JavaScript Timing Events

The **global window object** offers a few little functions that are great if you want to create the effect of movement on the web.

In this lesson, you will learn about:

- **setTimeout()**;
- **clearTimeout()**;
- **setInterval()**;
- **clearInterval()**.

In the process, you will also learn how to **preload images** with JavaScript and how to **build a simple photo gallery application**.

How do I use setTimeout()?

If your program has a chunk of code that needs to be executed after a specified time, then
setTimeout(actionToExecute, timeInMilliseconds) is your function.

Let's see it in action. In this simple example, we will call up an alertbox 2 seconds after page load. Type the code below in enclosing <script> tags on a simple HTML page:

```
//Package the code to insert in the timer in a simple function:
function sayHello(){
	alert("Hello");
}

//Package the timer into its own function
function timeGreeting(){
	//assign the timer to a variable:
	//you need to do this so the timer can be
	//referenced by another function when you want to stop it:
	var greeting = setTimeout(sayHello, 2000);
	//Notice how the time is set in milliseconds.
}

//Now call the timer function
timeGreeting();
```

Save your work and run it in a browser. After 2 seconds, an alertbox should pop up to greet you.

How do I use setInterval()?

If you need to execute a chunk of code at specified time intervals, then
setInterval(codeToExecute, timeIntervalInMilliseconds) is your function.

To see it in action in its most annoying manifestation, just replace **setTimeout** with **setInterval** in the previous example and you're done.

Now you should see an alertbox greeting you every 2 seconds! That's enough, just close that window in your browser and let's modify our script so that we can put a stop to that nuisance.

How do I stop the timer?

JavaScript offers 2 handy methods to get rid of timers: **clearTimeout(timerVariable)** if your timer uses setTimeOut(), and **clearInterval(timerVariable)** if your timer uses setInterval(). Let's put *clearInterval()* to good use right away by stopping that annoying alertbox. First of all, add an inputbox to your HTML page, like so:

```
<input type="button" value="Stop timer" onclick="stopTimer()" />
```

Next, rewrite the preceding JavaScript code and then add the new function as follows:

```
//Make the variable holding the timer global,
//so that it can be accessed both by the function setting
//the timer and by the function stopping the timer:
var greeting;

//sayHello() remains unchanged from the previous example
function sayHello(){
        alert("Hello");
}

//we increase the time interval to give more time to the user to
//click the button that stops the alert:
function timeGreeting(){
        greeting = setInterval(sayHello, 5000);
}

timeGreeting();


//package the code that cancels the timer
//in its own function that will be called when
//the button on the page is clicked:
function stopTimer(){
        //you call clearInterval() and pass the variable
        //containing the timer as argument
        clearInterval(greeting);
}
```

Save your work and preview it in a browser. Just click the button and the annyoing alertbox disappears: that's much better, great! I leave you to experiment with clearTimeout() on your own.

A photo gallery application

JavaScript timers are often used to produce animation effects on the web. Photo galleries are one of those widely used applications that often employ JavaScript timers.

At this stage, your photo gallery will consist of a photo container and 2 buttons: one button to stop the gallery, and one button to restart the gallery. As soon as the page loads, the gallery automatically displays its images one at a time every 2 seconds.

You will also need a few images, possibly all of the same size. For this demo, I prepared four 620px X 378px graphics and saved them in their own *images* folder.

Write an HTML page, which is made of a , an <img> tag, and 2 <input> tags. Also included are:

1) a small style declaration in the <head> of the document to the effect that the element that contains the gallery will be centered and sized to the same width and height as the gallery graphics; and

2) a reference to an external JavaScript file at the bottom of the <body> element of the HTML page. This location is the most appropriate one because we need to make sure the HTML page and, most importantly, the image referenced by the <img> tag, are fully loaded before the script kicks in.

```
<!DOCTYPE html>
<html>
<head>
        <title>JavaScript Timing Events</title>
        <style type="text/css">
                #gallery{
                        …
                }
        </style>
</head>
<body>
  <h1>My Photo Gallery</h1>
  <div id="gallery">
                …
  </div>
  <script src="galleryScript.js"></script>
</body>
</html>
```

Now prepare the **galleryScript.js** file. Your JavaScript code consists of 3 functions:

1. **init()** contains initialization routines: it preloads the photo gallery images so that they will be ready for display as soon as the script calls them. In addition, it binds event handlers to the 2 button elements to stop and restart the gallery;
2. **startGallery()** displays each graphic in the gallery every 2 seconds;
3. **stopGallery()** stops the gallery so that the photo that comes next with respect to the current photo is not displayed.

Furthermore, the code contains a few global variables at the top that need to be accessed by all the functions in the script. Let's get started.

```
//Global vars: a reference to the photo currently displayed on the page,
var curImage = document.getElementById("photo");

//a variable to store the timer,
var galleryStarter;

//a variable to store a true/false value indicating to the program whether the gallery is on or off,
var isGalleryOn = true;

//an array containing 4 strings representing the filepaths to the image files in the images folder,
var images = ["./images/1.jpg", "./images/2.jpg", "./images/3.jpg", "./images/4.jpg"];

//an empty array that will be filled with 4 preloaded image objects: the src property
//of these image objects will correspond to the filepaths contained in the images[] array,
var preloadedImgs = [];

//a variable that works as our counter to advance from one image to the next.  It starts at 0.
var counter = 0;

/*****************************************************************************************************/
//init() starts with the image preloading routine. First fill the preloadedImgs[] array
//with a number of image objects corresponding to the length of the images[] array:
function init(){
        …
    //now assign the value of the strings contained in the images[] array to the
    //src property of each image object in the preloadedImgs[] array, using one more loop:
        …
    //Now, assign event handlers to the 2 buttons to restart and stop the gallery:
    var btnStart = document.getElementById("btnStart");
    var btnStop = document.getElementById("btnStop");
    btnStart.onclick = startGallery;
    btnStop.onclick = stopGallery;
    //Finally, check the isGalleryOn flag is true.  If it is call the function that starts the gallery:
    if (isGalleryOn){
        startGallery();
    }
}
/*****************************************************************************************************/
//Assign the init() function to the onload event
onload = init;
```

```
/*****************************************************************************************************/
        //startGallery() contains the functionality to extract the photos from
        //the preloadedImgs[] array for display and to set the timer in motion:
        function startGallery(){
                //extract the image filepath using the counter variable as array index
                //and assign it to the src property of the curImage variable:
                …
                //advance the counter by 1:
                …
                //if the counter has reached the length of the preloadedImgs[] array,
                //take it back to 0, so the photo gallery redisplays the images from the start:
                …
                //Set the timer using this same function as one
                //of the arguments and 2000 (2 milliseconds) as the other argument.
                …
                //Signal that the gallery is on to the rest of the program:
                isGalleryOn = true;
        }

        /*****************************************************************************************************/
        //stopGallery() uses clearTimeout() to stop the gallery
        function stopGallery(){
                …
                //signal that the gallery has stopped to the rest of the program:
                isGalleryOn = false;
        }
```