Arrays/Branching Statements Tutorial:

In the last tutorial, you created a button that, when you clicked on it (the onclick event), changed another image on the page.

What if you have a series of pictures and you want to view the next picture in the series every time you click on the button? You need some structure that will hold on to all the names of each of the pictures you want to view, and then you need to be able to access each of the pictures in the structure. We can do this using Arrays.

Arrays are variables, or boxes, that hold more than one item. If you think of a variable as a box that holds something, and we use what's in the box by referring to the name of the box, then you can think of an array as one big long box with cubbyholes in it.

0	1	2	3	4	5	6	7
“pic1.jpg”	“pic2.jpg”	“pic3.jpg”	“pic4.jpg”	“pic5.jpg”	“pic6.jpg”	“pic7.jpg”	“pic8.jpg”

arrayholder

Now you can put things into each of the compartments in the Array. When you want to refer to one of the items in the array, you just refer to which cubbyhole of the Array it's in.

WARNING: Computer scientists always start counting at 0. (It has to do with space – if you want more of an explanation, ask me). So to refer to what is in the first cubbyhole in the Array arrayholder, I'd use arrayholder[0].

Creating an Array:

To create an Array, you must first tell the browser that you're creating an Array. You can call my Array anything that starts with a letter and has no special characters (including spaces). In this case I'll call the Array I'm creating arrayholder:

```
arrayholder = new Array()    // we just created a brand new array, and we called
                             // it arrayholder.  Note the space between new and
                             // Array()!! People tend to forget that space.
```

Now you can put things into my Array. Most of the time we purposely put things into an Array in order, largely so the array doesn't have a bunch of empty boxes and so we can find items more easily. To put the names of different images into the array, you'd do the following:

```
arrayholder[0] = “pic1.jpg”
arrayholder[1] = “pic2.jpg”
arrayholder[2] = “pic3.jpg”
arrayholder[3] = “pic4.jpg”
arrayholder[4] = “pic5.jpg”
arrayholder[5] = “pic6.jpg”
arrayholder[6] = “pic7.jpg”
arrayholder[7] = “pic8.jpg”
```

Now to refer to one of the pictures in the array, you'd do this:

document.images[idholder].src = arrayholder[2]

So to put this together in your code, you'd do the following:

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

  arrayHolder = new Array()
  arrayHolder[0] = "pic1.jpg"
  arrayHolder[1] = "pic2.jpg"
  arrayHolder[2] = "pic3.jpg"
  arrayHolder[3] = "pic4.jpg"
  arrayHolder[4] = "pic5.jpg"

  function ChangePicToArray(idholder)
  {
      document.images[idholder].src = arrayHolder[3]
  }

</script>
</head>
<body>

   <img src = "mypic.jpg" id = "thispic" height = "164"
          width = "194">
   <img src = "button.gif" id = "buttonpic" height = "20"
          width = "50" onClick = "ChangePicToArray('thispic')" />
</body>
```

Your Turn: Create an html page with a javascript that contains an array of images. Include in the body of your web page a picture and a button. When you click on the button (onClick), it should call the function you wrote to display a different image in the array (your choice of which image). In other words, make the above code work.

First thing you'll notice is that we put the array outside of the function, but inside the script. That's for efficiency and convenience, and conceptually it has to do with how your computer actually works. Every time a function is called, the parameters and variables inside it are created in memory (e.g., your parameter box and variable box magically comes into existence in specific space inside your computer) and they leave memory (pouf! They disappear) when the function is done running. An array is a whole bunch of cubbyholes. It is somewhat inefficient to make all those cubbyholes every time the function is run, and then they disappear so you have to remake them again the next time you run the function.

In our code, whenever you call a function, the code inside it runs. So every time we click on the button, we're calling the function ChangePicToArray(). Now when we call it, we want the picture's src to change, so that code needs to be inside the function. However, once we've created the array, we don't necessarily need to recreate the array every time we call the function. We can create it once, and then whenever we call the function, the function can use the array, but doesn't need to recreate the array. If, however, we'd put the creation of the array inside the function, then every time the function was called, the array would be recreated. That would be a waste of computer time.

Anything created outside of a function is available inside the function. The reverse is not true. If we created something inside a function, it would not be available for use outside the function. So, for instance, if we had another function that needed to use the same array, it can use it because the array was created outside of all functions.

Index Variable:

At this moment, this function probably doesn't do exactly what you were hoping it would do. You probably wanted the function to display the next image in the array every time you click on the button, so it would cycle through the pictures in the array.

We can do this using variable that holds a single value, this one holding a number that increases every time we click on the button. Then we can see the next image in the array every time we click on the button.

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

  arrayHolder = new Array()
  arrayHolder[0] = "pic1.jpg"
  arrayHolder[1] = "pic2.jpg"
  arrayHolder[2] = "pic3.jpg"
  arrayHolder[3] = "pic4.jpg"
  arrayHolder[4] = "pic5.jpg"

  indexholder = 0             // a variable, or box we named indexholder, set to
                              // hold 0 to start with…

  function ChangePicToArray(idholder)
  {
      document.images[idholder].src = arrayHolder[indexholder]
      indexholder = indexholder + 1      // adding 1 to the current value inside
                                         // the indexholder variable (box)
                                         // This means that it will start at 0.
                                         // When the function is called, it adds
                                         // 1 to that value, so now the value
                                         // inside indexholder is 1.  The next
                                         // time the function is called, it will
                                         // add 1 to the value inside indexholder
                                         // so indexholder will now hold 2.  Next
                                         // time the function is called, it adds
                                         // 1 to the value in indexholder, so the
                                         // value becomes 3, etc.
  }

</script>
```

Your Turn: Add a counting variable to your code. Click the button on your web page multiple times to see what happens. Does it do what you think it should do? Do you understand why?

Notice how now when you click on the button, the first image in the array appears. Then when you click on the next button, the next image appears. This is because indexholder starts out holding 0. Then when we click on the button, ChangePicToArray () is called, the document.images[idholder].src is changed to arrayHolder[0] (because indexholder holds 0). And then we add 1 to indexholder, so now indexholder holds 1.

Next time through document.images[idholder].src is changed to arrayHolder[1] (because indexholder holds 1). And then we add 1 to indexholder, so now indexholder holds 2. This continues until we've seen all the images in the arrayHolder array.

If (Branching) Condition:

Maybe what you'd want is, when we hit the end of the array, to start back over at the beginning. In English, you might say something like,

If the index is at the last box in the array or past it, reset the index to 0.

We say something similar in JavaScript:

```
if (indexholder >= arrayHolder.length)
{
      indexholder = 0
}
```

Now this says that if the value in indexholder is equal to the length of the array (length is the number of elements in the array), then reset the value in indexholder back to 0.

Altogether, your code will now look like:

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

  arrayHolder = new Array()
  arrayHolder[0] = "pic1.jpg"
  arrayHolder[1] = "pic2.jpg"
  arrayHolder[2] = "pic3.jpg"
  arrayHolder[3] = "pic4.jpg"
  arrayHolder[4] = "pic5.jpg"

  indexholder = 0

  function ChangePicToArray(idholder)
  {
      document.images[idholder].src = arrayHolder[indexholder]
      indexholder = indexholder + 1
      if (indexholder == arrayHolder.length)
      {
         indexholder = 0
      }
  }
</script>
</head>
<body>

   <img src = "mypic.jpg" id = "thispic" height = "164" width = "194">
   <img src = "button.gif" id = "buttonpic" height = "20" width = "50"
                onClick = "ChangePicToArray('thispic')" />
</body>
```

Your Turn:

1. Add the if (branching) statement to your code so that you can continuously click through the images and the images will loop around.

2. Add 2 more images to the end of your arrayHolder array. Before you run it, what do you think will happen now? Will those two images show up? Why or why not?

(To be continued)