OneClick

User's Guide

WestCode Software, Inc. · http://www.westcodesoft.com
619–487–9200 · fax 619–487–9255

The OneClick Product Team

Alan Bird
Leonard Rosenthol
Rob Renstrom
John Oberrick
Jeff Jungblut
Mark Brooks

Manual and Layout

Jeff Jungblut

Cover and Package Design

Steve Sharp, Sharp Advertising & Design

Copyright

Trademarks

Fourth Printing, November 1999
Printed in the United States of America.

Contents

Chapter 1
Introduction

Welcome to OneClick

OneClick™ is software that lets you use and create customized, floating button bars for all your favorite Macintosh applications. If you have used other applications that offer button bars, such as Microsoft Excel or ClarisWorks, you know how convenient buttons can be for performing complex tasks with a single mouse click. With OneClick, you get the benefits of consistent, easy-to-use button bars for all your applications, and you can control what buttons appear on the bars and what those buttons do.

The button bar advantage

The advantage of using button bars (*palettes*, actually) is that they are always visible and available. You don't have to remember convoluted keyboard commands, such as Shift-F7 for "Check Spelling." You also don't have to search through menus to find a command—depending on the application you're using, the Check Spelling command might be in the Edit menu, the Tools menu, or the Options menu. With OneClick, you can click the same Check Spelling button in different applications without having to remember the right keyboard or menu command. The Check Spelling button looks the same and works the same, no matter what.

Ready-to-use button palettes

OneClick offers pre-designed button palettes for many popular Macintosh applications, such as word processing, graphics, desktop publishing, the Finder, and other applications. These palettes let you instantly make use of OneClick from the moment you install it. See the Read Me First file on the OneClick disk for an up-to-date list of application palettes and libraries supplied with OneClick.

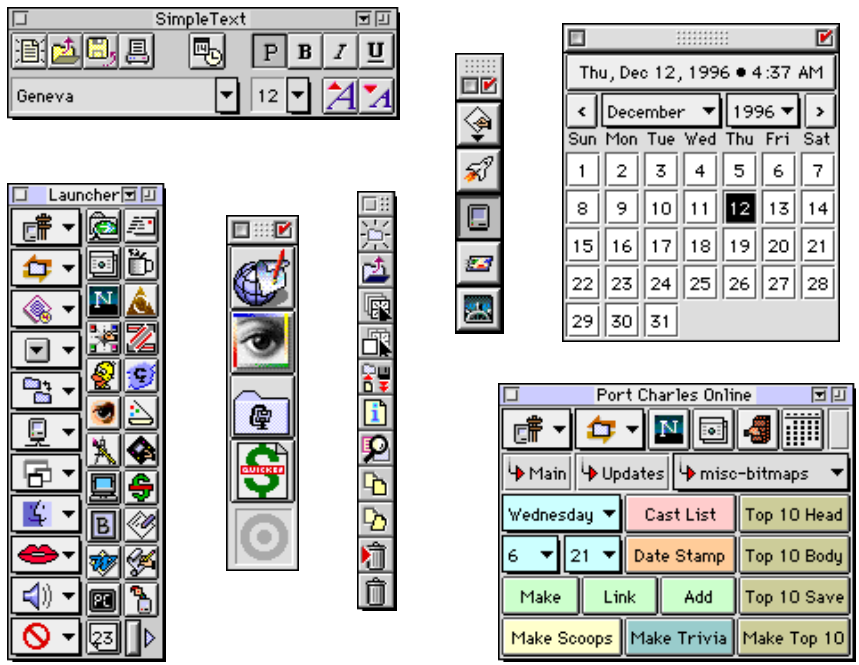
The freedom to customize

OneClick lets you add palettes to any application you wish. Each palette can contain any number of buttons; each button performs a specific task, such as creating a new document, addressing an envelope, or enhancing a scanned photo.

You can add buttons from the Button Library to any of the pre-made palettes and to new palettes you create. Simply choose the buttons you want from the Button Library and drag them to a palette.

When you create custom buttons for a palette, you can customize the appearance of the buttons by applying different colors, border styles, icons, and text. Palettes can have different background colors, patterns, and pictures. These customizing features let you create your own personal look for the palettes and buttons you create.

These sample palettes show a few of the many varieties of styles you can use:



The power to create

If you want to create buttons for applications that aren't listed in the Button Library, you can use the script recorder to create custom buttons.

OneClick buttons derive their functionality and versatility from a powerful scripting language built into OneClick. When you click a button, it performs the associated script which controls the button's action.

OneClick includes an intelligent script recorder that you can use to record a series of mouse and keyboard actions, such as clicking buttons, choosing menu commands, and typing. When you turn on the script recorder, OneClick watches all the actions you perform and saves the series of actions in a button's script. When you click the button, OneClick plays back the script and repeats the actions for you. You can modify any script using the built-in Script Editor, and you can also write scripts from scratch.

If you want to go beyond using pre-built or recorded buttons, you can extend OneClick's functionality using EasyScript, the OneClick scripting language. You can add advanced features and functionality to your buttons using over 200 built-in commands, functions, objects, and properties.

If you're a system integrator, you'll appreciate EasyScript's ability to interoperate with AppleScript, Apple Computer's scripting language. OneClick lets you embed AppleScript code within an EasyScript script and share data between scripts written in the two languages. A OneClick palette makes a perfect, easy-to-use "front-end" to a set of AppleScript programs or scripts.

Note You don't need to learn or understand EasyScript if you just want to use the pre-made buttons and palettes that come with OneClick.

What OneClick can do for you

The kinds of activities you can perform with OneClick are virtually unlimited. OneClick's Button Library includes many buttons for common actions, and you can create custom button actions using the script recorder and OneClick's scripting language. Following are a few examples of the kinds of tasks you can perform with OneClick buttons.

Pre-designed button palettes and libraries can...

- Open frequently-used applications, documents, folders, and control panels.
- Perform common system tasks, such as dragging files to the trash, setting sound or color levels, switching between active applications, and so on.
- Manage Finder and application windows, including opening, closing, moving, resizing, cascading, tiling, and zooming windows.
- Add new functionality to applications, such as word count, sorting a list of words, inserting the date or time, and more.

Custom buttons you create with the Script Recorder can...

- Choose commands and select options from pull-down menus, checkboxes, radio buttons, pop-up menus, and other controls.
- Repeat complex tasks or frequent actions, such typing often-used text, performing a series of commands or steps, and so on.

By exploiting the power of scripting, buttons and palettes can...

- Take advantage of new software controls provided by OneClick, such as a pop-up menu of all windows in any application, a pop-up font menu, and more.
- Display system and application status information, such as the date and time; the free space available on a disk; the font, style, and size; and other information.
- Create custom front-ends that provide alternative user interfaces for applications, or standardize and simplify interfaces across various applications.
- Provide “mini-applications,” like a calendar or phone dialer database, that are instantly available from within any application.
- Schedule tasks to occur periodically or at specific times, allowing your Mac to perform daily file backups, check e-mail for new messages, and other tasks.
- Install “agents” that silently wait in the background and automatically perform their tasks when a certain action occurs, such as automatically zooming windows, updating status information, moving files from one folder to another, and so on.
- Provide a visual interface to execute AppleScript programs or scripts.

About these manuals

If you're the type that likes to explore and start using software without first reading the manual, feel free to skip ahead to Chapter 2, "Installing OneClick," which explains how to install OneClick on your startup disk.

Once you have installed the software, read Chapter 3, "Getting Started with OneClick." You'll learn how to use the OneClick menu and the pre-designed palettes and buttons. Then, in a few brief tutorial exercises, you'll learn how to create a new palette, how to add buttons from the Button Library, and how to record scripts for new, custom buttons.

- Appendix A, "Control Panel Settings," explains the options available in the OneClick control panel.
- Appendix B, "Shortcuts," lists all the keyboard and mouse shortcuts you can use while working with palettes, buttons, and the various OneClick Editor windows.
- Appendix C, "Customer Support," contains information about WestCode Software's customer support services.

OneClick Authoring Guide

The separately available manual, *OneClick Authoring Guide*, shows how to create palettes and buttons using the OneClick Editor and how to write scripts using EasyScript. You don't need to read *OneClick Authoring Guide* to use OneClick, but you should read it if you want to create buttons that do more than just play back your recorded actions. The manual explains how to use the Script Editor and the EasyScript language, and it includes a complete language reference with dozens of sample scripts.

Do you know the basics?

This manual assumes that you are comfortable working with your computer and know how to select menu items, open files, click on buttons, and so forth. If terms such as *double-click*, *icons,* and *system software* are new to you, refer to the manuals that came with your Macintosh.

Chapter 2

Installing OneClick

This chapter shows you how to install OneClick on your startup hard disk. It also describes the kinds of files OneClick uses and where all the files get installed.

System requirements

To use OneClick you'll need the following:

- A Macintosh computer with a 68020 or later processor
- At least 4 megabytes of RAM
- Mac OS 7.5.3 or later (8.0 or later recommended)
- Macintosh Drag and Drop™ and AppleScript™ (included in Mac OS 7.5 and later) are optional

What's in this package

The following items should be in your OneClick package:

- One OneClick 2.0 CD-ROM
- OneClick User's Guide (this book)
- Registration card

What's on the OneClick CD

The OneClick CD contains the Installer program, a file containing release notes, and the OneClick User's Guide and Authoring Guide in Adobe Acrobat PDF format.

Be sure to read the **Read Me First** file before installing OneClick. This file contains important technical and compatibility information that wasn't available before this manual was printed.

Installing or upgrading OneClick

The OneClick Installer lets you install the OneClick software. You can install all components of the software or choose specific features to install.

WestCode recommends choosing the **Easy Install** option to install all components. If you are upgrading from a previous version of OneClick or re-installing, the Installer will save your existing palette files, so it's safe to install over your current version.

Note Before installing OneClick, you should restart your Macintosh with extensions turned off. (Restart the computer and hold down the Shift key until the "Extensions off" message appears.) Some extensions, such as virus protection software, may interfere with the Installer if they aren't disabled first.

▶ **To start the OneClick Installer**

1. Double-click the **OneClick 2.0 Installer** icon in the CD window.
2. Click **Continue** when the welcome screen appears.
3. Read the license agreement, then click **Accept**.
4. Review the Read Me file, then click **Continue**.

   The Installer window appears.

5 If your startup disk icon doesn't appear in the Install Location menu, select your startup disk from the menu.

Because OneClick is a control panel/system software extension, you must install it on your startup disk, not on a secondary hard disk or file server volume.

6 Click **Install**.

7 When the installation is complete, click **Restart**.

Where to go from here

After installing OneClick and restarting the computer, button palettes will appear automatically for many common applications, including the Finder. These palettes are ready to use immediately, just as if they were built into the applications. For an up-to-date list of applications with pre-made palettes, see the **Read Me First** file on the OneClick disk.

Turn to Chapter 3, "Getting Started with OneClick," to learn how to use the standard OneClick palettes and buttons.

Where the Installer puts files

Icon	File or folder	Location	What it's for
	OneClick	Control Panels folder	User interface/scripting engine—required to use OneClick
	Button Palettes folder	OneClick Folder in Preferences folder	Stores palettes you create for the system and various applications
	Libraries folder	OneClick Folder in Preferences folder	Stores library files containing pre-designed buttons
	OneClick Preferences	OneClick Folder in Preferences folder	Contains settings made in the OneClick control panel
	OneClick Help	OneClick Folder in Preferences folder	Contains online help for script keywords
	Extensions folder	OneClick Folder in Preferences folder	External script commands, functions, and system variables
	OneClick Scripting Additions	Scripting Additions folder in Extensions folder	Provides access to OneClick scripts and variables from AppleScript
	OneClick 2.0	Main (root) window of the startup disk	Contains additional palettes, technical information, and other supplementary files

Re-installing OneClick

If for any reason you ever need to re-install OneClick, you should first make a backup copy of the Button Palettes and Libraries folders inside the OneClick Folder (in Preferences). If you make any changes to the palettes that OneClick installs, or you create or change any global palettes, these changes are lost when you re-install OneClick. After re-installing the software, you can use the backup copies of your palette files to import your customized palettes back into the active palette files.

Removing OneClick

If you ever need to remove OneClick for any reason, you can use the Installer to remove the OneClick software and support files. The installer's Custom Remove option works just like the Custom Install option.

▶ **To remove OneClick or any of its support files**

1. Double-click the OneClick Installer icon on the OneClick disk.
2. Click **Continue** when the welcome screen appears.
3. Choose **Custom Remove** from the pop-up menu.
4. Click the triangles next to each feature to expand or collapse the feature list.
5. Select items to remove by clicking the checkboxes next to each feature.
6. Click **Remove**.
7. When the operation is complete, click **Restart**.

Chapter 3

Getting Started with OneClick

This chapter shows you how to get up and running with OneClick. You'll learn how to do the following:

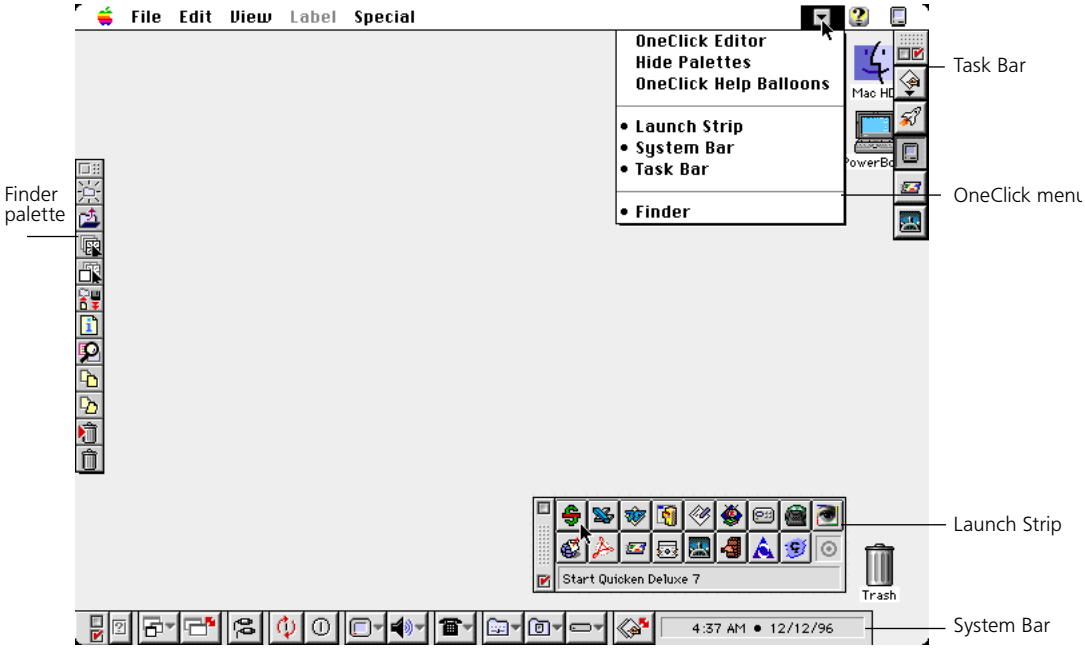
- Work with some of the pre-designed palettes, such as the System Bar, Task Bar, Finder, and Launch Strip palettes
- Use the OneClick menu to access palettes and get help for buttons
- Create a new palette and add some buttons to it from the Button Library
- Make some custom buttons and record scripts for the buttons
- Add keyboard shortcuts to buttons

All the basic concepts of using OneClick are covered in this chapter. Other topics, such as designing custom icons and using advanced features, are covered in later chapters.

About the standard palettes

After installing OneClick and restarting your computer, you'll notice some new button palettes on the screen, along with a OneClick welcome screen. Click the **Next** button on the welcome screen to browse through the Quick Tour. When you're done reading, click **Done** to close the tour.

Besides the new palettes, you'll also notice a new menu on the right side of the menu bar. The OneClick menu lets you turn palettes on and off; access the OneClick Editor window to change palettes and buttons; and get help for buttons.



The System Bar contains buttons and pop-up menus you can use in any program. The System Bar is always available no matter what application you're using.

The Finder palette contains buttons that perform common Finder operations, such as closing all windows, making an alias, and emptying the Trash.

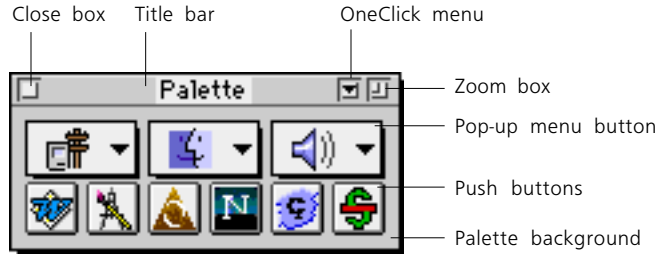
The Task Bar shows an icon for each running application. You can switch applications by clicking an application button.

The Launch Strip contains buttons for your frequently-used applications, documents, and folders. You can add buttons by dragging icons to the target button. (The Launch Strip on your screen won't look like the one shown here. A newly-installed Launch Strip doesn't have any launch buttons added to it yet.)

These pre-designed palettes, like any OneClick palettes, can be changed to meet your needs. You can add, remove, and rearrange buttons; change a button's functionality; change the size of palettes and move them around on the screen, and more. If you want these palettes to look or work differently, you have the freedom to customize them as much as you want.

Parts of a palette

OneClick palettes are similar to button bars in some applications, but some parts of a OneClick palette may work a little different than the button bars or palettes you've used before.



- **The close box** hides the palette. To show the palette again, choose its name from the OneClick menu in the menu bar.
- **The title bar** shows the palette's name and lets you drag the palette around on the screen. To shrink the palette, double-click its title bar; to unshrink the palette, double-click the title bar again. Not all palettes have title bars.
- **The OneClick menu** is a pop-up menu containing the same commands as the OneClick menu in the menu bar. Use the OneClick menu to show and hide palettes or open the OneClick Editor window.
- **The zoom box** shrinks a palette down to its title bar and moves it to the upper-right corner of the screen. To unshrink the palette and move it back, click the zoom box again.
- **The palette background** contains all the palette's buttons. If a palette doesn't have a title bar, you can move the palette by holding down the Option key and dragging the palette's background.
- **Pop-up menu buttons** display a menu of choices. To use a pop-up menu button, click and hold down the mouse on the button, then choose an item from the menu that appears.
- **Push buttons** perform their action when you click and release the mouse button.

Using the OneClick menu

The OneClick menu, represented by the  icon in the menu bar, lets you do the following:

- open the OneClick editor window, where you can make changes to palettes, buttons, icons, and scripts
- turn Balloon Help on or off for buttons on OneClick palettes
- hide or show all the palettes
- hide or show individual palettes

The OneClick menu also appears in the title bar of palettes that have title bars, and you can set the OneClick menu to appear as a submenu in the Apple menu (select the **In Apple Menu** option in the OneClick control panel).

Showing and hiding palettes

The bottom of the OneClick menu lists all the global palettes and any palettes for the active application. A bullet (•) next to a palette's name means that the palette is open. Palettes listed in the second section of the menu are global palettes which appear in all applications. Palettes listed at the end of the menu are application-specific palettes.

▶ **To display a palette**

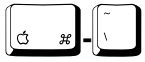
- Choose the palette's name from the OneClick menu.

  The chosen palette appears in the same position on the screen where you last used it.

▶ **To hide a palette**

■ Click the palette's close box or choose its name from the OneClick menu.

▶ **To hide all open palettes**



■ Choose **Hide Palettes** from the OneClick menu or press the Show/Hide Palettes hot key (Command-`).

All palettes remain hidden until you choose **Show Palettes** or choose a specific palette name from the OneClick menu.

When you quit an application, any application-specific palettes close automatically. They'll reappear in the same position the next time you open the application.

Moving, shrinking, and zooming palettes

Because palettes always float on top of other windows, they can sometimes get in the way of what you're doing, especially if you have a small screen. OneClick provides several ways to make palettes less obtrusive.

▶ **To move a palette**

■ Drag the palette's title bar, just as you drag other windows. If the palette doesn't have a title bar, hold down any modifier key (Command, Shift, Option, or Control) and click the palette's background (not on a button) to drag it around.

▶ **To shrink a palette down to its title bar**

■ Double-click the palette's title bar. To restore the palette, double-click the title bar again or click the palette's zoom box on the right side of the title bar.

▶ **To zoom a palette out of the way**

■ Click the palette's zoom box.

The palette shrinks down to its title bar and moves to the upper-right corner of the screen near the hard disk icon.

▶ **To zoom all palettes**

■ Hold down the Option key and click a palette's zoom box.

▶ **To restore a zoomed palette**

■ Click the zoom box again or double-click the palette's title bar.

Getting help

OneClick provides Balloon Help for all the buttons on the pre-made palettes and the buttons in the Button Library. You can get Balloon Help for buttons three different ways:

- **Hold down the Shift and Option keys while pointing to a OneClick button.** Balloons appear only while the keys are held down. Balloons appear only when you point to OneClick buttons, not when you point to other parts of the desktop.
- **Choose OneClick Help Balloons from the OneClick menu.** Balloons appear when you move the mouse over OneClick buttons, but they don't appear for other parts of the desktop. To turn the balloons off, choose OneClick Help Balloons again.
- **Choose Balloon Help from the Help menu.** Balloons appear when you point to OneClick buttons or any other desktop element that provides help balloons (menus, icons, and so on).

The Help Status area on the System Bar displays the help message for any button you point to. The message changes to show the current date and time when you're not pointing to a button. The Help Status button is also included in the Universal button library; adding this button to a palette is a convenient way to have help messages always available without having to toggle help balloons on and off.

Making a custom palette

To learn how to make custom buttons and palettes, you'll make a new palette for SimpleText, a simple text editor. SimpleText is an enhanced version of Apple's

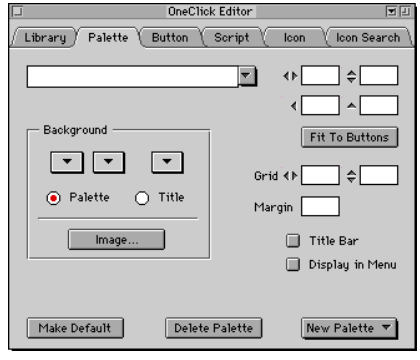
TeachText, the original Macintosh text editor you're probably already familiar with. You'll find SimpleText in the OneClick Goodies folder on your hard disk.

▶ To create a new palette for SimpleText

1. Double-click the **SimpleText** icon to open it.
2. Choose **OneClick Editor** from the OneClick menu.

The OneClick Editor window appears with six tabs across the top. Clicking each tab displays a different set of options in the window. Notice that like palettes, the OneClick Editor window floats on top of other windows.

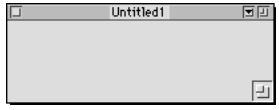
3. Click the **Palette** tab to display the Palette Editor.



The Palette Editor contains controls for creating and naming palettes; changing a palette's size, color, or background picture; and other options.

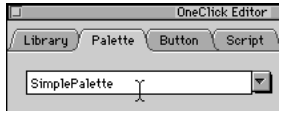
4 Choose **New Application Palette** from the New Palette pop-up menu.

An empty, untitled palette appears.



Creating an application palette while SimpleText is active, as opposed to creating a global palette, means that the palette is available only when SimpleText is the active application. When you switch to another application, the palette disappears; the palette reappears when you switch back to SimpleText.

5 In the Palette Editor, select the word **Untitled1** and type **SimplePalette**. (Type a more creative name if you want.)



The name in the palette's title bar changes as you type.

Note You need to click in the Palette Editor before you can type the palette's name. If you don't click the text box in the Palette Editor first, your keystrokes go to the SimpleText window, not the Palette Editor window. The Palette Editor's title bar frame appears darkened when it's selected for keystrokes.

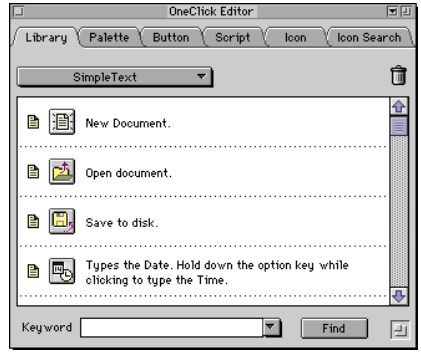
That's all there is to creating a new palette. At this point, the new palette isn't too useful without any buttons on it. You'll add buttons in the following sections.

Adding buttons from the Button Library

To make the empty SimpleText palette useful, you'll add some pre-defined buttons to it from the Button Library. The Button Library contains scores of ready-made buttons for both general and specialized tasks.

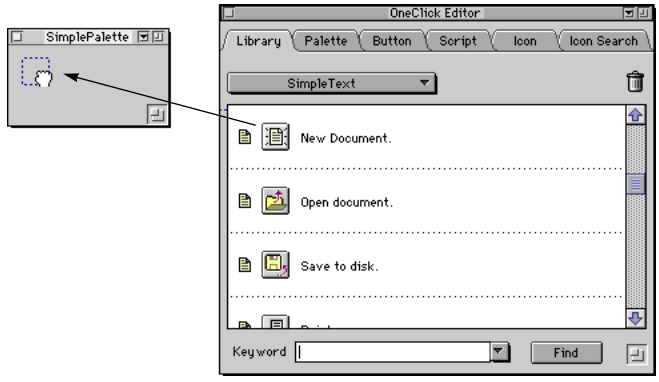
▶ To add buttons from the Button Library

1 In the OneClick Editor window, click the **Library** tab to display the Button Library.



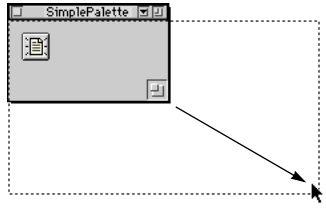
The SimpleText library appears, and all the pre-designed buttons for the SimpleText application appear in the button list. A brief description of what each button does appears next to each button.

2 Drag the  button from the Button Library to your empty palette. Drop it on the palette anywhere you want.

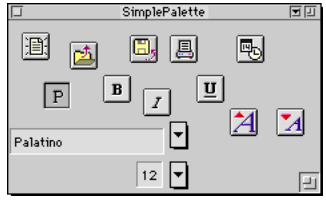


As you drag the button over the palette, notice that its outline snaps to an invisible grid location every few pixels. By having buttons snap to this alignment grid, you can easily line up several buttons in rows or columns.

3 Click the palette's resize box (in the lower-right corner of the palette) and drag to change the palette's size. Make the palette large enough to hold about a dozen square buttons.



4 Drag any other buttons from the list in the Button Library to the palette.



5 Arrange the buttons on the palette however you prefer.

If you want to move any buttons you've already placed, just drag them to a new location. You can move several buttons at a time by dragging a rectangle around a group of buttons, then dragging the selected group. You can also Shift-click buttons to select multiple buttons before dragging them.

6 To more precisely position the buttons, select one or more buttons, then press the arrow keys to nudge them one pixel in the direction you press. Resize the palette if you need to.

7 Click the **Palette** tab to return to the Palette Editor.

8 Click **Fit To Buttons**.

The palette changes size. Fit To Buttons makes a palette just large enough to show all the buttons.

Trying out the new buttons

You now have a palette with several working buttons! To try out the buttons, you'll first need to close the OneClick Editor window.

Note Clicking a button doesn't have any effect until you close the OneClick Editor window. When the OneClick Editor window is open, clicking a button just selects the button; when the OneClick Editor window is closed, clicking a button runs the button's script.

1 Click the close box in the OneClick Editor window, then click the button on the palette.

A new untitled SimpleText window should appear.

2 Type the following text: **Monday is my favorite day of the week.** If you don't agree with this statement, go ahead and type any other sentence you want.

3 Select some of the text you typed. Try experimenting with the buttons you added to the palette, such as , , and .

You've now seen what some of the pre-made buttons can do. While some of the buttons offer alternatives to using the menu commands, other buttons actually add new features to SimpleText. For example, the and buttons increase and decrease the font size of selected text, and the button inserts the current date. (If you Option-click the button, it inserts the current time.) These are simple examples of new features that OneClick can add to applications.

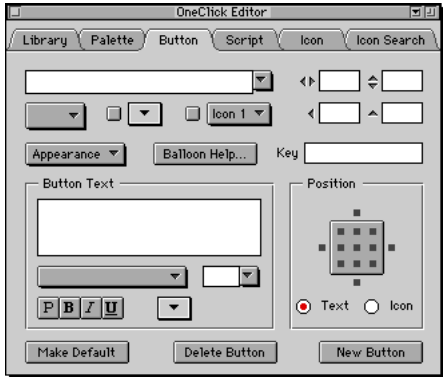
In the next section, you'll learn how to create your own buttons and record scripts for the buttons. Recording button scripts lets you design your own personal features for the applications you use.

Making custom buttons

The buttons you've added to the SimpleText palette so far are rather generic—that is, they can be used by anyone in just about any program, not just SimpleText. Now it's time to create some custom buttons for your own personal use. You'll start off by creating a button that types your return address in a SimpleText window.

1 Choose **OneClick Editor** from the OneClick menu.

2 Click the **Button** tab in the OneClick Editor window.



The Button Editor appears with options for changing a button's name, color, border style, size, label text and position, and other options.

3 Click your palette to select it.

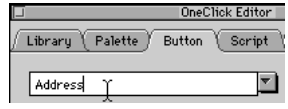
Remember, you need to select a palette so OneClick knows which palette you want to work with.

4 Click **New Button** in the Button Editor.

A new button appears on the SimpleText palette. The button has square handles on its four corners to show that it's selected.

The button is placed wherever there is room—going first from left to right, then top to bottom. If there's no room for the new button, the palette resizes itself to hold the new button. You can resize the palette and drag the button anywhere you want if you don't like where OneClick places it by default.

5 With the new, blank button selected on the palette, select the word **Untitled1** in the Button Name box and type **Address**.

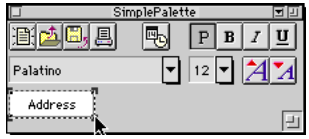


The name you type for this button doesn't really matter unless this button is referred to by another button's script. We'll talk more about how you use buttons from within scripts in later chapters.

6 Click the **Button Text** box, then type **Address**.

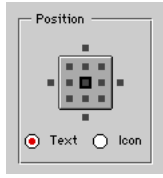
The word Address appears in the button. Part of the word, anyway—the default size for the new button is too small to show the whole word.

7 To resize the Address button so the word Address fits inside it, drag one of the button's corner resizing handles. Resize the palette by dragging the palette's size box if you need to make more room for the button.



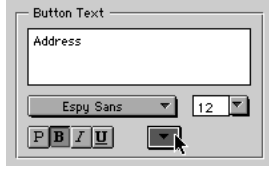
This text label is optional, and you don't have to use a text label if you'd prefer to use an icon instead. For this example, we're using a text label because a mailing address isn't easily represented by a small icon. (Well, at least we don't think so.)

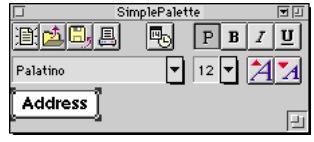
The grid in the Position box lets you control where the label appears in or around the button. The default position for the label is in the center of the button.



8 Experiment with different label positions by clicking the small squares in the Position grid. (Make the button larger if you need to.) Watch where the label appears on the Address button each time you click points in the grid.

9 Try changing the font, style, size, and color of the text in the button. The buttons and pop-up menus in the Button Text box let you change all the text formatting of the button's label.





10 When you're done resizing and positioning the button and its label, close the OneClick Editor window.

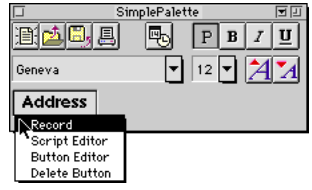
The Address button you just created doesn't do anything yet because you haven't recorded a script for it. If you click the Address button now, nothing happens.

Recording a button script

You can record just one action or several steps in a script. When you click the button to play back the script, OneClick repeats all the actions in the order you performed them.

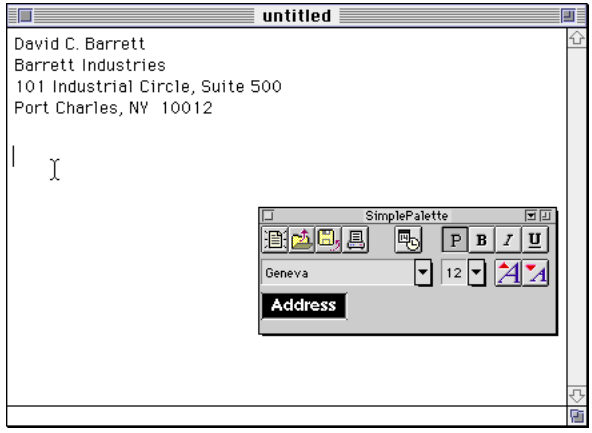
To make the Address button type your mailing address, you'll record a script.

1 Hold down the Command and Option keys, then click the Address button and choose **Record** from the pop-up menu that appears. (Make sure you close the OneClick Editor window first, if it's open.)



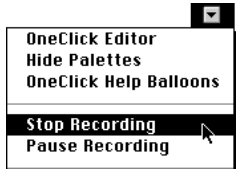
The Address button blinks and a microphone icon blinks in the menu bar to show that you're recording.

2 In the SimpleText window, type your return address. Press Return twice at the end of the last line in the address.



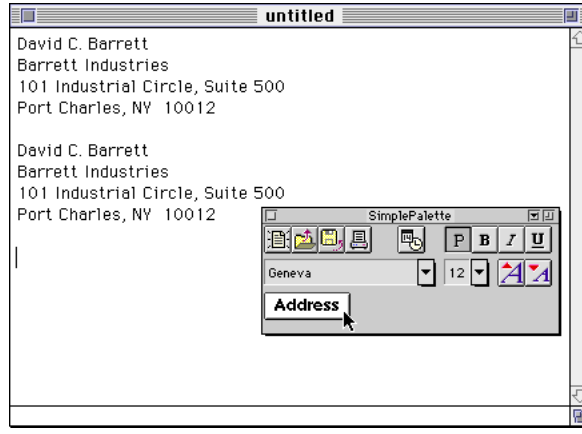
3 Click the blinking Address button when you're done typing.

Clicking the button stops recording and saves the script. You can also choose **Stop Recording** from the OneClick menu when you're done recording.



Now you're ready to try out the button you just recorded.

4 Click the Address button once.



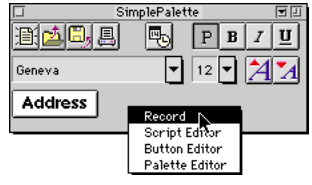
Whenever you click the Address button, OneClick types your mailing address. The script in the Address button simply repeats all the actions you performed while you were recording. That's all there is to recording a script!

Recording menu commands

OneClick can record other actions besides typing text and pressing command keys. You can also record different kinds of mouse actions, such as choosing menu items; clicking buttons, scroll bars and other controls; selecting checkboxes; and dragging.

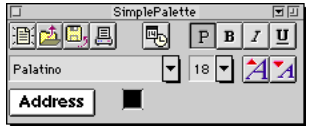
Now you'll create a new button that chooses menu commands. The commands will change all the text in the SimpleText window to a different font, size, and style.

1 Hold down the Command and Option keys, then click in an empty area on the palette (**not** on a button) and choose **Record** from the pop-up menu.



OneClick adds a new, blank button to the palette and immediately begins recording a script for the new button. Command-Option-clicking a palette and

choosing **Record** is a shortcut you can use instead of adding a new button in the Button Editor and then recording.



The new button and microphone icon blink to indicate recording is in progress.

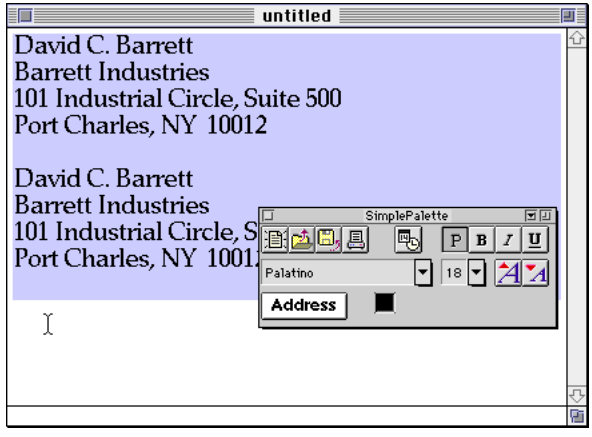
2 Press Command-A to select all the text in the window.

3 Choose **Palatino** from the Font menu in the menu bar.

Use the Font menu in the menu bar, not the pop-up font menu on the palette. You can't click other buttons on OneClick palettes while recording.

Choose a different font if you don't have Palatino installed (or if you just don't like Palatino).

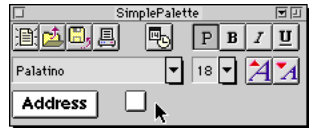
4 Choose **18 Point** from the Size menu.



5 Press the Right Arrow key.

When all the text is selected, pressing the Right Arrow key deselects the text and moves the cursor to the end of the document.

6 Click the new button (the one that's blinking) to stop recording.



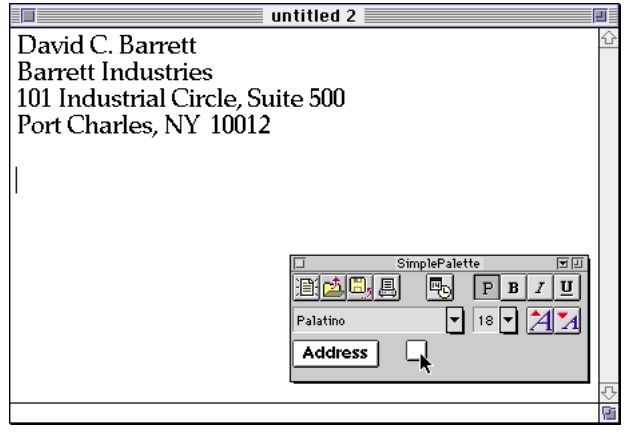
The button stops blinking. It's ready to use.

7 Click the  button to open another untitled SimpleText window.

8 Click the Address button you created earlier.

9 Click the new button you just recorded to change the font and size.

Your second SimpleText window should look similar to this:

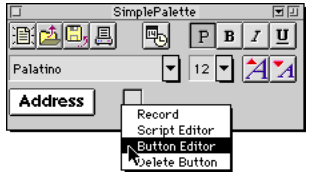


Note When you record a script and play it back, the script may not do *exactly* what you expect, depending on the actions you performed while recording. Scripts that type text, press command keys, and choose menu items are generally more reliable than scripts that perform mouse actions, such as clicking and dragging in a window. For more information on how to record reliable scripts, see Tips for recording a script in Chapter 5, "Using the Script Editor" in the *OneClick Authoring Guide*.

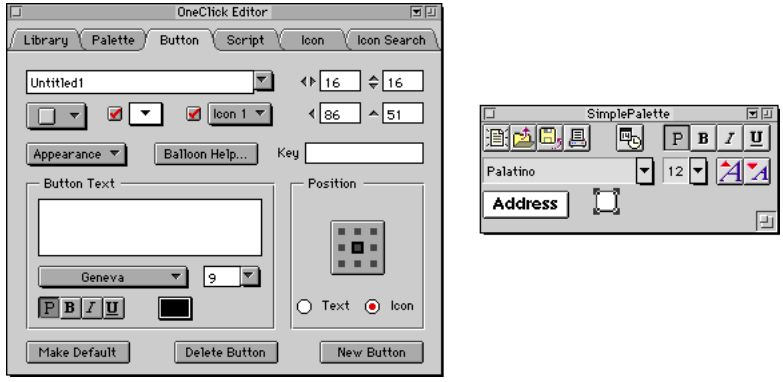
Changing a button's formatting

If you want, you can now go to the Button Editor and change the new button's color, border style, size, text label, and other options.

1 Hold down the Command and Option keys, then click the new button and choose **Button Editor** from the pop-up menu.



2 Try changing different options in the Button Editor to see how they affect the button's appearance.



The border style pop-up menu ( ) lets you choose a different border style for the selected button.

Buttons on the SimpleText palette use three different border styles—normal push buttons, like the two buttons you created; inset "status message" buttons for the font and size indicators; and pop-up menu buttons for the font and size menus.

Other options are covered in detail in Chapter 4, "Using the Button Editor" in the *OneClick Authoring Guide*.

3 Close the OneClick Editor when you're done.

Adding keyboard shortcuts to buttons

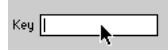
If you prefer using the keyboard instead of clicking buttons for some tasks, then OneClick lets you assign shortcut keys to buttons. When you assign a shortcut key to a button, pressing the shortcut key plays back the button's script, just as if you had clicked the button. This lets you use OneClick as a keyboard macro program.

Now you'll add keyboard shortcuts to the Address and font style buttons you created in the previous exercises.

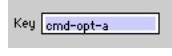
1 Do one of the following:

- If the OneClick Editor window is closed, Command-Option-click the Address button, then choose **Button Editor** from the pop-up menu.
- If the OneClick Editor is already open, click the Address button to select it, then click the **Button** tab (if necessary) to switch to the Button Editor.

2 Click the **Key** box to make it active.



3 Press Command-Option-A.



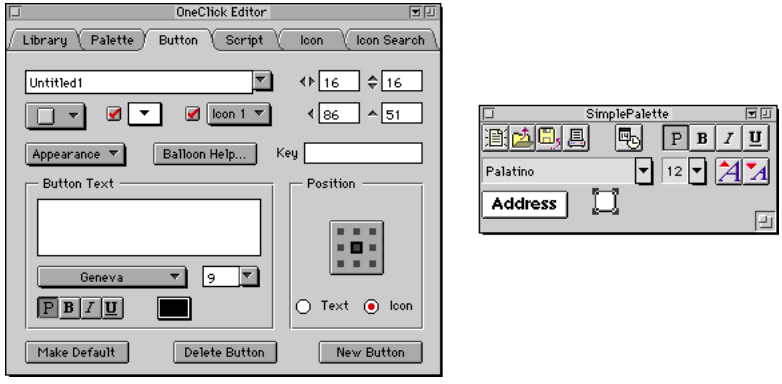
The text "cmd-opt-a" appears in the Key box. This means that when you press Command-Option-A in SimpleText, OneClick will type your address into the active window. (The shortcut doesn't work until you close the OneClick Editor.)

You can assign any key or key combination as a shortcut key. If you have an extended keyboard, you can use any of the function keys (F1–F15) as shortcut keys.

Be careful not to choose a shortcut key that might be used by the application. If you use a key that conflicts with a command key used in the application, such as

Command-N (for New), the OneClick shortcut takes precedence and overrides the application's command key.

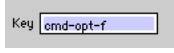
4 Click the font style button you created earlier.



The Button Editor's settings change to show the options for the selected button.

5 Click the **Key** box to make it active.

6 Press **Command-Option-F**.



The text "cmd-opt-f" appears in the Key box. Now when you press Command-Option-F in SimpleText, all the text in the active window will change to Palatino 18-point.

7 Close the OneClick Editor window.

8 Click the  button to open a new untitled SimpleText window.

9 Press **Command-Option-A** (for Address), then **Command-Option-F** (for Font).

The new window contains the same text as the previous window, except you performed the task with the keyboard. Depending on the task, shortcut keys can be a real time saver!

Where to go from here

If you're interested in learning about the Script Editor and writing scripts, then turn to Chapter 4, "Scripting Tutorial" in the *OneClick Authoring Guide*. You'll learn more about the scripts you recorded earlier: what they look like, how they work, and how you can enhance them using commands in the EasyScript scripting language.

Chapter 4
Scripting Tutorial

This chapter contains a few short tutorials that show you how to get up and running with the Script Editor and how to write some basic scripts.

These tutorials assume that you've read Chapter 3, "Getting Started with OneClick" in the *OneClick User's Guide*, which explains the basics of how to work with palettes and buttons. This chapter builds upon the SimpleText palette you created in that chapter, so if you haven't yet created your SimpleText palette, you'll need to perform the exercises in Making a custom palette in Chapter 3 of the *OneClick User's Guide* first. Then return here when you're ready to uncover the scripts on the SimpleText palette and see how they work.

We recommend that you go through these exercises in the order presented. The exercises cover the following topics:

- Viewing a button's script
- Making simple changes to a script
- Correcting errors in a script
- Getting help for script keywords
- Inserting parameters for script keywords
- Copying a script from one button to another
- Displaying information in message boxes

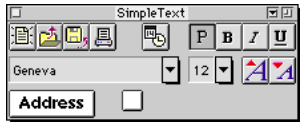
Along the way, you'll also learn a bit of the EasyScript language, such as how to use simple EasyScript commands and functions to type text, press command keys, and choose menu items.

Viewing a button's script

In the first part of this chapter, we'll take a look at the scripts you recorded earlier and make some changes to them.

1 If SimpleText is not already open, double-click the **SimpleText** icon to open it.

Your palette appears when SimpleText is open.



2 If there are no document windows open, click the  button to open a new untitled window.

3 Click the **Address** button once to type your address in the window.

The button you recorded to type your address probably works great, but what if you had made a typing error while recording? When you click the button to play back the recorded script, the button would faithfully reproduce the mistake.

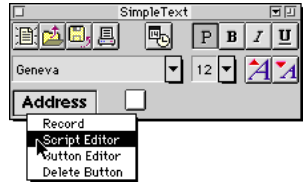
The script recorder accurately records all your actions—including any mistakes, such as typographical errors, misspellings, clicking the wrong button, or choosing the wrong menu item. To correct these kinds of errors, you can start over and record the button's script again, but that's probably not the most practical way to do it—especially if you were recording a long sequence of actions.

An easier way to correct errors made while recording is to edit the script and fix the mistakes directly. The Script Editor shows all your recorded actions in easy-to-understand *statements*, so you can quickly locate the error and fix it.

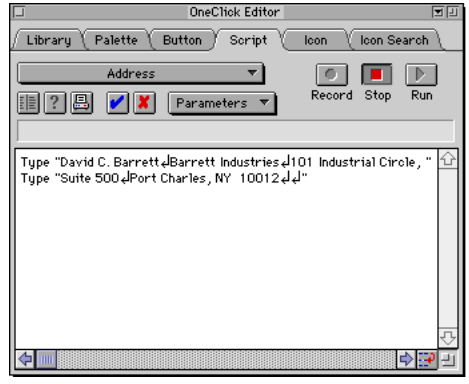
4 Do one of the following:

- If the OneClick Editor is closed, hold down the Command and Option keys, then click the **Address** button and choose **Script Editor** from the pop-up menu.

- If the OneClick Editor is open, hold down the Command and Option keys, then click the **Address** button. (No pop-up menu appears if the OneClick Editor is already open.)



The Script Editor appears. The script for the selected button (the Address button) appears in the text box at the bottom of the window.



5 If lines in the script run past the right edge of the window, drag the size box in the bottom-right corner to resize the window so you can see the rest of the script.

Checking out the script statements

Because you most likely typed your own address instead of David Barrett's address, you see some different information in the script window than the address shown above. Your script should contain one or more lines beginning with the word Type, followed by the information you typed.

Each <return> you see in the script represents a Return character. When you press a nonprintable key while recording, the script recorder shows the keystroke as the key

name in angle brackets. Other nonprintable keystrokes are represented by the following tags.

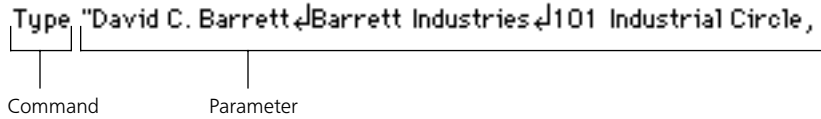
<return>	**<enter>**	**<tab>**
<esc>	**<delete>**	**<help>**
<fwddelete>	**<home>**	**<end>**
<pageup>	**<pagedown>**	**<leftarrow>**
<rightarrow>	**<uparrow>**	**<downarrow>**
<f1>	**<f2>**	**<f3>**
<f4>	**<f5>**	**<f6>**
<f7>	**<f8>**	**<f9>**
<f10>	**<f11>**	**<f12>**
<f13>	**<f14>**	**<f15>**

A script is like a mini-program: *statements* in the script tell OneClick what actions to perform when you click the button. In this script, a Type statement tells OneClick to type something as if you typed it yourself using the keyboard.

Your script may appear as one or more statements. If you type a lot of information while recording, OneClick divides the information into multiple Type statements. OneClick plays back the statements in the order in which they appear in the script.

Understanding commands and parameters

Each statement in the script contains two parts: a *command* and a *parameter*.



The *command* part of the statement, Type, tells OneClick what to do—that is, type text or other keystrokes. The Type command needs to know what text or keystrokes to type.

A *parameter* is a piece of information that you give to a command when the command needs more information to work with. The information between quotation marks (") is the text that OneClick types—the parameter for the Type command. A space separates the command name and the parameter so they don't run together. In

programmer-speak, giving information (such as the quoted text) to a command is called “passing a parameter.”

Not all commands require a parameter. The Beep command, for example, plays the Macintosh system beep. The command doesn’t need additional information to play the beep sound.

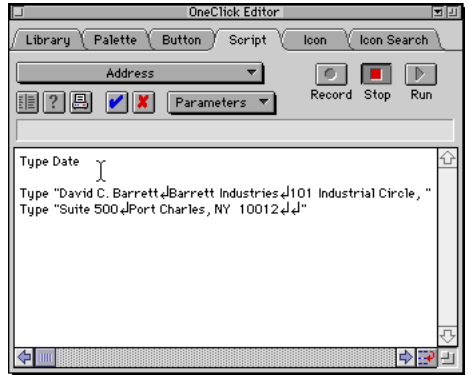
Making simple changes to a script

Now that you’ve seen the anatomy of a script, you’re ready to make some changes.

1. Click before the first **Type** statement in the script so that the cursor appears at the beginning of the line.
2. Press Return to insert a blank like.
3. Press the Up Arrow key to move the cursor back up to the first line.
4. Type the following statement in the script, then press Return to insert another blank line:

```
Type Date
```

The script window should now look something like this:



You can insert blank lines in a script to separate groups of statements, making the script easier to read. Blank lines aren’t required.

5 Click the **Run** button in the upper-right corner of the Script Editor.

Clicking the Run button plays back the script. It's the same as clicking the Address button when the OneClick Editor window is closed.

Correcting errors in a script

The script now types the current date, followed by your address, in the SimpleText window. However, the date and the first line of the address all appear on the same line. You'll need to edit the script so that it presses the Return key a few times to put some blank lines between the date and the address.

1 Click in the Script Editor window and move the cursor to the blank line following the Type Date statement.

2 Type the command **Type**, followed by a space and a quotation mark (").

```
Type "
```

3 Hold down the Option key and press Return three times.

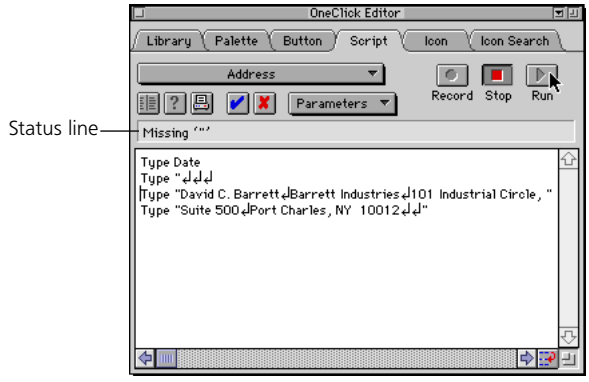
```
Type "<return><return><return>
```

Three <return> tags appear in the script. (Remember, the <return> tag in a Type statement tells OneClick to press the Return key.)

Note Holding down the Option key lets you type other special characters in a script, not just Return. You could press Option-Delete and Option-Left Arrow to insert the tags for the Delete and Left Arrow keys, for example.

4 Click the **Run** button.

The script won't run. Instead, the Script Editor beeps and displays an error message on the status line.



The message, Missing '"', means that there is a quotation mark (") missing somewhere in the script. The cursor blinks at the beginning of line below the new line you just typed, indicating that the quotation mark is probably missing on the previous line.

The parameter for the Type command, the three <return> tags, must be enclosed in quotation marks. The quotation marks tell OneClick where the parameter's text begins and ends.

OneClick won't run the script until all the errors in the script are corrected.

5 Move the cursor to the end of the line you typed in step 3, then type a quotation mark (").

6 Click the **Run** button.

The script now types the current date, followed by two blank lines and your address. The message "No errors" appears on the status line to indicate that no errors were found.

Note If you want to check a script for errors without running it, click the ☑ button in the Script Editor. The status line shows "No errors" if the script is OK, otherwise it displays an error message and highlights the error in the script.

There are additional error messages that can appear on the status line. Error messages are discussed at the end of Chapter 5, "Using the Script Editor."

OneClick always checks a changed script for errors when you select a different button, switch to another editor, or close the Script Editor window. You must correct any errors in the script before leaving the Script Editor, or else discard the changes you made to the script.

Understanding functions

The word **Date** you used in the previous exercise is called a *function*. A function gives a value, such as the current date, to a command (such as the Type command) or to another function. The Type command uses the Date function as its parameter. It takes the value given by the Date function and types it in the SimpleText window. The Date function itself doesn't do the actual typing, it just returns its value to the Type command.

The Date function you used earlier did not include a parameter. You can pass a number as a parameter to the Date function, however, that tells Date the format in which you want the date returned. For example, the statement "Date 3" returns the date in the short format, 7/23/95; the statement "Date 12" returns the date in the long format: Sunday, July 23, 1995. Without a parameter, the Date function returns the date in the default short format.

Note The names of commands, functions, and all the other words that OneClick understands are called *keywords*. A keyword is simply a single word in the EasyScript language. Type and Date are both examples of keywords.

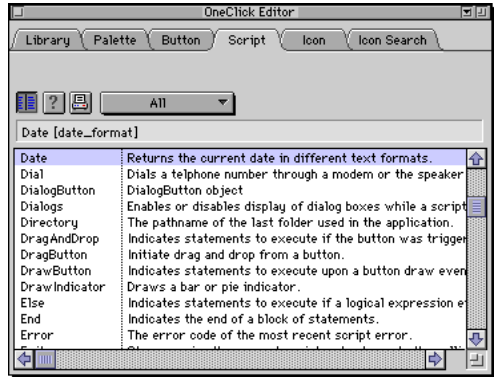
Getting help for script keywords

To find out how to tell the Date function to return the date in a different format, you need to know what numbers the Date function can use in its parameter. You can look up the information in Chapter 8, "EasyScript Reference" or you can use the online help available in the Script Editor.

1 Click the  button in the Script Editor.

A list of keywords appears in the window. This list shows all the keywords in the EasyScript language. A one-line description appears next to each keyword.

2 Type the letter "**D**" to scroll down to the keywords beginning with D.

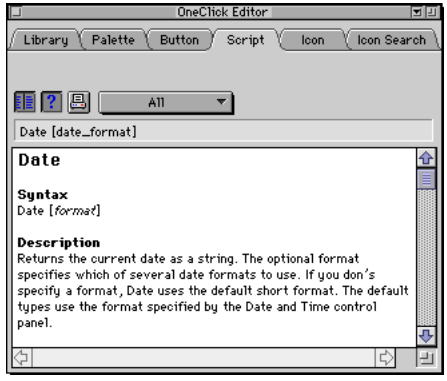


The status line shows the Date function, followed by the words "date_format" in square brackets. Date_format is the name of the Date function's parameter; the name tells you that the parameter you pass to the Date function indicates the date format to use. The square brackets around the parameter name mean that the parameter is optional—you don't need to supply a parameter to use the Date function.

The keyword list gives brief descriptions of all the keywords, but it doesn't tell you what values you can use for the format parameter.

3 Click the  button, or double-click the **Date** keyword in the list box.

Either method displays additional help for the Date function.



4 Scroll down past the Description paragraph in the help text to see a list of values for the format parameter.

As you can see in the Script Editor window on your screen, there are a variety of date formats you can use, and each format is represented by a number. Using a different number with the Date function causes the Date function to return the date in a different format. You can even add numbers together to create more formats, such as a short format with a dash separator (7-23-95).

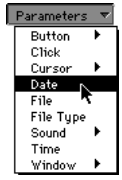
Looking up the different format numbers for the Date function may seem tedious if you want to use the function often. Fortunately, the Script Editor provides an easier way to figure out the parameter to use for the Date function (and other keywords). Also, the Date function is an exception; parameters for other commands and functions are typically much easier to remember and use.

Inserting parameters for script keywords

For the handful of commands and functions that have difficult-to-remember parameters, such as Date, the Script Editor provides an intuitive method for specifying the parameters.

1 Click the  button in the Script Editor to close the help window and return to the script.

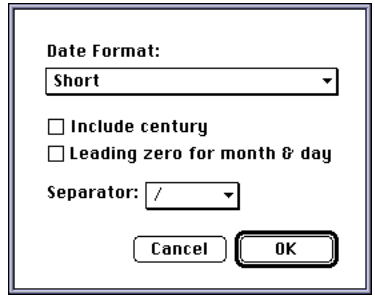
2 Move the cursor to the end of the **Type Date** statement in your script, then press the space bar to insert a space.

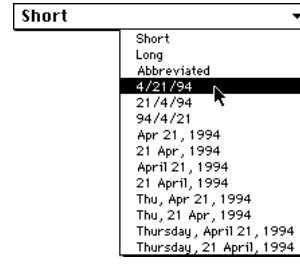


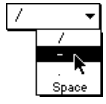
3 Choose **Date** from the Parameters pop-up menu in the Script Editor.

The Date Format dialog box appears.

4 Choose **4/21/94** from the Date Format pop-up menu.



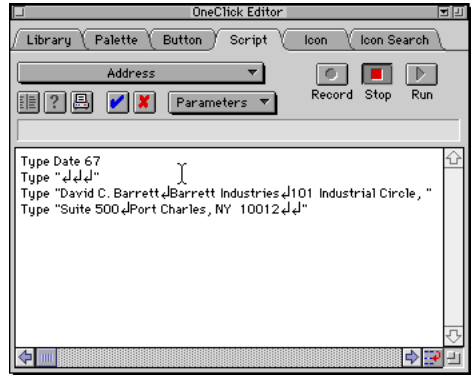




5 Choose the dash (–) from the Separator pop-up menu.

6 Click **OK**.

OneClick inserts the number 67 after the Date keyword in the script.



The number 67 is the format number that corresponds to the options you chose in the dialog box— the short M/D/YY format, using a dash (instead of a slash) for the separator character.

Note that the number 67 is *not* enclosed in quotation marks. Parameters that contain text (such as the parameters in the Type statements) must be enclosed in

quotation marks, but numeric parameters are not. If the number 67 were inside quotation marks, then OneClick might try to interpret the value as the characters "6" and "7", not the number 67.

7 Click the **Run** button to run the script.

OneClick types the date in the format M-D-YY, followed by two blank lines and your address.

Other options in the Parameters pop-up menu

Other items in the Parameters pop-up menu let you insert parameters for different keywords. For example, if you want to play a sound using the Sound command, you can either type the sound's name yourself, or you can use the Sound submenu in the Parameters menu. The Sound submenu lets you pick a sound from all the sounds available in your System file. When you choose a sound, OneClick inserts the sound's name in the script, so you don't have to type the name yourself or remember its exact spelling.

The Time option in the Parameters menu is similar to the Date option: it displays a dialog box that lets you choose options for the parameter used by the Time function. By the way, the Time function works just like the Date function—it returns the current time in a variety of formats, or a default format if you don't specify a parameter.

For information about the other options in the Parameters menu, see Chapter 5, "Using the Script Editor."

Copying a script from one button to another

You can use copy and paste to edit statements in the Script Editor. In this exercise, you'll combine the two scripts you've created into a single script for the Address button. When you run the combined script, OneClick will do the following:

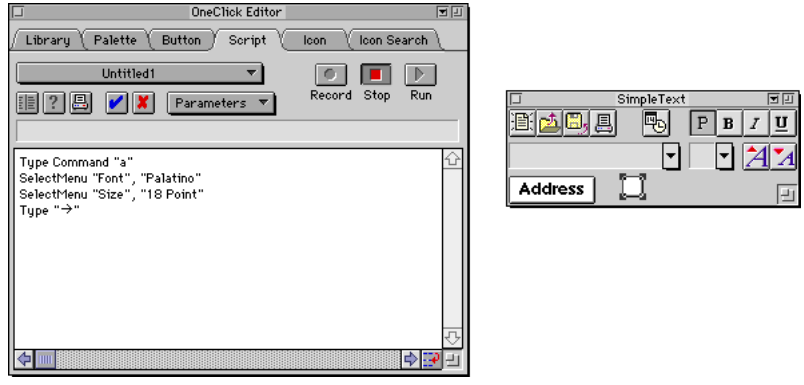
- Type the current date in the SimpleText window, followed by your address
- Select all the text in the window
- Change the font and size
- Move the cursor to the end of the document

First we'll take a look at the script that chooses commands from the Font and Size menus.

1 Open the OneClick Editor if it isn't already open.

2 Click the second button you created on the SimpleText palette (the button that selects all the text and changes the font and size).

The Script Editor shows the script for the selected button.



Note If an error message appears when you click the button, it means that the script for the Address button (the one you were just editing) contains an error. Click **Edit** to return to the Script Editor and correct the error, then try again.

There are two new keywords in this script: the **Command** keyword in the first line, and the **SelectMenu** keyword in the second and third lines.

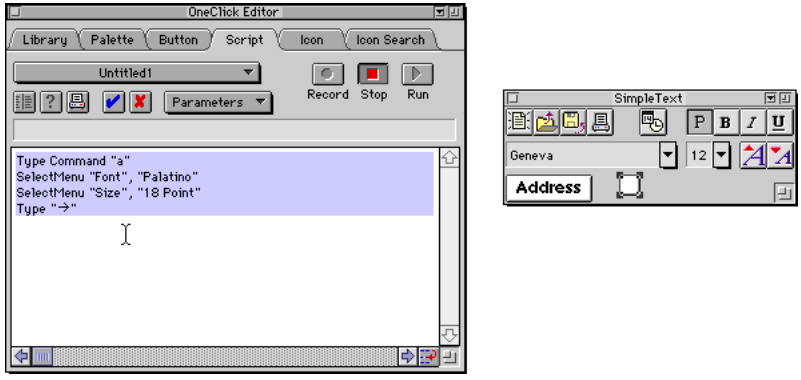
The Command keyword in the Type statement is called a *modifier*. The modifier changes the way the Type statement works. In this script, **Type Command "a"** means to hold down the Command key while typing the letter "a". When you run the script, OneClick presses Command-A to select all the text in the SimpleText window, instead of simply typing the letter "a" as text.

The SelectMenu command chooses an item from a pull-down menu in the menu bar. SelectMenu needs two parameters: the name of the menu in the menu bar and the name of the menu item to choose. In this script, the second statement

chooses Palatino from the Font menu and 18 Point from the Size menu. The menu and menu item names, like other text parameters, are enclosed in quotation marks.

3 Use the mouse to select all the lines in the script.

Shortcut To select a whole line in the script, triple-click the line. To select the whole script, quadruple-click the script or press Command-A.



4 Press Command-C to copy the script to the Clipboard.

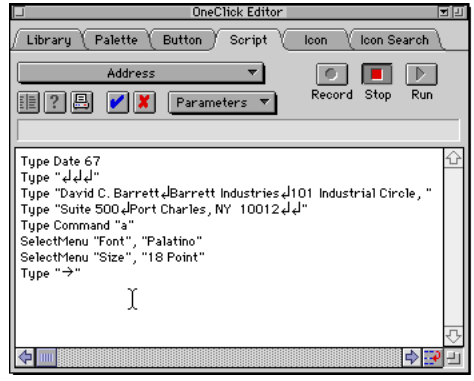
You must use the keyboard command, not the Copy command in the Edit menu. The command in the Edit menu copies text in the SimpleText window, not in the Script Editor window.

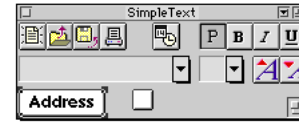
5 Click the Address button to display the button's script in the Script Editor.

6 In the Script Editor, move the cursor down to the blank line following the last Type statement.

7 Press Command-V to paste the script from the Clipboard into the script for the Address button.

Your script should now look something like this:





8 Choose **New** from the File menu to open another untitled SimpleText window.

9 Click the **Run** button in the Script Editor.

Your script now types the date, two blank lines and your address, then changes all the text to Palatino 18-point and moves the cursor to the end of the document.

Using the Check and Uncheck modifiers

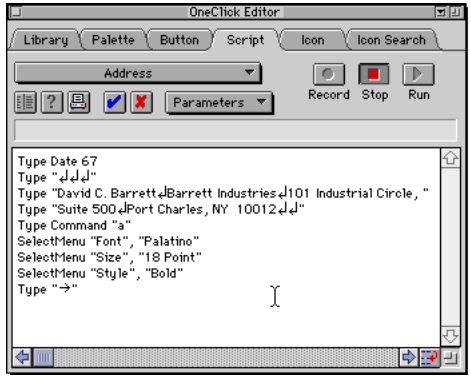
Now you'll add a new SelectMenu statement to the script. The statement will choose the Bold item from the Style menu.

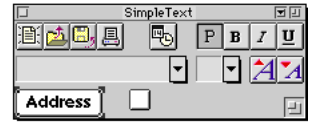
1 Move the cursor to the beginning of the last line in the script (the statement that types the Right Arrow key), then press Return to insert a blank line.

2 Type the following statement on the blank line before the last Type statement:

```
SelectMenu "Style", "Bold"
```

Your script should now look similar to this:





3 Close the OneClick Editor window.

If an error message appears, click **Edit** to return to the Script Editor. Make sure you typed the SelectMenu statement exactly as it appears above, then try again.

4 Click the Address button three or four times to run the script.

The text in the SimpleText window changes to Bold the first time you click the button. But every other time you click the button, the text alternates between Plain and Bold.

This happens because the Bold item in the Style menu is a checkmarked item. Each time the SelectMenu command chooses the menu item, it toggles the item on and off. This is a good example of why it's sometimes necessary to edit a script after you've recorded it—if you had simply recorded the menu choice, then ran the script a few times, the SelectMenu command would always toggle the Bold item on and off each time you ran the script. In cases like this, you'll need to do a little fine-tuning to make the script work the way you want.

To make the Bold option turn on and *stay* on, you'll add a modifier to the SelectMenu statement.

5 Hold down the Command and Option keys and click the Address button, then choose **Script Editor** from the pop-up menu.

The script for the Address button appears in the Script Editor.

6 Type the word **Check** in the last SelectMenu statement so that the statement reads as follows:

```
SelectMenu Check "Style", "Bold"
```

The Check keyword is a modifier that changes the way the SelectMenu command works. Using Check in a SelectMenu statement means that SelectMenu turns on the Bold item if it wasn't already on (checked). If the Bold item is already checked, then the SelectMenu statement does *not* choose the Bold item, because doing so would turn it off.

7 Click the **Run** button in the Script Editor a few more times.

Now when you run the script, the text in the SimpleText window changes to Bold and stays that way each time you run the script.

The SelectMenu command has another modifier, **Uncheck**, that you can use to turn off a checked menu item instead of turning it on (like Check) or toggling the menu item. You don't need to use Check and Uncheck with regular menu items, just checkmarked items when you want to force the checkmark on or off.

The SelectButton command

The **SelectButton** command is a command you'll see often in recorded scripts, but we haven't actually used this command in any of the scripts in this chapter.

SelectButton clicks a button or a checkbox in a dialog box or window. The command needs one parameter, the name of the button to click. Here are some examples:

```
SelectButton "OK"
SelectButton "Cancel"
SelectButton Uncheck "Smooth Graphics"
```

The first two statements click the OK and Cancel buttons in a dialog box. The third statement clicks a checkbox named Smooth Graphics (from the Page Setup Options dialog box).

As you see in the third statement, the SelectButton command can also use the Uncheck (and Check) modifiers. When the SelectButton command clicks a checkbox, it turns the checkbox on or off like a toggle. You can use the Check or Uncheck modifier to force the checkbox either on or off, just as you do with the SelectMenu command and checked menu items.

Displaying information in message boxes

The scripts you record and write aren't limited to choosing menu items, clicking buttons, pressing command keys, and typing text. Many other keywords let your scripts make decisions based on certain conditions, get input while the script runs, display information in message boxes, display pop-up menus, and more. Here's an example of one of the commands.

1 Close the OneClick Editor window if it's open.

2 Command-Option-click on an empty space on the SimpleText palette (not on a button), then choose Script Editor from the pop-up menu.

When you click on the palette's background and choose Script Editor, OneClick creates a new button and opens the Script Editor for the new button. It doesn't start recording.

3 Type the following line in the script window:

```
Message "Hello, World!"
```

The **Message** keyword is a command that shows a message box on the screen. When you run the script, the box contains the message "Hello, World!" and an OK button.

4 Click the **Run** button in the Script Editor.

A message box appears.

5 Click **OK** to close the message box.

Where to go from here

You've now learned the basic principles of scripting with OneClick. While working on your Address button, you learned how to do the following:

- Access the Script Editor and view a button's script
- Edit a script
- Get help for script commands
- Use the Parameters menu
- Run a script
- Correct mistakes

You have also learned a bit about the EasyScript language, such as how to use commands, functions, parameters, and modifiers.

There are many more features available in the EasyScript language than the few that you worked with in this chapter. Also, we haven't covered every single option available in the Script Editor. The following chapters cover the Script Editor and the EasyScript language in greater detail.

Appendix A

Control Panel Settings

Changing settings in the OneClick control panel

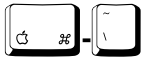
Changing the hot keys

OneClick provides two default hot keys you can use to show or hide all palettes and open the OneClick Editor.

▶ **To change the hot key for showing or hiding all OneClick palettes**

1 Click **Show/Hide Palettes**.

2 Type a new hot key, then click **OK**.

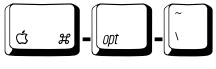


The default hot key is Command-`.

▶ **To change the hot key for opening the OneClick Editor window**

1 Click **Show/Hide Editor**.

2 Type a new hot key, then click **OK**.



The default hot key is Command-Option-`.

Choosing where the OneClick menu appears

OneClick gives you two choices of where the OneClick menu appears: an icon menu in the menu bar and a submenu in the Apple menu.

▶ **To turn on or off the OneClick menu in the menu bar or Apple menu**

■ Check or uncheck the **in Menu Bar** or **in Apple Menu** checkboxes.

In a button's script, you can use the PaletteMenu script command to make the OneClick menu appear when you click the button.

Disabling the OneClick Editor

You can disable the OneClick Editor to prevent other users from modifying palettes. This feature is useful for network administrators who don't want inexperienced OneClick users to be able to modify standard palettes used by their workgroup or department.

▶ **To disable the OneClick Editor**

■ Uncheck the **Enable OneClick Editor** checkbox.

The OneClick Editor option appears dimmed in all OneClick menus.

Disabling the startup icon

The OneClick icon appears each time your Macintosh starts up.

▶ **To turn off the startup icon**

■ Uncheck the **Show Startup Icon** checkbox.

Appendix B
Shortcuts

Button shortcuts

The following shortcuts work only when the OneClick Editor is closed.

To do this	Do this
Open the Script Editor for a button	Command-Option-click a button, then choose Script Editor from the pop-up menu.
Open the Button Editor for a button	Command-Option-click a button, then choose Button Editor from the pop-up menu.
Create a new button and record a script	Command-Option-click the palette's background where you want the new button to appear, then choose Record from the pop-up menu. Click the button to stop recording.
Record a new script for an existing button	Command-Option-click a button, then choose Record from the pop-up menu. Click the button to stop recording.
Delete a button from a palette	Command-Option-click a button, then choose Delete from the pop-up menu.
Get Balloon Help for a button	Hold down Shift and Option, then point to a button.
Stop execution of a button's script	Press Command-period.
Stop recording a script	Press Command-` (the Hide/Show Palettes shortcut).

Palette shortcuts

The following shortcuts work only when the OneClick Editor is closed.

To do this	Do this
Open the Palette Editor for a palette	Command-Option-click a palette's background, then choose Palette Editor from the pop-up menu.
Close all palettes	Option-click a palette's close box.
Zoom all palettes	Option-click a palette's zoom box.
Bring a palette in front of other palettes	Click the palette's title bar or background, or hold down Option and choose the palette's name from the OneClick menu.
Move a palette without a title bar	Hold down Shift, Option, Command, or Control and drag the palette's background.
Click "through" a palette to the underlying window or dialog box	Hold down Shift and Control, then click in the palette. The click passes through the palette to the window underneath, allowing you to click in a window when a palette is in the way.
Hide all palettes or show hidden palettes	Press Command-` (accent). You can change this shortcut in the OneClick Control Panel.
Hide or show the OneClick Editor window	Press Command-Option-` (accent). You can change this shortcut in the OneClick Control Panel.
Display the OneClick menu as a menu of tear-off palettes	Hold down Command and Option, then choose a palette from the OneClick menu.

OneClick Editor shortcuts

These shortcuts work only when the OneClick Editor window is open; it doesn't matter which editor is active.

To do this	Do this (when the OneClick Editor is open)
Select a palette for editing and target the palette for keystrokes	Click the palette.
Switch to the Palette Editor	Double-click the palette's background.

To do this	Do this (when the OneClick Editor is open)
Resize a palette	Drag the palette's grow box.
Move a palette without a title bar	Option-drag the palette's background.
Select one or more buttons for editing	Click a button to select it. Shift-click to select additional buttons. (Or drag to surround the buttons you want to select.)
Select all buttons on a palette	Click a button or the palette's background to select the palette, then press Command-A.
Switch to the Button Editor	Double-click a button.
Move buttons one pixel	Select the button(s) to move, then press the arrow keys.
Move buttons by the amount specified for the palette grid	Select the button(s) to move, then hold down Command and press the arrow keys.
Resize a button	Select the button(s) to resize, then drag one of the four handles on the button.
Duplicate a button	Select the button(s) to duplicate, then press Command-D.
Create a new button	Click a palette's background to select the palette, then press Command-N.
Delete a button	Select the button(s) to delete, then press Delete.
Restore the most recently deleted button(s)	Select the palette from which the buttons were deleted, then press Command-Z to undo. (Undo works only if deleting a button was the most recent action you performed.)
Display the standard 34 Apple icon colors (instead of 256) in a pop-up color menu	Option-click the pop-up color menu. (This shortcut works only when your monitor is set for 256 or more colors.) These are colors that Apple recommends for color icons.
Cut text from any text field to the clipboard	Select text, then press Command-X.
Copy text from any text field to the clipboard	Select text, then press Command-C.
Paste text from the clipboard into any text field	Click the text field, then press Command-V.
Close the OneClick Editor window	Press Command-W. (This works only if the editor is targeted for keystrokes.)

Button Library shortcuts

To use these Button Library shortcuts, first make the library active by clicking one of the fields in the Button Library window. A dark outline surrounding the window's title bar shows that the window is active.

To do this	Do this
Select all of a button's Balloon Help text	Click in the text, then press Command-A.
Press the Find button after typing a keyword	Press Return.

Icon Editor shortcuts

To use these Icon Editor shortcuts, first make the editor active by clicking one of the fields in the Icon Editor window. A dark outline surrounding the window's title bar shows that the window is active.

To do this	Do this
Select the entire icon with the selection tool	Press Command-A.
Copy the selection to the clipboard	Press Command-C.
Paste the selection from the clipboard	Press Command-V.
Undo the last editing action	Press Command-Z.
Delete the selection	Press Delete.
Drag the selection without erasing it (drag a copy of the selection)	Option-drag the selection.
Fill all pixels that are the same color as the pixel you click	Select the fill tool (the paint bucket), then Command-click a pixel.
Temporarily switch from any tool to the Dropper to pick up a color	Hold down the Option key, then click a color to change the current draw color. If the Eraser is the selected tool, then the Dropper changes the current erase color, not the draw color.

To do this	Do this
Use the Dropper to change the erase color instead of the draw color	With the Dropper tool selected, hold down any modifier key (Command, Shift, Option, or Control) and click a color.
Save the icon and update the button	Press Command-S.

Script Editor shortcuts

To use these Script Editor shortcuts, first make the editor active by clicking one of the fields in the Script Editor window. A dark outline surrounding the window's title bar shows that the window is active.

To do this	Do this
Select a word in a script	Double-click the word.
Select a line in a script	Triple-click the line.
Select all text in a script	Press Command-A or quadruple-click the script.
Cut text to the clipboard	Press Command-X.
Copy text to the clipboard	Press Command-C.
Paste text from the clipboard	Press Command-V.
Undo the last editing action	Press Command-Z.
Check the script's syntax, save the script and update the button	Press Command-S.
Check the script's syntax and run the script	Press Command-R.
Print the script	Press Command-P.
Toggle between the script window and the keyword list	Press Command-Tab.
Insert special characters in a script (such as Return, Delete, or arrows)	Hold down Option and type the character (Option-Return, Option-Delete, Option-Left Arrow, and so on).

Appendix C
Customer Support

At WestCode Software, our company mission is to create outstanding, innovative products that improve the way you use your Macintosh. This package reflects our dedication to producing well-designed software combined with comprehensive user documentation that helps you get the most from your purchase.

Our goal is to ensure that you're satisfied with our products and service. If you ever have a question or problem with one of our products, and you can't find an answer in the manual, send us email or call us for a quick solution.

Support by email

We prefer that you receive your tech support through email to **support@westcodesoft.com**. Email is usually faster for both parties.

Most tech reports lack vital information. In your report to tech support, please include:

- your name, email address, or other contact information
- product version and your registration number (if any)
- symptoms of the problem
- the steps you have taken to isolate the problem

Most of our products have a **Create Report For WestCode Support** command in their Configure menu. The report created by this command gives us some comprehensive system information that can help tech support diagnose the problem, including your Mac OS version and machine type, OneClick version, whether or not related extensions (such as AppleScript) are installed, and a list of your extensions and control panels.

If you think that your question is a simple one and does not require much system information, feel free to email us directly.

From time to time, WestCode Software makes new OneClick palettes, button libraries, and plug-in extensions available online. To access WestCode's web site, visit http://www.westcodesoft.com.

Support by phone or mail

Our knowledgeable and friendly technical support staff is available to help you from 9:00 A.M. to 5:00 P.M. Pacific time, Monday through Friday. Call or write to:

(619) 487–9200 General information, sales, and technical support
(619) 487–9255 Fax

WestCode Software, Inc.
Attn: Technical Support
11835 Carmel Mountain Rd, Suite 1304
San Diego, CA 92128

When calling, you should be at your computer, have this manual handy, and be prepared to give the information provided in the Report for WestCode Support (see the previous section).

Product registration

Please take a moment to complete the Registration Card included with this package. Registration will allow us to notify you of product improvements and to tell you about new products. Remember, product support is available only to registered users.

Customer comments

We're always looking for ways to improve our products and we frequently add new features requested by our customers. We welcome and encourage your comments for making our products and service even better. We're also interested in ideas for new products that you'd like to see produced.

Index

Acknowledgments

Button Icon Design

Suling Wang
David Lai
Brian McMurdo

Button Scripts

John Oberrick
Jeff Jungblut
Mark Brooks
Rob Renstrom
Alan Bird

Beta Testing

John Andrews
Randy Brandt
Ralph Brown
Dave Case
Ed Glassgow
Richard Kephart
Guy McLimore
Bobby Saha
Larry Sherman
Martin Tschofen
Abbi Vakil
Dan Verkade
Chuck Walker
Norm Weiner

Dedicated To

Anna, Zachary, and Jacob Renstrom
Fiona and Connor Oberrick
Melanie, Chris, Alisha, Cameron, Jared and Ryan Bird

Special Thanks

David Barrett
Jimmy and Kathy Dial
Clayton and Leslie Evans
Phil Feder
Daniel Foote
Ward Graham
Van Haynie
Jack and Marilyn Oberrick
Ken and Marie Renstrom
Gerry and Karen Renstrom
Derek and Connie Smith
Jeff Taylor
Brian and Carol Turney
Don and Gloria Wuckert

Programming Supplies

Taco Bell®
In-N-Out Burger®
Coca-Cola Company®