

JavaScript Tutorial 2:

So far we've seen `document.write` for writing to your web page document, creating variables using `var`, reading in variables from the user using the `prompt` command, creating conditional (`if`) branches, creating `if/else if /else` branching, and generating random numbers. Now we will continue to add to the set of JavaScript tools you can use to create dynamic web pages.

Nested If branching:

You can put `if` branches inside of `if` branches. Keep in mind, we first check the outer `if` conditions, and only do the first outer `if` condition that is true. Then, once we have determined which is the first outer `if` condition to be true, we look at the inner `if` condition. So in the example below, if the user answers, "yes" to the prompt, "Do you want to buy a dog collar??", then (and only then do we prompt with, "small, medium, or large?". So if the user types in "yes" to the first prompt AND then the user types in "medium" to the second prompt, JavaScript will write out, "Your total is 13 dollars".

If, however, the user types in, "no" in response to the prompt, "Do you want to buy a dog collar", the user will never get the prompt, "small, medium, or large?". Instead, s/he will get only, "Sorry I couldn't be of more help."

In all cases (regardless of what the user answers in response to the prompt(s)), the JavaScript will write out, "Shop again soon!". This is not inside of any of the `if` conditions, so it happens regardless of which, if any, of the `if` conditions gets executed.

```
<!DOCTYPE html><html><head><meta charset= "utf-8" /></head>
  <body>
    <h1> Dog Collars</h1>
    <script>
      var ans = prompt("Do you want to buy a dog collar??")
      if (ans.toLowerCase() == "yes")
      {
        var collarsize = prompt("small, medium, or large?")
        if (collarsize.toLowerCase() == "large")
        {
          document.write("<p> Your total is 17 dollars</p>")
        }
        else if (collarsize.toLowerCase() == "medium")
        {
          document.write("<p> Your total is 13 dollars </p>")
        }
        else if (collarsize.toLowerCase() == "small")
        {
          document.write("<p>Your total is 8 dollars</p>")
        }
      }
      else
      {
        document.write("<p>Sorry I couldn't be of more help.</p>")
      }
      document.write("<p><em> Shop again soon!</em></p>")
    </script>
  </body></html>
```

ans.toLowerCase(): Note the `toLowerCase()` attached to our variable `ans`. This takes whatever string the user entered in response to the `prompt` command and converts the entire thing to lower case. I've included this in the above code because the computer thinks that the word, "Yes" and the word "yes" are two entirely different words. I want to make sure that if the user types in "Yes" (or "YES", or even "yEs"), the computer will recognize that what the user typed in is equivalent to "yes".

The following code is an example of a very basic running app. In this script, the user is prompted for the number of miles run. If the user types in a number that is 6 or greater, a random number between 0 and 3 (not including 3) is generated and, based on that random number, the corresponding encouraging phrase is printed out. If, however, the number is less than 6, a random number between 0 and 4 (not including 4) is generated and the encouraging phrase corresponding to the random number is written.

```
<!DOCTYPE html><html><head><meta charset= "utf-8" /></head>
  <body>
    <h1> Nike App</h1>
    <script>
      var ans = parseInt(prompt("How far did you run?"))
      if (ans >= 6)
      {
        var x = Math.floor(Math.random() * 3)
        if (x == 0)
        {
          document.write("<p> Way to go with the long distances! </p>")
        }
        else if (x == 1)
        {
          document.write("<p> You're showing strong endurance! </p>")
        }
        else if (x == 2)
        {
          document.write("<p>Those distances are impressive! Keep it up! </p>")
        }
      }
      else
      {
        var y = Math.floor(Math.random() * 4)
        if (y == 0)
        {
          document.write("<p> Those short runs help you build for the longer ones! </p>")
        }
        else if (y == 1)
        {
          document.write("<p> Good hustle! </p>")
        }
        else if (y == 2)
        {
          document.write("<p>Even short runs are better than sitting on the couch! </p>")
        }
        else if (y == 3)
        {
          document.write("<p>Keep running! That's what counts! </p>")
        }
      }
    }
  </script>
</body></html>
```

The following script is an example of poorly written nested if statements. Why is it poorly written? First, you must remember that we first analyze the outer if condition and ONLY if that is true do we even look at the inner if conditions. So if we enter the first part of the outer if branch (the if (ans >=6) branch), the ans variable MUST contain a value that is greater than or equal to 6. That means that we will NEVER ever print out "That's a pretty impressive distance!" because if the ans variable is >= to 6, it can't possibly be less than 4.

That's just one problem. What if the user typed in 7 in response to the prompt, "How far did you run?"? The ans is greater than or equal to 6, so we'd enter the first of the outer branches. Then we'd look at the inner if conditions. Well, the ans variable isn't greater than 10, so we wouldn't enter that branch. It isn't greater than 7, so we wouldn't enter that branch. And it isn't less than 4, so we wouldn't enter that branch. But they are the only branches inside the first of the outer if branches (if (ans >= 6)). So there is no branch for the condition in which ans holds 7. So nothing would be written out.

Do you see why if the user typed in 3 in response to the question, "How far did you run?", the script would write out,

“Keep running!”?

```
<!DOCTYPE html><html><head><meta charset= "utf-8" /></head>
  <body>
    <h1> Lousy Nike App</h1>
    <script>
      var ans = prompt("How far did you run?")
      if (ans >= 6)
      {
        if (ans > 10)
        { document.write("<p> Wow! You really went far! </p>")
        }
        else if (ans > 7)
        { document.write("<p> That's a pretty impressive distance! </p>")
        }
        else if (ans < 4 )
        { document.write("<p>Short runs build muscle! </p>")
        }
      }
      else
      { if (ans > 10)
        { document.write("<p> Awesome run! </p>")
        }
        else if (ans < 2)
        { document.write("<p> You burnt calories! </p>")
        }
        else
        { document.write("<p>Keep running! </p>")
        }
      }
    </script>
  </body></html>
```

Arrays: Storing more than one thing

So far we've created and used variables that hold only one value at a time. For example:

- var num = 3
- var ans = "yes"
- var color = "blue"

Each of these variables holds one thing. If I put something else into the variable, the old value is removed and the new value replaces it. So, if a script contains:

```
var num = 3
num = 7
document.write("<p> The number is " + num + "</p>")
```

What this script would write out is:

The number is 7

However, there are times when we want to keep a set of things together in one variable, like maybe a set of colors, or phrases, or products, or even pictures. We will use an array for that! An array is a variable with more than one space in which to put something.

Creating an array

To create the array (to let the browser know we want to have an array available to use), we must first define it. With variables, we only had to say, `var x = 3`. However with arrays, we have to first say that we're creating a brand new array. So before we can put any values into it, we must say,

```
var arrayHolder = new Array();
```

Now we have an array called `arrayHolder`, and it is empty. You can call the array variable anything you want, as long as it follows the variable naming rules. In this case we called the new array `arrayHolder`.

So far the `arrayHolder` array is empty. We haven't put any values into it yet. In order to put values into it, we'd do the following:

```
arrayHolder[0] = "hello"  
arrayHolder[1] = "howdie"  
arrayHolder[2] = "greetings"  
arrayHolder[3] = "hey there"  
arrayHolder[4] = "hi"  
arrayHolder[5] = "wassup"
```

The three things to remember when putting values into an array are:

1. you must first create the array (e.g., `var arrayHolder = new Array()`, above).
2. You must always start putting values into your array at location 0 (e.g., `arrayHolder[0] = "hello"`, above)
3. You must always add values to your array sequentially, meaning you must first put something into `arrayHolder[3]` before you can put something into `arrayHolder[4]`.

Now to access a value in the array, you refer to the name of the array and the number of the location of the value in `[]` (square brackets). For example, the following code:

```
document.write("<p> " + myArray[3] + "</p>")
```

writes out:

```
"greetings"
```

You can use a variable containing a number as if it is a number, so you can do the following:

```
var num = 2  
document.write("<p> " + myArray[num] + "</p>")
```

this would write out:

```
"wassup"
```

You can even generate a number randomly between 0 and the length of the array (Note that the number is 5, yet there are only values in the array in locations 0 through 4. This is because the random number generator generates a random number up to but not including that multiplication number. So in the following code, the `var num` can hold values between 0 and 4, inclusive, only.)

```
var num = Math.floor(Math.random() * 5)  
document.write("<p> " + myArray[num] + "</p>")
```

In the above code, a random number between 0 and 4, inclusive, is generated, and placed into the variable `num`. That random number is used as the numeric index into `myArray`, and whatever value happens to be at `myArray` at the location of the random number is what is written out.

Given that knowledge, can you follow what the following script does?

```
<script>
  var EngArr = new Array()
  EngArr[0] = "dog"
  EngArr[1] = "cat"
  EngArr[2] = "hello"
  EngArr[3] = "Thank you"
  EngArr[4] = "bunny"

  var FrenchArr = new Array()
  FrenchArr[0] = "chien"
  FrenchArr[1] = "chat"
  FrenchArr[2] = "bonjour"
  FrenchArr[3] = "merci"
  FrenchArr[4] = "lapin"

  var ans = parseInt(prompt("Pick a number (0 - 4)"))
  var frenchans = prompt("What is the French translation of " + EngArr[ans] + "?")
  if (frenchans.toLowerCase() == FrenchArr[ans])
  { document.write("<p> You're right! </p>")
  }
  else
  { document.write("<p> Sorry. You're no good at this. </p>")
  }
</script>
```

In the above script, you create two arrays, one with English words in it, and one with their French equivalent words (I hope.) The user is prompted to pick a number between 0 and 4, and the answer goes into the ans variable. That number is then used to choose which value in EngArr is printed in the prompt, "What is the French translation of...?" The user should (hopefully) then type in the French word for the English word that was printed and that French word will go into the variable frenchans. If that word in frenchans is the same as the word from the FrenchArr that is at the same location as the word displayed in the EngArr, then "You're right!" is written. Otherwise, "Sorry, You're no good at this." is written.

Can you figure out what this script does?

```
<script>
  var PicArr = new Array()
  PicArr[0] = "cute1.jpg"
  PicArr[1] = "cute2.jpg"
  PicArr[2] = "cute3.jpg"
  PicArr[3] = "cute4.jpg"
  PicArr[4] = "cute5.jpg"

  var NameArr = new Array()
  NameArr[0] = "lion cubs"
  NameArr[1] = "hedgehogs"
  NameArr[2] = "otter pup"
  NameArr[3] = "kitten"
  NameArr[4] = "panda babies"
  var num = Math.floor(Math.random() * 5)
  document.write("<p><img src = \""+PicArr[num]+"\"></p>")
  document.write("<p> " + NameArr[num] + "</p>")
</script>
```

Changing existing image or element in a web page:

We can use `document.write` to write new html code to a web page, e.g.,

```
document.write("<h1>Zombie page</h1>")
document.write("<p> This is a paragraph about zombies.  They're mostly dead.</p>")
```

What if we want to change something that already exists on a web page? Instead of `document.write`, we can use:

document.getElementById()

This can be used to change any tag with an **id** on your web page. Remember, we gave different elements unique ids? We can use those ids to change something about the element with that id. For example, let's look at an image on a web page. An image would be included on a web page in the following way:

```
<img src = "cute1.jpg" height="100" width = "100" alt = "really cute picture" id = "pic1" >
```

We can use `document.getElementById()` to change the:

- src
- width
- height
- alt

For example, given the following web page and javascript:

```
<!DOCTYPE html><html><head>  <meta charset= "utf-8" /></head>
  <body style = "background-color:#000000; color: red;">
    <h1> Hello</h1>
    <p> <img src = "ball.jpg" width = "100" height = "100" alt = "a ball picture"
      id = "ball1"></p>

    <script>
      var x = parseInt(prompt("What size should the ball's width be?"))
      document.getElementById("ball1").width = x
    </script>
  </body></html>
```

In the above web page, there's an image with an id of "ball1". Below it in the web page is a script, in which the user is asked to enter the size of the ball's width. What is typed in goes into x. Then `document.getElementById("ball1").width` is set to that number, meaning that the element on the web page with the id of "ball1" has its width reset to whatever number is in the x variable.

The following example uses `getElementById` to change the src of the image. This, in essence, changes the actual picture being displayed on the page:

```
<!DOCTYPE html><html><head><meta charset= "utf-8" /></head>
  <body style = "background-color:#000000; color: red;">
    <h1> Hello</h1>
    <p> <img src = "kittenasleep.jpg" width = "200" height = "200"
      alt = "a picture" id = "pic1"></p>

    <script>
      var ans = prompt("Want to see something else?")
      if (ans.toLowerCase() == "yes")
      {  document.getElementById("pic1").src = "kittybelly.jpg"
      }
    </script>
  </body></html>
```

Using `getElementById`, we can change properties of any element on the web page with an id. We've seen how to use it to change the width, the height, and the src of an image, as long as the image has an id. We can also use `getElementById` to change CSS style:

```
<!DOCTYPE html><html><head>   <meta charset= "utf-8" /></head>
  <body>
    <p id = "firstp"> This is a paragraph</p>
    <p id = "secondp">This is a second paragraph</p>
  <script>
    document.getElementById("firstp").style.color = "#9999FF";
    document.getElementById("firstp").style.fontSize = "120%";
    document.getElementById("firstp").style.backgroundColor="#332277";
    document.getElementById("firstp").style.textAlign = "right";
    document.getElementById("firstp").style.padding = "30px";
    document.getElementById("firstp").style.borderWidth = "8px";
    document.getElementById("firstp").style.borderColor = "green";
    document.getElementById("firstp").style.borderStyle = "inset";
  </script>
</body></html>
```

Other styles you can change using `getElementById`

backgroundColor	color
backgroundImage	fontFamily
backgroundPosition	fontSize
backgroundRepeat	fontWeight
borderColor	fontStyle
borderStyle	lineHeight
borderWidth	textAlign
margin	And more...
padding	http://www.w3schools.com/jsref/dom_obj_style.asp
width	
position	

InnerHTML

What if we want to change the text of a paragraph or header on a page? You can use:

`document.getElementById().innerHTML`

For example, in the following code, if you want to change the text between the opening and closing `<p>` tags for the paragraph with the id, "firstp", you could do the following:

```
<body>
  <p id = "firstp"> This is a paragraph</p>
  <p id = "secondp">This is a second paragraph</p>
<script>
  var x = prompt("Do you want to see new text?")
  if (x == "yes")
  {
    document.getElementById("firstp").innerHTML = "Some new text";
  }
</script></body>
```

The **innerHTML** is what is between the opening and the closing tag, regardless of what the tag is:

```
<p id = "firstp"> This is the innerHTML text between the opening and closing tag</p>
```

Above, the innerHTML is: "This is the innerHTML text between the opening and closing tag"

To change it:

```
document.getElementById("firstp").innerHTML = "new text for paragraph"
```

```
<h1 id = "firsth">Title goes here </h1>
```

Above: innerHTML is: Title goes here

To change it:

```
document.getElementById("firsth").innerHTML = "New Title"
```

Try:

In the following example:

```
<p id = "linked"> <a href = "udel.edu" id = "firstlink"> link to udel </a></p>
```

- What is the innerHTML of linked?
- What is the innerHTML of firstlink?
- How would you change the innerHTML of linked to a new link?

```
<ol id = "list1">
```

```
  <li id = "firstItem"> cats </li>
```

```
  <li id="seconditem"> dogs </li>
```

```
</ol>
```

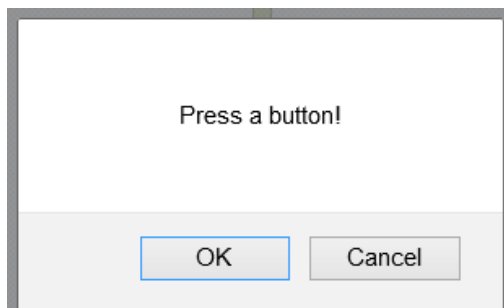
- What is the innerHTML of list1?
- What is the innerHTML of firstitem?
- How would you change the innerHTML of list1 to a new list?

Confirm box

The confirm box is like the prompt box, only in this case you only have the choice of choosing "ok" or "cancel". To create a confirm box, you'd write the following code:

```
var x = confirm("Press a button!")
```

A confirm box will pop up that looks like this:



If you choose ok, the x variable will hold true. If you choose cancel, the x variable will hold false.

Aside: true and false are known as Boolean values

Here is an example of using the confirm box:

```
<p id = "p1"> hi there </p>
var x=confirm("Press a button")
if (x==true)
{   document.getElementById("p1").innerHTML="You pressed OK!"
}
else
{   document.getElementById("p1").innerHTML = "You pressed Cancel!"
}
```

Now, given all you know, can you follow the following script?

```
<body>
  <h1>Learn about <span id = "animal">animals</span></h1>
  <p><img width = "230" height = "200" id = "pic1" alt = "animal pic">
  </p>
  <p id = "firstp"> You can learn so much about every animal in the zoo!</p>
  <script>
    var animals = new Array()
    animals[0] = "lions"
    animals[1] = "giraffes"
    animals[2] = "penguins"
    animals[3] = "polar bears"

    var pics = new Array()
    pics[0] = "Images/lion.jpg"
    pics[1] = "Images/giraffe.jpg"
    pics[2] = "Images/penguin.jpg"
    pics[3] = "Images/polarbear.jpg"

    var info = new Array()
    info[0] = "Lions are the second largest big cat species in the world (behind tigers). The
    roar of a lion can be heard from 8 kilometers (5.0 miles) away."
    info[1] = "A male giraffe can weigh as much as a pick up truck! Although a giraffe's neck
    is 1.5 - 1.8 metres, it contains the same number of vertebrae at a human neck."
    info[2] = "Penguins spend around half their time in water and the other half on land.
    Penguins spend around half their time in water and the other half on land."
    info[3] = "Polar bears keep warm thanks to nearly 10 cm of blubber under the skin. Polar
    bears have black fur under their outer layer of white fur."

    var x = confirm("Would you like to learn about an animal?")
    if (x == true)
    {   var num = Math.floor(Math.random()*4)
        document.getElementById("animal").innerHTML = animals[num];
        document.getElementById("pic1").src = pics[num];
        document.getElementById("firstp").innerHTML = info[num];
    }
  </script>
</body>
```

Comments

Comments are a way of including comments to yourself and to other people. Comments are not executed by the browser (the browser completely ignores them! The ONLY time they show up is when you load your web page into textWrangler or notepad++. They never show up in the browser's display.

Comments start with `/*` and end with `*/`. Everything between these opening and closing markers is ignored by the browser, so anything between them won't be run by javascript. For example:

```
<script>
  /* This script asks the user for a number. It uses that number to set the
     element with the id "ball1" to the width of the number the user
     entered */

  var x = parseInt(prompt("What size should the ball's width be?"))
  document.getElementById("ball1").width = x

</script>
```

Comments are designed for leaving information for people to read (as opposed to the browser). However, we can use comments to isolate what code is working and what isn't

Debugging:

When your code doesn't work, we say, there is a "bug" in your code. Finding the bug and fixing it, so your code works, is known as "debugging". There are 2 types of errors in code:

1. It could be a "syntax error"
 - ▶ These include typos, which happen a lot.
 - ▶ Syntax errors include capitalizing a letter that should be small, e.g.,
var num = 3
Num = 4
 - ▶ e.g., `document.getElementbById('pic1').src = "cat.jpg";`
 - ▶ e.g., forgetting an opening or closing `{ }` or `()`
 - ▶ e.g., `if (par1 = 'pic1')`
2. Could be a logic error
 - ▶ These are the hardest to find!
 - ▶ When what you're trying to do won't be done in the way you're trying to do it.

How do you go about finding the bug? It can be tricky and frustrating. But there is a systematic way to narrow down exactly where the problem is.

Finding the bug (Debugging)

1. Is your web site showing up as you want it to?
 - ▶ Probably an html error
 1. Make sure the page is valid (you've got an opening and closing `<html>` tag and an opening and closing `<body>` tag inside it.

2. Check to make sure that if you opened a tag, you closed it properly.
2. Is the javaScript just not working at all?
 - ▶ Check to make sure everything opened is closed
 1. Make sure your quotes (" ") open and close properly
 1. Make sure the quotes are " " and not "" (from copying from ppt or word)
 2. Go through and check for opening and closing () and {}
 3. **CHECK CAREFULLY FOR PROPER CAPITALIZATION**
 4. If nothing shows up, check to make sure you've properly entered,
`<script>`
`</script>`
(I've seen a lot of <scirpt> tags)

If none of that works, you can always try isolating the bug by putting comments around parts of your code to see where the code is working and where it isn't. For instance, if the script isn't working at all, put comments around a section. If the part that's remaining works, it means that the problem is inside the part that you commented out.