JavaScript - Timers and timeouts

This article explains how to use JavaScript timer functions

Introduction

JavaScript has useful methods for performing delays and timings, countdowns, these are:

setInterval() clearInterval() setTimeout() clearTimeout()

The setTimeout and setInterval methods belong to the window object and are useful to perform some actionat a later point in time. One example would be a game where the player needs to e.g. answer a question or perform an move within a specific amount of time and if fails to do that, we could then display a message that the time to answer has been run out.

Timer functions

For this example purpose we will use the window.setTimeout() method. Our message will be displayed within 5 seconds. The syntax for for the timeout method is :

setTimeout (expression, timeout);

When the Timeout period is running, the script can continue normal working and only when the timeout timehas passed, what has been set to it will execute. This method returns a timeout ID (numeric) for timeout tracking.

Example code

```
<input type="button" name="Hit the GO button to waste 5 seconds of your life." value=""onclick="setTimeout('alert(\'That was it, you
```

To call a your custom function instead of the alert, you would use:

```
setTimeout ( "wasteFiveSeconds()", 5000 );
```

The accompanying method to use is clearTimeout(). This is what you would like to have in case that you want tocancel the timeout that has been launched. In case of the casual gamer, the response time awaited was 5 seconds,but if the player succeeds to comply before that, it would make the timer's resulting message useless. So whatwe would do is to use the ID of the timeout what was returned when calling the setTimeout() method.

For this we would use clearTimeout (timeoutId) syntax. e.g. like when we are awaiting the user to hit theGO! button, and if he actually does do that, well then we better cancel the timer right there.

The example to demonstrate this is below. The code will use the button once again. Now after the player has pressed it to let the timer roll, the text will change to "Give up!". This means that the player is in doubt and does not want to wait the five seconds. If player again hits the same button, the timer is stopped, the countdown is canceled and this is brought to the attention of the player.

```
<script type="text/javascript">
var myTimerID = 0;

function playTheGame()
{
  if (document.getElementById("startTheGameTimer").value == "GO!")
  {
    document.getElementById("startTheGameTimer").value = "Give up!";
    myTimerID = setTimeout ( "displayMessage()",5000 );
  }
  else
  {
    document.getElementById("startTheGameTimer").value = "GO!";
    clearTimeout ( myTimerID );
  }
}

function displayMessage ()
{
  alert ("You did not give it up early enough. Sorry!");
  document.getElementById("startTheGameTimer").value = "GO!";
}

</script>

<input type="button" name="clickMe" id="startTheGameTimer" value="GO!" onclick="playTheGame()"/>
```

Now let us check out the other timing functionality, called setInterval() method.
This is useful when you have e.g. game running, usually you want that certain things will take place in a regular manner, e.g in 500ms intervals.

There is not much difference between the setTimeout() and setInterval(), their syntax is the same.

```
setInterval ( expression, interval );
```


(C) Copyright Nokia 2013. All rights reserved.

The usefulness in the setInterval() is in it's ability to keep on repeating, unless it is cancelled.

Example:

```
setInterval ("updateGameBoard()",500);

function updateGameBoard()
{
  // Draw the player lunar module and calculate decending speed.
}
```

To cancel this from happening, you must then use the method clearInterval(), and pass in the ID of the interval, the one that was returned when the setInterval() gets called.

```
<script type="text/javascript">
var myIntervalID = 0;

function gameRunner()
{
  if (document.getElementById("gameButton").value == "Click me!")
  {
    // Start the timer
    document.getElementById("gameButton").value = "Enough already!";
    myIntervalID = setInterval ("gameUpdate()", 500);
  }
  else
  {
    document.getElementById("gameUpdateMessage").innerHTML = "";
    document.getElementById("gameButton").value = "Click me!";
    clearInterval ( myIntervalID );
  }
}

function gameUpdate ()
{
  if (Math.random() > .5)
  {
    document.getElementById("gameUpdateMessage").innerHTML = "You won!";
  }
  else
  {
    document.getElementById("gameUpdateMessage").innerHTML = "You lost!";
  }
  setTimeout ('document.getElementById("gameUpdateMessage").innerHTML = ""', 500);
}
</script>

<div id="gameUpdateMessage"></div>
<input type="button" name="go" id="gameButton" value="GO!" onclick="gameRunner()"/>
```

Summary

Basically the code messages above are just fictional, and the code itself does not play any game. The only purpose is to demonstrate the timing functions. Ofcourse, you could utilize the presentedmethods in your own code the way you wish.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.