JavaScript Timers with setTimeout and setInterval

JavaScript features a handy couple of methods of the window object: setTimeout() and setInterval(). These let you run a piece of JavaScript code at some point in the future. In this tutorial we'll explain how these two methods work, and give some practical examples.

## *setTimeout()*

window.setTimeout() allows you to specify that a piece of JavaScript code (called an **expression**) will be run a specified number of milliseconds from when the setTimeout() method was called. The general syntax of the method is:

setTimeout ( *expression*, *timeout* );

where expression is the JavaScript code to run after timeout milliseconds have elapsed.

setTimeout() also returns a numeric timeout ID that can be used to track the timeout. This is most commonly used with the clearTimeout() method (see below).

Here's a simple example:

<input type="button" name="clickMe" value="Click me and wait!"

onclick="setTimeout('alert(\'Surprise!\')', 5000)"/>

When a visitor clicks the button, the setTimeout() method is called, passing in the expression that we want to run after the time delay, and the value of the time delay itself - 5,000 milliseconds or 5 seconds.

Try it yourself! Click the button below and wait 5 seconds:

It's worth pointing out that setTimeout() doesn't halt the execution of the script during the timeout period; it merely schedules the specified expression to be run at the specified time. After the call to setTimeout() the script continues normally, with the timer running in the background.

In the above simple example we embedded the entire code for our JavaScript alert box in the setTimeout() call. More usually, you'd call a function instead. In this next example, clicking a button calls a function that changes the button's text colour to red. At the same time, this function also sets up a timed function using setTimeout() that sets the text colour back to black after 2 seconds:

<script type="text/javascript">

function setToRed ( )

{

document.getElementById("colourButton").style.color = "#FF0000";

setTimeout ( "setToBlack()", 2000 );

}

function setToBlack ( )

{

document.getElementById("colourButton").style.color = "#000000";

}

</script>

<input type="button" name="clickMe" id="colourButton" value="Click me and wait!" onclick="setToRed()"/>

Click the button below to see it in action:

## *clearTimeout()*

Sometimes it's useful to be able to cancel a timer before it goes off. The clearTimeout() method lets us do exactly that. Its syntax is:

clearTimeout ( timeoutId );

where timeoutId is the ID of the timeout as returned from the setTimeout() method call.

For example, the following code sets up an alert box to appear after 3 seconds when a button is clicked, but the visitor can click the same button before the alert appears and cancel the timeout:

<script type="text/javascript">

var alertTimerId = 0;

function alertTimerClickHandler ( )

{

if ( document.getElementById("alertTimerButton").value == "Click me and wait!" )

{

// Start the timer

document.getElementById("alertTimerButton").value = "Click me to stop the timer!";

alertTimerId = setTimeout ( "showAlert()", 3000 );

}

else

{

document.getElementById("alertTimerButton").value = "Click me and wait!";

clearTimeout ( alertTimerId );

}

}

function showAlert ( )

{

alert ( "Too late! You didn't stop the timer." );

document.getElementById("alertTimerButton").value = "Click me and wait!";

}

</script>

<input type="button" name="clickMe" id="alertTimerButton" value="Click me and wait!" onclick="alertTimerClickHandler()"/>

Try it out! Click the button below to start the timer, and click it again within 3 seconds to cancel it.

## *setInterval()*

The setInterval() function is very closely related to setTimeout() - they even share similar syntax:

setInterval ( expression, interval );

The important difference is that, whereas setTimeout() triggersexpression only once, setInterval() keeps triggering expressionagain and again (unless you tell it to stop).

So when should you use it? Essentially, if you ever find yourself writing code like:

setTimeout ( "doSomething()", 5000 );

function doSomething ( )

{

// (do something here)

setTimeout ( "doSomething()", 5000 );

}

...then you should probably be using setInterval() instead:

setInterval ( "doSomething()", 5000 );

function doSomething ( )

{

// (do something here)

}

Why? Well, for one thing you don't need to keep remembering to call setTimeout() at the end of your timed function. Also, when using setInterval() there's virtually no delay between one triggering of the expression and the next. With setTimeout(), there's a relatively long delay while the expression is evaluated, the function called, and the new setTimeout() being set up. So if you need regular, precise timing - or you need to do something at, well, set intervals - setInterval() is your best bet.

## *clearInterval()*

As you've probably guessed, if you want to cancel a setInterval() then you need to call clearInterval(), passing in the interval ID returned by the call to setInterval().

Here's a simple example of setInterval() and clearInterval() in action. When you press a button, the following code displays "Woo!" and "Yay!" randomly every second until you tell it to stop. (I said a simple example, not a useful one.) Notice how we've also used a setTimeout()within the wooYay() function to make the "Woo!" or "Yay!" disappear after half a second:

<script type="text/javascript">

var wooYayIntervalId = 0;

function wooYayClickHandler ( )

{

if ( document.getElementById("wooYayButton").value == "Click me!" )

{

// Start the timer

document.getElementById("wooYayButton").value = "Enough already!";

wooYayIntervalId = setInterval ( "wooYay()", 1000 );

}

else

{

document.getElementById("wooYayMessage").innerHTML = "";

document.getElementById("wooYayButton").value = "Click me!";

clearInterval ( wooYayIntervalId );

}

}

function wooYay ( )

{

if ( Math.random ( ) > .5 )

{

document.getElementById("wooYayMessage").innerHTML = "Woo!";

}

else

{

document.getElementById("wooYayMessage").innerHTML = "Yay!";

}

setTimeout ( 'document.getElementById("wooYayMessage").innerHTML = ""', 500 );

}

</script>

<div id="wooYayMessage" style="height: 1.5em; font-size: 2em; color: red;"></div>

<input type="button" name="clickMe" id="wooYayButton" value="Click me!" onclick="wooYayClickHandler()"/>

And here it is in action. Click the button below to kick it off:

## *Summary*

We've now covered the four methods that you can use to create timed events in JavaScript: setTimeout() and clearTimeout() for controlling one-off events, and setInterval() and clearInterval() for setting up repeating events. Armed with these tools, you should have no problem creating timed events in your own scripts.