**Lesson 7 - JavaScript: A Solid Introduction**

**Lecture Notes**

**Activity**

* CTRL + SHIFT + J (or F12 and click)
* Google chrome has a javascript command line!
* Let’s get familiar with the javascript console, and with the debugger.

**Lecture**

**JavaScript Introduction**

* Makes your HTML dynamic!
  + It can change HTML Content
  + It can change HTML Attributes
  + It can change HTML Styles (CSS)
  + It can validate data
  + (and much much more!)

**JavaScript Where To**

* <script> tag
  + Usually a good idea to specify the type of script
  + <script type=”text/javascript”></script>
* <head> or <body>
  + If they go in the <body> they have to be at the very end
  + That way the html will have time to show up on the page.
  + Usually a good idea to put all your scripts right before </body>
  + Some external scripts must be in the <head> tag.
* Inline-scripts
  + No. just no
  + But seriously, it’s arguably worse than inline styles
* External scripts
  + Use these pretty much all the time
  + <script type=”text/javascript” src=”myScript.js”></script>
* Internal Scripts
  + Just line internal styles, these can be nice for when you’re putting a page together.
  + Kind of nice if you’re feeling a little lazy
  + <script type=”text/javascript”>console.log(‘hello world’);</script>

**JavaScript Output**

* window.alert()
  + Don’t ever use these.
  + Say you have a lot of code, and you forget to delete a few “debugging” lines, these will pop up if a user hits an edge case scenario…
* document.write()
  + Not super useful, as it will delete all existing html
* innerHtml
  + Definitely pretty good, as you can put html content inside of existing html content.
* console.log()
  + Easily the most useful.
  + Use this all the time
  + Seriously pretty great
  + But for real, this is your bread-and-butter when it comes to javascript.
  + You can see the output by pushing CTRL + SHIFT + J
  + Or F12 and then click console

**JavaScript Syntax**

* **Variables**
  + Usually a good idea to declare and assign at the same time.
  + 1 equal sign means the word “gets”
  + var x = 10;
  + “The variable x gets the value of 10”
* **Operators**
  + + - \* / are for computation
  + = is for assignment
* **Keywords**
  + Can’t use var as a variable name because it’s a keyword
  + Can’t use for, switch, break, continue, etc…
  + <http://www.w3schools.com/js/js_statements.asp>
* **JavaScript comments**
  + // should be used for one-line comments
  + /\* \*/ should be used for multi-line comments
  + /\* \*/ should not be used for one-line comments
    - Makes it difficult to comment out big blocks of code
    - I use this comment style quite sparingly.
  + Comments are frequently used to prevent execution
* **Case Sensitive**
  + LASTNAME and lastName are two totally different variables
* **Naming conventions**
  + Pascal case
    - Each new word is capitalized, including the first word
    - ThisBigLongVariableNameIsInPascalCase
    - Usually used for object names
    - Things that are more important get pascal case
  + CamelCase
    - Each new word is capitalized, but not the first word
    - Usually used for local variables, scope variables, small things.
    - thisVariableNameIsInCamelCase
  + Underscores
    - This\_variable\_has\_underscores\_damn\_this\_is\_annoying
    - Usually used with GLOBAL\_VARIABLES
    - Or constants

**JavaScript Statements**

* One instruction to be executed by the web-browser
* every statement ends in a semi-colon
  + Semi-colons are optional, but best practice says you should use them
* Whitespace is not a big deal. Just try to make things readable
* Usually a good idea to add a line break if it doesn’t fit on half of your screen
* Code blocks
  + Multiple statements can appear in { }
  + Usually a good idea to indent 4 spaces (or 1 tab) when starting a new block

**JavaScript Operators**

* Basic operators
  + + addition, - subtraction, \* multiplication, / division
  + % modulus
  + ++ increment
  + -- decrement
* Assignment operators
  + += is the same as
  + x = x + y
* String operators
  + + is used to concatenate
* Comparision and logical operators
  + == equal to
  + === equal to and equal type
  + != not equal to
  + !== not equal value or not equal type
  + > greater than
  + < less than
  + >= greater than or equal to
  + <= less than or equal to
  + ? : ternary operator (we’ll cover this later)

**Javascript Data Types**

* JavaScript will process anything after an equal sign from left to right.
  + It does account for precedence
    - <http://www.w3schools.com/js/js_arithmetic.asp>
  + When in doubt of which order things will take place….
  + Just use parenthesis!
* **Pretty much everything is one of the following**
  + String
  + Number
  + Object
  + Array
  + Undefined
  + Boolean

**Javascript Functions**

* Functions optionally return something
  + It’s okay if they don’t return anything
* Functions are great for re-using code!
  + No need to keep writing the same code twice!
  + Also can provide different results

<http://www.w3schools.com/js/js_arrays.asp>

<http://www.w3schools.com/js/js_array_methods.asp>.

<http://www.w3schools.com/js/js_loop_for.asp>