imeout. 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() triggers expression only once, setInterval() keeps triggering expression again 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.

FILED UNDER: JAVASCRIPT TAGGED WITH: COUNTDOWN, REGULAR, TIME DELAYS, TIMED EVENTS, TRIGGER, TRIGGERING

Reader Interactions

Comments

akeane says

3 March 2010 at 3:54 am

Excellent explanation of timers and your examples are great.
Thank you.

Reply

matt says

3 March 2010 at 4:34 pm

@akeane: Thanks for your kind words. I'm glad you enjoyed the tutorial. 🙂

Cheers,
Matt

Reply

davehardwick says

10 April 2010 at 2:57 pm

Another thing to talk about here with setInterval is setting a loop counter and clearing the interval when that loop count is hit.

```
var counter = 0
intervalId = 0;

function myLoopFunction()
{
        intervalId = setInterval(function() {
        //make this is the first thing you do in the set interval function
        counter++;

        //do you code for setInterval() and clearInterval under normal conditions here...

        //okay, if we have tried for 5 minutes (30 x 10 seconds ), then lets stop trying because we did not reach the clearInterval under number means

       //make this the last check in your set interval function
        if ( counter > 30 ) {
            clearInterval(polliFrameSourceIntervalId);
        }

        //end setInterval
       } , 10000);
```

Reply
anupam says

7 December 2010 at 12:12 am

Very useful post, and the examples are just superb, thanks, it helped a lot!!

Reply
matt says

9 December 2010 at 7:22 pm

@anupam: Thanks, glad you found it helpful 🙂

Reply
NT_Jester says

19 January 2011 at 8:27 am

I Just want to thank you for such a great and simple to follow tutorial. I have looked at a lot over the past few weeks and I have got to say this is the best one I have seen for ages. Keep up the good work this has helped me heaps.

Reply
matt says

20 January 2011 at 2:20 pm

@NT_Jester: You're welcome – thanks for the kind words. 🙂 Hope life is good up in the NT!

Reply
ecommy says

19 August 2011 at 7:47 am

Great and detailed explanations, I took the time to create an account just to thank you as it helped debugging a JavaScript issue I had since a while.

Reply
matt says

23 August 2011 at 10:41 pm

@ecommy: Great stuff 🙂

Reply
froopy says

8 February 2012 at 8:16 am

Thank you for this post.
I was able to modify this for a music writing application.
I will link back.

Reply
matt says

13 February 2012 at 4:10 am

@froopy: You're welcome – thanks for the link back 🙂

Reply
jacky says

30 April 2012 at 4:10 pm

thank you for taking the time to create this clear and useful post!

Reply
frikson says

6 May 2015 at 6:47 am

Start a new thread please rather than hijacking this Article discussion.

Thank You

[Edited by chrishirst on 06-May-15 10:01]

Reply
frikson says

19 May 2015 at 5:39 am

Hello,

this is my script

```
<script>function setToRed ( ) { document.getElementById("moondoge").style.backgroundColor
= "#FF0000";
  setTimeout ( "setToBlack()", 10000 );
} function setToBlack ( )
{ document.getElementById("moondoge").style.backgroundColor= "#009933";
}
</script>

<script>function setToRed ( ) { document.getElementById("freedoge").style.backgroundColor =
"#FF0000";
  setTimeout ( "setToBlack()", 20000 );
```

} function setToBlack ( )
{ document.getElementById("freedoge").style.backgroundColor= "#009933";
}
</script>
and this is my html

<a href="http://moondoge.co.in/" target="_blank">
<input type="button" name="clickMe" id="moondoge" value="Moon Doge"
onclick="setToRed()"/>

<a href="https://freedoge.co.in" target="_blank">
<input type="button" name="clickMe" id="freedoge" value="Free Doge" onclick="setToRed()"/>
After click on a moondoge button the second button changes the color into red and not the moondoge.

Any suggestions?

Thanks in advance

Reply
chrishirst says

19 May 2015 at 5:05 pm

Just look at it logically, you have two function sets, BOTH sets of functions have the same names, which is going to be called when you “click” one of the buttons?

Reply
frikson says

19 May 2015 at 6:36 pm

Hello Chris, thanks for your replay.
I’m sorry, but I’m a totally new in this script/html language. I know that it sound crazy, but I really don’t know what you mean.
I try to add 1 at the end of function setToRed1( ) …

<script type="text/javascript">
<script>function setToRed( ) { document.getElementById("moondoge").style.backgroundColor
= "#FF0000";
  setTimeout ("setToBlack()", 10000);
} function setToBlack ( )
{ document.getElementById("moondoge").style.backgroundColor= "#009933";

```
}
</script>

<script>function setToRed1(  ) {
document.getElementById("freedogecoin").style.backgroundColor = "#FF0000";
  setTimeout ( "setToBlack()", 10000 );
} function setToBlack (  )
{ document.getElementById("freedogecoin").style.backgroundColor= "#009933";
}
</script>
```
html code

```
<a class="button" href="http://moondoge.co.in/"  target="_blank"><input type="button"
name="clickMe" id="moondoge" value="Moon Doge" onclick="setToRed( )"/>

<a href="http://freedoge.co.in"  target="_blank">
<input type="button" name="clickMe" id="freedogecoin" value="Free Doge Coin"
onclick="setToRed1( )"/>
```
First button doesn’t change the color at all, the second button Free Doge Coin works perfectly.

When I try to add the 3rd button

```
<script>function setToRed2(  ) { document.getElementById("dogefree").style.backgroundColor =
"#FF0000";
  setTimeout ( "setToBlack()", 10000 );
} function setToBlack (  )
{ document.getElementById("dogefree").style.backgroundColor= "#009933";
}
</script>
<a href="http://dogefree.net/"  target="_blank">
<input type="button" name="clickMe" id="dogefree" value="Doge Free" onclick="setToRed2(
)"/>
```
The second button changes the color into red, but after 10 sec. the 3rd button changes color to green and not the second one. The second button is red all the time, now the 3rd button works perfectly.
I’m totally confuse.

I have 38 buttons to set to work this way.
Could you, please, correct my code/function set to work perfectly, I will be very grateful.

Reply
froopy says

20 May 2015 at 11:29 pm

Just use one set of functions, one to red and one to black. Pass the name of the buttons when you call the function.

```
<script>
var dogebutton="";
function setToRed (dogebutton) {
document.getElementById(dogebutton).style.backgroundColor = "#FF0000";
setTimeout(function(){ setToBlack(dogebutton) }, 10000);
}
function setToBlack (dogebutton){
document.getElementById(dogebutton).style.backgroundColor= "#009933";
}
</script>



<a href="http://froopysnotebook.com/"  target="_blank">
<input type="button" name="clickMe" id="moondoge" value="Moon Doge"
onclick="setToRed('moondoge')"/>

<a href="http://froopysnotebook.com/pr"  target="_blank">
<input type="button" name="clickMe" id="freedoge" value="Free Doge"
onclick="setToRed('freedoge')"/>
```

Reply
frikson says

21 May 2015 at 11:51 am

Many many thanks for your reply, froopy, it works perfectly.

I just forgot to mention that I have different time interval for different buttons; 90 sec. for Moon Doge, 60 sec. for Free Doge, 10 min. for …
I have 10 buttons with 90 sec. interval,
20 buttons with 60 sec. interval,
15 buttons with 10 minutes interval, …

I guess that I have to add some additional function and I try it, but it’s not working.
Could you help me with this, please!

Reply

chrishirst says

21 May 2015 at 5:25 pm

You pass values to ONE function as parameters, and those values determine which element they affect or manipulate and how they it.

I'm going to write a full answer with examples at my forum simply because it has much better editing tools and I can write a whole article rather than a cramped up forum post and I will provide a link to the article here.

With any luck it will be a quiet Friday and I'll had it done for late tomorrow.

I'll be back!

Reply
frikson says

21 May 2015 at 6:25 pm

Great, thanks in advance.

Have a nice day!

Reply
chrishirst says

22 May 2015 at 6:02 pm

Bit later than anticipated, but

http://webmaster-talk.eu/articles/8-website-development-and-design/45-javascript-functions-write-once-use-many

Reply
sajjadhira says

30 August 2017 at 6:24 am

This is an example of JavaScript automatic popup closing after 10 seconds with countdown,

This window will close automatically within 10 second(s).

```
<script type="text/javascript">



 function countdown() {

    var i = document.getElementById('counter');

    i.innerHTML = parseInt(i.innerHTML)-1;

 if (parseInt(i.innerHTML)<=0) {

  window.close();

 }

}

setInterval(function(){ countdown(); },1000);

</script>
```

I have found it into PHPAns Forum. Hope this will be helpful for you.

[Edited by sajjadhira on 30-Aug-17 06:25]

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

To include a block of code in your comment, surround it with <pre> ... </pre> tags. You can include smaller code snippets inside some normal text by surrounding them with <code> ... </code> tags.

Allowed tags in comments: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <s> <strike> <pre> .

Primary Sidebar
HIRE MATT!
Matt Doyle headshot
Need a little help with your website? I have over 20 years of web development experience under my belt. Let's chat!

SEND ME AN EMAIL

```
/* A Backtracking program in
Java to solve Sudoku problem */

class GFG
{

public static boolean isSafe(int[][] board,

                             int row, int col,

                             int num)
{

    // row has the unique (row-clash)

    for (int d = 0; d < board.length; d++)

    {

        // if the number we are trying to

        // place is already present in

        // that row, return false;

        if (board[row][d] == num)

        {

            return false;
```

```
    }
}

// column has the unique numbers (column-clash)
for (int r = 0; r < board.length; r++)
{
    // if the number we are trying to
    // place is already present in
    // that column, return false;

    if (board[r][col] == num)
    {
        return false;
    }
}

// corresponding square has
// unique number (box-clash)
int sqrt = (int) Math.sqrt(board.length);
int boxRowStart = row - row % sqrt;
int boxColStart = col - col % sqrt;
```

```
    for (int r = boxRowStart;
            r < boxRowStart + sqrt; r++)
    {
        for (int d = boxColStart;
                d < boxColStart + sqrt; d++)
        {
            if (board[r][d] == num)
            {
                return false;
            }
        }
    }

        // if there is no clash, it's safe
    return true;
}

public static boolean solveSudoku(int[][] board, int n)
{
    int row = -1;
    int col = -1;
    boolean isEmpty = true;
```

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        if (board[i][j] == 0)
        {
            row = i;
            col = j;

            // we still have some remaining
            // missing values in Sudoku
            isEmpty = false;
            break;
        }
    }
    if (!isEmpty)
    {
        break;
    }
}
```

```
// no empty space left
if (isEmpty)
{
    return true;
}

// else for each-row backtrack
for (int num = 1; num <= n; num++)
{
    if (isSafe(board, row, col, num))
    {
        board[row][col] = num;
        if (solveSudoku(board, n))
        {
            // print(board, n);
            return true;
        }
        else
        {
            board[row][col] = 0; // replace it
        }
    }
```

```
    }

    return false;
}



public static void print(int[][] board, int N)
{

    // we got the answer, just print it

    for (int r = 0; r < N; r++)

    {

        for (int d = 0; d < N; d++)

        {

            System.out.print(board[r][d]);

            System.out.print(" ");

        }

        System.out.print("\n");



        if ((r + 1) % (int) Math.sqrt(N) == 0)

        {

            System.out.print("");

        }

    }
}
```

```
// Driver Code

public static void main(String args[])
{


    int[][] board = new int[][]

    {

            {3, 0, 6, 5, 0, 8, 4, 0, 0},

            {5, 2, 0, 0, 0, 0, 0, 0, 0},

            {0, 8, 7, 0, 0, 0, 0, 3, 1},

            {0, 0, 3, 0, 1, 0, 0, 8, 0},

            {9, 0, 0, 8, 6, 3, 0, 0, 5},

            {0, 5, 0, 0, 9, 0, 6, 0, 0},

            {1, 3, 0, 0, 0, 0, 2, 5, 0},

            {0, 0, 0, 0, 0, 0, 0, 7, 4},

            {0, 0, 5, 2, 0, 6, 3, 0, 0}

    };

    int N = board.length;


    if (solveSudoku(board, N))

    {

        print(board, N); // print solution

    }
```

```
    else

    {

        System.out.println("No solution");

    }
}
}


// This code is contributed
// by MohanDas
```