Loops/Confirm Tutorial:

What you've learned so far:

- 3 ways to call a function
- how to write a function
- how to send values into parameters in a function
- How to create an array (of pictures, of sentences, of numbers, etc.)
- How to change an image (its src, its width, height, alt, etc.)
- How to change text (with innerHTML)
- how to generate a random number
- how to go through an array (using a count variable)
- how to use an if condition to control what code is executed in a function

In this tutorial, you'll learn 2 more things: How to loop, and how to use the confirm box to make your web page more interactive.

setTimeout() for Looping

What the setTimeout() function does:

setTimeout() is a built-in javascript function that in essence causes the JavaScript code to pause working for a certain amount of time, and then call a function to restart it. No matter what the JavaScript code is doing, when it hits the line that calls setTimeout(), it pauses. How long does it pause? As long as we tell it to. One of the parameters we send into the setTimeout() function is the number of milliseconds we want the javascript to pause for. So the call looks like this:

```
setTimeout("Function1()",5000)
```

the pause will be for 5000 milliseconds (or 5 seconds). Then after the pause, setTimeout() will call the function "Function1()".
Here is an example of using setTimeout() to change the color of a button back to black after 2000 msec. Try it:

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

function setToRed ( )
{
  document.getElementById("colorButton").style.color = "#FF0000";
  setTimeout ( "setToBlack()", 2000 );
}

function setToBlack ( )
{
  document.getElementById("colorButton").style.color = "#000000";
}

</script>

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

If this is working correctly, the button should turn red, and then after 2000 msec it should turn back to black.

You can use setTimeout() to make a slide show that automatically cycles through each image in an array, pause for a few seconds, then go to the next. So you've already written a JavaScript that cycles through each image in an array. The following code should look familiar:

```
<script type = "text/javascript">
           imgArray = new Array()
           imgArray [0] = "kittyfur-back.jpg"
           imgArray [1] = "kittyhimself.jpg"
           imgArray [2] = "kittybreakfast.jpg"
           imgArray [3] = "kittyno-regrets.jpg"
           imgArray [4] = "kttylemon.jpg"
           imgArray [5] = "kittypanda.jpg"

           arraycounter = 0

           function ChangePic()
          {
               document.images["kittypic1"].src = imgArray[arraycounter]
               arraycounter = arraycounter + 1
               if (arraycounter == imgArray.length)
               {
                      arraycounter = 0
               }
          }
 </script>
```

We'll modify this by adding a call to a function setTimeout().
For our slide show, I'll make the timeout last for 3000 milliseconds, and when it's over, I want to call the ChangePic() function again so that the next picture comes up. To do that, at the end of the ChangePic function (from page 1), I'll add:

```
setTimeout("ChangePic()",3000)
```

So my function now looks like this:

```
function ChangePic()
{
   document.images["kittypic1"].src = imgArray[arraycounter]
   arraycounter = arraycounter + 1
   if (arraycounter == imgArray.length)
   {
      arraycounter = 0
   }
   setTimeout("ChangePic()",3000)
 }
```

Now what happens is whenever the ChangePic() function is called, it sets the image with the id "kittypic1"'s source to whatever is in the imgArray[arraycounter]. I then add one to arraycounter and check to see if the arraycounter is equal to the length of the imgArray. If it is, I reset the arraycounter back to 0. And now I pause the javascript for 3000 msec. When 3000 msec is up, the function ChangePic() is started over from the beginning. Make sense?

The only thing left is to add an image in my web page with the id of "kittypic1".

```
<body >
<img src = "kittyjumping.jpg" width = "500" height = "250" alt = "a pic of a kitty
jumping" id = "kittpic1" onClick = "ChangePic()"/>
</body>
```

Below is another example: In this case clicking on the paragraph "Click here to start" will call the function myfunc1() . Inside myfunc1(), if count is less than 20, the onbulb.jpg picture will be displayed, count will increase, and setTimeout

will pause the code for 500 milliseconds, then call myfunc2(). In myfunc2, if the count is less than 20, theoffbulb.jpg picture will be displayed, the code will be paused, and then the myfunc1() function will be called. This will continue, back and forth, until the count is equal to 20. When that happens, none of the code inside the if condition will be done (including the setTimeout() function), and thus the loop will stop. The effect is a blinking light that blinks on and off 20 times total.

```
<script type = "text/javascript">
           count = 0
           function myfunc1()
           {
              if (count < 20)
              {    document.images['img1'].src = "onbulb.jpg"
                   count = count+1
                   setTimeout("myfunc2()",500)
              }
           }
           function myfunc2()
           {
              if (count < 20)
              {    document.images['img1'].src = "offbulb.jpg"
                   count = count + 1
                   setTimeout("myfunc1()",500)
               }
           }
       </script>
    </head>
    <body>
        <table align = "center">
            <tr><td width = "189" height = "267">
                   <img src = "" width = "189" height = "267"  id = "img1">
                </td>
                <td><p onClick = "myfunc1()">Click here to start</p>
            </tr>
        </table>
   </body>
```

Restarting loop?

In the above example, when you click on "Click here to start", the function myfunc1 will be called, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2, then myfunc1, then myfunc2 (see why it's called a loop?), and then, because count should now be 20, the loop stops. What if you want to restart the loop? You'd need to set the count value back to 0. Here's the problem. If you set it back to 0 in either of the functions, then every time you call the function, it will be reset to 0. To see what I mean, what if you had:

```
           function myfunc1()
           {  count = 0
              if (count < 20)
              {    document.images['img1'].src = "onbulb.jpg"
                   count = count+1
                   setTimeout("myfunc2()",500)
              }
           }
```

Now every single time you call myfunc1(), you'll reset count back to 0. So count will always be less than 20. So the loop will continue forever and ever. We don't necessarily want that. At the same time we do want to be able to restart the loop. The easiest way to do this is to create a third, initialization function.

```
<script type = "text/javascript">
           count = 0
           function init()
           {
```

```
            count = 0
            myfunc1()
         }

         function myfunc1()
         {
            if (count < 20)
            {    document.images['img1'].src = "onbulb.jpg"
                 count = count+1
                 setTimeout("myfunc2()",500)
            }
         }
         function myfunc2()
         {
            if (count < 20)
            {    document.images['img1'].src = "offbulb.jpg"
                 count = count + 1
                 setTimeout("myfunc1()",500)
             }
         }
      </script>
   </head>
   <body>
       <table align = "center">
           <tr><td width = "189" height = "267">
                  <img src = "" width = "189" height = "267"  id = "img1">
               </td>
               <td><p onClick = "init()">Click here to start</p>
           </tr>
       </table>
  </body>
```

Now every single time you click on the text "Click here to start", you'll call the init function, which will set count back to 0 and call myfunc1, which will call myfunc2, which will call myfunc1… 20 times, and then stop because count >= 20. To restart, you click on Click here to start, which will set count back to 0, and call myfunc1, which will call myfunc2… 20 times again. Every time you click on restart, you reset count back to 0 and start the loop over.

You can use the init function to reset other values as well. In the following code, the init function resets both the count and the total back to 0. Then the function getcards generates a random card (a number between 0 and 10), and adds that to the total. It pauses for 500 milliseconds, then calls getcards again and generates a second random card (number between 0 and 10), adds that to the total, pauses, and then calls getcards again. It does this a total of 5 times, generating 5 new cards and keeping track of their total value. After 5 cards are generated, it stops. To get a new hand, you must click on the "Click here to start" text, which calls the init function, which reinitializes both count and total to 0, and then calls getcards to loop 5 times and get a new hand.

```
<script type = "text/javascript">
    count = 0
    total = 0
    function init()
    {  count = 0
       total = 0
          getcards()
       }

       function getcards()
       {
          if (count < 5)
          {  num = Math.floor(Math.random() * 10)
             total = total + num
             document.getElementById('para1').innerHTML="you drew a "+num+", your total is now "+total
```

```
            count = count+1
            setTimeout("getcards()",500)
        }
    }
    </script>
 </head>
 <body>
     <p onClick = “init()”>Click here to get a card hand</p>
     <p id = “para1”>Hand will show here.</p>
</body>
```

Aside: onload

You can make a function start when you click on the paragraph with the text, “Click her to start”, but in some cases, you might want the code to start all on its own, without having to click on anything. Think of slide shows on web pages that seem to happen automatically. To do this, in the <body> tag you can use onload to call the function. It will look like this:

```
<body onload = "myfunc1()">
```

Now when the web page loads into the browser, it automatically calls the function myfunc1(). So now you’ve got yet another way to call a function.

While Loop for looping

So far you’ve used an if condition to control when we do particular statements. In an if condition, you say something like,

```
if (par1 == “cat”)
{   document.write(“<p>It’s a cat!</p>”)
}
```

In this case the document.write code is only executed when (par1 == “cat”) is true (meaning par1 does, in fact, hold “cat”).

A variant on this is a while loop. With a while loop, javascript ***continues to do the statements as long as the condition remains true***. So, for instance, in javascript a loop would look like this:

```
line 1.   countvar = 1;
line 2.   while (countvar < 5)
line 3.   {
line 4.           document.write(“<p>really </p>”);
line 5.           countvar = countvar + 1;
line 6.   }
line 7.   document.write(“<p> cute! </p>”);
```

In the above code:

line 1. the number 1 is first placed into the variable countvar.
line 2. while the value inside of countvar is less than 5, we do the code inside the { and }, which is:
line 3. defining the start of the code that will be done repeatedly until the countvar is >= to 5
line 4. the word “really” is written to the html document
line 5. the value inside the countvar increases by 1 (to hold 2).
line 6. This is the end of the code that is done repeatedly until countvar is >= to 5.

At this point, we go back to line 2 and check to see: is the value in countvar < 5? If it is (if this condition is true), we do lines 3 through 6 again, so the word “really” will be written again, and the countvar will change to 3.

Again, we go back to line 2 (remember, it’s a loop. It happens over and over again, until countvar >= to 5). Again we check, and find that countvar is still less than 5, so we do lines 3 through 6 again, the word really is written out again, and countvar becomes 4.

Again, we check. Is countvar <5? Yes, so we write “really”, and change countvar to 5

We check. is countvar < 5? NO! countvar now is equal to 5. So we don’t do the code in lines 3-6 again. We now skip down to line 7
line 7. When the loop is done and the loop’s condition is no longer true, we write the word “cute!” to the web page.

That’s a loop. Whatever code is between the { and the } will happen again and again until the condition is no longer true. Try it:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
      <title> First Javascript </title>
      <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
      <script type = "text/javascript">

        function JS()
        {   document.write("<html><body>”)
             countvar = 0
            while (countvar < 5)
            {     document.write("<p> really </p>");
                  countvar = countvar + 1;
            }
            document.write("<p> cute </p>");
           document.write("</body></html>”)
      }
      </script>
</head>
<body>
<p id = "here" onClick = "JS()">Click to see the secret message </p>

</body>
</html>
```

Here is another example. In this case, the loop continues while continuevar holds true. It will stop happening when continuevar holds false:

```
<html>
<head>
   <script type = "text/javascript">
       function loopfun()
       {
          continuevar = true
          while (continuevar == true)
          {
             document.write("<p> Because </p>");
             continuevar = confirm("But Why?")
          }
          document.write("<p> Because I said so and that's final! </p>");
        }
      </script>
   </head>
   <body>
        <p onclick = "loopfun()">Why?</p>
   </body>
</html>
```

In this case, we’re using the confirm box to get input from the user of your function. What is a confirm box? A confirm box is a pop-up box that asks the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

- If the user clicks "OK", the box returns true.
- If the user clicks "Cancel", the box returns false.

Example:

```
<html>
<head>
<script type = "text/javascript">
function myFunction()
{
   r=confirm("Press a button!");
   if (r==true)
   {
      document.getElementById("demo").innerHTML="You pressed OK!";
   }
   else
   {
      document.getElementById("demo").innerHTML="You pressed Cancel!";
   }
}
</script>
</head>
<body>
<p onclick="myFunction()">Click to display a confirm box </p>
<p id="demo"></p>

</body>
</html>
```

When you learned about setTimeout(), you generated a card hand of 5 cards. You can do the same thing using a while loop. Notice that in this case you can initialize the count to 0 and the total to 0 within the function, because it is outside of the while loop. Inside the loop, you must increase count each time through. You're also generating a new card, and adding its value to the total each time through, as well as printing it out. But in this case, the while loop goes so fast, you'll only see the results, not each card as you generate it. This does let you click again and again, and each time you'll generate a new hand.

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

       function getcards()
       {  count = 0
          total = 0
          while (count < 5)
          {  num = Math.floor(Math.random() * 10)
             total = total + num
             document.getElementById('para1').innerHTML="you drew a "+num+", your total is now "+total
             count = count+1
          }
       }
       </script>
    </head>
    <body>
        <p onClick = "getcards()">Click here to get a card hand</p>
        <p id = "para1">Hand will show here.</p>
   </body>
</html>
```

That's it! You've learned loops!