**JAVASCRIPT BASICS**

**JAVASCRIPT DEFINITION:**

**-**A programing language that’s used to make webpages interactive.

-Interpreted language

-Runs on the client’s computer/browser

-Object Based

-Doesn’t need to be compiled.

-Executed on the client’s computer. (This means we don’t need any kind of special server or anything like that.)

-Scripting language: Interpreted by a web browser when the page is loaded

PROGRAMMING FUNDAMENTALS:

-Variables: Items of data that have a name and a value

-Arrays: Groups of variables

-Loops: Used to iterate through data, and do what ever you’d like

-Conditionals: If statements

-Comparison and Operators

-Objects

-Functions

-Program Flow

-Data Types: Strings, numbers

Prototypes

-Events: Events or actions associated with a particular element.

WHAT CAN YOU DO WITH JAVASCRIPT

-Put content in an HTML page on the fly

-Make pages interactive and responsive

-Detect a User’s Browser & Other info

-Create Cookies

-Validate Forms

-Create Animations, Slideshows, Scrollers, etc

-Build Apps With JavaScript Frameworks (AngularJS, ReactJS, BackboneJS, etc)

-Node JS. JavaScript running on a server

HOW TO ACTIVATE JS

-Either link a JavaScript file, or we can just put it right in the html using script tags

Adding it directly to the html file:

<script>

All your JavaScript goes here.

</script>

Linking it to the html page:

<head>

<script src=”main.js”></script>

</head>

VARIABLES and DATATYPES

The Different Datatypes:

-Number

- String

-Array

-Object

**VARIABLES**

**Things to do with Variables:**

var myName; (This is considered a variable declaration since I put a semicolon right after the variable name. This variable would also be considered to have an undefined value since, It doesn’t have a follow up value)

var myVariable = 5; (This is considered to be a number variable, for obvious reasons.)

var myVariable = true;

var myVariable = false; (These variables are considered to have what’s called a Boolean value which means true or false )

**Example:**

Var number1 = 35;

(since it doesn’t have quotes around it, it’s going to be looked at as a number. If it had double or single quotes, it would be looked at as a string.)(Also keep in mind that when you use an alert, if you use quotations within the parenthesis, it’ll display what ever is inside the quotations as text, but if there’s no quotations inside the parenthesis, then it’ll display it as a value

**EX:**

Alert(“number1”) this will display as “number1” in the alert.

Alert(number1) this will display the value assigned to that variable which in this case is 35

Performing a math operation:

var number1 = 35

var number2 = 15

alert (number1 + number2):

This will display the alert as “50”

-note that the variables have to be next to each other. There could be spaces between them, but there can’t be another variable interrupting their flow

**Correct Example:**

var number1 = 35

var number2 = 15

alert(number1 + number2)

this will equal 50

**Correct Example 2:**

Var number1 = 35

Var number2 = 15

Alert(number1 + number2)

This will equal 50

**Incorrect Example:**

Var number1 = 35

Alert(number1 + number2)

Var number2 = 15

STRINGS

**STRINGS:** Strings are stored as objects. Because they’re considered objects, **they have properties** (such as length).

Properties can be accessed by putting the name of the variable followed by a period, followed by the property name. You can also create new variables by using the original variable’s properties as its’ value.

**Example:**

var alpha = “Hey there”;

var length = alpha.length;

document.write(length);

So if you add document.write as the example above, the result will show you the number of characters in the string. So the variable above will show like this: 9

If you did the same thing to an array it would show you the number of variables inside the array instead of the actual number of total letters.

**Strings also have Methods:** The difference between properties and methods, is that a property is just a piece of information. It’s a value. It tells you something about the object. A method on the other hand can take input. Do computations, and then give output to the outside.

One method is **Substring** It will just extract a portion of the string. You command methods the same way you command properties; by adding a period after the variable name, and then you give it input by using the parenthesis.

**Substring: Takes two index values, and it extracts the portion of the string between the values you select**

**Example:**

var alpha = “lkdhwr, ”

var result =alpha.substring(1, 4);

document.write(result);

**Changing them from a number datatype to a string**

Var number1 = “35”; (these numbers have now become strings)

Var number2 = “15”;

Alert(number1 + number2)

This will display as the following: 3515 (This is known as concactenating the two strings together)

**Concactenation Example 2:**

Alert (“my favorite number is” + number1)

This would show as: my favorite number is35

Alert (“my favorite number is” + “ ”+ number1)

This would show as: my favorite number is 35

Alert (“my favorite number is “ + number1)

This would show as : my favorite number is 35

(notice how there’s two ways to add a space to it)

**Acceptable forms of variables:**

-Letters, numbers, underscores, dollar signs

-These are the only character that you should be using

-They should begin with a letter (Just for proper form)

-They’re also case sensitive

**Example:**

Var test = “This is a test”;

Var Test = “This is another test”;

**Variable styles:**

**-Camel Case:** thisIsAnExampleOfCamelCase (camel case starts with a lowercase)

**-Partial Case:** ThisIsAnotherExample (Notice how partial case is identical to camel case, except the first word also starts with a capital letter)

**-Underscore:** this\_is\_yet\_another\_example

**Final Examples:**

* var myNameIs
* var MyNameIs
* var my\_name\_is

If you initialize the variable once, you don’t have to re-type it again

**Example:**

var myNameIs

(later on you don’t have to include the var)

**Example:**

myNameIs = Jahaziel

OPERATORS

**Concactenating Different Variables:**

**Example:**

var num1 = 39;

var num2 = 293;

var total = num1 \* num2;

document.write(total);

<This will multiply var num1 and var num2;

**Example2:**

var num5 = 9283;

var num6 = 932;

var total3 = num5 + num6;

console.log(total3);

<this will add num5 and num6>

**Example3:**

var num3 = 838;

var num4 = 9348;

var total2 = num3 / num4;

document.write(total2);

<this will divide num3 between num4>

**Example4:**

var num19 = 847;

var num20 = 99834;

var finalTotal = ++num19 \* num20;

console.log(finalTotal);

<if you add the ++before the desired number, it’ll increase that number by one before it calculates and combines that number with other numbers. Adding the ++ after the number name will make it so that it adds the one after the equation.

**You can also do – but it has to be on the left side of the variable**

**Example5:**

var num21 = 373;

var num22 = 23;

var finalTotal = **--**num21 + num22

ARRAYS

**Arrays Definition:**

-They’re used to store multiple values in a variable.

-You declare an array as a variable:

**Example:**

Var colors = [];

(Note that the brackets define it as an array. So this would be considered an initialized array)

**Example2:**

var colors = [“red”, “blue”, “green”];

(this is considered an array that has 3 different values in it)

**2nd Method for creating an Array:**

var a = new Array (6);

a[0] = “Cat”;

a[1] = “Dog”;

a[2] = 4;

a[5] = true;

<This is how we define the size of the array>

**Targeting Specific Values inside an Array:**

-You have to create an extra bracket inside next to the array name inside the alert parenthesis. And inside the brackets you have to assign a value by placing a number inside it corresponding to the variable you want to use. (Please note that the values start with 0 not with 1)

**Example:**

var colors = (“red”, “ blue”, “ green”)

alert(colors[1])

(so in this case the alert would display the blue, and not the red because of the 1 inside the brackets)

**Method number 2 for creating an Array:**

**Example:**

var colors = new array (“ red”, “ yellow”, “ orange”);

(so the 2 differences between this second method and the first method are: in this method you add the word “new array” right after the equal sign, and instead of using brackets to put your content in, you use the parenthesis, like you would in a regular string.)(IMPORTANT: when you add the command “new Array”, make sure that the word “new” is all lowercase, and the word “Array” begins with a cap. Otherwise it’s not going to work)

**Adding Extra Variables to Arrays, Directly Through The Alert Section:**

**Example:**

var puppies = [“Chato”, “ Kai”, “ Pelusa”]

alert(puppies[3] = “tiempo”)

This would show up like this “tiempo”

By adding the = followed by the variable name which in this example is “tiempo” It’ll declare it as if you had added it to the var array section.

You can also just add the extra variable by simply including the array name followed by the new variable name, and then later on make it appear in an alert

**Example:**

Var puppies = [“Chato”, “ Kai”, “ Pelusa”]

Puppies[3] = “Tiempo”;

alert(pupies)

This alert will show up as follows: Chato, Kai, Pelusa, Tiempo

**Example2(Preferred Method):**

Var puppies = [“Chato”, “ Kai”, “ Pelusa”]

Puppies.push(“Tiempo”)

This second method is called the push method.

**.Syntax:** The previous method we saw is an example of what’s called .syntax

**Example:**

Array.function

**Array with numbers.**

**Example:**

Var numbers = [5,77,6,43];

Alert(numbers[0] + numbers[2]);

This will make it so it adds the specified numbers which in this case are 5 and 6, which will = 11

You can also mix and match, numbers with strings

**Example:**

Var numbers = [5, 77, 6, “Seven”]

**Some synonyms:**

Methods and functions (These have to have parenthesis)

Properties and Attributes

**Some Properties:**

**-**Push: adds a variable or number to an existing array

-Length: Tells us the number of values that exist in an array.

-Sort: This function (I think it’s a function) sorts the values inside your array by value

**Example:**

Var numbers = [5,77,6,43];

alert(numbers.sort())

**Result:** 43, 5, 77, 6

LOOPS

**The Basic Types of Loops: Loops iterate (do something repeatedly) through something.**

-For Loop

-While Loop

-For Each Loop (These loops are built to work with an array)

**For Loop:**

**Example(3 different parameters which are separated by semicolons: 1- set a variable, 2- set the condition EX: I < 10 here you’re saying “as long as i is less than 10” 3- Increment EX: i++) (Inside the { is what we want to happen through each iteration. In this particular example we’ll do a console log, which is a way to print stuff out kind of like alert}):**

for(var i = 0; i < 10; i++){

console.log(i);

}

**While Loop:**

**Example(Here we’ll start with our condition):**

Var i = 1

while(i < 11){

console.log(i);

i++;

}

**For Each Loop:**

**Example:**

FUNCTIONS

**Functions:** They’re very useful if you have code that you want to re-use. You put it in the function, and then you just call the function as many times as you want.

**How to use the function:** First you use the function keyword, and then the name you want to give your function. Then we open and close parenthesis, and then open and close curly brackets, and put the code you want to store inside the curly brackets

**Example:**

Function sayHello (){

document.write(“Hello”);

}

You can put these functions in the head tag, and then you can just activate them by adding the name of the function followed by the open and close parenthesis, and closing it out with the semicolon.

**Example:**

<head>

<script>

Function sayHello (){

document.write(“Hello”);

}

</script>

</head>

<body>

<script>

sayHello();

</script>

</body>

You can also add **parameters** to the parenthesis to give it further inputs.

**Parameters:** A parameter of a function is an independent variable. It belongs to the function and it has no contact with the outside world. Note that you can have more than one parameter, if I separate them with commas. Than you can use this variable inside of a function. When we call the function, we give the variable a value.

**Example:**

<head>

<script>

Function sayHello (**who**){

document.write(“Hello” + who);

}

</script>

</head>

<body>

<script>

sayHello(“**Bob”**);

document.write(“<br>”);

sayHello(“**Mary**”);

</script>

</body>

Notice that you can include a document write between code in order to input some html such as the <br> or the <hr>

FLOW CONTROL STATEMENTS

If Statement: An if statement will only execute code, if a particular condition is true.

**Example:**

a = 7;

if(a > 10) {

alert(a);

}

So in this case, it wouldn’t execute because a is not greater than 10. But If you were to make it greater than 10, then the alert would appear.

Example:

a = 20;

if(a > 10) {

alert(a);

}

PRACTICE ROUNDS (Proper setup)

**Var:**

var variableName = “String”

var redCar = “This is a beautiful red car!”

MISC INFO

**Typing Text Directly on the Page:**

**Example:**

Document.write(“Hello World”);

<Using the semicolon creates a statement>

10 THINGS TO MASTER IN JAVASCRIPT IN ORDER TO BE PROFECIENT

10- How does JavaScript Work

9- How to test JavaScript

JSON

**JSON:** This is the most common way to store data in Javascript.

It all boils down to 2 things: Objects and Arrays.

**Objects:** Let’s say I want to store data about a cat, and I want all of this data to live within one variable, or one object.

**Example:**

var myCat = {

“name”: “Meawsalot”,

“species”: “cat”,

“FavFood”: “tuna”

}

By adding the curly brackets, I’m saying that it’s an object.

And then I can assign it different properties and values.

Notice that “name” on the left of this variable is considered a property. And the actual name is the value which in this case is called “Meawsalot” Also notice that the comma is added to indicate that there’s another property we’re going to assign to it. So this is considered an object.

In order to access one of the properties, I would have to type the variable name, dot the property name, and it would give me the value

**Example:**

myCat.name

and this would return the value of ”Meawsalot”

This is an example of how to access data that lives in an object.

**Arrays:** Let’s imagine I want to create a single variable that contains all of my favorite colors.

Example:

var myFavColors = [“Green”, “Blue”, “Black”];

**Project:** We’re going to combine both object and array into one big line of code. We start with our variable and telling it to be an array, except instead of making it an array of simple text, we’ll make it an array of objects. We define the object by opening a pair of curly brackets, and we separate the objects by using commas after the curly brackets

**Example:**

var thePets =[

{

"name": "Meawsalot",

"species" : "cat",

"favFood": "tuna"

},

{

"name": "Barky",

"species" : "dog",

"favFood": "carrots"

}

]

**Example2:**

var cars = [

{

"make": "Honda",

"model": "Accord",

"cylinders": "4 cyl"

},

{

"make": "Nissan",

"model": "Altima",

"cylinders": "4 cyl"

},

{

"make": "Nissan",

"model": "Maxima",

"cylinders": "6 cyl"

}

]

This code above is an example of JSON. JSON are objects and arrays nested inside each other.

JSON is just a popular and agreed upon format for sending and receiving, and storing data.

Java

Script

Object

Notation

In order to access different parts of the code above, you have to first type out the variable name and then specify that it’s an array by using the square brackets next to the var name. **REMEMBER:** In an array of objects, the unit numbers count for the objects, and not the text inside

**Example:**

var thePets =[

{

"name": "Meawsalot",

"species" : "cat",

"favFood": "tuna"

},

{

"name": "Barky",

"species" : "dog",

"favFood": "carrots"

}

]

If we wanted to access value dog. It would look like this

thePets[1].species

Keep in mind that in the real world you probably won’t have JSON data readily available in your text editor like the one we just constructed. You’re probably going to have to access is from some database. We’re going to learn how to do this by using AJAX

AJAX

**Exercise:** We’re going to use the instructor’s database for now in order to do these exercises.

https://learnwebcode.github.io/json-example/animals-1.json

AJAX: The process of sending and receiving data without reloading the page.

Web browsers have a built in tool named **XMLHttpRequest:** This tool will do the heavy lifting for us. It will establish a connection with the url that we specify, and it lets us send or receive data.

We’ll start by creating a variable, which in this case we’ll name ourRequest.

**Example:**

var ourRequest = new XMLHttpRequest();

Now we need to tell our variable to actually do something. The web browser will expect our variable to use a method called **open**

To specify that we want our variable to do something, we’ll use the variable name followed by a period (which is how we command the system to take an action) followed by the input “open”

**Example:**

ourRequest.open();

We want to give this open method 2 arguments. The first argument is whether we want to send, or receive data

**Receiving Data: “GET**”

**Sending Data: “POST**”

The second argument is simply the url that we want to talk to.

**Example:**

**ourRequest.open(“GET”, “https://learnwebcode.github.io**

**/json-example/animals-1.json”);**

So now we need to actually do something with that data. So we’ll type out our variable name again, and this time we’re going to use a method called onload once again using the period after the variable name. this method determines what should happen once the data is loaded.

**Example:**

ourRequest.onload = function() {

console.log(ourRequest.responseText)

}

within the body of this function, we can do anything we’d like.

In this case we’re going to log out the data on the external site onto the console.

So at this point we’ve defined our variable and what we want it to do. Now it’s time to actually send the request.

**Example:**

ourRequest.send();

**Saving our data to a variable:** Now we’ll save all this data to a variable. We create a new variable name, and then = ourRequest.responseText;

**Example:**

**ourData = ourRequest.responseText;**

We’ll be using this instead of console.log(ourRequest.responseText)

In order to only log one of the objects within our ourData variable we’ll do the following:

**Example:**

Console.log(ourData[0] );

**NOTE: If you do this, our browser still doesn’t know that this is json data. So It’ll only show the opening bracket in our console area. Our web browser doesn’t know that it’s an object, or that the variable is an array of objects.**

Also Note that the **responseText** is the text that lives within our browser.

**SPECIAL EXAMPLE:**

So we originally did this:

var ourData = ourRequest.responseText;

instead of doing that, **we’ll do the following:**

var ourData = JSON.parse(ourRequest.responseText);

So now the content that lives within our url and which in this case has the name of ourRequest, will first go through our JSON filter.

CONGRATS! WE JUST PERFORMED OUR FIRST BIT OF AJAX!!!

Asynchronous

JavaScript

And

XML