28. JavaScript Timing Events

JavaScript can be executed in time-intervals.

This is called timing events.

28.1. Javascript Timing Events

With JavaScript, it is possible to execute some code at specified time-intervals. This is called timing events.

It's very easy to time events in JavaScript. The two key methods that are used are:

- setInterval() - executes a function, over and over again, at specified time intervals
- setTimeout() - executes a function, once, after waiting a specified number of milliseconds

Note: The setInterval() and setTimeout() are both methods of the HTML DOM Window object.

28.2. The setInterval() Method

The setInterval() method will wait a specified number of milliseconds, and then execute a specified function, and it will continue to execute the function, once at every given time-interval.

Syntax

```
window.setInterval("javascript function",milliseconds);
```

The **window.setInterval()** method can be written without the window prefix.

The first parameter of setInterval() should be a function.

The second parameter indicates the length of the time-intervals between each execution.

Note: There are 1000 milliseconds in one second.

Example

Alert "hello" every 3 seconds:

```
setInterval(function(){alert("Hello")},3000);
```

The example show you how the setInterval() method works, but it is not very likely that you want to alert a message every 3 seconds.

Below is an example that will display the current time. The setInterval() method is used to execute the function once every 1 second, just like a digital watch.

Example

Display the current time:

```
var myVar=setInterval(function(){myTimer()},1000);

function myTimer()
{
var d=new Date();
var t=d.toLocaleTimeString();
document.getElementById("demo").innerHTML=t;
}
```

28.3. How to Stop the Execution?

The clearInterval() method is used to stop further executions of the function specified in the setInterval() method.

Syntax

```
window.clearInterval(intervalVariable)
```

The **window.clearInterval()** method can be written without the window prefix.

To be able to use the clearInterval() method, you must use a global variable when creating the interval method:

```
myVar=setInterval("javascript function",milliseconds);
```

Then you will be able to stop the execution by calling the clearInterval() method.

Example

Same example as above, but we have added a "Stop time" button:

```
<p id="demo"></p>
<button onclick="myStopFunction()">Stop
time</button>

<script>
var myVar=setInterval(function(){myTimer()},1000);
function myTimer()
{
var d=new Date();
```

```
    var t=d.toLocaleTimeString();
    document.getElementById("demo").innerHTML=t;
    }
    function myStopFunction()
    {
    clearInterval(myVar);
    }
    </script>
```

28.4. The setTimeout() Method

Syntax

```
    window.setTimeout("javascript function",milliseconds);
```

The **window.setTimeout()** method can be written without the window prefix.

The setTimeout() method will wait the specified number of milliseconds, and then execute the specified function.

The first parameter of setTimeout() should be a function.

The second parameter indicates how many milliseconds, from now, you want to execute the first parameter.

Example

Wait 3 seconds, then alert "Hello":

```
    setTimeout(function(){alert("Hello")},3000);
```

28.5. How to Stop the Execution?

The clearTimeout() method is used to stop the execution of the function specified in the setTimeout() method.

Syntax

```
    window.clearTimeout(timeoutVariable)
```

The **window.clearTimeout()** method can be written without the window prefix.

To be able to use the clearTimeout() method, you must use a global variable when creating the timeout method:

```
myVar=setTimeout("javascript function",milliseconds);
```

Then, if the function has not already been executed, you will be able to stop the execution by calling the clearTimeout() method.

Example

Same example as above, but we have added a "Stop the alert" button:

```
var myVar;

function myFunction()
{
myVar=setTimeout(function(){alert("Hello")},3000);
}

function myStopFunction()
{
clearTimeout(myVar);
}
```

28.6. More Examples

a) Another simple timing

```
<!DOCTYPE html>
<html>
<head>
<script>
function timedText()
{
var x=document.getElementById('txt');
var t1=setTimeout(function(){x.value="2
seconds"},2000);
var t2=setTimeout(function(){x.value="4
seconds"},4000);
var t3=setTimeout(function(){x.value="6
seconds"},6000);
}
</script>
</head>
```

```
<body>
<form>
<input type="button" value="Display timed text!"
onclick="timedText()" />
<input type="text" id="txt" />
</form>
<p>Click on the button above. The input field will
tell you when two, four, and six seconds have
passed.</p>
</body>

</html>
```

b) A clock created with a timing event

```
<!DOCTYPE html>
<html>
<head>
<script>
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":
"+s;
t=setTimeout(function(){startTime()},500);
}

function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
```

```
    </body>
    </html>
```