menus 21
- Python 305
 - configuration 307
 - Python scripts 305

Q

- Quick Search
 - see Live Search 150
- QUOTATION tag 241

R

- range end indicators 82
- recording scripts 282
- rectangular selection 87
- Redo command 72
- Reflow Document 248
- reflowing paragraphs 99
- reformatting HTML 246
- refresh open files 198
- regular expressions
 - see grep
- #RELATIVE# placeholder 351
- remember recently used items 198
- Remove Line Breaks command 97
- removing comments 247
- Rendezvous. see Bonjour 57
- Reopen Documents 199
- Reopen documents, preventing 199
- repetition metacharacters 165
- Replace 139
- Replace & Find Again command 140, 153
- Replace All 139, 143, 150, 153
- Replace to End 140
- replacing text 70
 - see also searching
- Return key 22
- Ruby 305

S

- SAMP tag 242
- Save a Copy command 43
- Save a Copy to FTP Server command 60
- Save As command 42

- Save As options
 - line breaks 43
 - Save As Stationery 43
- Save as Styled HTML 45
- Save as Styled Text 45
- Save command 42
- Save Selection command 42
- Save to FTP Server command 59
- Saved Sources.xml 36
- Scratchpad 134
- #script# placeholder 262, 266
- script systems 111
- SCRIPT tag 234, 240
- Scripts 310
- Scripts palette 32, 325
- scrolling, synchronized 135
- search results window 143, 191
- search sources 36
- searching 138
 - all open documents 145
 - case sensitive 140
 - for non-printing characters 140
 - for whole words 140
 - grep 140
 - see also grep
 - in a folder 145
 - in multiple files 141
 - in results of a previous search 146
 - in selection only 140
 - menu reference 151
 - non-printing characters 163
 - on a Web site 146
 - replacing in multiple files 149
 - results window 143, 191
 - search set 144
 - wrap around 140
- Select All command 22, 70
- Select Line command 70
- Select Paragraph command 70
- #select# placeholder 262
- SELECT tag 238
- selected text 70
- selecting text 70, 86
 - by clicking 86
 - extending the selection 87
 - rectangular selection 87
- #selend# placeholder 262
- #selstart# placeholder 262
- Services menu 42
- Set Jump Mark 155
- Set Marker command 106
- Set Menu Keys. see Menus & Shortcuts preference panel 22
- setting markers 106
 - using grep 107

- SGML 216
 - prologue 218
- Shell scripts 305
- shell scripts 305
- Shell Worksheet, creating 42
- shell worksheets 300
- shifting text 113
- Show Files Starting with "." 58
- Show Hidden Items 50
- Show Invisibles command 74
- Show Page Guide 74, 196
- simple includes 352
- SMALL tag 243
- Soft Wrap Text command 74
- soft wrapping 95, 97, 98
 - as default 98
- Software Update 198
- solid diamond 74
- Sort Lines plug-in 115
- SPAN tag 241
- spell checking 107
 - user dictionary 109
- split bar 74
 - in browsers 188
- splitting a window. see split bar 74
- startup
 - window handling 199
- startup items 40
- stationery 65
 - creating 43, 65
 - Stationery folder 34
 - using 65
- status bar
 - hiding 95
 - in browsers 188
 - in disk browsers 189
- status bar. see tool bar 73
- STRIKE tag 243
- STRONG tag 242
- STYLE tag 234
- stylesheets 234
- SUB (subscript) tag 241
- subpatterns 166
- substitution
 - in glossaries 261
- SUP (superscript) tag 241
- Synchro Scrolling command 135
- syntax checking 244
- syntax coloring 96
 - on printout 67

T

- tab width 196
- TABLE tag 236
- Tables submenu 236

- tabs
 - converting to and from spaces 120
- Tag Maker command 224
- tarballs 50
- TBODY tag 237
- TD tag 236
- templates
 - for HTML documents 220, 254
 - scripts 354
 - see also stationery 220
 - variables 353
- Text Display menu 83
- Text Document, creating 41
- text encoding
 - choosing 46
- Text Encodings preference panel 46
- text factories 121
- text folding
 - disclosure triangles 82
 - fold indicator 82
 - gutter 81
 - nested folds 82
 - range end indicators 82
- text highlighting 70
- Text Options popup 74
- text transformation 97
- text wrapping 97
- TEXTAREA tag 239
- TFOOT tag 237
- TH tag 236
- THEAD tag 237
- Tidy
 - Clean Document 247
 - Reflow Document 248
- #time# placeholder 262
- time stamps 68
- tool bar 73
- TR tag 236
- transformations, case 172
- translation
 - HTML 246, 253
- TT tag 243
- typing text 70
- typographer's quotes 94

U

- U (underline) tag 243
- UL tag 236
- Un/Comment plug-in 121
- Undo command 72
- Unicode 43, 46, 54, 111
- universal-argument 346
- Unix line breaks 43
- Unix shell scripts 305
- unordered lists 236

- Update submenu 245
- URL clippings 59
- user interface 21
- Using Language Modules 322
- UTF-16 46, 55
- UTF-8 46, 55
- Utilities submenu 246

V

- validation 244, 245
- VAR tag 242
- variables 353
- verify open files 198
- VMWare Fusion 206

W

- wildcards 160
- window list 133
- windows
 - arranging 135
 - exchanging with next 135
 - split bar 74
- Windows floating window 46
- Windows line breaks 43
- Windows menu 131
- worksheets, shell 300
- wrap around 140
- Wrap while Typing option 97
- wrapping text 95, 97

X

- XML declaration 219

Y

- yank-pop 346

Z

- Zap Gremlins command 119