7 Editing Code

Then, arising with Aurora's light,
The Muse invoked, sit down to write;
Blot out, correct, insert, refine,
Enlarge, diminish, interline.

On Poetry
Jonathan Swift

I write until beer o'clock.

Stephen King

Moving, copying, deleting, and replacing code

1 **Select the code to be copied or moved.**

2 **Choose the appropriate command from the Edit menu:**

- **Copy**

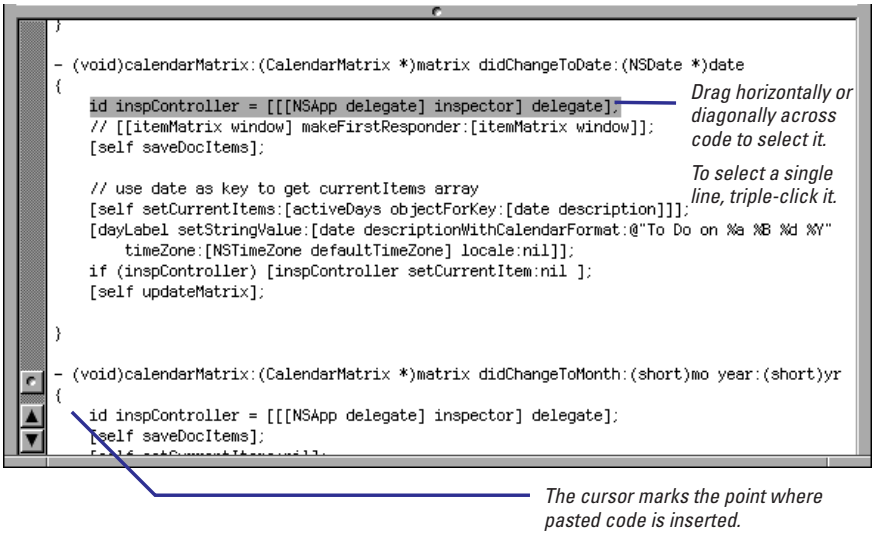
  Or

- **Cut**

3 **Insert the cursor where you want the code to go.**

4 **Choose Paste from the Edit menu.**

As you can in other OpenStep applications, you can use the Copy, Move, and Paste commands to delete, move, copy, and replace code.



When you choose the Copy or Cut commands, the selected code goes into a *kill buffer*. The Paste command puts the contents of this buffer into the stream of characters at the location marked by the cursor. You can issue multiple Paste commands to copy the same contents multiple times. Of course, if you only want to delete code, don't follow the Cut command with Paste.

Several techniques can help you move and copy code:

- If the "Indent pasted lines" option is checked in the Indentation preferences, pasted code will be automatically indented.
- To replace code rather than inserting it, select the destination code before pasting the new code in place of it.
- To delete code without copying it to the kill buffer, select the code and then press the Delete key.

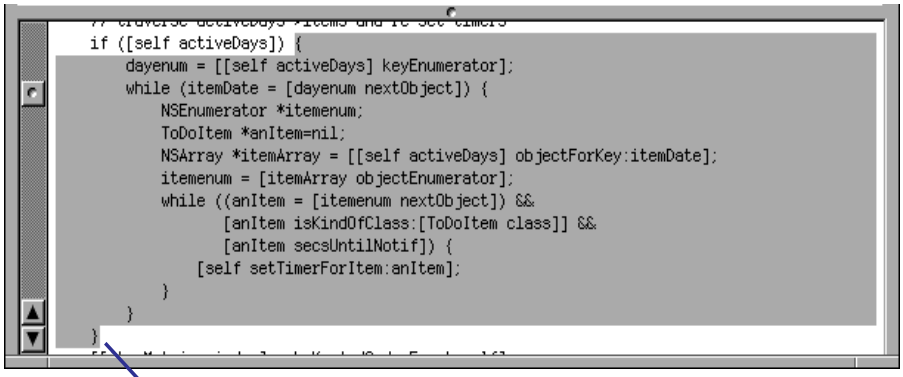
You can use the delimiter-checking feature as a shortcut for selecting messages or blocks of code prior to copying, moving, or replacing them. See "Checking delimiters" on page 158.

Several Emacs commands also allow you to copy, cut, and paste code. See "Emacs Key Bindings" on page 162 for details.

Checking delimiters

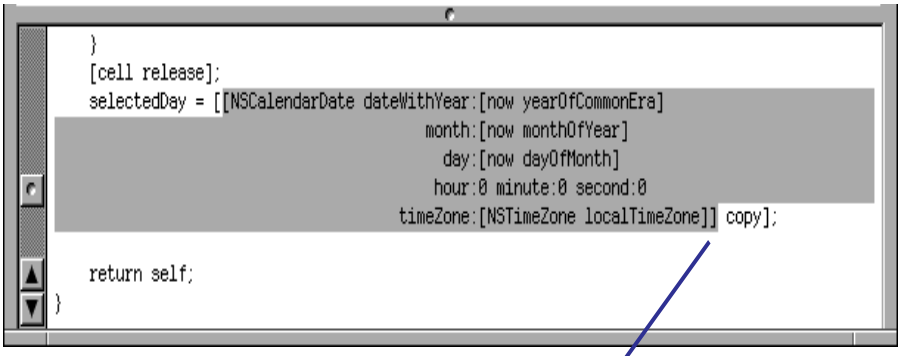
- **Double-click a brace to highlight the block it delimits.**
- **Double-click a square bracket to highlight the message it delimits.**
- **Double-click a parenthesis to highlight the function arguments or C expression delimited.**

One common source of compiler errors is mismatched code delimiters: braces for blocks of code, brackets for message expressions, and parentheses for C expressions and function arguments. Project Builder's code editor allows you to check which delimiters match just by double-clicking.



You can double-click either the starting delimiter or the ending delimiter to highlight code.

Put the cursor as close as possible over the delimiter



This is a good way to check for nested messages.

In addition to checking for missing or extra delimiters, you can use this technique for selecting the code in between (and including) the delimiters.

If a matching delimiter is missing, the code editor displays the error in the status area in its upper-right corner. Project Builder also checks for delimiters as you type code. When you type a delimiter of any type, the cursor briefly jumps to the opposite delimiter. If the opposite delimiter is out of view, the line it's on appears in the status area.

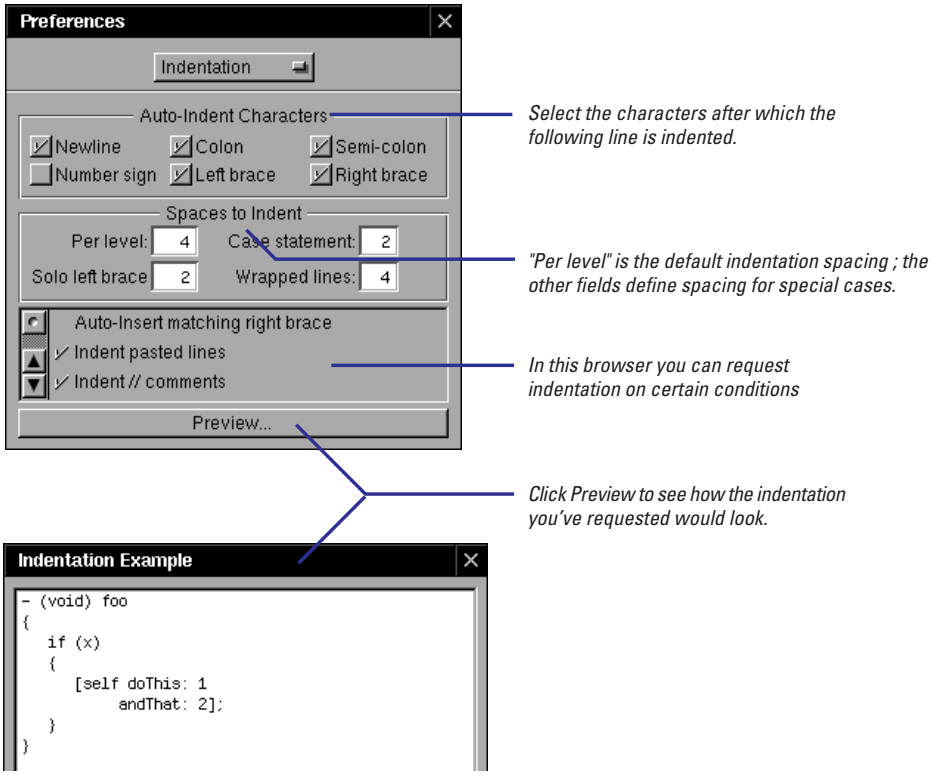
You can also enable your right mouse button so that when you click anywhere in a type or message expression, that type or message is selected and *then* the Project Find panel performs a definition lookup. See <<x-ref?>> for more on this feature.

Indenting code

- **To customize automatic code indentation, choose Preferences from the Info menu and select the desired Indentation preferences.**
- **To indent a line of code, click anywhere in the line and choose Edit ▶ Indentation▶ Indent.**
- **To force indentation left or right, select code and then chose Shift Left or Shift Right from the Indentation menu.**
- **To format messages with many arguments, choose Expand Message Expression from the Indentation menu.**

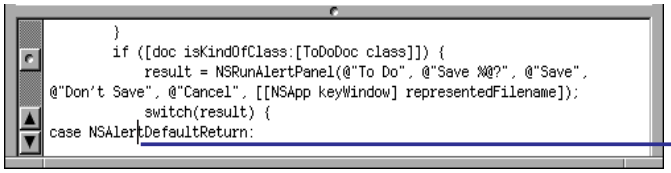
One of the more tedious tasks programmers must do is indenting code: aligning statements and blocks of code with the same scope on the same tab stop. Project Builder alleviates much of this tedium by providing options for both automatic and manual indentation.

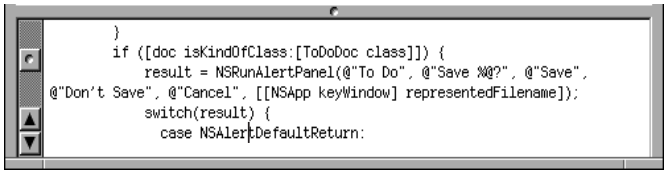
In the Indentation preferences display, you can tell the code editor when to indent each line of code based on the final character of the previous line. You can also specify how many spaces lines should be indented, based on certain conditions.



Tip: The Key Bindings preference display has options related to using the Tab key for indentation: "Indent only at beginning of line" and "Indent always." Use the second option if you want to use Tab to indent a line when the cursor is anywhere in that line.

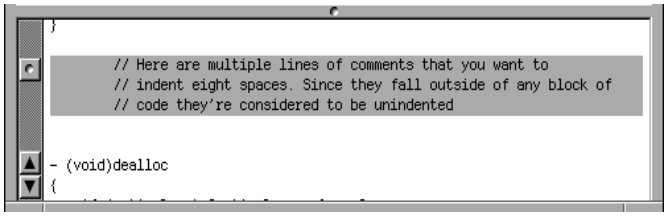
For both single lines and ranges of lines, you can you can issue commands to indent according to the indentation characteristics specified in Preferences; or you can give commands to force a certain indentation.





When indenting single lines of code, you can Insert the cursor anywhere.

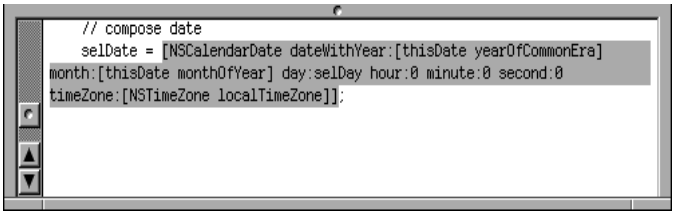
Choose Indent from the Indentation menu to indent the line according to preferences



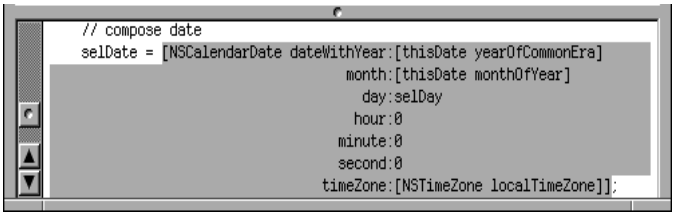
Select multiple lines of code to indent them as a group.

This example shows multiple lines force-indented right after Shift Right is chosen twice from the Indentation menu.

Messages with multiple arguments can be hard to read. Project Builder gives you a command on the Indentation menu with which to format them.



Select a message expression, preferably one with many arguments.



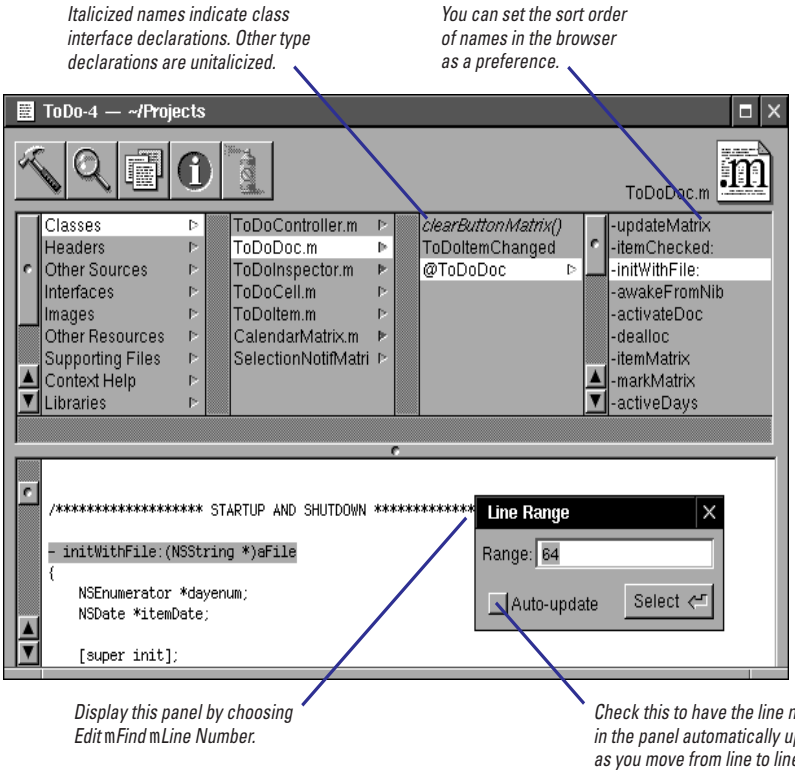
When you choose the Expand Message Expression command, the message is formatted so arguments are aligned.

To return a formatted message expression to an unformatted message expression (in other words, reverse the sequence in the above example), select the expression and choose Edit ▸ Indentation ▸ Compress White space.

Navigating within code files

- **Go directly to methods and functions by selecting their names in the project browser.**
- **Enter a line number in the Line Range panel to go that line.**

You can, of course, go from one place in a source-code file to another place in the same file by scrolling the code editor. Although this mode of navigation is sometimes inescapable, you have other navigation techniques at your disposal.



You can use Emacs key bindings to move around in code without ever touching the mouse. See “Emacs Key Bindings” on page 162 for a list of enabled Emacs commands.

The incremental-search feature (Emacs binding Control-s) is a powerful code-navigation tool for locating text strings within a file. You can also use the Find panel and especially the Project Find panel to find (and replace) specific definitions, reference, and text strings. See the chapter “Finding Information” for details on all of these search features.

Methods, functions, and types appear in the project browser only if the project has been indexed. In the Indexing preferences, you can specify how items in the browser should be sorted: by position in the file, by symbol name (that is, alphabetically), or by symbol name within type.

As its name suggests, you can use the Line Range panel not only to navigate to specific line numbers, but to select ranges of text by specifying colon-separated line numbers. The panel is also a useful tool for learning the current line number. One place where this might be useful is **gdb** (run from the command line) where you can set breakpoints within methods by *file*:*line number*.

Tip: You can “visit” a line of code and return directly to your original location with a couple of Emacs commands. First set a mark by pressing Escape, then the spacebar. Naviagate to the other line, view it (or copy it, or whatever), and press Control-x Control-x to return the marked line.

Emacs Key Bindings

Emacs is an interactive, customizable, richly featured code editor that is popular among many programmers, especially UNIX programmers. Project Builder's code editor incorporates many common Emacs commands.

You issue Emacs commands with the Control key (Ctl) or the Escape (Esc) key, but how you give the command differs with each key. You press the Control key just before the character key, and keep pressing them together. For Escape commands, press the Escape key first, then press the character key. Also, some commands begin with Control-x and are followed by a separate key press. The separation of the final character in Control-x and Escape commands is represented by a space.

Notes:

- On OpenStep for Windows, many Emacs key bindings conflict with the standard bindings for Control keys on Windows applications (for example, Ctl-v is scroll forward in Emacs but is Paste in Windows). Windows key binding override any corresponding Emacs key bindings.
- To use the Emacs commands that begin with the Escape key (Esc), you must select the "Act as Emacs Meta key" option in the Key Bindings preferences display.

Moving Around

Command	What It Does
Ctl-f	Move forward one character
Ctl-b	Move backward one character
Esc f	Move forward one word
Esc b	Move backward one word
Ctl-n	Move to the next line
Ctl-p	Move to the previous line
Ctl-e	Move to the end of the line
Ctl-a	Move to the beginning of the line
Ctl-v	Scroll foward a "page"
Esc v	Scroll backward a "page"
Esc >	Go to the end of the edited file
Esc <	Go to the beginning of the edited file
Ctl-l	Center cursor in middle of displayed code
Ctl-x Ctl-x	Exchange point and mark (return to mark)
Ctl-s	Search forward incrementally
Ctl-r	Search backward incrementally

Editing, Deleting, and Copying

Command	What It Does
Ctl-d	Delete character under cursor
Ctl-k	Delete (kill) to end of line
Ctl-y	Paste (yank) contents of kill buffer
Esc-d	Delete next word or to end of current word
Esc-Del	Delete previous word
Ctl-x u	Undo last change (applies successive undos)
Ctl-i	Indent line
Esc w	Copy region
Esc y	Yank-pop: paste previously cut text in kill buffer

Files and Views (Buffers)

Command	What It Does
Ctl-x 2	Split current view into two views (Split command)
Ctl-x 1	Make one view (Maximize command)
Ctl-x o	Edit in other view
Ctl-x Ctl-b	Open Loaded Files panel
Ctl-x b	Next loaded file
Ctl-x Ctl-f	Display Open Quickly panel
Ctl-x Ctl-s	Save current view to file
Ctl-x Ctl-w	Write to file (Save As)
Ctl-x s	Save all loaded files
Ctl-x i	Insert file
Ctl-x k	Close current file

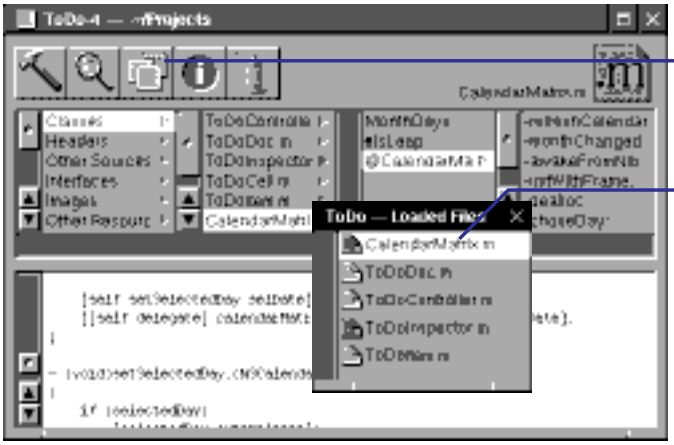
Miscellaneous

Command	What It Does
Ctl-x space	Sets a mark which, with point, marks a region.
Ctl-x '	Go to next error (as displayed in Build panel exception browser)
Ctl-x p	Go to previous error
Esc .	Find definition of current symbol using Project Find panel (as identified by location of cursor)
Ctl-q	Quote next character (for example, a control sequence)
Ctl-g	Quit current command

Navigating between code files

- **Select souce-code files in the project browser.**
- **Use the Loaded Files browser to navigate among opened files**
- **Use the "Open File or Project" panel to locate and open project or non-project files.**

When you select header files, Objective-C implementation files, and other source-code files in the project browser, those files are displayed in the code editor. Although this is a useful feature, it can sometimes require complicated mouse work, especially if you have many project files spread across many categories. The Loaded Files browser provides a navigational focus for the set of files you're most interested in—the files that you've already opened.



To display the Loaded Files browser, click here or choose Tools m Loaded Files.

Click a file to redisplay it in the code editor.

The default sort order is by time visitied. You can change the order to alphabetical by choosing Tools m Loaded File m Sort by Name.

To remove a file from the Loaded Files browser, select it and choose Close from the Edit menu.

A quick way to locate files in the file system, especially good for non-project files, is to use the "Open File or Project" panel. To display this panel, choose Open Quickly from the File menu.

Press the spacebar or Escape to invoke name completion. The next possibly matching file or directory is underlined.

Tip: You can use several Emacs commands to edit the path in the Open File field: Control-a (beginning of line), Control-e (end of line), Control-k (delete to end of line), Control-f (forward character), Escape b (backward "word), and so on.

You can also drag document icons from the File Viewer and drop them over the code editor to open and display them.

Using name completion in editing

- **To cycle through the symbols *local* to a given scope and having thesame prefix, type the prefix and press Escape repeatedly.**
- **To cycle through *all* project symbols with the same prefix, type the prefix and press Alternate-Escape repeatedly**
- **To display a list of all global symbols with the same prefix, type the prefix and press Alternate-I.**

Name completion is a feature that displays all completions of a partial symbol name, including classes, methods, functions, constants, structures, and even local variables. Its has several uses in code editing: It allows you to locate symbols that are only vaguely familiar; it also helps to prevent compilation errors due to misspellings; and it simply a convenient way to insert symbols without having to type them. With name completion, you can obtain symbols local to a file or global to the project.

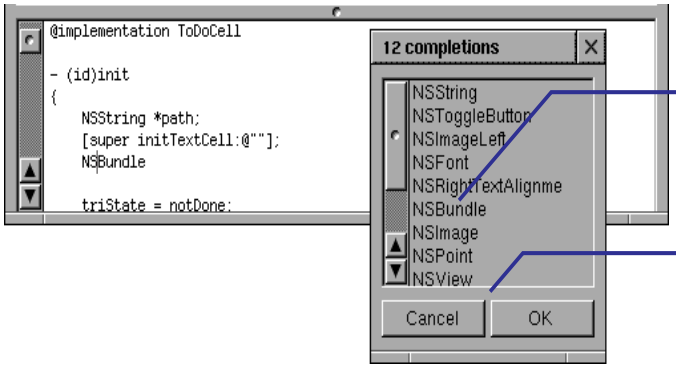
The part of the symbol that completes the prefix is underlined.

Choose a name by clicking the mouse or pressing any key other than Escape. Cancel by pressing the Backspace key.

You must type at least a one-character prefix or insert the cursor within an existing symbol.

If you insert the cursor within an existing symbol, any symbol chosen from name completion will be inserted before the existing symbol.

If you prefer not to cycle through all symbols with a given prefix, you can display a panel that lists possible completions from among the project's global symbols.



When you select a symbol name, it's immediately inserted.

Click Cancel to undo the insertion, click OK to confirm it.

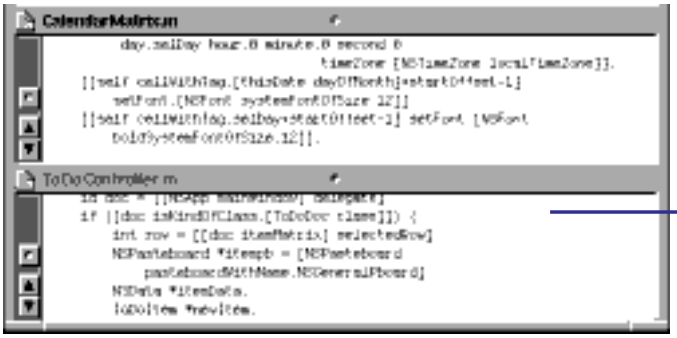
To use name completion, the project must be indexed. When a project is indexed, it "knows" about all symbols—both those that the project internally defines and those that it imports.

Name completion is available in many other contexts, including the fields of the Project Find and Find panels, and in some fields of the Project Inspector panel. In addition, you can use name completion to complete file names and pathnames in all Open and Save panels and in the Open Quickly file (where the space bar rather than Escape is used). See chapter 9, "Finding Information," to learn how name completion is used in find and replace operations.

Displaying multiple views of code

- **To open a new view in the code editor, chose File ▶ View ▶ Split.**
- **To tear off a window from the code editor, chose File ▶ View ▶ Tear Off.**

You can edit code in multiple views in the code editor. The views can display different areas of the same file or different files. The multiple-view feature permits you to view and edit related sections of code—like method declarations in a header file and their implementations in the .m file—without have to navigate among files, and lose context in the process.



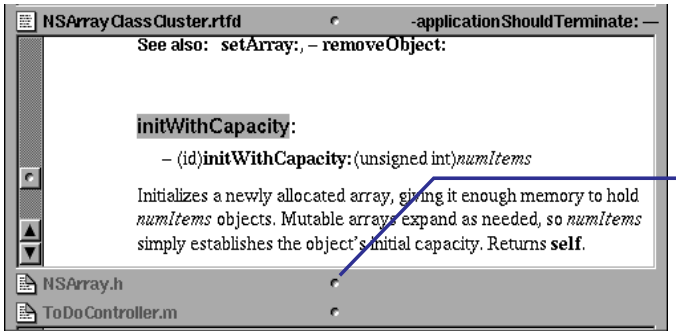
When you split a view, the new view occupies the lower half of the previous view and displays the same general contents.

Click in a view to select it. Open a file or select it from the project or Loaded Files browser to display it.

You can split views repeatedly, with each split view being halved. As you edit in one view, your changes are reflected in all other views of that same file.

Tip: Press Control-x o to cycle through the current views, selecting each one in turn.

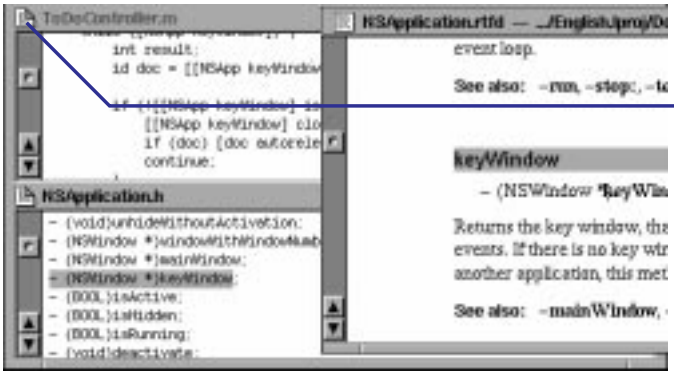
You can enlarge the editing area temporarily by "closing" views. To do this, move a divider (a bar with a dimple in its center) to the top or bottom of the code editor, or to an adjacent divider.



These two views are dragged "closed," enlarging the editing area. To restore them, you'd drag the dividers upward.

To remove a specific view, select it and choose File ▶ View ▶ Close (the "parent" view is automatically selected next). To remove all views except the one you're working in, select that view and choose File ▶ View ▶ Maximize.

Instead of cluttering the code editor with views that become smaller and smaller as you add each subsequent view, you can “tear off” a view and put it in its own window. To tear off a view, select it and choose File ▶ View ▶ Tear Off.



You can also create a tear-off window by Alternate-dragging the file icon off the Project Builder main window.

The window of the tear-off view behaves like any other window, except that editing of its contents is synchronized with any other view displaying the same file. To close a tear-off window, click the close button. Do not choose Close from the View menu.

Note: You cannot display files in tear-off windows by dragging file icons into them and you cannot split tear-off windows. You also cannot use the Emacs command Control-x o to jump to other tear-off windows or views in the code editor.

Undoing and redoing changes

- **To undo a change, choose Edit ▶ Undo ▶ Undo .**
- **To redo an undone change, choose Edit ▶ Undo ▶ Redo .**

Project Builder saves every editing change you make to a *kill buffer*. If you make a mistake, or decide that a modification you made earlier is not what you want, you can undo the change. Because the kill buffer is a stack, when you give the Undo command, you're undoing the most recent change; the next Undo command (without any other intervening command) undoes the previous modification, and so on until the beginning of the editing session (that is, when the file last had no unsaved modifications).

You delete these variables and their initializations.

Choose the Undo command once.

After deselecting the restored text , choose Undo again.

If you leave the text selected, the change is left undone when you perform the next Undo.

In undoing changes, deselect code if you want to retain it; keep the code selected to continue cycling through Undos.

To reinstate a change that you've just undone, give the Redo command. For example, if you decide that you don't want the **currentMonth** variable you've just restored by undoing (last illustration, above), choosing Redo will yield this:

Successive Redo commands reverse the effects of each successive Undo.

Instead of undoing changes in succession by repeatedly choosing the Undo command, you can simultaneously undo all changes made to a region of code since the beginning of the last editing session. Simply select the region and choose Edit ▶ Undo ▶ Undo Region.

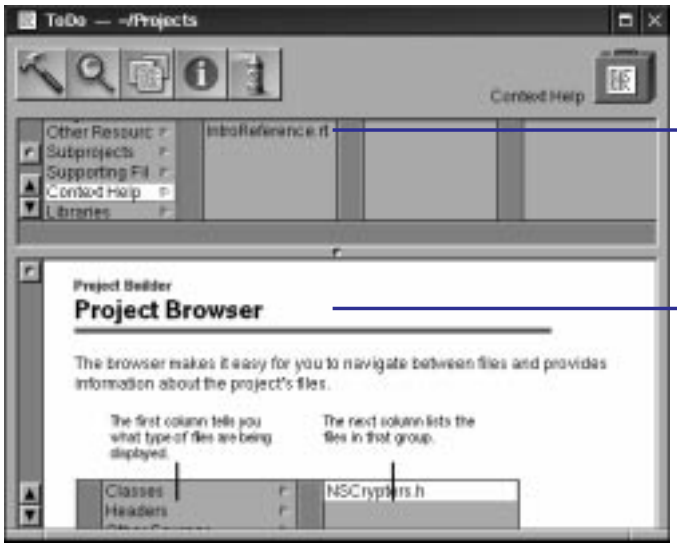
Formatting and adding graphics to RTF text

1 **Create or add an RTF or RTFD file as a project or non-project file.**

2 **Format the text using the commands on the Format menu or on one of its submenus.**

3 **If you want to add graphics, drag and image from the File Manager or from the Images suitcase and drop it over the code editor.**

You can create and format RTF (rich text) and RTFD files in Project Builder. These files can be project or non-project files. Typical RTF project files are context-sensitive help files and, for framework projects, reference documentation. A common non-project RTF file might be a README file.



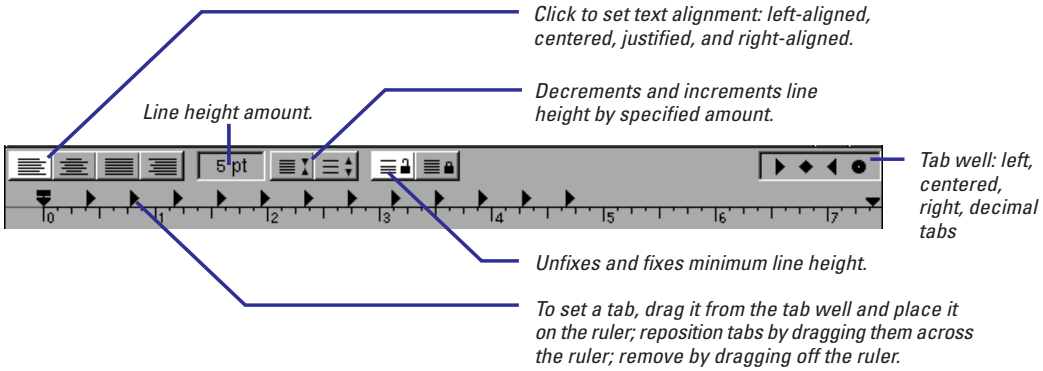
Create or add context help files here. Non-project files go in the Non Project Files "suitcase" of the project browser.

Text can have various attributes, including font, size, style, and color. It can also include graphical images.

You can find RTF editing commands on the Format menu and its submenus. Possibly the most important of these is the one that displays the Font panel (Format ► Font ► Font Panel). The Font menu also has submenus of commands for performing fairly sophisticated typographical operations

Font Submenus	Description
Kern	Adjusts the spacing between selected letters.
Ligature	Use All or Use Default joins certain combinations of characters, such as “fl”.
Baseline	Superscript and Subscript commands lower and raise the selected text’s baseline by a set amount. The Raise and Lower commands adjusts the baseline incrementally.

You can display a ruler above RTF text by choosing Format ▶ Text ▶ Show Ruler. This ruler enables you to set margins and tabs in selected text, to adjust line height (spacing between lines), to set alignment, and to lock changes.



Note: The unfixed (open lock icon) and fixed (closed lock icon) buttons affect the minimum line height. If the fixed button is selected, users can decrement the line height below the limit set by the highest character. If the unfixed button is selected, the line height cannot be adjusted below that limit.

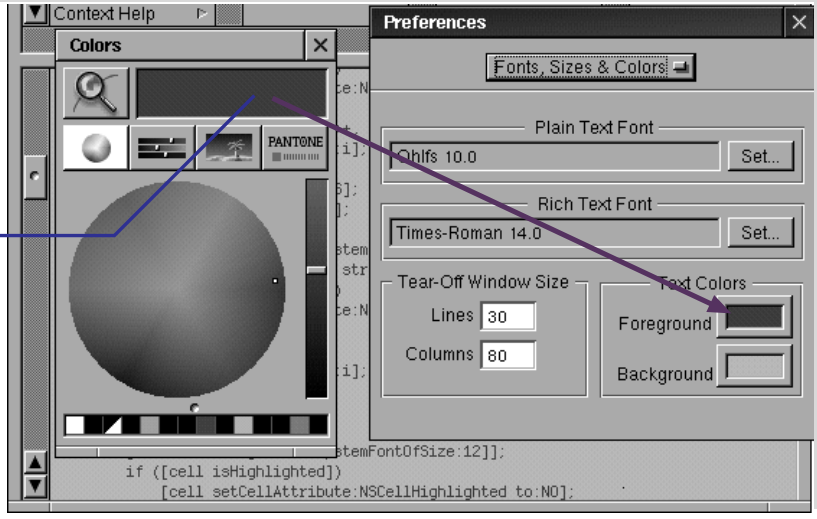
Code files are ASCII files by default. You can, however, convert them to RTF by inserting the cursor anywhere in the file and choosing Make Rich Text from the Format menu. You can then give portions of the code their own attributes; for instance, comments can be in blue and recently modified text can be in bold face. Code with RTF attributes can be safely compiled.

Customizing the Editing Environment

In the Fonts, Sizes & Colors preferences panel, you can customize the default attributes of the code editor, including text color and font, background color, and the size of tear-off windows. Attributes take effect when the next file is opened or when you create the next tear-off window.

Drag a color into one of the Text Colors color wells to set the default foreground (text) and background colors.

The plain text font should be a fixed-pitch font to ensure that indented lines are aligned properly.



RTFD files are Rich Text Format files that can display graphical images. You can easily add EPS and TIFF images to RTF text displayed in Project Builder.

You add images by dragging them from the File Viewer or Project Builder and dropping them in the code editor; they're inserted where the cursor is. (These files become RTFD files in the process, if they're not already.).