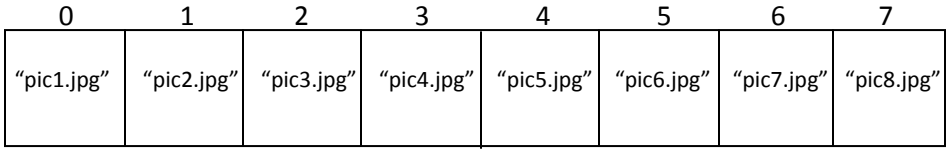
Arrays/Branching Statements Tutorial:

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 like variables, only they 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. Like variables, we can name arrays just about anything we want (must be only letters and numbers, must start with a letter… the usual rules that you know by now about naming things).



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

Creating an Array:

To create an Array, I must first tell the browser I'm creating an Array. In this case I'll call the Array I'm creating arrayholder: (Note that this is new. We're actually creating the array before we use it.)

Note 2: You must put a space between the word new and the word Array();

```
arrayholder = new Array();
```

Once the array has been created we'll want to put things into my Array. Remember how we put things in to variables? One way was to pass things in when we called a function with parameters (variables). The other was to assign a value to the variable using the =. We did this:

```
aNumVar = 4;
```

```
aStringVar = "kittypic.jpg";
```

That put the value 4 into aNumVar and "kittypic.jpg" into aStringVar. We'll use this method to put values into the Array we created. Remember, an array has more than one cubbyhole in which we can put something. So how do we know which cubbyhole we're putting something in? We number the cubbyholes. Most of the time we purposely put things into an Array in order, largely so the array doesn't have a bunch of empty cubbyholes and so

we can find items more easily. To put the names of different images into the array we created, we'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, I can do this:

document.images["kit1"].src = **arrayholder[2];**

The image with the id of "kit1" just got what is in arrayholder's cubbyhole #2, or "pic3.jpg". Does this make sense to you?

So to put this together in our code, we'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.

First thing you'll notice is that we put the array outside of the function, but inside the script. That makes the array available to all functions whenever we want to use the array. In other words, if we had 3 or 4 functions, every one of them could use the array we created. We can create it once, and then whenever we call any of the functions, it can use the array.

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. If, however, we created an array inside a function, then other functions that might need to use that same array can't use it.

Good

```
<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];
    }
    function ChangetoAnotherPic(idholder)
    {
        document.images[idholder].src = arrayHolder[1];
    }
</script>
</head>
<body>
   <img src = "mypic.jpg" id = "thispic" height = "164"
    width = "194“ onClick = “ChangetoAnotherPic(‘thispic’)”/>
   <img src = "button.gif" id = "buttonpic" height = "20“
    width = "50“ onClick = "ChangePicToArray('thispic')" />
</body>
```

Bad

```
<head>
<script type = "text/javascript">
    function ChangePicToArray(idholder)
    {
        arrayHolder = new Array();
        arrayHolder[0] = "pic1.jpg";
        arrayHolder[1] = "pic2.jpg";
        arrayHolder[2] = "pic3.jpg";
        arrayHolder[3] = "pic4.jpg";
        arrayHolder[4] = "pic5.jpg";
        document.images[idholder].src = arrayHolder[3];
    }
    function ChangetoAnotherPic(idholder)
    {
         document.images[idholder].src = arrayHolder[1];
    }
</script>
</head>
<body>
   <img src = "mypic.jpg" id = "thispic" height = "164"
    width = "194“ onClick = “ChangetoAnotherPic(‘thispic’)”/>
   <img src = "button.gif" id = "buttonpic" height = "20“
    width = "50“ onClick = "ChangePicToArray('thispic')" />
</body>
```

Variable used as an index:

That said, as of this moment, this function probably doesn't do exactly what you wanted it to do. You probably wanted the function to display the next image in the array every time you click on the button.

We can do this using another variable, 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";

  indexnum = 0;

  function ChangePicToArray(idholder)
  {
      document.images[idholder].src = arrayHolder[indexnum];
      indexnum = indexnum + 1;
  }

</script>
```

Step By Step:

```
<script type = "text/javascript">
     arrayHolder = new Array();
     arrayHolder[0] = "relaxed11.jpg";
     arrayHolder[1] = "relaxed6.jpg";
     arrayHolder[2] = "relaxed9.jpg";
     arrayHolder[3] = "relaxed4.jpg";
     arrayHolder[4] = "relaxed5.jpg";
     indexnum = 0;
     function
ChangePicToArray(idholder)
     {

document.images[idholder].src =
arrayHolder[indexnum];
          indexnum = indexnum + 1;
     }
</script>
</head>
<body>
   <img src = "relaxed12.jpg"    id =
"thispic"   height = "164"  width =
"194">
   <img src = "button.jpg" id =
"buttonpic" height = "60"
         width = "53" onClick =
"ChangePicToArray('thispic')" />
```

1. When the browser loads the html page with the javascript in it, the array is created and indexnum is set to 0.
2. When you click on the buttonpic image, we call the function ChangePicToArray. We set the image "thispic" to be arrayHolder[indexnum]. Indexnum holds 0 now, so the image "thispic" is set to "relaxed11.jpg".
3. We then add 1 to the value inside indexnum. Indexnum is 0, so indexnum+1 will be 1
4. Next time we click on the image "buttonpic", we go into the function ChangePicToArray. We set the image "thispic" to arrayHolder[indexnum]. Indexnum holds 1 at the moment, so thispic is set to relaxed6.jpg.
5. We then add 1 to the value inside indexnum. Indexnum is 1, so indexnum + 1 will be 2.
6. Etc.

What happens if we change the line to:
Indexnum = indexnum + 2?

Your Turn: Add a counting variable to your code. Click the button on your web page multiple times to see what happens.

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 indexnum 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 indexnum holds 0). And then we add 1 to indexnum, so now indexnum holds 1.

Next time through document.images[idholder].src is changed to arrayHolder[1] (because indexnum holds 1). And then we add 1 to indexnum, so now indexnum 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 cubbyhole in the array, reset the index to 0.

What we want is to use an if statement.

Basics Form

if (expression)
{
statement1;
statement2;
etc.
}

The statements get executed if the expression is true

Example:

```
aNumVar = 10;
if (aNumVar > 5)
{
      document.write("<p> That is a great big number! </p>");
}
```

Example2:

```
aStringVar = "wilbur";
if (aStringVar == "wilbur")
{
      document.write("<p> We are talking about a pig! </p>");
}
```

Back to our example, translating "If the index is at the last cubbyhole in the array, reset the index to 0" into javascript:

```
if (indexnum == arrayHolder.length)

{
        indexnum = 0;
}
```

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

NOTE: Javascript allows us to always know the length of an array we've created by using the name of the array.length. In the above case, the length of the array arrayHolder will be arrayHolder.length.

NOTE 2: Be aware that the length of the array is the number of cubbyholes in the array, not the index of the last value. It's a computer thing. We always start numbering the cubbyholes at 0. That means that in our array above, the length would be 5, even though the last index is 4. That's because there are 5 cubbyholes in our array.

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";

  indexnum = 0;

  function ChangePicToArray(idholder)
  {
      document.images[idholder].src = arrayHolder[indexnum];
      indexnum = indexnum + 1;
      if (indexnum == arrayHolder.length)
      {
         indexnum = 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?

3. Add another button to your web page for going back in the set of images (in other words, if you're viewing the image in arrayHolder[4], by hitting the back button, you should see the image in arrayHolder[3]. Now add a function that goes backwards through the images instead of forwards. Start with indexnum set to arrayHolder.length.

4. Modify the function you just wrote so that if indexnum gets to 0, it loops back to arrayHolder.length.