24U SimpleHelp Plug-In 3.2

User's Guide

24U Software

July 7, 2006

Contents

List of Figures

Chapter 1

Introduction

Welcome to 24U SimpleHelp Plug-In. This FileMaker plug-in allows you to make your FileMaker databases easier to use by implementing help tags (also known as tool tips), guided tours, roll-over effects and other context sensitive help. Doing so will save you a lot of time you would otherwise spend by answering frequent technical support calls or writing complex manuals.

1.1 Using this manual

This User's Guide contains an introduction to 24U SimpleHelp Plug-In's features and provides step-by-step instructions on implementing the most common guidance into FileMaker solutions. The main part of this manual closely corresponds with the **SimpleHelpTutorial.fp7** file which you can use to test the described functions directly in FileMaker Pro or FileMaker Pro Advanced. You will also find a complete reference of 24U SimpleHelp Plug-In's functions, their parameters, and possible returned results at the end of this guide.

Note: All information referring to "FileMaker Pro" in this guide applies to both FileMaker Pro and FileMaker Pro Advanced products unless explicitly specified otherwise.

1.2 Using other documentation

The 24U SimpleHelp Plug-In's documentation provides a variety of different learning paths to making your FileMaker development easier and more efficient by implementing 24U SimpleHelp Plug-In into your solutions.

The following documents are included with the plug-in

- **ReadMe.html** is an intentionally brief introduction to the plug-in, which you can easily read and quickly find basic general information there. It also contains a short Troubleshooting section with the most frequently asked support questions answered.
- **UsingFMPlugIns.html** is an introduction to FileMaker plug-ins in general. If you have never used FileMaker plug-ins before, read this document to see how easy and safe it is to use plug-ins to extend FileMaker's abilities.

- **SimpleHelpUsersGuide.pdf** (this manual) contains comprehensive printable instructions for using 24U SimpleHelp Plug-In. The guide is structured the same way as the tutorial mentioned below, so that you can easily test the techniques you read about by jumping to the tutorial.
- **SimpleHelpTutorial.fp7** is a FileMaker document that will guide you through all the different features of 24U SimpleHelp Plug-in, while letting you immediately try them out and practice by modifying the predefined fields and sample scripts.
- **ReleaseNotes.html** lists all changes and feature additions across past version of 24U SimpleHelp Plug-in. This information will help you to maintain compatibility of your solutions when updating to a new version of the plug-in.
- **HowToRegister.html** contains detailed description of all available licensing options, along with instructions for obtaining your own license for 24U SimpleHelp Plug-In.
- **LicenseAgreement.html** defines the legal terms under which you can use 24U SimpleHelp Plug-In within your license. If you get a developer license, see also the Support folder for the **SampleAboutScreen.pdf** and **LicensePasteIn.html** files, which serve as examples of about screen and license agreement terms for your solution.

Even more useful learning resources can be found in the following folders:

- **Examples** contains simple but meaningful FileMaker solutions which utilize the features offered by 24U SimpleHelp Plug-In.
- **Support** contains additional tools that make implementing 24U SimpleHelp Plug-In into your solutions even easier.

In addition to the documentation bundled with the plug-in, you can always find new and updated information on the `http://www.24uSoftware.com` web site. Make sure to check it regularly to make your work simple.

1.3 24U SimpleHelp Plug-In Overview

Here is an overview of the new abilities you add to FileMaker Pro by installing 24U SimpleHelp Plug-In.

1.3.1 Help Tags (Tool Tips)

With 24U SimpleHelp Plug-In your solution can display short onscreen hints for user interface elements to the user. This context sensitive help is displayed in small boxes usually called tool tips or help tags. We will use the term help tags further in this manual.

Help tag is typically displayed when the user moves the mouse pointer over the user interface element, such as button or data field, and waits a few seconds. This indicates to the application that the user is not sure about the meaning of the object, and so the application responds by displaying a help tag. The example of such a bad user experience is shown at figure 1.1 and figure 1.2 shows the expected behavior.

In solutions with graphical user interface based on icons and pictures it is essential to provide short descriptions for buttons to the user. Even one-word description can save the user a lot of time.

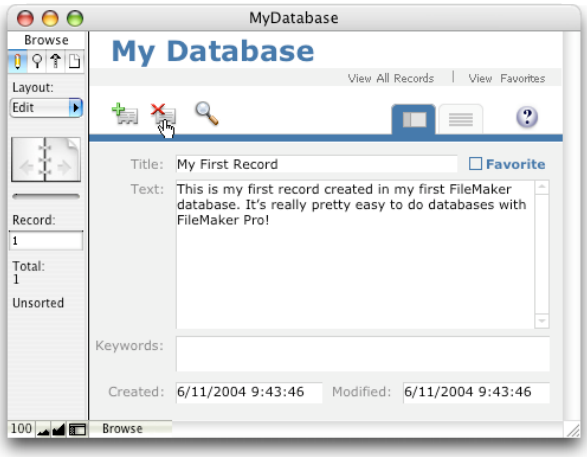

Figure 1.1: A confusing graphical button

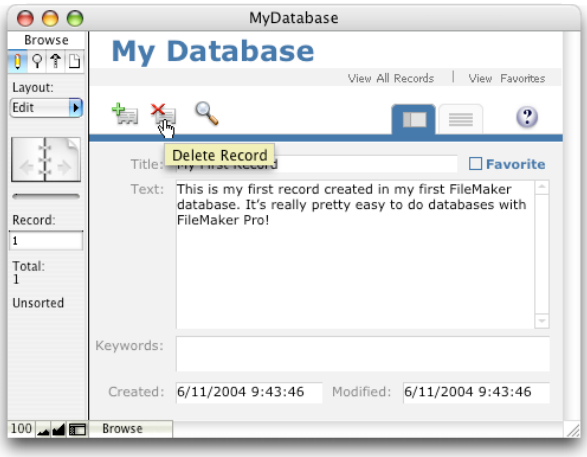

Figure 1.2: A graphical button explained by help tag

However, if a more detailed description is necessary, 24U SimpleHelp Plug-In can still help. 24U SimpleHelp Plug-In lets you define help tags for all kinds of layout objects, such as buttons, fields, rectangles, ovals, or pictures. You can define different sets of help tags for different layouts, you can calculate the help tag's text on-the-fly at the time when it is to be displayed. You can also easily display different hints for the same object based on the modifier keys currently pressed.

For more information about using help tags, see section 2.1 at page 17.

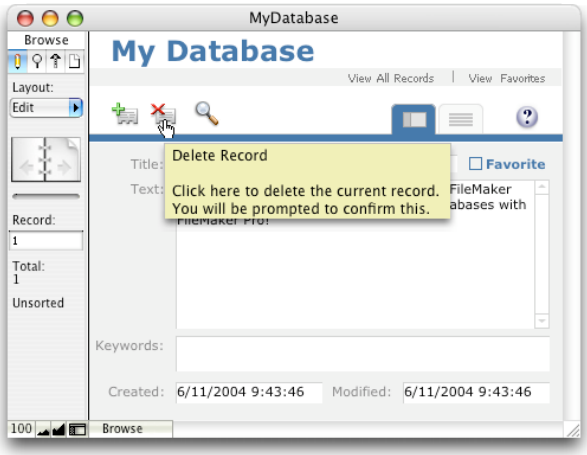

Figure 1.3: A graphical button explained in detail

1.3.2 Coachmarks

A coachmark is an onscreen graphic that circles or points to an item on the screen. With 24U SimpleHelp Plug-In you can use coachmarks in FileMaker solutions to guide the user's attention to screen areas described in a help instruction, typically if the user needs to perform an action (for example, to click on a button or type information into a field). Sample coachmark is depicted at the following picture:

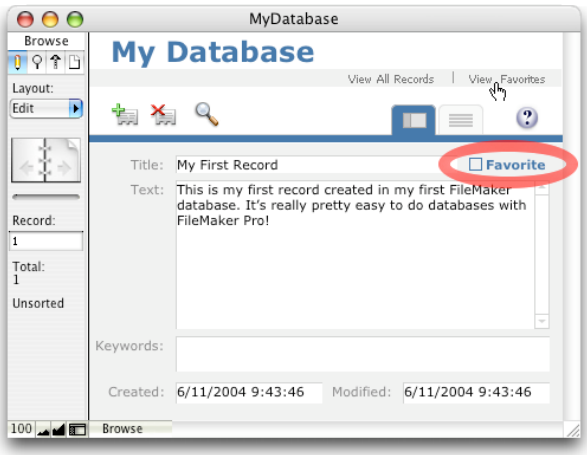

Figure 1.4: A coachmark guiding the user to a checkbox

With the help of coachmarks, you can develop step-by-step guided instructions for difficult processes, and teach the user effectively by emphasizing important user interface elements.

For more information about using coachmarks, see section 2.2 at page 23.

1.3.3 Roll-over effects

The world wide web probably made you used to see and/or hear some effect when you roll the mouse pointer over an active link. We call this a "roll-over effect." The only roll-over effect FileMaker supports

by its nature is changing cursor's shape after moving the mouse pointer over a button. 24U SimpleHelp Plug-In extends this ability to any other effect you can implement by FileMaker scripts.

With 24U SimpleHelp Plug-In you can have a script triggered when the mouse pointer enters a defined "hot rectangle" on your layout, and another script can be triggered when the mouse pointer leaves this "hot rectangle".

For example, if you define a text object as a button, you can make the text to change its color and style in response to moving the mouse pointer over it, just like web browsers usually highlight links.

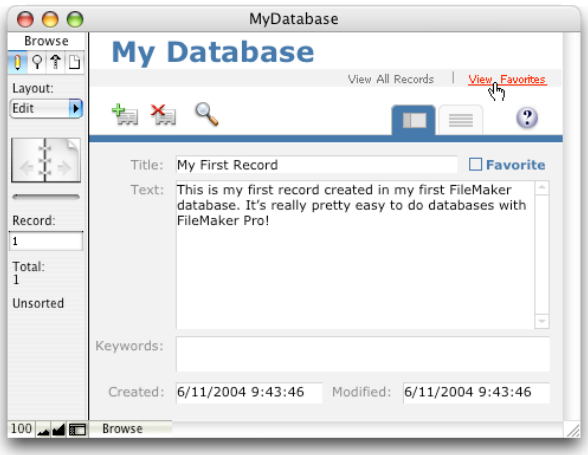

Figure 1.5: A text button activated by roll-over

For more information about using roll-over effects, see section 2.3 at page 25.

1.3.4 Special features

In order to build an effective help system, you may want to combine help tags, coachmarks and roll-over effects with additional functions. A few such special features are already available in 24U SimpleHelp Plug-In.

- When displaying a dynamic help tag, or activating a roll-over effect, 24U SimpleHelp Plug-In lets you check the current mouse position, so that you can react more precisely to the user's action.
- After displaying a coachmark, you may require the user to click somewhere. To check if the user has really clicked on the right position, 24U SimpleHelp Plug-In lets you check where the mouse pointer was at the time of last mouse click.
- Although these special features were added to 24U SimpleHelp Plug-In mainly to extend the usability of its main features, of course you can use them also separately, for other specific purposes. For example, you can easily implement a clickable image map, that is a single button triggering a single script but performing different actions base on the actual position within the button where the user clicked.

For more information about special features, see chapter 4 at page 35. The actual way you will utilize them is limited only by your imagination.

1.4 Installation

For a plug-in to be recognized, you have to copy it to a folder where FileMaker Pro will recognize it. This is the Extensions folder inside the FileMaker application's folder.

To install 24U SimpleHelp Plug-In, find the plug-in's version for your operating system, and copy it to the FileMaker's Extensions folder. Use the plug-in with the "fmplugin" extension for Mac OS X, and the one with the "fmx" extension for Windows.

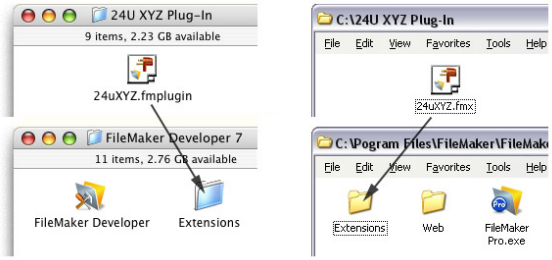

Figure 1.6: Installing plug-in on Mac OS X and Windows

1.5 Registration

24U SimpleHelp Plug-In is shareware and if you don't register it, it will stop working when your trial period is over. After purchasing a license, you will receive a code, which you can use to register your copy of 24U SimpleHelp Plug-In.

There are *two* ways how to register 24U SimpleHelp Plug-In. If you have an **end-user** license (different from a developer license), you can enter your registration code into the plug-in's registration dialog, and have it stored permanently on your hard disk.

Figure 1.7: Entering an **end-user** license code in the registration dialog

The registration dialog should appear automatically when you use 24U SimpleHelp Plug-In for the first time after launching FileMaker Pro. If the registration dialog does not appear automatically, you can force it to be displayed by opening Application Preferences, going to Plug-Ins, selecting 24U SimpleHelp Plug-In, and clicking on its Configure button.

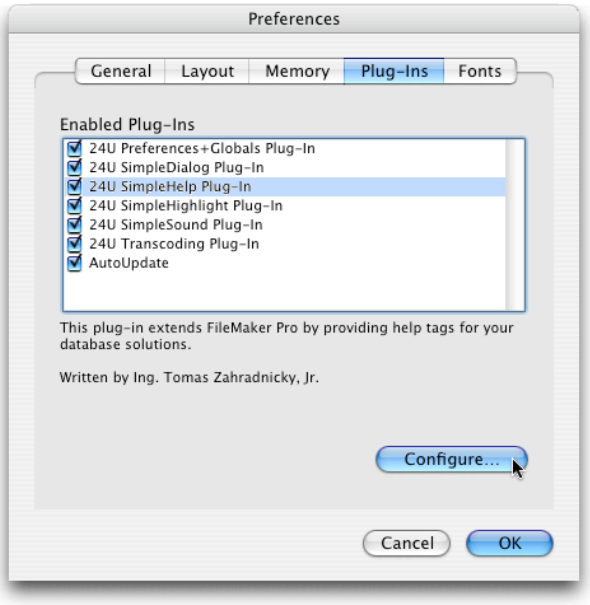

Figure 1.8: Invoking the registration dialog manually

If you have a **developer** license, or if you want to automate installation of 24U SimpleHelp Plug-In in a networked multi-user solution, you can register the plug-in also programatically from within a script.

To make sure that the plug-in gets registered before it is used for the first time, call the following script from within your solution's startup script:

```
If [SHelp_Register("put your registration code here") <> 0]
 # Registration failed, plug-in will not work
Else
 # The plug-in has been granted to work
End If
```

To define a startup script, choose the File Options item from the File menu, then check the "Perform script" checkbox in the "When opening this file" group, and select your startup script from the pop-up menu.

Important: If you pass an invalid code to the `SHelp_Register` function, a registration dialog will appear, and so you still have a chance to register the plug-in. But if you fail to register it, the plug-in will stop working and return an error code as a result of the registration function. So make sure to include an appropriate handling code in your registration script for the case when the plug-in refuses to work due to an invalid registration.

Note: Using a developer license registration code may cause a splash dialog to appear on the screen for about 2 seconds at the quit time of your solution. The splash helps you to comply with our License Agreement without having to design your own about screen. If you don't want your solution to show the splash dialog, send us samples of your about screen and license agreement to `registration@24uSoftware.com`. We will send you instructions for disabling the splash as soon as your solution is proven to comply with the developer license's sublicensing conditions.

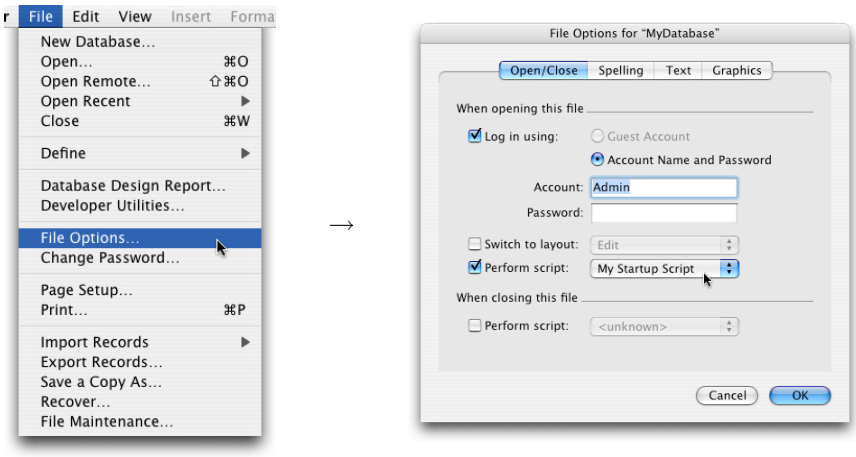

Figure 1.9: Selecting a startup script in the File Options window

1.6 Support

Before you contact our technical support regarding problems or difficulties using 24U SimpleHelp Plug-In, please check if your issue is not listed in the Troubleshooting section of the **ReadMe.html** document.

If your problem is not listed in the **ReadMe.html** document or if the recommended solution does not work for you, please visit our technical support web page at `http://www.24uSoftware.com/support.php` for assistance. If you don't find a solution for your issue in our support database, you can use our on-line contact form to send a question to our technical support. You can also send your question directly to our support e-mail address `support@24uSoftware.com`.

Please make sure to specify exact steps to reproduce your issue, as well as relevant detailed information about your system configuration, when contacting our technical support. If we are able to reproduce your situation, we will not have to request more information from you, and so we will likely be able to provide a solution to your problem in the very first response.

1.7 Updating from previous version

Before updating your copy of 24U SimpleHelp Plug-In to a newer version, you should consult the **ReleaseNotes.html** document for any changes that might affect compatibility of your solution with the new version.

Although we are trying to make all new plug-ins compatible with their previous versions, you should always check if the currently available version of 24U SimpleHelp Plug-In is the one you expect. You should perform this check before using any of the plug-in's functions, as using a wrong version may result in unexpected behavior.

To actually update the plug-in, first quit FileMaker Pro, then remove the old version of the plug-in, and then follow the installation instructions. Always make sure that you don't have two different copies or versions of the same plug-in installed in the same FileMaker Pro application at the same time.

The best way to check 24U SimpleHelp Plug-In's version is to call a script like this from within your solution's startup script:

```
If [SHelp_Version("") = "" or SHelp_Version("") = "?"]
  Show Custom Dialog ["Error"; "24U SimpleHelp Plug-In is missing or not active."]
  # Make sure the plug-in is not used
Else
  If [SHelp_Version("") <> "3.2"]
    Show Custom Dialog ["Error"; "An unsupported version of 24U SimpleHelp Plug-In
    is installed."]
    # Make sure the plug-in is not used
  Else
    # It is safe to use the plug-in
  End If
End If
```

Tip: You can get more information from the `SHelp_Version` function by passing a parameter to it. The `SHelp_Version("long")` call will return the plug-in's official name followed by version number (i.e. "24U SimpleHelp Plug-In 3.2"). The `SHelp_Version("platform")` call will return an identification of the plug-in's code currently running. Although you will probably never use this value in your solutions, it may be helpful as an additional information when specifying a question for our technical support.

1.8 Automated Installation

Each required plug-in must be installed and registered on every client (every computer running FileMaker Pro and connecting to FileMaker Server) calling its functions. This may be very tiring when you have tens of clients in a network. Another pain is re-doing the installation each time the plug-in gets updated.

FileMaker Server cannot host plug-ins to FileMaker clients, but it can distribute plug-ins to your clients using an Auto Update feature. By putting all your plug-ins into a special folder on the server, you make them available through the FileMaker's Auto Update plug-in, which can automatically install any required plug-in if it is not installed for a local client or if the local version is not the right version. See the documentation accompanied with FileMaker Server for more information on using the Auto Update feature.

You can ease the installation and updates of 24U SimpleHelp Plug-In and other plug-ins to all your networked FileMaker clients from FileMaker Server by using 24U Plug-In AutoInstaller, which comes pre-configured with 24U SimpleHelp Plug-In for your convenience.

After being incorporated into your solution 24U Plug-In AutoInstaller will maintain a list of all plug-ins required by your solution, alongwith their required versions and registration codes, and remove all your pain from using plug-ins. This utility will automate installations, registrations, and updates of all the plug-ins you use in your multi-user FileMaker solutions. It comes pre-configured for 24U FileMaker Plug-Ins, but it can be used for other 3rd party plug-ins as well.

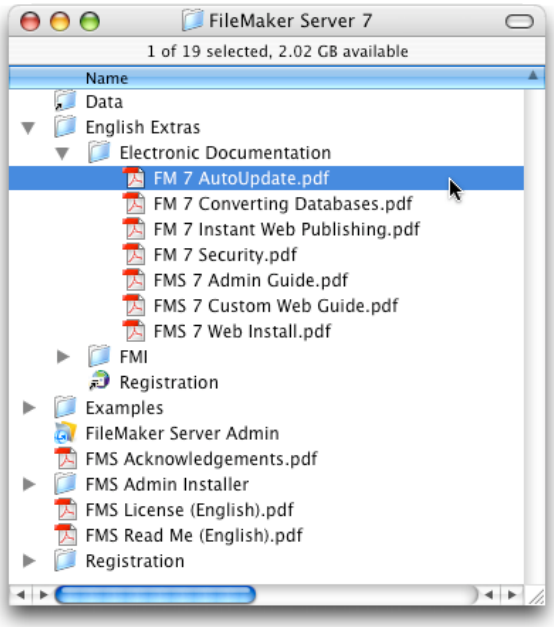

Figure 1.10: FileMaker's documentation on Auto Update

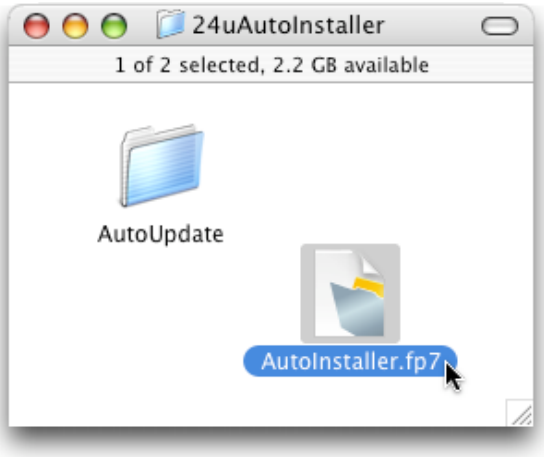

Figure 1.11: 24U Plug-In AutoInstaller in the Support folder

Chapter 2

Using 24U SimpleHelp Plug-In

This chapter explains how you can easily use 24U SimpleHelp Plug-In to:

- define tool tips (help tags) for buttons, fields, and other layout objects
- display different context-sensitive help based on current conditions
- guide the user through a procedure using persistent help tags
- guide the user to a specific object by drawing a coachmark around it
- indicate active areas by providing roll-over effects to the user

Note: Examine the **24uSimpleHelpTutorial.fp7** file to see the individual features explained here actually implemented. Try to edit the file's scripts to modify the plug-in's behavior.

2.1 Using Help Tags

The basic feature of 24U SimpleHelp Plug-In is the ability to display a help tag (also known as tool tip) when you move the mouse pointer over a specific object, such as button or field. This works by first defining rectangular areas that are sensitive to mouse pointer location. We call them hot rectangles.

As long as the mouse pointer is outside of a hot rectangle, nothing special happens. After moving the mouse pointer over the hot rectangle and leaving it there for some time, a help tag appears and stays on the screen until the mouse pointer leaves the hot rectangle.

Figure 2.1: Mouse pointer outside and inside a hot rectangle

2.1.1 Static Tags

The simplest kind of help tag supported by 24U SimpleHelp Plug-In is a static help tag. It is permanently attached to a single hot rectangle of a single layout in a single FileMaker database. Its definition (hot rectangle and text to display) stays unchanged until it is detached.

To define a hot rectangle and attach a help tag to it, use the `SHelp_AttachTags` function. In the following script, a hot rectangle with coordinates 100, 100, 200, 130 gets attached a help tag with the text "This is the sample tag".

```
If [SHelp_AttachTags("100,100,200,130"; "This is the sample tag") <> 0]
  # Attaching a tag failed because of an error (or maybe invalid parameters)
End If
```

Note that this script attaches the help tag only for the current layout of the frontmost database. To make the same hot rectangle work also in another layout, you need to switch to that layout, and attach the help tag again. Alternatively, you can utilize named layouts described in section 2.1.6 at page 22, and a global layout described in section 2.1.7 at page 23.

All attached help tags get automatically detached (removed) when you close the database, or quit the FileMaker application. If you want the help tags to work the next time you open the database, create a startup script, and call your attaching script(s) from it.

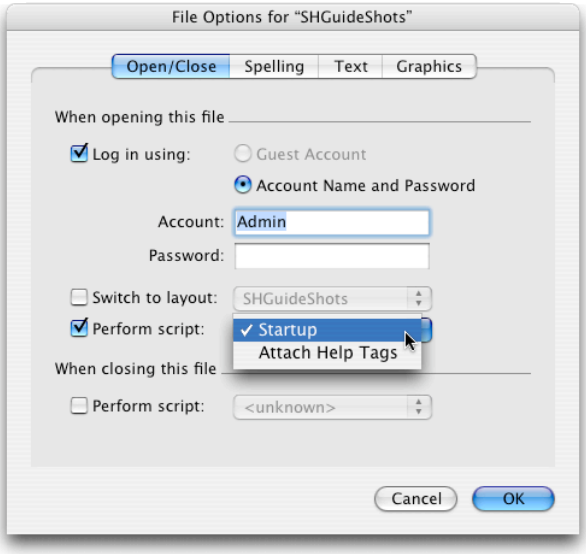

Figure 2.2: Choosing a startup script for a database

To manually remove an attached tag, use the `SHelp_DetachTag` function with exactly the same coordinates you used to attach it:

```
If [SHelp_DetachTags("100,100,200,130";) <> 0]
  # Detaching the tag failed because of an error (or maybe invalid parameters)
End If
```

To manually remove all help tags attached to the current layout, specify the word `all` instead of the hot rectangle coordinates:

```
If [SHelp_DetachTags("all";) <> 0]
  # Detaching the tag failed because of an error (or maybe invalid parameters)
End If
```

Both the `SHelp_AttachTags` and `SHelp_DetachTags` functions return 0 to indicate success, or a non-zero number to indicate an error. You may want to check for the result and perform some additional actions in the case of failure.

You can use 24U SimpleHelp Utility (explained later in a separate chapter) to easily define help tags for layout objects without having to manually find and enter their coordinates. If you want to define hot rectangles by hand anyway, you can use the Object Size palette in FileMaker's layout mode.

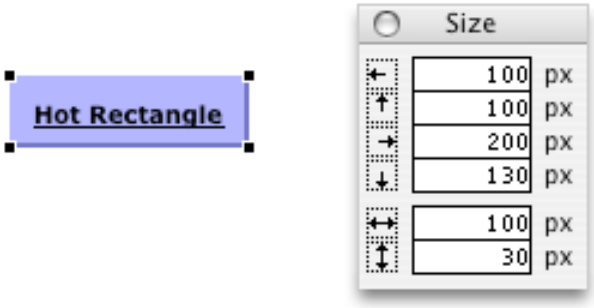

Figure 2.3: Obtaining hot rectangle's coordinates

Important: 24U SimpleHelp Plug-In uses "screen coordinates", which are always the same, while FileMaker displays "printing coordinates" in the Object Size palette. The printing coordinates are dependent on the current printer selection and page setup, so they usually do not match the screen coordinates. To sync these two coordinate systems, and so simplify defining SimpleHelp's hot rectangles, set the layout's printing options to use fixed margins, and set the left and top margins to 0. This will make the printing coordinates independent on the printer selection, so they will match the screen coordinates.

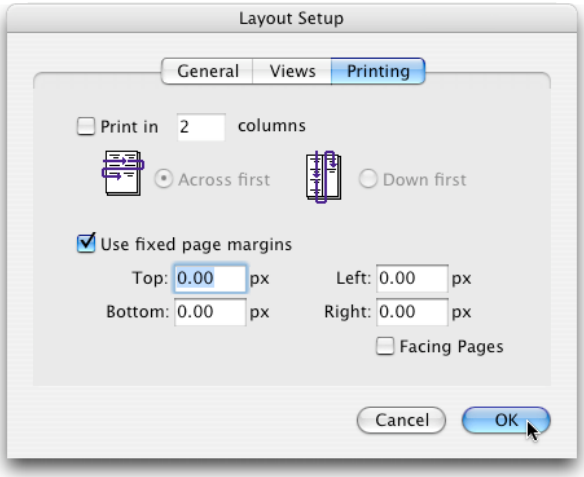

Figure 2.4: Syncing printing and screen coordinate systems

Note: As the function names advise, you can attach or detach multiple help tags at once. To do that, simply add their definitions as additional parameters.

Tip: Since every hot rectangle is attached only to the current layout, you may prefer to define hot rectangles for all your layouts at once from within your startup script, and then let the 24U SimpleHelp Plug-In do all the remaining work for you.

2.1.2 Dynamic Tags

Although it is pretty helpful to display a static help text description in a help tag, sometimes it would be even more useful if you had a chance do decide what to display in the help tag right at the time it is to be displayed. To display such a dynamic tag, attach a script name instead of your help text to your hot rectangle, preceded with the "$$" prefix:

```
If [SHelp_AttachTags("100,100,200,130"; "$$Display a Dynamic Tag") <> 0]
  # Attaching a tag failed because of an error (or maybe invalid parameters)
End If
```

Moving the mouse pointer over a hot rectangle defined by the above script will not display a help tag immediately. Instead, it will try to find a script named "Display a Dynamic Tag", and if it is found, SimpleHelp will trigger it. The script can then evaluate current conditions, and display a help tag with a dynamically calculated text.

For example, the following script will display current time in the help tag, provided it is triggered by SimpleHelp as a reaction to touching a hot rectangle with the mouse pointer:

```
If [SHelp_ShowTag("this"; GetAsText(Get(CurrentTime))) <> 0]
  # Displaying a tag failed because of an error (or maybe invalid parameters)
End If
```

2.1.3 Alternate Tags

Once you have defined a help tag to calculate its contents dynamically, it's quite easy to have it display different messages based on the current modifier keys combination. This way you can display just short button/field descriptions by default, and still deliver more detailed instructions on demand.

You can use the `Get(ActiveModifierKeys)` function to check what modifiers are currently down:

```
If [SHelp_ShowTag("this";
  Case(Get(ActiveModifierKeys); "Alternate Help"; "Default Help")) <> 0]
End If
```

The problem with this tag, however, is that it does not update itself when you change the modifier keys combination. To see the alternative help, you have to leave the hot rectangle, press a modifier key, and move the mouse pointer back to the hot rectangle while holding the modifier key.

To have a dynamic help tag updated each time the modifier keys are pressed or depressed, without having to leave the hot rectangle, you can set up a one-time trigger with the `SHelp_Set` function:

```
If [SHelp_ShowTag("this";
      Case(Get(ActiveModifierKeys); "Alternate Help"; "Default Help"))]
End If
# The following script step will cause the current script to be re-triggered
# when modifier keys are (de)pressed while still being in the hot rectangle
If [SHelp_Set("on-modifiers"; Get(ScriptName))]
End If
```

The above script not only displays a help tag, but also schedules itself to be triggered again when any of the modifier keys is pressed or depressed.

2.1.4 Persistent Tags

Besides displaying help tags under current mouse position, you may also want to create guiding scripts which display their own help tags at specific positions. 24U SimpleHelp Plug-In lets you do this using the `SHelp_Set` function, specifying the tag's position in the first parameter:

```
If [SHelp_ShowTag("100,100"; "This is persistent help tag")) <> 0]
  # Displaying a tag failed because of an error (or maybe invalid parameters)
End If
```

Important: All help tags displayed this way are persistent. They don't disappear from the screen until you ask them to disappear. To remove all persistent tags, use the `SHelp_ShowTag` function with empty parameters:

```
If [SHelp_ShowTag("";) <> 0]
    # Help tags could not be removed because of an error
End If
```

Tip: You can take advantage of the FileMaker 8's ability to pass parameters to scripts, and use just a single script to display many different persistent tags in response to different button being clicked.

2.1.5 Customizing Help Tags

If you still want more from help tags, there are couple of things you can easily customize with the `SHelp_Set` function. This function takes a customizable option's name as the first parameter, and its requested new value as the second parameter:

```
SHelp_Set(option name; value)
```

You can also check the current value of any customizable option by using the `SHelp_Get` function:

```
SHelp_Get(option name)
```

With the version 3.2 of 24U SimpleHelp Plug-In you can easily customize all the following attributes of help tags:

- Background color
- Frame color (only on Windows)
- Text color
- Popup delay (the time to wait after entering a hot rectangle before a help tag finally appears)

To modify any of these options, the `SHelp_Set` can be used. For example, the following function sets the background of help tags to light blue:

```
SHelp_Set("tag-background-color"; "#99CCFF")
```

On Windows, you can also use the following special function to make help tags reflect the system's settings for tool tips:

```
SHelp_Set("system-like";)
```

In addition to controlling the above options, you can also turn displaying of help tags off temporarily, without having to detach them, by using special values for the popup delay option:

```
SHelp_Set("tag-popup-delay"; "off")

SHelp_Set("tag-popup-delay"; "on")
```

For exact names and possible values of the customizable options, please refer to the Function Reference, section 4.5 at page 42, or see the Customizing Tags section in the **24uSimpleHelpTutorial.fp7** file.

2.1.6 Named Layouts

By default, 24U SimpleHelp Plug-Ins attaches and displays help tags for the current layout of the frontmost FileMaker database window. This is quite convenient when you develop a simple solution with 1 to 20 layouts, but starts to be annoying when you have to go through tens or hundreds of layouts to define your help tags.

With the introduction of the version 3.1 of 24U SimpleHelp Plug-In, you can use the following function to make SimpleHelp think the current layout is what you want it to think, not what it really is:

```
SHelp_Set("current-layout"; "layout name")
```

After evaluating this function, SimpleHelp starts ignoring the current layout and sticks with the specified "layout name" layout until you tell it otherwise. To re-activate SimpleHelp's default behavior of being sensitive to current layout, call the same function with an empty layout name:

```
SHelp_Set("current-layout"; "")
```

Important: Although you can cheat on SimpleHelp's recognition of the current layout, it is still sensitive to the current (frontmost) database file. So only the current file will get cheated, and current layout will still be correctly recognized in other files. To cheat on current layout after bringing another file's window to front, you have to use the above described function again.

Tip: SimpleHelp does not care if the layout name supplied to it via the `SHelp_Set("current-layout"; "layout name")` function really exists in the current file. You can use any string as a layout name, creating a virtual layout this way. Then you can easily switch between different help configurations by simply switching your virtual layouts.

2.1.7 Global Tags

When your solution shares the same user interface elements, such as navigation buttons, across multiple layouts, you know how annoying it is to do the same work again and again for each layout. To avoid defining the same set of help tags for each individual layout, 24U SimpleHelp Plug-In 3.2 lets you set a named layout to behave as being always active:

```
SHelp_Set("global-layout"; "layout name")
```

After specifying a global layout with the above function, SimpleHelp will consider all help tags defined for the "layout name" layout to be defined for all layouts. This way you can only define the global help tags once, and add just the layout-specific tags to each individual layout.

To deactivate the global tags, simply set the global layout to none using the following function:

```
SHelp_Set("global-layout"; "")
```

Important: Just like named layous, also the global layout setting is specific to the current file and does not affect SimpleHelp's behavior for other files.

Tip: Since you can define help tags for non-existing virtual layouts, it is a good idea to use a virtual layout for your global tags. You can even have multiple help tag sets attached for multiple virtual layouts and easily switch between them based on current user context.

2.2 Using Coachmarks

With 24U SimpleHelp Plug-In you can easily develop a guiding script which takes the user and drives him or her through a specific procedure step-by-step, while displaying instructions for each step in a persistent help tag. In addition to using persistent help tags, you can navigate the user even better by using coachmarks.

A coachmark is an onscreen graphic that circles an item on the screen, or an object on a database layout. You can use coachmarks to guide the user's attention to layout objects described in a help instruction, typically if the user needs to perform an action (for example, to click on a button or type text into a field).

2.2.1 Displaying Coachmarks

Let's say you have a Print button on your layout, and you want to show the user where the Print button is.

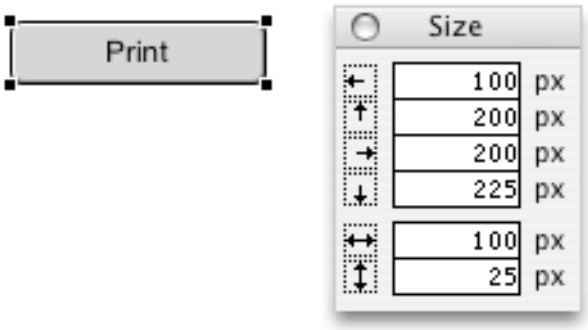

Figure 2.5: A Print button

To draw a coachmark around a layout object, use the `SHelp_ShowCoach` function, passing it coordinates of a rectangular area you want to have circled:

```
If [SHelp_ShowCoach("100,200,200,225") <> 0]
    # Coachmark could not be drawn because of an error
End If
```

Running the above script will cause the Print button to get circled with a colored oval, called coachmark. After drawing a coachmark you will probably want the user to click on the circled object. When you

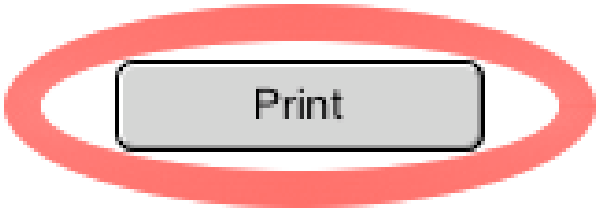

Figure 2.6: A button circled with a coachmark

discover that the user has successfully done that (for example clicked on a button and started your script), you can hide the coachmark, as it is not needed any more.

To hide all coachmarks, simply use the same `SHelp_ShowCoach` function with an empty string as a parameter.

```
If [SHelp_ShowCoach("") <> 0]
    # Coachmark could not be hidden because of an error
End If
```

Tip: Use a coachmark to guide the user's attention to a button, and a help tag to inform him what action is attached to the button. To see an example of such combination, open the **Contacts.fp7** file in the Examples folder, and run the script "Help: Switching between phone, address and e-mail".

2.2.2 Customizing Coachmarks

Just like help tags, also coachmarks can be customized with the `SHelp_Set` function. The version 3.2 of 24U SimpleHelp Plug-In allows you to customize the color used to draw coachmarks:

```
SHelp_Set("coach-color"; color code)
```

Please refer to the Function Reference in this manual for more information on valid color codes accepted by the `SHelp_Set` function. See the section Customizing Coachmarks of the **24uSimpleHelpTutorial.fp7** file to immediately test differently colored coachmarks.

2.3 Using Roll-over Effects

To implement a roll-over effect, you have to define a hot rectangle just like you would do it for a dynamic help tag. The only difference will be the script you attach to the hot rectangle:

```
If [SHelp_AttachTags("100,100,200,130"; "$$Activate Roll-Over") <> 0]
  # Attaching a tag failed because of an error (or maybe invalid parameters)
End If
```

Now you need to write the "Activate Roll-Over" script to perform the action(s) you want to have performed as a response to entering the hot rectangle. For example, you can have a graphical button represented by a container field, and you may want to change the button's picture:

```
Set Field [GUI::ButtonPicture; GUI::ActiveButton]
```

or, in the case of text buttons, you may want to store the button's number:

```
Set Field [GUI::ButtonNumber; 1]
```

and create an unstored calculation field for each button, with formatting based on that value:

```
Button1 =

    Let([ buttonname = "Print" ];
    Case(ButtonNumber = 1;
    TextStyleAdd(TextColor(linkname; RGB(255; 0; 0)); Underline);
    linkname))
```

Now you have a hot rectangle defined to activate your roll-over effect. But you also need the roll-over deactivated when the mouse pointer leaves the hot rectangle. To make the effect complete, you have to set up another script to be triggered when you leave the hot rectangle. Do this as a last step of your "Activate Roll-Over" script:

```
If [SHelp_Set("on-hide"; "Deactivate Roll-Over")]
End If
```

Finally, write your "Deactivate Roll-Over" script similarly to your activation script. You may even save some time by duplicating the activation script and modifying it.

For graphical button, the deactivation script may look like this:

```
Set Field [GUI::ButtonPicture; GUI::InactiveButton]
```

And this can be your deactivation script for text buttons:

```
Set Field [GUI::ButtonNumber; 0]
```

Tip: Set the `tag-popup-delay` configurable option to 0.02 to have your roll-over effects work instantly. See the Function Reference for the `SHelp_Set` function later in this manual, or examine the script Make Help Tags Appear Instantly in the **24uSimpleHelpTutorial.fp7** file for an example.

Chapter 3

Using 24U SimpleHelp Utility

The **24uSimpleHelpUtility.fp7** file that comes with 24U SimpleHelp Plug-In will help you to add visual aids, such as help tags, roll-over effects and guided tours, to your solutions quickly and easily. Instead of examining coordinates of every single object and defining your help tags and other effects by hand, you can simply run a few scripts, and have all the necessary code generated for you automatically.

To make your work really easy, 24U SimpleHelp Utility will appreciate if you have also 24U SimpleDialog Plug-In 2.7 installed. With this additional plug-in, the utility's user interface will be much more efficient and comfortable.

Before you start using this utility's scrips in your own solutions, you can try them out on the Sample layout to learn how they work and what you can do with them. You can create and edit help tags for this layout, then have their definitions generated, and then see the results on the Results layout.

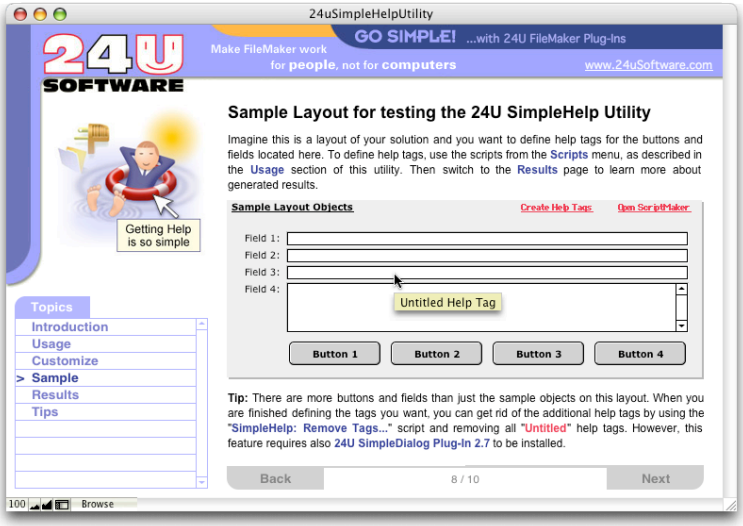

Figure 3.1: The Sample Layout

3.1 Preparing your solution

Before you can use 24U SimpleHelp Utility to add help tags and other visual aids to your FileMaker solution, you need to prepare your solution by following these steps:

1. Make sure you have 24U SimpleHelp Plug-In installed.
2. To make your work with the utility even easier, install also 24U SimpleDialog Plug-In 2.7.
3. Open your solution file.
4. Open the Define Database window, and create a temporary global text field. You will use this field to store results generated by 24U SimpleHelp Utility.
5. Open ScriptMaker, then click on "Import...", and open the 24uSimpleHelpUtility.fp7 file.
6. In the Import Scritps dialog, select all scripts with the "SimpleHelp:" prefix, and click on the OK button to import them.

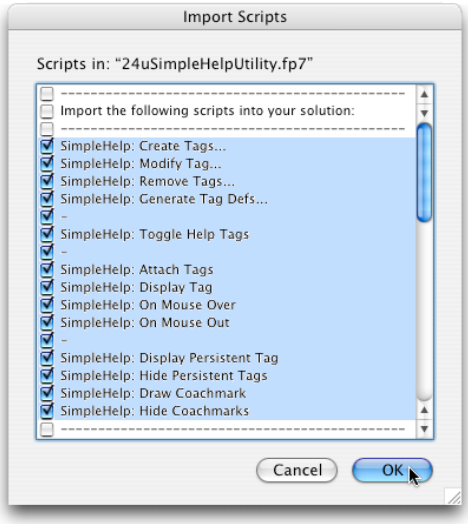

Figure 3.2: Importing utility scripts

7. Select the imported script "SimpleHelp: Generate Tag Defs..." and click "Edit...".
8. Locate the Set Field script step preceded by "MODIFY THE FOLLOWING..." comments, and modify it to store its result to the temporary field you created in the step 4. Now you should be ready to start using the imported utility scripts in your solution.

Once you have imported the utility scripts and selected the target field for your help tag definitions, you are ready to start adding help to your layouts.

3.2 Defining Static Help Tags

24U SimpleHelp Plug-In has an ability to search the current layout for objects, and batch-generate help tags for them. 24U SimpleHelp Utility takes advantage of this feature and lets you generate help tags

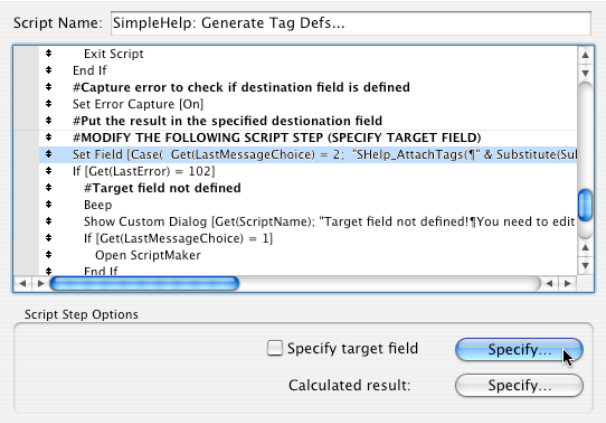

Figure 3.3: Selecting target field

for all objects of a specified kind by running a single script. Another script then lets you edit the text of each help tag, and yet another script then generates the definitions for you.

After preparing your solution as described in section 3.1 at page 28, you can easily define static help tags by applying these steps on each layout:

1. Switch to the layout you want to define help tags for.
2. Run the "SimpleHelp: Create Tags..." script to create help tags with default text for your layout's objects. If you have 24U SimpleDialog Plug-In installed, you will get prompted for initial text message and kind of objects to attach the help tags to. Without 24U SimpleDialog Plug-In, you will only be able to generate help tags for fields and buttons.

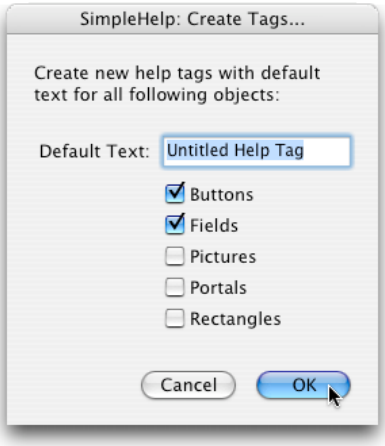

Figure 3.4: Options for generating help tags

Figure 3.5: Help tag editing dialog

3. Now move the mouse pointer over an object so that its default help tag appears, then run the "SimpleHelp: Modify Tag..." script. With SimpleDialog, you will be able to edit the help tag's text immediately in a pop-up dialog. Without SimpleDialog, you will be offered to change the tag's text to a time stamp, so that you can later recognize the tag in the generated definitions, and edit it. Repeat this step for every help tag.
4. If you have 24U SimpleDialog Plug-In installed, you can now help yourself by running the "SimpleHelp: Remove Tags..." script to remove any remaining unwanted help tags.

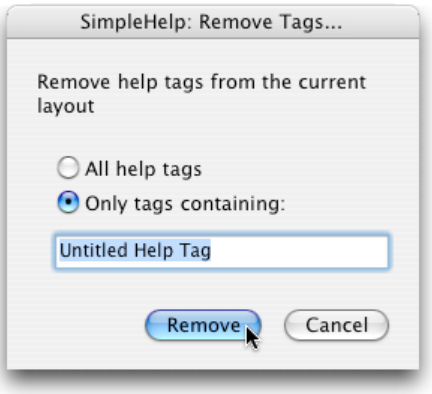

Figure 3.6: Removing unwanted help tags

5. When you are done editing your help tags, run the "SimpleHelp: Generate Tag Defs..." script to generate your tag definitions and put them into your results field.

6. Copy the generated definitions from your results field to clipboard, and open ScriptMaker.

7. Duplicate the "SimpleHelp: Attach Tags" script (you need one for each layout).

8. Modify the duplicated script as advised by the included comments. There are only two modifications needed. First, you need to make the script go to the correct layout. Then you paste your definitions from the clipboard into the If script step containing the `SHelp_AttachTags` function. Alternatively, you can store the tag definitions in a field, and use the field as a parameter for the `SHelp_AttachTags` function.

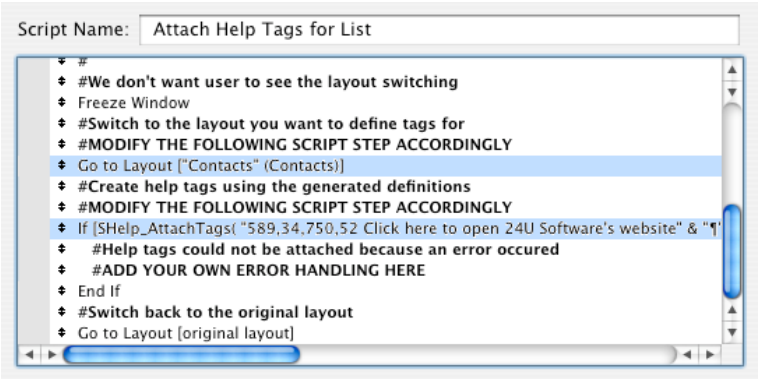

Figure 3.7: Editing the attaching script

9. Make sure your "Attach Tags" script(s) get called from your solution's startup script.

Tip: The "SimpleHelp: Modify Tag..." script always edits the last displayed help tag. Therefore running the script by choosing if from the menu is not always good as you can accidentally activate another help tag while going to the menu. Much better is running the script with a keybord shortcut.

3.3 Defining Dynamic Help Tags

To make a help tag calculate its text message dynamically, you start by defining a script to get called instead of displaying a help tag. Within the script, you can then decide on-the-fly what message you want to display.

24U SimpleHelp Utility has a template script made especially for this purpose. This script not only displays a help tag, but also schedules itself to be run again when the user presses or depresses a modifier keys. This way you can easily display different help messages based on the current modifier keys combination.

Follow these steps to define a dynamic help tag:

1. Create a copy of the "SimpleHelp: Display Tag" script and give it a unique name.
2. Modify the duplicated script to display the text of your desire.
3. When modifying your help tag with the "SimpleHelp: Modify Tag..." script, click on the "run a script..." radio button, and enter the name of the script you created in the step 1.

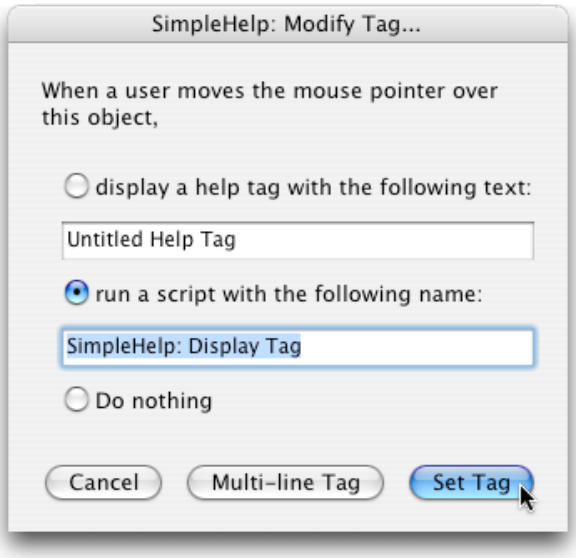

Figure 3.8: Selecting a dynamic tag handling script

Important: The "SimpleHelp: Modify Tag..." script requires 24U SimpleDialog Plug-In 2.7 to be installed to be able to offer you the "run a script..." option. If you don't have 24U SimpleDialog Plug-In 2.7 installed, you will have to edit the generated help tag definitions by hand. See section 2.1.2 at page 20 for more information.

3.4 Defining Roll-Over Effects

To create a roll-over effect, you define a script to get called instead of displaying a help tag, and have this script activate the effect. Then you define another script to get called when the mouse pointer leaves the hot rectangle, and de-activate the effect.

Follow these steps to define your roll-over effect:

1. Start by preparing the roll-over effect itself. This can be either a global container fields changing its content, or a text field changing its formatting, or a sound (such as spoken help) being played via 24U SimpleSound Plug-In. The actual technique you will use is limited only by your imagination. FileMaker 8 provides a very good set of tools for roll-over effects in its text formatting functions.
2. Create a copy of the "SimpleHelp: On Mouse Over" and "SimpleHelp: On Mouse Out" scripts and give them unique names. To make your life easier, name them equally, except for the "Mouse Over" and "Mouse Out" endings. For example, you can name the scripts "Effect A Mouse Over" and "Effect A Mouse Out".
3. Modify your new "Mouse Over" script to do whatever will activate your roll-over effect. Typically, you will set some global field to something. If you did not follow the naming convention advised above, modify also the `SHelp_Set` function at the end of the script, so that it has the "Mouse Out" script's name in the second parameter.
4. Modify your new "Mouse Out" script to do whatever will de-activate your roll-over effect. Typically, you will set the same global field to something else.
5. When modifying your help tag with the "SimpleHelp: Modify Tag..." script, click on the "run a script..." radio button, and enter the name of your new "Mouse Over" script.

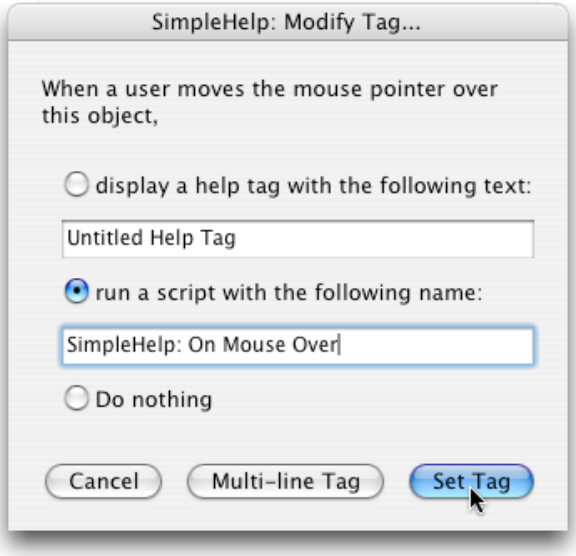

Figure 3.9: Selecting a roll-over activator

Important: The "SimpleHelp: Modify Tag..." script requires 24U SimpleDialog Plug-In 2.7 to be installed to be able to offer you the "run a script..." option. If you don't have 24U SimpleDialog Plug-In 2.7 installed, you will have to edit the generated help tag definitions by hand. See section 2.1.2 at page 20 for more information.

3.5 Creating Guided Tours

In addition to providing help to the user right under the mouse pointer, 24U SimpleHelp Plug-In lets you also create interactive guided tours, taking the user step-by-step through complex workflow procedures.

The 24U SimpleHelp Utility provides you with four template scripts which you can use as building blocks for such guided tours:

The “SimpleHelp: Display Persistent Tag” script will display a help tag with your supplied text at any specified position within the current layout. Use clones of this script to advise individual steps of your procedure, one at a time. Any tags displayed with this script or its clones will stay on the screen until you explicitly request to hide them.

Use the “SimpleHelp: Hide Persistent Tags” script to hide any persistent help tags. The “SimpleHelp: Draw Coachmark” script will draw a colored oval around any rectangle within the current layout you specify. Use clones of this script to complement the textual instructions displayed in help tags with graphical marks, driving the user’s attention to specific layout objects, such as buttons.

Use the “SimpleHelp: Hide Coachmarks” script to hide any drawn coachmarks.

By combining these four scripts and their clones in a wise sequence, you can effectively guide the user through any complex procedure.

Note: In the current implementation, hiding persistent help tags also hides all coachmarks, and vice versa. However, do not relay on this implementation as it may change in future.

Chapter 4

Function Reference

4.1 SHelp_AttachTags

Format

`SHelp_AttachTags( tag definition {; tag definition... } )`

Parameter(s)

`tag definition` — a help tag definition consisting of hot rectangle coordinates, and a help message; can be composed as a single textual parameter delimited by the pipe character "|", or as two adjacent textual parameters. One of keywords listed below in the Description can be used in place of hot rectangle coordinates to have layout objects automatically searched for. To attach a script trigger, precede the script's name with two dollar characters "$$".

Parameters in curly braces { } are optional.

Note: Multiple help tags can be defined within a single function. Additional definitions can be added as additional parameters, or all definitions can be composed into a single textual parameter, delimited by line breaks.

Result

number — 0 to indicate success, or an error code to indicate failure

Description

Defines one or more hot rectangles for the current layout, and attaches a help tag or a script trigger to each of them.

If any specified coordinates match an already defined hot rectangle, the older help tag definition is replaced with the specified one.

If one of the following keywords is used in place of hot rectangle coordinates, 24U SimpleHelp Plug-In will locate all objects of the specified type on the current layout, define a new hot rectangle for each of them, and attach the specified help message it. Warning! These keywords work only in Windows version of 24U SimpleHelp Plug-In.

`"all"` — every recognized layout object will get the specified help tag attached

`"last"` — the last displayed tag's definition will be replaced with the specified one

`"buttons"` — every button located on the current layout will get the specified help tag attached

`"ellipses"` — every ellipse located on the current layout will get the specified help tag attached

`"fields"` — every field located on the current layout will get the specified help tag attached

`"lines"` — every line located on the current layout will get the specified help tag attached

`"ovals"` — this is an alias for the `"ellipses"` keyword

`"pictures"` — every picture located on the current layout will get the specified help tag attached

`"portals"` — every portal located on the current layout will get the specified help tag attached

`"rectangles"` — every rectangle located on the current layout will get the specified help tag attached

`"rounded-rectangles"` — every rounded rectangle located on the current layout will get the specified help tag attached

`"text"` — every text object located on the current layout will get the specified help tag attached

Examples

Assume you have an "About" button in the upper left corner of your layout, and you want to attach a help tag to this button.

`SHelp_AttachTags("10,10,90,30"; "Developer credits")` attaches the specified message to our About button.

`SHelp_AttachTags("10,10,90,30|Developer credits";)` is a compact equivalent of the previous function, more compatible with older versions of 24U SimpleHelp Plug-In.

`SHelp_AttachTags("10,10,90,30"; "$$About")` attaches a script named About to our About button. Instead of displaying a help tag, the "About" script will be triggered when the user moves the mouse pointer over the About button.

`SHelp_AttachTags("10,10,90,30"; "Developer credits"; "100,10,180,30"; "Make new record")` defines two hot rectangles and attaches two help tags to them at once.

`SHelp_AttachTags("buttons"; "This is a button")` attaches a help tag with the text "This is a button" to every button on the current layout.

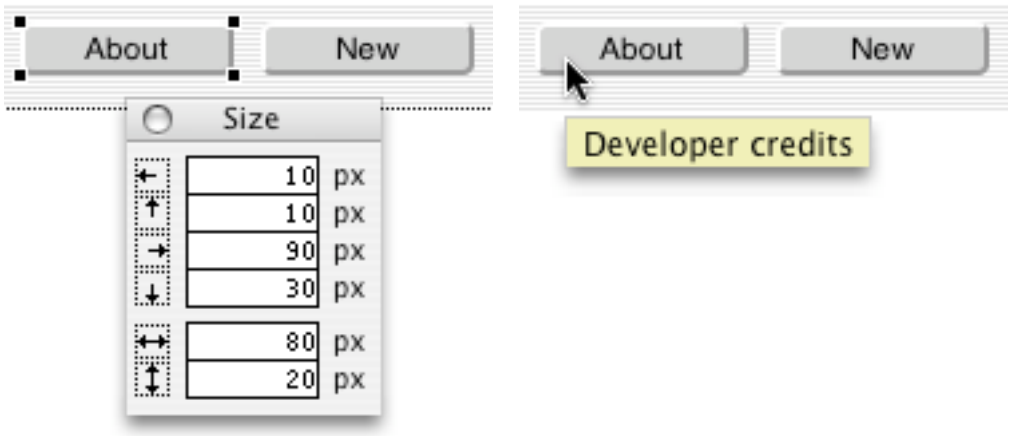

Figure 4.1: Sample help tag and its hot rectangle's coordinates

Special considerations

Although this function can have one or more parameters, FileMaker 8 requires the first semicolon even when using just one parameter for some reason. So instead of `SHelp_AttachTags("10,10,90,30|About")` you have to write the function as `SHelp_AttachTags("10,10,90,30|About";)`.

For compatibility with previous versions of 24U SimpleHelp Plug-In, help tag definitions can be combined in a single textual parameter, delimiting parameters by the pipe "|" and definitions by line breaks "¶". When using this compact parameter format, to use a line break in the help message, you must enclose the whole message into double quotes:

```
SHelp_AttachTags("10,10,90,30|\"First line¶Second line\"¶100,10,180,30|Second tag";)
```

The current version of 24U SimpleHelp Plug-In does not always recognize scripts whose names contain non-ascii characters. It is strongly recommended to use only ASCII characters for names of scripts which are supposed to be triggered by 24U SimpleHelp Plug-In.

4.2 SHelp_DetachTags

Format

`SHelp_DetachTags( tag specification {; tag specification... } )`

Parameter(s)

`tag specification` — a string containing either coordinates of an existing help tag's hot rectangle, or the word `"last"` representing the last displayed help tag, or the word `"all"` representing all hot rectangles on the current layout. Additional help tags can be specified either as additional parameters, or in a single parameter delimited by the pipe character "|".

Parameters in curly braces { } are optional.

Result

number — 0 to indicate success, or an error code to indicate failure

Description

Removes (detaches) specified help tag(s) (and roll-over effects) from the current layout.

Examples

Assume you have defined help tags for two layout buttons using the following calculation:
`SHelp_AttachTags("10,10,90,30"; "Developer credits"; "100,10,180,30"; "Create new record")`.

`SHelp_DetachTags("10,10,90,30"; "100,10,180,30")` undefines both help tags, so the 24U SimpleHelp Plug-In then behaves as if they were never defined.

`SHelp_DetachTags("10,10,90,30|100,10,180,30";)` is a compact equivalent of the above example, more compatible with older versions of 24U SimpleHelp Plug-In.

`SHelp_DetachTags("100,10,180,30";)` undefines the "Create new record" tag, while leaving the "Developer credits" tag intact.

`SHelp_DetachTags("all";)` undefines all help tags defined on the current layout

`SHelp_DetachTags("last";)` undefines the last help tag that was displayed, or the currently displayed help tag if one is visible

Special considerations

Although this function can have one or more parameters, FileMaker 8 requires the first semicolon even when using just one parameter for some reason. So instead of `SHelp_DetachTags("100,10,180,30")` you have to write the function as `SHelp_DetachTags("100,10,180,30";)`.

Previous versions of 24U SimpleHelp Plug-In allowed the same string to be used with both the attaching and detaching functions. If you use the old compact one-parameter approach, you can utilize this feature with this version as well. So the following calculation will safely ignore the help messages and detach the specified tags:

```
SHelp_DetachTags("10,10,90,30|\"First line¶Second line\"¶
100,10,180,30|Second tag";)
```

4.3 SHelp_Get

Format

`SHelp_Get( selector )`

Parameter(s)

`selector` — the name of a customizable option or status information (one of the values listed below in Description)

Result

number *or* text — the current value of the customizable option specified by selector

Description

Returns the current value of the specified customizable option or status information. One of the following text strings can be used as a customizable option selector. Use:

`"tag-background-color"` to get the default background color for help tags

`"tag-frame-color"` to get the default color for help tag's borders (Windows only)

`"tag-text-color"` to get the default color for text displayed in help tags

`"tag-popup-delay"` to get the default time that needs to elapse before a help tag appears after moving the mouse pointer into a hot rectangle; the value of 0 indicates that displaying help tags is turned off temporarily

`"coach-color"` to get the default color for coachmarks

`"current-layout"` to get the name of the current layout as recognized by SimpleHelp

`"global-layout"` to get the name of the layout specified as global

Any color value is represented by a string formatted as `"#RRGGBB"`, where `RR` is a hexadecimal value of the red component, `GG` is a hexadecimal value of the green component, and `BB` is a hexadecimal value of the blue component in the standard RGB color space. The same format is used by the HTML standard.

Each of these options can be modified with the `SHelp_Set` function.

In addition to the selectors listed above, you can also use the following special selectors to get status information related to SimpleHelp behavior:

`"list-tags|all"` to get definitions of all help tags defined for the current layout

`"list-tags|last"` to get the definition of the last or currently displayed help tag (or last or currently activated hot rectangle)

`"mouse-position"` to get the current mouse position in screen coordinates

`"mouse-click-position"` to get the mouse position at the time of the last click

Examples

Assume that none of the customizable options has been modified since launching the FileMaker application, so they all are set to their default values.

`SHelp_Get("tag-background-color")` returns `"#EEEEAA"` representing the default yellow background.

`SHelp_Get("tag-frame-color")` returns `"#000000"` for the default black frame used on Windows.

`SHelp_Get("tag-text-color")` returns `"#000000"` representing the black color used for text in help tags by default.

`SHelp_Get("tag-popup-delay")` returns `1.5`, the default one-and-half second delay.

`SHelp_Get("coach-color")` returns `"#FF0000"`, the default red color for coachmarks.

4.4 SHelp_Register

Format

`SHelp_Register( registration code )`

Parameter(s)

`registration code` — the registration code obtained from 24U Software for your license

Result

number *or* text — 0 to indicate success, an error code or an error message to indicate failure

Description

Attempts to unlock (register) your copy of 24U SimpleHelp Plug-In with the specified code. If the code is valid, 24U SimpleHelp Plug-In gets unlocked and stays in this state until you quit the FileMaker application, deactivate the plug-in, or the registration code expires. While unlocked, 24U SimpleHelp Plug-In is fully functional and does not bother the user with a shareware reminder.

Important: If an invalid code is specified, a registration dialog appears, allowing the user to enter a valid end-user registration code. If the user enters a valid code, or if the shareware trial period is not over yet, the function may indicate success even though the code supplied in its parameter was not valid.

Since the unlock is effective only for a single session, you will probably want to use it in a script which you set to run automatically right after opening your database solution.

This function is the only way how to apply developer license registration codes. However, you can find it useful also for multi-user registration codes to simplify installation of your solution on a network with FileMaker Server.

Examples

`SHelp_Register("SHL300-ABCDEF-GHIJKL")` will fail because the code is invalid. Please purchase a license to obtain a valid registration code.

Special considerations

If you install 24U SimpleHelp Plug-In and it expires before you get a chance to try it out, please request a time-limited evaluation code at `eval@24uSoftware.com`.

4.5 SHelp_Set

Format

```
SHelp_Set( selector; value )
```

Parameter(s)

`selector` — the name of a customizable option (one of the values listed below in Description)

`value` — the value you want to set the customizable option to

Result

number *or* text — same as value if successful, an error code to indicate failure

Description

Sets the specified customizable option's value to the specified value.

One of the following text strings can be used as an option selector. Use:

`"tag-background-color"` to set the default background color for help tags

`"tag-frame-color"` to set the default color for help tag's borders (Windows only)

`"tag-text-color"` to set the default color for text displayed in help tags

`"tag-popup-delay"` to set the time that needs to elapse before a help tag appears after moving the mouse pointer into a hot rectangle; use "off" to turn displaying help tags off temporarily, use "on" to turn them back on and restore the original value

`"coach-color"` to set the default color for coachmarks

`"current-layout"` to consider the specified name being the current layout

`"global-layout"` to use the specified layout's help tags for all layouts in the current file

To specify a color value, represent it by a string formatted as `"#RRGGBB"`, where `RR` is a hexadecimal value of the red component, `GG` is a hexadecimal value of the green component, and `BB` is a hexadecimal value of the blue component in the standard RGB color space. The same format is used by the HTML standard.

You can use the `SHelp_Get` function to check the current value of any of the above options.

In addition to the selectors listed above, you can also use the following special calculation on Windows to have 24U SimpleHelp Plug-In adopt the system's default look of tool tips:

```
SHelp_Set("system-like"; "")
```

Examples

`SHelp_Set("tag-text-color"; "#FFFF00")` sets the color for drawing text in help tags to yellow.

`SHelp_Set("tag-popup-delay"; 0.02)` sets the pop-up delay to the lowest possible value, makes help tags appear instantly.

`SHelp_Set("coach-color"; "#0000FF")` sets the color used for drawing coachmarks to blue.

4.6 SHelp_ShowCoach

Format

```
SHelp_ShowCoach( coachmark coordinates )
```

Parameter(s)

`coachmark coordinates` — screen coordinates of a rectangle to be circled with a coachmark, specified by a string formatted as "L,T,R,B" where L is left, T is top, R is right, and B is bottom coordinate; or an empty string.

Result

number — 0 to indicate success, or an error code to indicate failure

Description

Circles the specified rectangle with a colored oval called coachmark. If an empty string is passed instead of coordinates, all currently visible coachmarks get removed.

Examples

Assume you have an "About" button in the upper left corner of your layout, and you want to circle it to drive the user's attention to it.

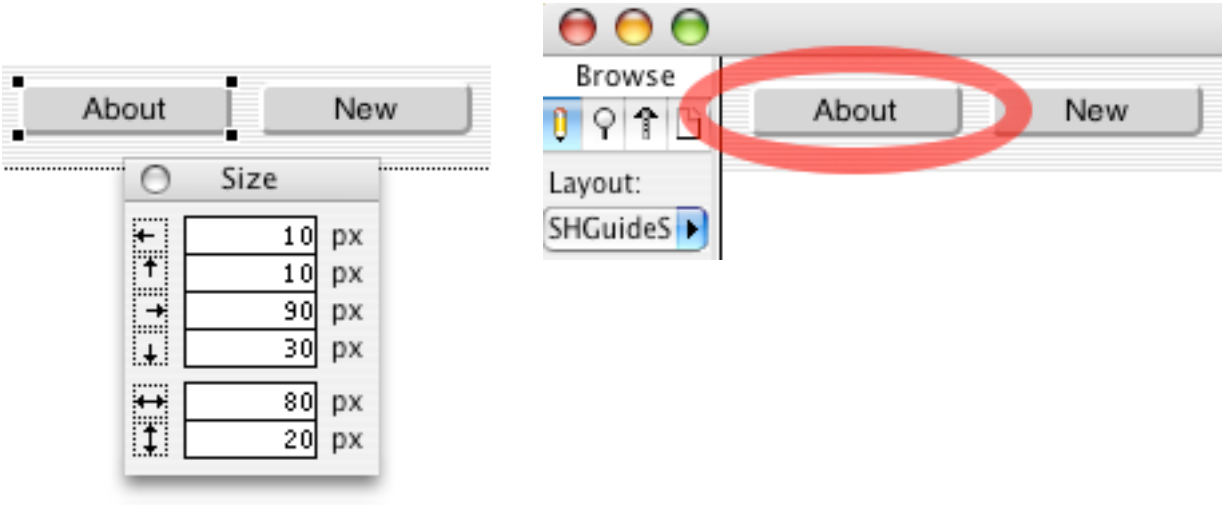

Figure 4.2: Sample buttons and coach mark

`SHelp_ShowCoach("10,10,90,30")` draws a coachmark around your About button.

`SHelp_ShowCoach("")` removes any currently displayed coachmarks.

Special considerations

It is recommended that you don't get more than two coachmarks drawn at the same time, because their handling routines do not get much time from FileMaker to do their work.

4.7 SHelp_ShowTag

Format

`SHelp_ShowTag( tag specification {; tag text} )`

Parameter(s)

`tag specification` — either coordinates of the upper left corner for displaying a persistent help tag, or the word `"this"` to indicate the currently active hot rectangle.

`tag text` — a text to display in the help tag; omit this parameter when hiding tags (see Description below).

Parameters in curly braces { } are optional.

Result

number — 0 to indicate success, or an error code to indicate failure

Description

Displays a help tag on demand. If coordinates are specified as the first parameter in the format `"L,T"` where `L` is desired left and `T` is desired top coordinate of the help tag, a persistent help tag gets displayed with the text specified by the second parameter.

If the word `"this"` is passed as the first parameter and the current script was triggered in a response to entering a hot rectangle, a help tag appears on the same place where it would normally appear if a help message was attached to the hot rectangle instead of the script trigger.

If the tag specification is empty, all currently displayed help tags disappear.

Examples

Assume you have an "About" button in the upper left corner of your layout, and you have attached a script named "About" to the button using the function `SHelp_AttachTags("10,10,90,30"; "$$About")`.

`SHelp_ShowTag("10,40"; "Click the About button above to see developer's credits.")` displays a persistent help tag right below the About button.

`SHelp_ShowTag("";)` removes the persistent help tag displayed with the previous function.

`SHelp_ShowTag("this"; "Not available yet")` when used in the "About" script, displays a help tag with the text "Not available yet" each time the mouse pointer moves over the About button.

4.8 SHelp_Version

Format

`SHelp_Version( version format )`

Parameter(s)

`version format` — one of the strings listed below in Description, or an empty string

Result

text — version string of the active 24U SimpleHelp Plug-In, formatted as requested by the parameter

Description

Returns the version information about the currently installed and active copy of the 24U SimpleHelp Plug-In. One of the following selectors can be used to obtain different results. If an unknown selector is used, the short version info is returned by default. Use:

`"short"` to get just the version number
`"long"` to get the plug-in name followed by its version number
`"platform"` to get platform of the code currently running

Examples

Assume you have the version 3.2 of 24U SimpleHelp Plug-In installed and active.

`SHelp_Version("")` returns “3.2”

`SHelp_Version("short")` returns “3.2”

`SHelp_Version("long")` returns “24U SimpleHelp Plug-In 3.2”

`SHelp_Version("platform")` returns “Mac OS X” of “Windows” based on what operating system you are using