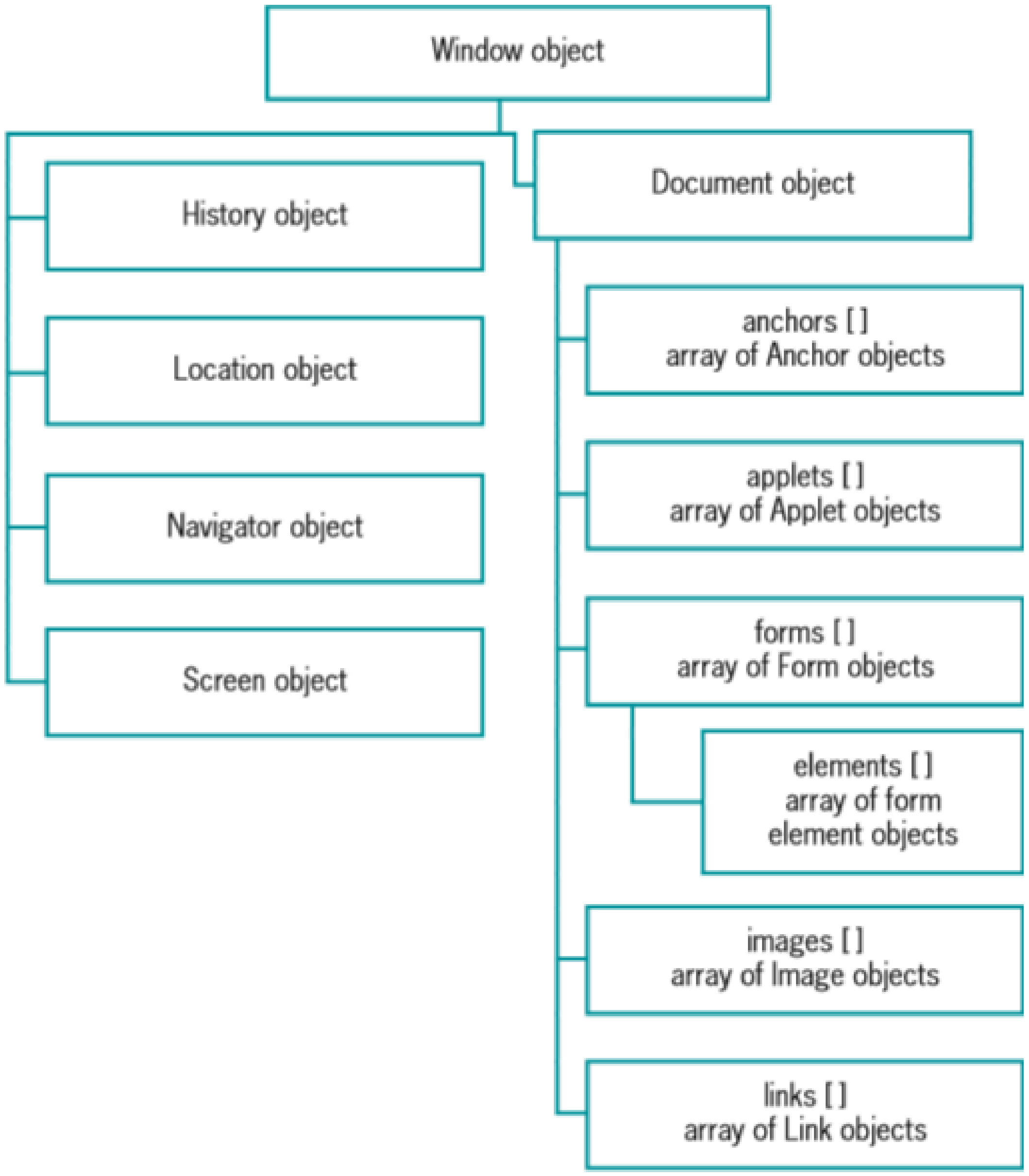
Browser Object Model

Browser Object Model

- The browser object model (BOM) is a hierarchy of objects, each of which provides programmatic access to a different aspect of the Web browser window or the Web page.
- You can use the methods and properties of objects in the browser object model to manipulate the window and elements displayed in a Web browser.
- The Window object, which represents a Web browser window.
- You use the methods and properties of the Window object to control the Web browser window.
- The Document object tat you use its methods and properties to control the Web page

Browser Object Model


Window object
History object
Location object
Navigator object
Screen object
Document object
anchors [ ] array of Anchor objects
applets [ ] array of Applet objects
forms [ ] array of Form objects
elements [ ] array of form element objects
images [ ] array of Image objects
links [ ] array of Link objects

The Document Object

- The Document object is arguably the most important object in the browser object model because it represents the Web page displayed in a browser.
  - Its common methods are the write() and writeln() methods
- All elements on a Web page are contained within the Document object, and each element is represented in JavaScript by its own object.
  - The Form object, which is used by JavaScript to represent forms created with the <form> element.
  - The Radio object, which is used by JavaScript to represent a radio button created with an <input> element,

Referencing JavaScript Objects

- Some of the objects in the browser object model represent arrays. For example,
  - The images[]array contains Image objects that represent all the <img> elements on a Web page.
  - Image objects for each <img> element are assigned to the elements of the images[] array in the order that they appear on theWeb page.
    - The first Image object is represented by images[0], the second Image object is represented by images[1], and so on.
  - Example,

  <img src="company_logo.gif" name="**companyLogo**" onclick="window.alert('This image is located at the following URL: '+ **document.companyLogo.src**);"alt="Image of a company logo." />

  same as

  <img src="company_logo.gif" onclick="window.alert('This image is located at the following URL:'+ document.images[0].src);"alt="Image of a company logo." />

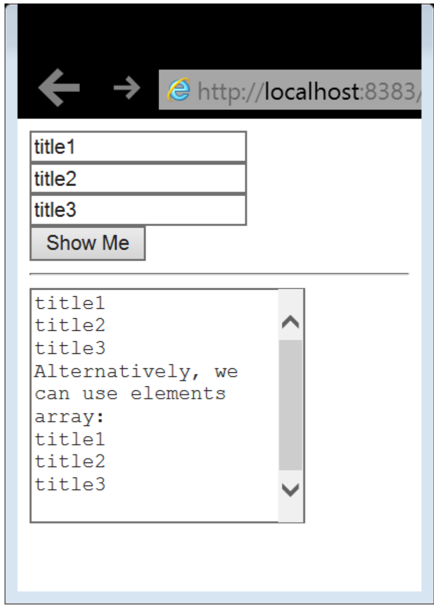
- When referring to the current object (in this case, the Image object for the preceding statement), you can also simply use the this keyword

  <img src="company_logo.gif" onclick="window.alert('This image is located at the following URL:'+ **this.src**);"alt="Image of a company logo." />

The referencing_javascript_objects.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Referencing Javascript Objects</title>
    <script type="text/javascript">
      function showMe() {
        var message;
        message = document.myForm.book1.value + "\r";
        message += document.myForm.book2.value + "\r";
        message += document.myForm.book3.value + "\r";
        message += "Alternatively, we can use elements array:" + "\r";
        message += document.myForm.elements[0].value + "\r";
        message += document.myForm.elements[1].value + "\r";
        message += document.myForm.elements[2].value + "\r";
        document.myForm.info.value = message;
      }
    </script>
  </head>
  <body>
    <form method="post" action="" name="myForm">
      <input type="text" name="book1" value="title1" /><br />
      <input type="text" name="book2" value="title2" /><br />
      <input type="text" name="book3" value="title3" /><br />
      <input type="button" name="show" onclick="showMe()" value="Show Me" />
      <hr />
      <textarea name="info" rows="10"></textarea>
    </form>

  </body>
</html>
```



```
title1
title2
title3
Alternatively, we
can use elements
array:
title1
title2
title3
```

The Window Object's Properties

- The Window object includes several properties that contain information about the Web browser window. The common properties are:
  - closed - Returns a Boolean value that indicates whether a window has been closed
  - defaultStatus - Sets the default text that is written to the status bar
  - document - Returns a reference to the Document object
  - history - Returns a reference to the History object
  - location - Returns a reference to the Location object
  - name - Returns the name of the window
  - opener - Refers to the window that opened the current window •
  - parent - Refers to a frame within the same frame set
  - self - Returns a self-reference to the Window object; identical to the window property
  - status - Specifies temporary text that is written to the status bar
  - top - Returns the topmost Window object
  - window - Returns a self-reference to the Window object; identical to the self property

The Window Object's Methods

- The Window object has various methods that allow you to manipulate the Web browser window itself
    - alert() displays a simple message dialog box with an OK button
    - blur() removes focus from a window
    - clearInterval() cancels an interval that was set with setInterval()
    - clearTimeout() cancels a timeout that was set with setTimeout()close() Closes a Web browser window
    - confirm() displays a confirmation dialog box with OK and Cancel buttons
    - focus() makes a Window object the active window
    - moveBy() moves the window relative to the current position
    - moveTo() moves the window to an absolute position
    - open() opens a new Web browser window
    - print() prints the document displayed in the current window
    - prompt() displays a dialog box prompting a user to enter information
    - resizeBy() resizes a window by a specifi ed amount
    - resizeTo() resizes a window to a specifi ed size
    - scrollBy() scrolls the window by a specifi ed amount
    - scrollTo() scrolls the window to a specifi ed position
    - setInterval() repeatedly executes a function after a specifi ed number of milliseconds haveelapsed
    - setTimeout() executes a function once after a specified number of milliseconds have elapsed

Windows and Events

- Events are particularly important when it comes to working with the browser object model because they allow you to execute the methods and change the properties of objects in the browser object model.

The click and dblclick Events

- Normally the click event is used with form controls, such as radio buttons, to execute JavaScript code.
- The click event can also be used with other types of elements. For example,
  - You can use the click event to change the image displayed on a Web page.

```
<img src="images/banner1.gif"  alt="Banner ads"
onclick="this.src='images/banner2.gif';" />
```

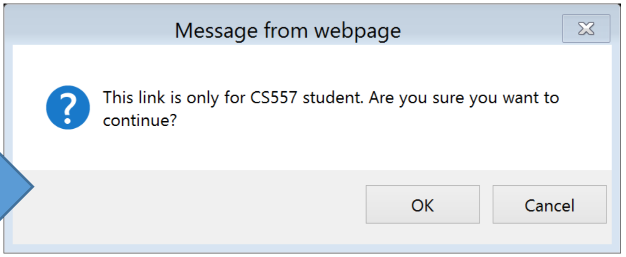
  - You can use it to override an anchor element’s automatic onclick event handler with your own code.

```
<script type="text/javascript">
    function warnUser() {
        return window.confirm("This link is only for CS557 students.
Are you sure you want to continue?");}
</script>
<body>
    <a href="CS557.html "onclick="return warnUser();">CS557 Course
</body>
```

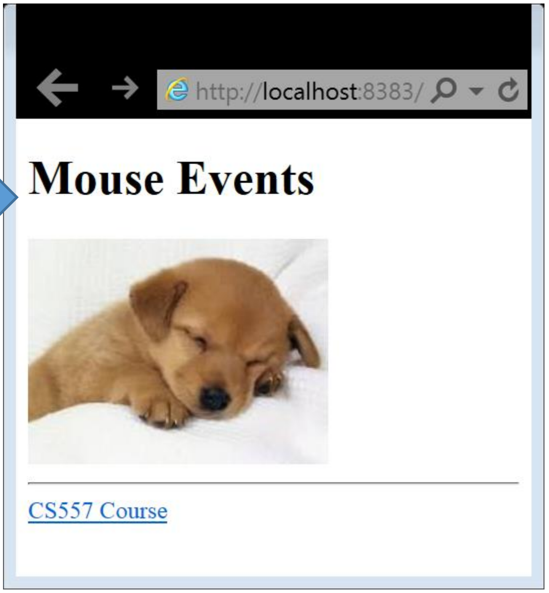
  - The dblclick event works the same as the click event, except that users need to double-click the mouse instead of single-clicking it.

The mouse_clicks.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Mouse Events</title>
      <script type="text/javascript">
         var img = "dog1.png";
         function warnUser() {
            return window.confirm("This link
is only for CS557 student. Are you sure you
want to continue?");
         }
      </script>
   </head>
   <body>
      <h1>Mouse Events</h1>
      <img src="dog1.png"  alt="Banner ads"
           onclick="img = (img == 'dog1.png')?
'dog2.png' : 'dog1.png';
           this.src=img;" />
      <hr />
      <a href="http://www.npu.edu"
onclick="return warnUser();">CS557
Course</a>
   </body>
</html>
```


Message from webpage

This link is only for CS557 student. Are you sure you want to continue?



Mouse Events

CS557 Course

The mouseover and mouseout Events

- You use the mouseover and mouseout events to create rollover effects.
- A rollover is an effect that occurs when your mouse moves over an element.
  - The mouseover event occurs when the mouse passes over an element and
  - The mouseout event occurs when the mouse moves off an element. •
- Examples:

```
<a href="npu.html" onmouseover="this.style.textDecoration='underline';"
onmouseout="this.style.textDecoration='none';">NPU Club</a>


<a href="townhouse.html"
onmouseover="this.src='townhouse.jpg';this.style.color='Red '"
onmouseout="this.src='noselection.jpg';this.style.color='Bl
ue'">Townhouse</a>


<a href="DRG5000.html"
onmouseover="document.messageForm.carLink.value='Click for more info on
the DRG 5000 SUV.'"
onmouseout="document.messageForm.carLink.value=''">DRG 5000 SUV</a>
```

The mousedown and mouseup Events

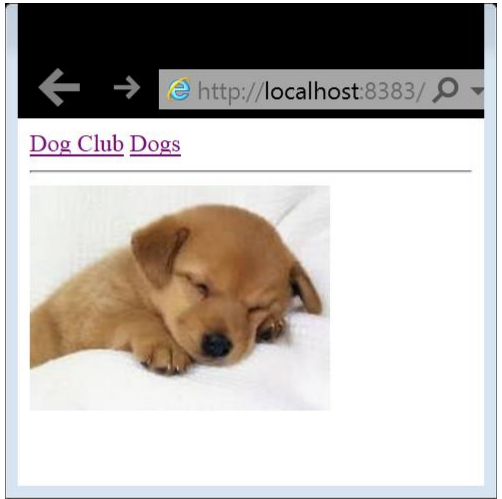
- The mousedown event occurs when you point to an element and hold the mouse button down;
- The mouseup event occurs when you release the mouse button.
- The following code shows the <img>element that displays the motorcycle and showroom images, this time using mousedown and mouseup events:

```
<img src="v500tec.gif" alt="Banner images"
onmousedown="this.src='showroom.gif'"
onmouseup="this.src='v500tec.gif'" />
```

The mouse_over.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Mouseover and mouseout</title>
   </head>
   <body>
      <a href="dog.html" onmouseover="this.style.textDecoration = 'underline';"
        onmouseout="this.style.textDecoration = 'none';">Dog Club</a>

      <a href="dogs.html" onmouseover="this.src = 'dog1.png';
            this.style.color = 'Red '" onmouseout="this.src = 'dog2.png';
                  this.style.color = 'Blue'">Dogs</a>
      <hr>
      <img src="dog1.png" alt="Banner images" onmousedown="this.src='dog2.png'"
         onmouseup="this.src='dog1.png'" />
   </body>
</html>
```



Referring to Frames and Windows

- When working with multiple frames and windows, you can refer to individual frames and windows in JavaScript code.
- The frames[] array contains all the frames in a window. The first frame in a window is referred to as frames[0], the second frame is referred to as frames[1], and so on.
- You can open a link in a new window by using the <a> element’s target attribute. For example, the following link opens the Wikipedia home page in a new window, named wiki-Window:

  <a href="http://www.wikipedia.org/"target="wikiWindow">Wikipedia home page</a>
- The target attribute is deprecated in XHTML. Instead, you normally use the open() method of the Window object. The syntax for the open()method is as follows:
  - window.open(url, name, options, replace);
  - URL represents the Web address or filename to be opened
  - name assigns a value to the name property of the new Window object
  - options represents a string that allows you to customize the new Web browser window’s appearance
  - replace is a Boolean value that determines whether the URL should create a new entry in the Web browser’s history list or replace the entry

The open() window options

- When you open a new Web browser window, you can customize its appearance by using the options argument of the window.open() method.
  - height sets the window’s height
  - left sets the horizontal coordinate of the left of the window,in pixels
  - location includes the URL Location text box
  - menubar includes the menu bar
  - resizable determines if the new window can be resized
  - scrollbars includes scroll bars
  - status includes the status bar
  - toolbar includes the Standard toolbar
  - top sets the vertical coordinate of the top of the window, in pixels
  - width sets the window’s width
  - You can use the focus() method of the Window object to make a window the active window,
  - Example:

```
var carWindow;
function showCar(linkTarget) {
    carWindow = window.open(linkTarget,
"carInfo","toolbar=no,menubar=no,location=no,scrollbars=no,resiza
ble=no,width=400,height=375");

carWindow.focus(); }
```

The close() method

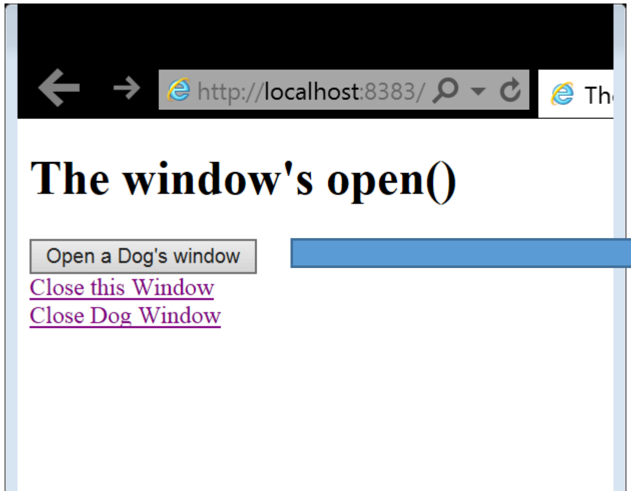
- The close() method that closes a Web browser window.
  - To close the Web browser window represented by the OpenWin variable for example, you use the statement OpenWin.close();.
  - To close the current window, you use the statement window.close() or self.close().
- For example, the onclick event handler in the anchor element calls the close() method, which will close the window.

```
<a href="" onclick="self.close();">Close Window</ a>
```

The window_open()

```
<!DOCTYPE html>
<html>
   <head>
      <title>The window's open()</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <script type="text/javascript">
         var dogWindow;
         function showDog(linkTarget) {
            dogWindow = window.open(linkTarget, "dogInfo",
                "toolbar=no,menubar=no,location=no,scrollbars=no,resizable=no,width=400,height=375");

            dogWindow.focus();
         }
      </script
   </head>
   <body>
      <h1>The window's open()</h1>
      <input type="button" name="openWindow" onclick="showDog('mouse_clicks.html')" value="Open a
Dog's window" />
      <br>
      <a href="" onclick="self.close();">Close this Window</ a><br>
      <a href="" onclick="dogWindow.close();">Close Dog Window</ a><br>
   </body>
 </html>
```

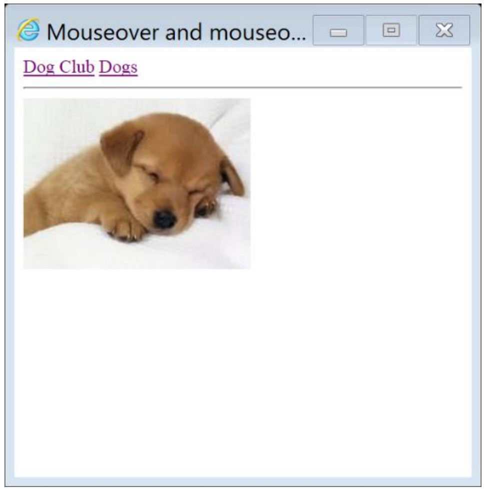


The window's open()

Open a Dog's window

Close this Window

Close Dog Window



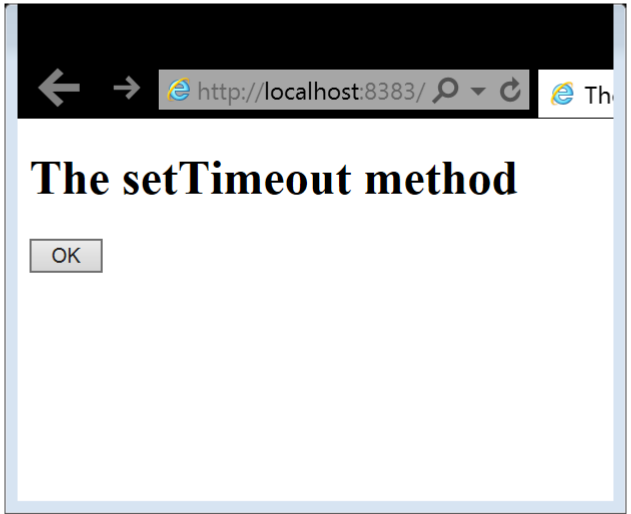
Dog Club Dogs

The Timeouts and Intervals

- Sometimes, you may need to have some JavaScript code execute repeatedly, without user intervention.
- You may also want to create animation or allow for some kind of repetitive task that executes automatically. For example,
  - To include an advertising image that changes automatically every few seconds. Or,
  - To use animation to change the ticking hands of an online analog clock.
- The setTimeout() method is used to execute code after a specific amount of time has elapsed. Code executed with the setTimeout() method executes only once.
  - var variable =setTimeout("code", milliseconds);.
  - code argument must be enclosed in double or single quotation marks and can be a single JavaScript statement, a series of JavaScript statements, or a function call.
  - milliseconds is the amount of time the Web browser should wait before executing the code argument
- The clearTimeout() method is used to cancel a setTimeout() method before its code executes. Th e clearTimeout() method receives a single argument, which is the variable that represents a setTimeout() method call

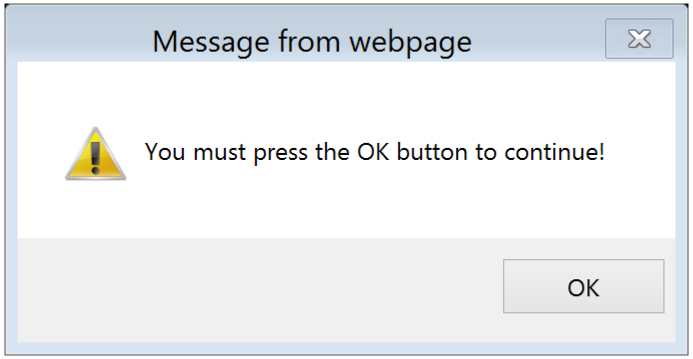
Example of Timeouts and Intervals

```
<!DOCTYPE html>
<html>
  <head>
    <title>The setTimeout method</title>
    <script type="text/javascript">
      var buttonNotPressed = setTimeout("window.alert('You must press the OK button to
continue!')",10000);
      function buttonPressed() {
        clearTimeout(buttonNotPressed);window.alert("The setTimeout() method was cancelled!");
      }
    </script>
  </head>
  <body>
    <h1>The setTimeout method</h1>
    <input type="button" value=" OK "onclick="buttonPressed();" />
  </body>
</html>
```



The setTimeout method

After 10 seconds



Message from webpage

You must press the OK button to continue!

The setInterval() method and the clearInterval()

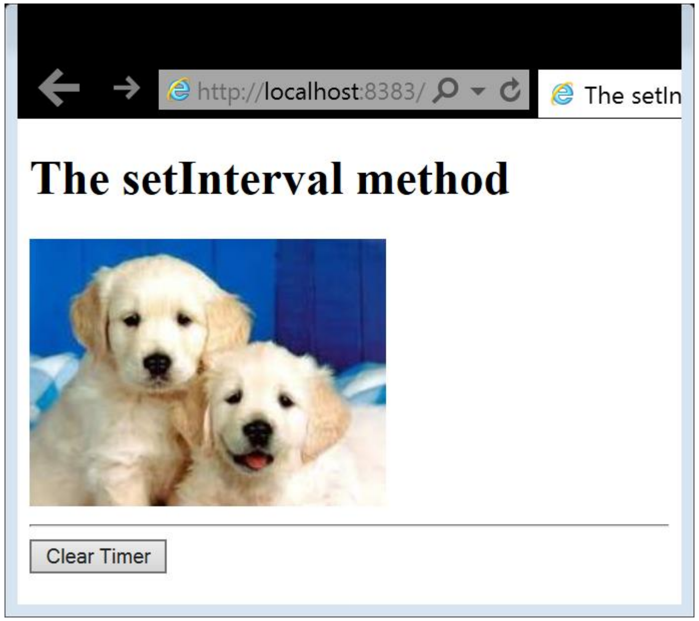
- The setInterval() method is similar to the setTimeout() method, except that it repeatedly executes the same code after being called only once.
- The clearInterval() method is used to clear a setInterval() method call.
- The setInterval() and clearInterval() methods are most often used for starting animation code that executes repeatedly.

The example of setInterval() method and the clearInterval()

- The following code uses the setInterval() method to automatically swap the images every two seconds

```
<!DOCTYPE html>
<html>
   <head>
      <title>The setInterval method</title>
      <script type="text/javascript">
         var curBanner="cycle1";
         var begin;
         function changeBanner() {
            if (curBanner == "cycle2") {
               document.images[0].src = "dog1.png";
               curBanner = "cycle1";
            }else {
               document.images[0].src = "dog2.png";
               curBanner = "cycle2";
            }
         }

      </script>
   </head>
   <body onload="begin=setInterval('changeBanner()',2000);">
      <h1>The setInterval method</h1>
      <img src="dog1.png"  alt="Banner images" />
      <hr />
      <input type="button" name="cleartimer" value="Clear Timer"
onclick="clearInterval(begin);" />
   </body>
</html>
```



The History Object

- The History object maintains an internal list (known as a history list) of all the documents that have been opened during the current Webbrowser session.
- Each Web browser window contains its own internal History object.
- Two important security features of the History Object:
  1. The History object will not actually display the URLs contained in the history list. In other words, it prevents others from viewing the URLs in a History list.
  2. The Javascript code can use the history list to navigate to Web pages that have been opened during a Web browser session. But only Web pages exist within the same domain as the Web page containing the JavaScript code that is attempting to move through the list.

The History Object's methods

- The History object includes three methods:
  - back() produces the same result as clicking a Web browser's Back button.
  - forward() produces the same result as clicking a Web browser's Forward button.
  - go() opens a specific document in the history list.
    - The argument is an integer that indicates how many pages in the history list, forward or backward. For example,
      - history.go(-2);   // Two pages back
      - history.go(3);   // Three pages forward
      - history.go(1);   // same as forward()
      - history.go(-1);   // same as back()
    - The length property that contains the specifc number of documents that have been opened during the current browser session. For example,

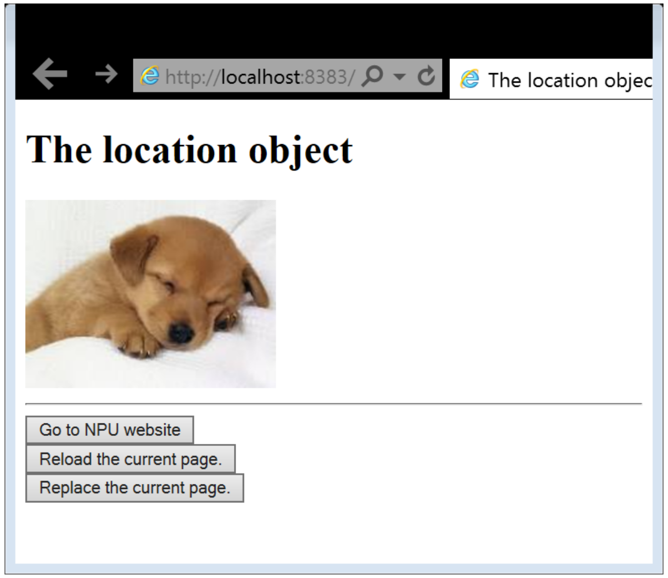
```
window.alert("You have visited " + history.length+ " Web pages.");
```

The Location Object

- You can use JavaScript code and the Location object to open Web pages. For example, you may want to redirect your Web site visitors to a different or updated URL.
- The Location object contains several properties and methods for working with the URL of the current document in a Web browser window.
- The Location Object's properties:
  - hash is a URL’s anchor
  - host is the host and domain name (or IP address) of a network host
  - hostname is a combination of the URL’s host name and port sections
  - href is the full URL addresspathname The URL’s path
  - port is the URL’s port
  - protocol is the URL’s protocol
  - search is a URL’s search or query portion
  - The Location Object's methods:
    - assign() loads a new Web page
    - reload() causes the page that currently appears in the Web browser to open again
    - replace() replaces the currently loaded URL with a different one

The location.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>The location object</title>
  </head>
  <body>
    <h1>The location object</h1>
    <img src="dog1.png"  alt="Banner ads" onclick="img = (img == 'dog1.png')? 'dog2.png' : 'dog1.png';
this.src=img;" />
    <hr />
    <input type="button" name="assign" onclick="location.assign('http://www.npu.edu');" value="Go to NPU
website"/><br />
    <input type="button" name="reload" onclick="location.reload(true);" value="Reload the current page."
/><br />
    <input type="button" name="replace" onclick="location.replace('http://secure.npu.edu');" value="Replace
the current page." /><br />
  </body>
</html>
```


http://localhost:8383/
The location objec
The location object
Go to NPU website
Reload the current page.
Replace the current page.