More cool scripts:

Slide Show:

So far the user has controlled the action. However, you can make a slide show that automatically cycles through each image in an array and goes to the next image at an incremental time. You do this by adding a call to a "setTimeout()" function (already written by Javascript) that basically does nothing for a set amount of time, then calls your first function over again. 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>
```

First, we'll modify this by adding a call to a function setTimeout().

What the setTimeout() function does:

setTimeout() is a function that in essence causes the JavaScript code to pause working. 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 too. One of the "parameters" we send into the setTimeout() function is the number of milliseconds we want it to pause for. So the call looks like this:

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

the pause will be for 5000 milliseconds. 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 5000 msec. Try it:

```
<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()"/>
```

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 so that the next picture comes up. To do that, at the end of the ChangePic function, 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". I could make the slideshow start when I clicked on the image associated with the id "kittypic1" (you know how to do this, right?), but I think in this case I would like the slideshow to start as soon as the web page loads. So I am going to add to the <body> tag an event handler for onload. It will look like this:

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

Now when my web page loads into the browser, the function ChangePic() will be called and the slideshow will start.

Your Turn: Make a slide show. Put together the JavaScript at the top, with an array, a counter, and a change image function, then add the setTimeout() call. Make sure there's an image in your web page with the correct id, and then add the onload event to the body.

Text for slideshow:

You may wish to cycle through text as you cycle through your slideshow (think of vacation pictures, each with their own text associated with them).

To do this, you will need to create an array of text. The array will need to be the exact same length as the array of images if you want them to synch up. If you don't, it doesn't matter how long the text array is. For now, I'll make them be the same length:

```
textArray = new Array()
textArray[0] = "picture of grumpy cat and a big pile of fur"
textArray[1] = "baby kitten wanting to hold bottle of milk"
textArray[2] = "cat looking at clock and demanding breakfast"
textArray[3] = "kitty has gone through a whole roll of toilet paper"
textArray[4] = "kitty with a very sour look on his face because he ate a lemon"
textArray[5] = "Panda fell head-first off a slide"
```

Then inside the ChangePic() function, you need to add the code that changes the innerHTML of the id associated with the text you want to change. Thus if, in your code you've got

```
<p id = "piclabels">Text associated with pictures </p>
```

You'll add to your ChangePic() function:

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

That should do it. Now when you load your page, both the picture and the text below it should cycle through

Your turn: Make sure this works for you. Get a page with an image and text rotating through a slide show when the web page loads into the browser.

Changing Style

So far we've changed images and text, but we have only done a little with changing the style of a web page. JavaScript lets us do that as well. We can call a function that allows us to change the style of one or many elements on the web page. We can do that as we've been doing it, but instead, let's use a form's submit button.

First, create a web page with a submit button on it (remember from creating forms in html?):

```
<input type = "submit" id = "redbtn" value = "red" onclick = "ChangeStyle('red')" />
```

Next we create a function that changes the style of a particular element. For now, I am going to just change the background color of the entire body of the web page. So my function will look like this:

```
function ChangeStyle(colorholder)
{
   document.body.style.backgroundColor = "#FF0000"
}
```

Put it all together, and you get:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Our first javascript!</title>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />

        <script type = "text/javascript">

           function ChangeStyle(colorholder)
          {
               document.body.style.backgroundColor = "#FF0000"
          }
       </script>


    </head>
    <body>

     <input type = "submit" id = "redbtn" value = "red" onclick = "ChangeStyle('red')" />

    </body>
</html>
```

Your turn: Try this. Create a basic web page with a button that, when clicked on, turns the web page background color to red.

Different Colors:

Let's expand this. Let's create a bunch of buttons that allow us to change the background color to different colors. First, let's create some more buttons:

```
<input type = "submit" id = "redbtn" value = "red" onclick = "ChangeStyle('red')" />
<input type = "submit" id = "yellowbtn" value = "yellow" onclick = "ChangeStyle('yellow')" />
<input type = "submit" id = "greenbtn" value = "green" onclick = "ChangeStyle('green')" />
<input type = "submit" id = "cyanbtn" value = "cyan" onclick = "ChangeStyle('cyan')" />
<input type = "submit" id = "bluebtn" value = "blue" onclick = "ChangeStyle('blue')" />
<input type = "submit" id = "purplebtn" value = "purple" onclick = "ChangeStyle('purple')" />
```

Then let's modify the ChangeStyle() function so that when it is called with a particular variable, the body's background color changes to the appropriate color:

```
           function ChangeStyle(colorholder)
          {
               if (colorholder == "red")
               {
                  document.body.style.backgroundColor = "#FF3322"
               }
               else if (colorholder == "yellow")
               {
                  document.body.style.backgroundColor = "#FCEF04"
```

```
        }
        else if (colorholder == "green")
        {
           document.body.style.backgroundColor = "#22DE43"
        }
        else if (colorholder == "cyan")
        {
           document.body.style.backgroundColor = "#10FAE4"
        }
        else if (colorholder == "blue")
        {
           document.body.style.backgroundColor = "#4201D3"
        }
        else if (colorholder == "purple")
        {
           document.body.style.backgroundColor = "#B900CD"
        }
    }
```

Your turn: Modify your web page so that you've got a bunch of different buttons at the top, and each one changes the background color to a different color

Changing Style of other elements:

So far we've changed the background color of the web page. We can change the style of other elements on the page as well. Of course, we'll need other elements on the page.

First, create another element on the page. I'm going to add a header:

```
<body>


    <input type = "submit" id = "redbtn" value = "red" onclick = "ChangeStyle('red')" />
    <input type = "submit" id = "yellowbtn" value = "yellow" onclick = "ChangeStyle('yellow')" />
    <input type = "submit" id = "greenbtn" value = "green" onclick = "ChangeStyle('green')" />
    <input type = "submit" id = "cyanbtn" value = "cyan" onclick = "ChangeStyle('cyan')" />
    <input type = "submit" id = "bluebtn" value = "blue" onclick = "ChangeStyle('blue')" />
    <input type = "submit" id = "purplebtn" value = "purple" onclick = "ChangeStyle('purple')" />

    <h1 id = "firstheader"> Our Dynamic Style Page </h1>

</body>
```

Now, when a button is clicked, I'm going to alter the background color of the firstheader element as well as the background color of the entire page:

```
    function ChangeStyle(colorholder)
    {
        if (colorholder == "red")
        {
           document.body.style.backgroundColor = "#FF3322"
           document.getElementById('firstheader').style.backgroundColor = "#57AF33"
        }
        else if (colorholder == "yellow")
        {
           document.body.style.backgroundColor = "#FCEF04"
           document.getElementById('firstheader').style.backgroundColor = "#7788CC"
```

```
            }
            else if (colorholder == "green")
            {
               document.body.style.backgroundColor = "#22DE43"
               document.getElementById('firstheader').style.backgroundColor = "#0055FF"

            }
            else if (colorholder == "cyan")
            {
               document.body.style.backgroundColor = "#10FAE4"
               document.getElementById('firstheader').style.backgroundColor = "#E410FA"

            }
            else if (colorholder == "blue")
            {
               document.body.style.backgroundColor = "#4201D3"
               document.getElementById('firstheader').style.backgroundColor = "#F3F300"

            }
            else if (colorholder == "purple")
            {
               document.body.style.backgroundColor = "#B900CD"
               document.getElementById('firstheader').style.backgroundColor = "#BBAADD"

            }
      }
```

I'm going to modify some other style properties of the firstheader element by adding:

```
            document.getElementById('firstheader').style.fontSize = "45px"
            document.getElementById('firstheader').style.fontFamily = "arial"
            document.getElementById('firstheader').style.padding = "20px"
            document.getElementById('firstheader').style.fontStyle = "italic"
            document.getElementById('firstheader').style.textAlign = "right"
            document.getElementById('firstheader').style.borderStyle = "solid"
            document.getElementById('firstheader').style.borderWidth = "4px"
```

My function will now look like:

```
      function ChangeStyle(colorholder)
      {
            if (colorholder == "red")
            {
               document.body.style.backgroundColor = "#FF3322"
               document.getElementById('firstheader').style.backgroundColor = "#57AF33"
            }
            else if (colorholder == "yellow")
            {
               document.body.style.backgroundColor = "#FCEF04"
               document.getElementById('firstheader').style.backgroundColor = "#7788CC"
            }
            else if (colorholder == "green")
            {
               document.body.style.backgroundColor = "#22DE43"
               document.getElementById('firstheader').style.backgroundColor = "#0055FF"

            }
            else if (colorholder == "cyan")
            {
               document.body.style.backgroundColor = "#10FAE4"
               document.getElementById('firstheader').style.backgroundColor = "#E410FA"
```

```
            }
            else if (colorholder == "blue")
            {
               document.body.style.backgroundColor = "#4201D3"
               document.getElementById('firstheader').style.backgroundColor = "#F3F300"

            }
            else if (colorholder == "purple")
            {
               document.body.style.backgroundColor = "#B900CD"
               document.getElementById('firstheader').style.backgroundColor = "#BBAADD"

            }
            document.getElementById('firstheader').style.fontSize = "45px"
            document.getElementById('firstheader').style.fontFamily = "arial"
            document.getElementById('firstheader').style.padding = "20px"
            document.getElementById('firstheader').style.fontStyle = "italic"
            document.getElementById('firstheader').style.textAlign = "right"
            document.getElementById('firstheader').style.borderStyle = "solid"
            document.getElementById('firstheader').style.borderWidth = "4px"

      }
```

Your Turn: Add this to your web page. Cool, huh. Feel free to play around.

Assignment: Think about your average web page. Then think about adding a button that would adapt the page for those with red-green or blue-yellow color blindness. What about those with low vision who need a web page with high contrast?

Loop Code for selecting a background image:

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

```
if (var1 == "cat")
{
   document.write("<p>It's a cat!</p>")
}
```

In this case the document.write code is only executed when (var1 == "cat") is true (meaning var1 does, in fact, hold "cat"). A variant on this is a loop. With a 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:

```
countvar = 1;
newstring = "";
while (countvar < 5)
{
        document.write("<p>really </p>");
        countvar = countvar + 1;
}
document.write("<p> cute! </p>");
```

In the above code, the number 1 is first placed into the variable countvar. Then, while the value inside of countvar is less than 5, javascript will execute the code between { and } . Inside the loop, the word "really" is written to the html document and the value inside the countvar increases by 1 (to hold 2). Javascript then checks again: is the value in countvar < 5? If it is (if this condition is true) javascript writes the word "really" to the document again, and the value inside the countvar increases to 3. JavaScript checks again: Is the value in countvar < 5? JavaScript continues to execute the code between { and } until the condition (countvar < 5) is false (the value inside of countvar is no longer less than 5). 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(countvar)
        {
            countvar = parseInt(countvar);
            while (countvar < 5)
            {
                  document.write("<p> really </p>");
                  countvar = countvar + 1;
            }
            document.write("<p> cute </p>");
      }
      </script>
</head>
<body>


<p id = "here" onClick = "JS('1')">Click to see the secret message </p>

</body>
</html>
```

Question: (DO NOT TRY THIS!!!) What would happen if you didn't include the line "countvar = countvar + 1" inside the loop?

Note 2: The line, "countvar = parseInt(countvar)" converts a string to a number. In computer science, the numbers are located in one place and characters are located in another. so 5 actually would occur twice, once in the number area and once in the character area. To make absolutely sure we're using the 5 from the number area, we use "ParseInt() to convert the 5 in the character area into the 5 in the number area.

Another example of a loop would be:

```
continuevar = true
while (continuevar == true)
{
     count = count + 1
     document.write("<p>Because </p>")
     continuevar = confirm("But Why?")
}
```

In this case, the loop (all the code between { and } will continue to happen as long as the value inside of continuevar is true. So the value inside continuevar will change whenever the confirm box pops up and the user either hits "ok" (which is true), or "cancel" (which is false). Try this:

```
<!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>
```

```
        <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>
```

Now let's put this all together. You're going to write a javascript that cycles through potential pictures for a background image on your page. When the user decides on an image (By clicking cancel when asked, "Do you want to see another picture?") the current image being displayed becomes the background image.

```
<html>
<head>
<script type = "text/javascript">
      ls = new Array()
      ls[0] = "bg1.jpg"
      ls[1] = "bg2.jpg"
      ls[2] = "bg3.jpg"
      ls[3] = "bg4.jpg"
      ls[4] = "bg5.jpg"
      ls[5] = "bg6.jpg"
      count = -1
      function loopforbackground(idholder, idholder2)
      {
        document.body.style.backgroundImage = ""
        document.getElementById(idholder2).innerHTML = " "
        continuevar = true
            while (continuevar == true)
        {
            count = count + 1
            if (count == ls.length)
            {
               count = 0
            }
            document.images[idholder].src = ls[count]
            continuevar = confirm("Do you want to see another picture?")
        }
          document.body.style.backgroundImage = "url(\'" + ls[count] + "\')"
          document.getElementById(idholder2).innerHTML = "click here for other background options"
      }
    </script>
  </head>
  <body>
```

```
    <img width = "500" height = "374" alt = "area in which pics will be displayed" id = "display" />
    <h3 id = "firsttext" onClick = "loopforbackground('display','firsttext')">
          Click here to see potential pics... </h3>
  </body>
</html>
```

Note: What's going on here?

```
document.body.style.backgroundImage = "url(\'" + ls[count] + "\')"
```

Let's break it down. First, we're changing the body of the html document's background image. To change a background image, normally you'd use "url('backgroundpic.jpg')" In this case, the background image is inside the array at ls[count]. If we did this:

bad: `document.body.style.backgroundImage = "url('ls[count]')"`

Javascript would try to set the background to an image called "ls[count]" (and not what is inside ls[count]). Clearly that would not work because there is no image called "ls[count]". So what we do instead is join together three different things: The string "url(' " and ls[count] and " ')" We join these three strings together using the + sign. At this point we'd have:

better, but still won't work: `document.body.style.backgroundImage = "url(' " + ls[count] + " ')"`

We're almost there, but the problem is the ' is a special character in javascript that indicates a quote. To show that we want the ' to be included in the string (and not an indicator of the beginning or ending of a string), we use the \ before it. Then Javascript just includes it in the string of characters. We end up with:

good: `document.body.style.backgroundImage = "url(\' " + ls[count] + " \')"`

Project 2:

1. Magic 8 (sort of):

You're going to create a very odd version of the Magic 8 ball that sort of plays itself. With the physical magic 8 ball, someone asks a question and shakes a ball. The response that pops to the top of the ball is the answer to the question. So, for instance, the user might ask, "Am I going to have 20 kids with the man of my dreams?" and shake the ball, and an answer that might pop up would be, "It appears so". For our automatic magic 8 ball game, you will create an array with a list of potential answers (like, "Certainly", "No Way!", "It doesn't look good", etc.). You should also create an array (of equal length) of images that reflect the answer (a smiling face for "Certainly", a dejected face for "It doesn't look good", etc.) . You will also need to create an array of questions. This array MUST NOT be the same length as the length of the array of answers.

So far so good?

You will also need 2 counters, one for the array of questions, and one for the array of answers (we will also use this same counter for the array of pictures).

Now you will write a function that generates a random number between 0 and the length of the array of questions(remember how to generate a random number?), and uses that random number to display the question in the array associated with that random number. Your function will then generate another random number between 0 and the length of the array of the answers. It will use that random number to display both the text answer and the image associated with that random number.

Because this is a very bizarre Magic 8 ball game, your function will continue to generate new questions and new answers every 3000 milliseconds, and will start when the web page loads into the browser (<body onload =…) (You may need to make the time longer than 3000 milliseconds –that's up to you).

Make sure the html code in the web page includes a place for the question to be displayed, and for the answer and its associated image to be displayed.

(An equally fun assignment would be to have an array of head images, a different length array of body images, and an even different length array of leg images, then have a function that displayed a random head on top of a random body on top of a random set of legs. Unfortunately, it requires hunting down a lot more images. If you want to do this instead, though, that's acceptable).

2. BlackJack (sort of)

Overall Game: You're going to write a web page in which you get asked if you want another card continuously, and, if you say yes, a new card will be displayed in a different place on the web page. You're trying to make the cards come as close to but not go over 21.

Steps:

1. Create an array of images. The images (pictures) should be of the numbers 1 through 11
   (Note: the image of number 1 will go in the array at 0 because we always start our arrays at 0)
2. Write an html page with space for 12 images (probably a table with images in it.) Each image should have a unique id.
3. Create a counter variable.
4. Write a function GetCards()
5. Inside GetCards, write a confirm asking, "do you want a card".
6. Write a loop. Your loop will continue while the answer from confirm is true
7. Inside your loop, generate a random number between 0 and the length of your image array.
8. Slightly challenging part:
    1. Now (still inside your loop), you will create an if statement, that goes something like this:
    2. If the counter is equal to 0, use the random number generated to display the image from the array (like we've done before). You will display the image in the first id on the web page.
    3. If the counter is equal to 1, you'll display the image in the second id on the web page.
    4. If the counter is equal to 2, you'll display the image in the third id on the web page,
    5. …
    6. If the counter is equal to 11, you'll display the image on the twelfth id on the web page.
9. Now you will use the confirm to ask, "do you want another card".
10. End your loop
11. End your function

Extra Credit (5 pts): Keep track of the total number generated by adding each random number generated together.

If the number goes over 21, modify text on the page to say, "You lose!"

Extra Extra Credit (10 pts): After the loop, but before the end of the function, generate a random number between 12 and 25 (The dealer's hand). If the dealer's random number is >21, modify text on the page to say, "You win!" Otherwise, check to see whose value is closer to 21 and modify text to report who won, accordingly.