**Using JavaScript to Create a Timer**

May 14, 2015 by [stephen](http://logicalmoon.com/author/stephen/" \o "View all posts by stephen)

I’m currently working on a project that requires a timer. Nothing fancy – it just needs to count up the amount of elapsed time from a given start point to when it stops, showing that in terms of whole minutes and seconds. It’s not even that important that it is **exactly**accurate, but of course, that would be nice!

So how do we do it? Clearly we need to do this on the front-end in the browser because it would be painful (wasteful even?) to keep making round-trips to the server to find out the elapsed time, so that means JavaScript is an obvious choice. Fortunately, there are some in-built functions that make this quite easy.

**Setting up the HTML**

Before we begin though, let’s set up our HTML to be able to show the changes in time.



|  |  |
| --- | --- |
| 1  2  3  4 | <div class="timer">      <span id="minutes">0</span> minute(s)      <span id="seconds">0</span> seconds.  </div> |

You can see I have skipped all the other elements and just focussed on the parts that matter. Don’t worry though: you can see the whole source by clicking the link at the bottom of this page and then right-clicking in the page before choosing *view page source*.

Looking at what we have so far, you can see that I have created two placeholders using the span elements and labelled them as minutes and seconds. This is important because we will need to reference these when we come to fill in the changes.

Also, notice the spaces just after each closing span tag — without those, the numbers would touch the text, making it harder to read.

**Starting the Timer**

The next step is to start the timer and to do that we are going to use the in-built function: setInterval. Let’s take a look at the code and then I will explain what is happening afterwards.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <script type="text/javascript">      function startTimer() {          var seconds = 0;  timer = setInterval(function() {      seconds ++;      document.getElementById("seconds").innerText = seconds % 60;              document.getElementById("minutes").innerText = parseInt(seconds / 60);          }, 1000);  } |

I’ve wrapped up this code in a function called startTimer and immediately, I create a variable called seconds and set it to zero. This will hold the total elapsed number of seconds for our timer.

Next, I use the setInterval function and pass into that an anonymous function which will be called each time the timer fires. Firing is not exactly literal  I just mean when the event occurs. You can see in the second argument to that function that I want the event to fire every 1000 milliseconds (line 8). It’s worth mentioning that the interval is desired but not guaranteed. ! You can read more about the nuances of this [here](http://javascript.info/tutorial/settimeout-setinterval).

So what happens inside this anonymous function? We firstly increment seconds and then set the text inside our span element to be that number modulus 60. This simply means I want the remainder when we divide the total number of seconds by 60. To illustrate that, consider this example:

*Imagine seconds was 61. We’d expect our result to be 1 minute and 1 second wouldn’t we? So, what do we get when we divide*[*61 by 60*](https://www.google.co.uk/search?q=61%2F60&oq=61%2F60)*? 1.01666666667. That means, we were able to get 1 whole lot of 60 with a*[*teeny*](https://www.google.co.uk/search?&q=define+teeny)*bit left over. That teeny bit is basically 1 second (2/125 ish of 60). In term so the modulus operator, it will just result in 1 and not the fractional number here.*

If the maths is making your head spin, don’t worry too much. The kinds of answers we expect for remainders are between 0 and 59. Can you guess why it will never be 60? I’ll tell you at the bottom if you’re still not sure.

OK, so we have the number of seconds left. To get the number of whole minutes, we divide by 60 and take the whole (integer) amount of that. Using our previous example above, it’s the 1 (whole) part.

The actual filling in of the span text occurs with the document.getElementId…line. That simply tells the browser to go hunting for the element and then assign the required value to the text.

Lastly, to keep track of this timer, I store the handle to the event in the variable timer; we’ll need that later when we come to stop it.

**Stopping the Timer**

Hang on, what about stopping the timer? Again, we have a function we can use and that’s clearInterval.

Do you remember when we stored the handle to the event in the variable timer? We are going to need that now.



|  |  |
| --- | --- |
| 1  2  3  4  5 | <script type="text/javascript">      function stopTimer() {          clearInterval(timer);      }  } |

Shockingly easy, right?

**Adding Some Control Buttons**

Finally, we can add in two buttons to allow the user to start and stop all the timing goodness.



|  |  |
| --- | --- |
| 1  2 | <input type="button" value="START" onclick="startTimer()"></input>  <input type="button" value="STOP" onclick="stopTimer()"></input> |

Here’s what it looks like; I did say it would be basic, right?!

[](http://logicalmoon.com/wp-content/uploads/2015/05/timer-example.jpg)

*Screenshot*

**Did I Forget Anything?**

Maybe  A friend pointed out an interesting situation.

What do you think would happen if you repeatedly pressed the start button? I hadn’t thought about this since it’s only a small example, but in effect, what occurs is that multiple functions are assigned to that interval timer which can lead to all sorts of race and clashing conditions.

The solution is to disable the start button once it has been pressed and in the webpage below, you can see that I have done that. If you fancy having a go yourself, take a look at the disabled property; if you get stuck, you can always look at my source code.

That’s more or less it, then. Take a look at the example [here](http://apps.logicalmoon.com/timer.html). You can also find the source on my [GitHub](https://github.com/s-moon/TimerExample" \t "_blank) page.

Oh…before I forget…why can’t we get a remainder of 60? Well, that would make another whole minute, wouldn’t it