


Script and DHTML Examples

The examples in this reference demonstrate how you can use scripts and Dynamic HTML (DHTML) to add functionality to your help system.

Script examples:

- Create a pop-up window by using JavaScript to call the TextPopup method. This example is intended to be used within a help topic, and is not the same feature as context-sensitive help,

which opens in an external application. To implement context-sensitive help, you must work with the HTML Help API.

- Annotate an image by associating a series of pop-up windows with the regions of an image map. This example shows how you can annotate images in much the same way that Microsoft Help Workshop Hotspot Editor (Shed.exe) allows you to create and edit hypergraphics.
- Store text for pop-up windows in a text file instead of embedding the text in each HTML file. This example also shows how to link to the text file from an HTML file.
- Use JavaScript to create a secondary browser window that can be used to display text or graphics.
- Create a global script file to store all of your scripts in one easy-to-reference location. This can make it much easier to maintain scripts that affect multiple topics within a help system.
- Use JavaScript to link to a file outside of your help system. This is especially useful for linking to larger files, such as video clips.

DHTML examples:

- Use DHTML to create expandable sections that react to user input. This can help you to conserve screen space in your help topics.
- Create dynamic links that will liven up your help topics. This example shows the easiest way to add mouseover functionality to every hyperlink in your help project, and also shows how to create compact custom links using the `<SPAN>` tag.

About the HTML Help References

Example: Create a pop-up window

Using the HTML Help ActiveX control, you can create a pop-up window that appears when a user clicks a specific word, phrase, or graphic in a topic. This example is based on the TextPopup method.

The steps of the process are as follows:

1. Write the text for the pop-up window.
2. Insert the HTML Help ActiveX control in your HTML file.
3. Create a hyperlink to open the pop-up window.

Notes

- You can use this example with both uncompiled HTML files and compiled help (.chm) files.
- This procedure cannot be used to create context-sensitive help topics that users open through an external program.

About the script and DHTML examples

Example: To store text for pop-up windows in a text file

The text for pop-up windows can be stored in one text file with the .js file name extension. This enables you to store the text for all pop-up windows in your entire help system in a single location. This also enables you to globally define the font attributes to be used for pop-up windows.

1. Create your pop-up windows using any text editor, in the following format:

   ```
   popfont="Facename[, point size[, charset[, PLAIN BOLD ITALIC UNDERLINE]]]"
   Text1="The text for the first pop-up window."
   Text2="The text for the second pop-up window."
   ```

   where *popfont* is the name of the variable that specifies the font attributes for the pop-up text and *Text1*, *Text2*, and so on, are the variables that specify the text of each pop-up window.

2. Save the file with a .js file name extension.
3. Copy the following code into each HTML file from which you want to open a pop-up window. Place the code between the `<HEAD>` start and end tags:

   ```
   <OBJECT
   id=HHCTRL type="application/x-oleobject"
   classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" >
   </OBJECT>

   <SCRIPT language=javascript
   SRC="terms.js">
   </SCRIPT>
   ```

   where *HHCTRL* is the ID of the control that you are referencing, and *terms.js* is the name of your text file. You might want to include an error handler that is invoked if the specified text file cannot be found.

4. Also copy this code in your HTML file to call the TextPopup method of the HTML Help ActiveX control:

   ```
   <A HREF="JavaScript:HHCTRL.TextPopup(Text1,popfont,9,9,-1,-1)"
   Title="Click for pop-up definition">Word to be defined</A>
   ```

Notes

- You must add the text file to the `[FILES]` section of your project (.hhp) file.
- This procedure cannot be used to create context-sensitive help topics that users open through an external program.

About the script and DHTML examples

Example: Add text pop-ups to an image

You can add pop-up help topics to an image by using an image map in conjunction with the HTML Help ActiveX control. This lets you associate a pop-up window with each region of an image. When a user clicks an area of the image, a pop-up window will display the topic that goes with that area. This example is based on the TextPopup method.

The steps of the process are as follows:

1. Create an image map.
2. Write text for pop-up windows.
3. Add the image to your project (.hhp) file.
4. Insert the HTML Help ActiveX control in your HTML file.
5. Add hyperlinks to each region of the image map.

Note

- You can use this example with both uncompiled HTML files and compiled help (.chm) files.

About the script and DHTML examples

Example: Create a new window using JavaScript

You can use JavaScript to launch a new window (an instance of Internet Explorer). The code in this example can be used on both standard Web pages and within compiled help (.chm) files.

Create the link as follows:

```
<A HREF= "#" onClick="window.open('examples/sample.htm',
'Sample','toolbar=no,width=190,height=190,left=500,top=200,
status=no,scrollbars=no,resize=no');return false">
See the sample</A>.
```

where `examples/sample.htm` is the path to the file that will appear in the new window, `Sample` is the *`name`* argument for the new window, and the remaining values are the attributes for the pop-up window (width and height in pixels, toolbar, scroll bar, status bar, and resize). These attributes are separated by commas, and the entire line is enclosed in single quotes.

Notes

- The hashmark ("#") character ensures that the state of the current help window will be retained. `return false` prevents the hashmark from appearing in the location field of the main window, which would interfere with the functionality of the **Back** and **Forward** buttons.
- All of the code must be on the same line or it will not work. The lines in this example are broken for legibility.
- When working with compiled help (.chm) files, all HTML files referred to by this code must be included in your project before compiling. Also, because the compiler can only include graphics files in `<IMG>` tags, it will miss any graphics files that are included in this script. If you use this script in a compiled help file, make it point to an HTML file rather than a graphics file.

About the script and DHTML examples

Example: link to file outside of your help system

There are many reasons why you might want to link to a file located outside of a compiled help (.chm) file. Say you want to link to a video clip. Including such a large file in your help project could result in an enormous compiled help file. There may also be times when you need to link to a readme file that was not ready when the .chm file was built.

When you are designing a normal Web page, linking to files is a simple matter of specifying the location of the target file. The location of the page containing the link is irrelevant. However, things are different in a compiled help file. The main problem with compiled help is that you can only link to external files using an absolute path. This is fine if you are certain the location of the help file will always be the same. But what if a user installs your help file to an unexpected location? If this happens, any links using absolute paths will be broken. Similarly, if your help file is to be referenced from a compact disc, it is impossible to predict which drive letter will be in use.

You can use JScript to bypass these limitations. The following script uses the location object to find out where your .chm is located on the local computer, and parses an absolute path to the files you specify. All you need to do is install the target file(s) into the same directory as the .chm, and specify the target filenames in each respective anchor tag.

Following is a step-by-step explanation of how the script works.

Here, the function is begun and the variables are declared. Variable "fn" will be passed to the script from the link itself:

Here, the function is begun and the variables are declared. Variable "fn" will be passed to the script from the link itself:

```
<SCRIPT Language="JScript">

function parser(fn) {
```

```
var X, Y, sl, a, ra, link;
```

The JScript search method is used to find the index of the first colon in the string returned by the location object:

```
ra = /:/;

a = location.href.search(ra);
```

An IF statement determines which of the two possible moniker protocols is in use. If the value of "a" equals 2, then **mk:@MSITStore:** is the protocol in use and X equals 14 (used later to parse an offset of 14 characters). Otherwise, protocol **ms-its:** is in use and X equals 7.

```
if (a == 2)

X = 14;

else

X = 7;
```

Next, the JScript lastIndexOf method returns the index of the last backslash in the location.href string to use as a value for Y (the second character offset, used later in parsing). Variable "sl" contains the substring to search for (two backslashes are used to indicate a single backslash):

```
sl = "\\";

Y = location.href.lastIndexOf(sl) + 1;
```

This is where the values X and Y come into play. The substring method uses X and Y to parse the beginning and end of location.href, respectively, returning a substring (hence the name of the method). For example, say location.href returns the value *ms-its:C:\dir\file.chm::/topic.htm*. In the substring method, if X equals 7, then the first 7 characters are removed from the string. If Y equals 14, then everything after the 14th character is removed. This would result in *C:\dir*, the root path of file.chm. The string "file:///" is then added to the beginning of the string, and the target filename (variable "fn") is added to the end:

```
link = 'file:///' + location.href.substring(X, Y) + fn;
```

Finally, the location object is changed to the path value stored in the variable "link":

```
location.href = link;

}

</SCRIPT>
```

Here is the complete script without comments:

```
<SCRIPT Language="JScript">

function parser(fn) {

var X, Y, sl, a, ra, link;

ra = /:/;
```

```
a = location.href.search(ra);

if (a == 2)

X = 14;

else

X = 7;

sl = "\\";

Y = location.href.lastIndexOf(sl) + 1;

link = 'file:///' + location.href.substring(X, Y) + fn;

location.href = link;

}

</SCRIPT>
```

This is the syntax for the link:

```
<a onclick="parser('loremipsum.htm')" style="text-decoration: underline;

color: green; cursor: hand">Link to loremipsum.htm</a>
```

The target file name (in parentheses) is passed to the script as a string variable ("fn" in this example). This enables you to access the same script with multiple anchor tags so you can link to more than one file.

Notes

- The file you are linking to must be stored in the same directory as your compiled help file.
- The HREF attribute must be omitted from the <A> tag.
- All style attributes for the anchor tag must be defined manually so that the anchor text appears as a link.
- Text and HTML files will open in the Help Viewer window. If you are linking to a text file, be sure to use hard returns. Otherwise, the text will not wrap properly.
- All other files will open in the program with which they are associated. For example, clicking a link to an .avi file will launch an instance of Microsoft Media Viewer. However, if a user does not have the appropriate program installed, he or she will not be able to open the file.

About the script and DHTML examples

Example: Create a global script file

You can consolidate scripts in a global file that can be referenced from pages in both compiled and uncompiled help systems. This makes it easier to maintain scripts that are used by many different

pages, such as those used for image-swapping. This example shows a global file using JavaScript, but it will work with other Web scripting languages, such as Microsoft Visual Basic Scripting Edition.

There are three steps to implementing a global script file:

1. Create the script file using Notepad (or another text editor). This is the file that contains all of the script code. Save this file with a .js extension.
2. Add a `<SCRIPT>` tag to the `<HEAD>` tag of each HTML file. This links the code in your global file to the page. The tag should look like this:

   ```
   <script language="JavaScript" src="master.js"></script>
   ```

   Where `JavaScript` is the name of the scripting language you are using and `master.js` is the name of the global script file.

3. If you are creating a compiled help (.chm) file, add the file name of your master script file to the [FILES] section of your project (.hhp) file.

You can call scripts that are stored in a global file the same way you would call script code that is actually on the page. For example, if you created a global file with `function Foo()`, you could reference it like this: `<a onclick="Foo()">`.

Notes

- You can create more than one global script file.
- This procedure can be used with any scripting language.

 About the script and DHTML examples

---

Example: Create expandable sections with DHTML

You can use Dynamic HTML (DHTML) to create expandable sections in your HTML files. This can be helpful when you document screen shots and other items that occupy a large amount of screen space. Expandable sections give the user more control over the display of such elements, letting them display them as needed.

These topics use, as samples, the source code for the sidebars in the HTML Help documentation.

There are three steps to creating expandable sections:

1. Create the tag that will contain the contents for each section.
2. Insert the JavaScript code.
3. Update your style sheet to reflect the new section.

Notes

- You can use this example with both uncompiled HTML files and compiled help (.chm) files.

About the script and DHTML examples

Example: Create dynamic elements with DHTML

Dynamic HTML (DHTML) offers a wide range of possibilities to help you improve the function and appearance of your help system. The following examples explain two dynamic links that are used in the HTML Help documentation, and they provide sample code that will help you to implement similar links in your help system:

- Add mouseover functionality to all of the hyperlinks in your help system. All it takes is one small addition to your style sheet.
- Take your links a step further with dynamic inline elements.

For more information, go to the DHTML, HTML, and CSS Web site in Microsoft's Site Builder Network.

About the script and DHTML examples

HTML Help Viewer topics

The HTML Help Viewer topics contain basic information about how to use the viewer. Information on using the accessibility shortcut keys is included. If you are a help author, you can include these topics in your help file.

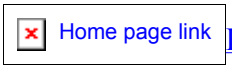
You can download a compiled help (.chm) file that contains the Help Viewer topics from the HTML Help Web site. You can then decompile the help file, revise the topics to conform to your guidelines, and include them in your help system.

What is HTML Help Workshop?

About MSDN scripting references

Microsoft publishes extensive scripting references for Microsoft JScript, Microsoft Visual Basic Scripting Edition, and Dynamic HTML (DHTML).

If you are a member of the Microsoft Developer Network, you can view these references on the Web.



HTML Help references

Send feedback to MSDN. Look here for MSDN Online resources.