JavaScript Window-Element Objects

The window Object

- The *window* object is at the top of the object hierarchy.
- A *window* object exists for each open browser window.
- The properties of this object describe the document in the window and provide information about the window.

The window Object

- Three of the *window* object's properties are child objects:
  - The location object stores the location (URL) that is displayed in the window.
  - The document object holds the Web page itself.
  - The history object contains a list of sites visited before and after the current site.
- In most cases, there is only one window object, so you can omit the *window* object name when referring to the current script's window.
  - Example: `status = "This is a test."`

The window Object

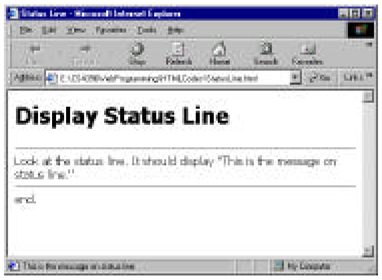
- There are several terms that refer to window objects:
  - window refers to the current window object.
  - self also refers to the current window object.
  - top refers to the topmost (usually, the first) browser window.
  - parent refers to the parent window when frames are used. Frames will be introduced later in this section.
  - opener is used for windows you create and refers to the window that opened the current window.

window Object Properties

- The window object has a variety of properties that specify information about the window and its components.
- The simplest property is the name property; this contains the name of the current window.
- A created window need to be assigned with a name for referencing.

Example -- Status Line

```
<HTML>
<HEAD><TITLE>Status Line </TITLE>
</HEAD>
<BODY>
<H1>Display Status Line</H1>
<HR>
  Look at the status line. It should display "This is the
   message on status line."
<HR>
<SCRIPT LANGUAGE="JavaScript">
window.status="This is the message on status line."
</SCRIPT>
end.
</BODY>
</HTML>
```



Display Status Line

Look at the status line. It should display "This is the message on status line."

end.

This is the message on status line

Status Line - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Stop Refresh Home Search Favorites
Address
Go Links
My Computer

Frames Property

- The window object has two properties that are used with frame documents:
  - The `frames` array is used to store information about each frame. It is made up of frame objects, which work as window objects in themselves. A frame's window object can be referred by name.
  - The `parent.frames.length` property specifies the number of frames in the window.

window Object Methods

- The **`window.open()`** method is used to open a new browser window.
  - `WindowName=window.open("URL", "WindowName", "Feature List");`
- The following are the components of the window.open() statement:
  - The **`WindowName`** variable is used to store the new window object.
  - The first parameter of the window.open() method is an URL, which will be loaded into the new window. If left blank, no Web page will be loaded.
  - The second parameter specifies a window name. This is assigned to the window object's name property and is used for *targets*.
  - The third parameter is a list of optional features, separated by commas: toolbar, status line, and other features.

window Object Methods

The features available in the third parameter of the **`window.open()`** method include

- width
- height
- toolbar
- location
- directories
- status
- menubar
- scrollbars
- resizable

```
SmallWin =
   window.open("","small","width=100,height=120,toolbar=0,status=0");
```

Example

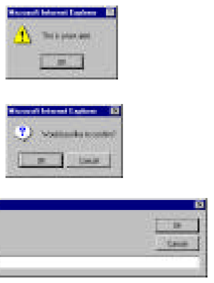
```
<HTML>
<HEAD><TITLE>Create a New Window</TITLE>
</HEAD>
<BODY>
<H1>Create a New Window</H1>
<HR>
Use the buttons below to test opening and closing windows in JavaScript.
<HR>
<FORM NAME="winform">
<INPUT TYPE="button" VALUE="Open New Window"
onClick="NewWin=window.open('ListElements.html','NewWin','toolbar=no,status=no,
         width=200,height=100'); ">
<P><INPUT TYPE="button" VALUE="Close New Window"
          onClick="NewWin.close();" >
<P><INPUT TYPE="button" VALUE="Close Main Window"
          onClick="window.close();">
</FORM>
<BR>Have fun!
<HR>
</BODY>
</HTML>
```

Displaying Dialogs

- The window object includes three methods that are useful for displaying messages and interacting with the user:
  - The **`alert()`** method displays an alert dialog box to simply gives the user a message.
  - The **`confirm()`** method displays a confirmation dialog. This displays a message and includes OK and Cancel buttons. This method returns true if OK is pressed and false if Cancel is pressed.
  - The **`prompt()`** method displays a message and prompts the user for input. It returns the text entered by the user.

Example

```
<HTML>
<HEAD><TITLE>Alerts, Confirmations, and Prompts</TITLE>
</HEAD>
<BODY>
<H1>Alerts, Confirmations, and Prompts</H1>
<HR>
Use the buttons below to test dialogs in JavaScript.
<HR>
<FORM NAME="winform">
<INPUT TYPE="button" VALUE="Display an Alert"
onClick="window.alert('This is a test alert.');  ">
<P><INPUT TYPE="button" VALUE="Display a Confirmation"
onClick="temp = window.confirm('Would you like to confirm?');
window.status=(temp)?'confirm: true':'confirm: false'; ">
<P><INPUT TYPE="button" VALUE="Display a Prompt"
onClick="var temp = window.prompt('Enter some Text:','This is the default value');
window.status=temp;  ">
</FORM>
<BR>Have fun!
<HR>
</BODY>
</HTML>
```



Timeout Method

- **`setTimeout()`** method. This method has two parameters:
  - JavaScript statement, or group of statements, enclosed in quotes.
  - the time to wait in milliseconds (thousandths of seconds).
  - For example, this statement displays an alert dialog after 10 seconds:

  ```
  ident=window.setTimeout("alert('Time's up!')",10000);
  ```
- **`clearTimeout()`** method, specifying the identifier of the timeout to stop:

  ```
  window.clearTimeout(ident);
  ```

Example

```
•   <HTML>
•   <HEAD><TITLE>Timeout Example</TITLE>
•   <SCRIPT>
•   var counter = 0;
•   // call Update function in 2 seconds after first load
•   ID=window.setTimeout("Update();",2000);
•   function Update() {
•      counter ++;
•      window.status="The counter is now at " + counter;
•      document.form1.input1.value="The counter is now at " + counter;
•   // set another timeout for the next count
•      ID=window.setTimeout("Update();",2000);
•   }
•   </SCRIPT>
•   </HEAD>
•   <BODY>
•   <H1>Timeout Example</H1>
•   <HR>
•   The text value below and the status line are being updated every two seconds.
•   Press the RESET button to restart the count, or the STOP button to stop it.
•   <HR>
•   <FORM NAME="form1">
•   <INPUT TYPE="text" NAME="input1" SIZE="40"><BR>
•   <INPUT TYPE="button" VALUE="RESET" onClick="counter = 0;"><BR>
•   <INPUT TYPE="button" VALUE="STOP" onClick="window.clearTimeout(ID);">
•   <HR>
•   </BODY>
•   </HTML>
```

document Object

- *document* object is a child of the window object.
- This object represents the contents of the current HTML Web page.
- *document* object includes the form, link, and anchor objects to describe forms, links, and anchors on the page.

document Object Properties

Information about document

- The **`URL`** property (formerly location) specifies the document's URL.
- The **`title`** property lists the title of the current page, defined by the HTML <TITLE> tag.
- The **`referrer`** property is the URL of the page the user was viewing prior to the current page-usually, the page with a link to the current page.
- The **`lastModified`** property is the date the document was last modified. This date is sent from the server along with the page.

Example

```
<HTML>
<HEAD>
    <TITLE>Test Document</TITLE>
</HEAD>
<BODY>
This page was last modified on:
<SCRIPT>
document.write(document.lastModified);
</SCRIPT>
<BR>
</HTML>
```

Link and Form Information

- The *form* objects include information about each <FORM> in the current document.
- The *anchors* array identifies each of the anchors (places that can be jumped to) in the current document.
- The *links* array includes information for each of the links in the current document.
- The *images* array contains information about each of the images in the document.

Controlling Document Appearance

- **`bgColor`** is the background color, specified with the BGCOLOR attribute.
- **`fgColor`** is the foreground (text) color, specified with the TEXT attribute.
- **`linkColor`** is the color used for nonvisited links, specified with the LINK attribute.
- **`vlinkColor`** is the color for visited links, specified with the VLINK attribute.

history Object

The history object's methods is used to send the user to other locations:

- **`history.back()`** goes back to the previous location. This is equivalent to the browser's back-arrow button.
- **`history.forward()`** goes forward to the next location. This is equivalent to the browser's forward-arrow button.
- **`history.go()`** goes to a specified location in the history list. You can specify a positive number to go forward, a negative number to go back, or a string to be searched for in the history list.

Example

```
<HTML>
<HEAD><TITLE>Back and Forward Example</TITLE>
</HEAD>
<BODY>
<H1>Back and Forward Example</H1>
<HR>
This page allows you to go back or forward to pages in the history list.
These should be equivalent to the back and forward arrow buttons in the
browser's toolbar.
<HR>
<FORM NAME="form1">
<INPUT TYPE="button" VALUE="< - BACK" onClick="history.back();">
...
<INPUT TYPE="button" VALUE="FORWARD - >" onClick="history.forward();">
</FORM>
<HR>
</BODY>
</HTML>
```

link Object

- *link* object is a child of document object. Each one includes information about a link to another location or anchor.
- You can access link objects with the links array.
- Each member of the array is one of the link objects in the current page.
- A property of the array, **`document.links.length`**, indicates the number of links in the page.

link Object

- Each link object (or member of the links array) has a list of properties defining the URL. These are the same properties as the location object, defined earlier in this chapter. You can refer to a property by indicating the link number and property name. For example, this statement assigns the variable link1 to the entire URL of the first link (index 0):

```
link1 = links[0].href;
```

link Object – Event Handlers

- The **`onMouseOver`** event happens when the mouse pointer moves over the link's text.
- The **`onClick`** event happens when the user clicks on the link.

```
<a href="#" onClick="window.alert('This is a test.');">

<a href="sound.wav" onClick="return window.confirm('play sound?');">
```

anchor Object

- Each *anchor* object represents an anchor in the current document-a particular location that can be jumped to directly.
- anchors can be accessed through an array: anchors. Each element of this array is an anchor object.
- The **`document.anchors.length`** property gives the number of elements in the anchors array.

form Objects

- *form* object represents an HTML form. There can be several form objects within a document.
- Forms can be reference by name (specified with the NAME attribute in the <FORM> tag).
- The form has many properties that are objects themselves: the *elements*, or components, of the form: text fields, buttons, and other objects that make up a form.