**How to write a string**

“This is how you write a string”

**How to count using length**

“something”.length

**To Comment**

// then type something

**How to make a pop up confirmation**

confirm(“Type something”);

**How to prompt an input**

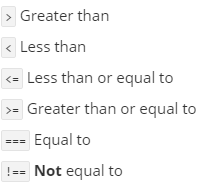
prompt(“Prompt something”);

**BOOLEAN** is true or false

**Console.log()**

console.log() will take whatever is inside the parentheses and log it to the console, it’s also known as *printing out.* How to use this type console.log(“Heyy there”) or console.log(2 \* 5)

**Operators**



**IF statement example**

*If (“this is nice” >= 3) {*

*console.log(“YAYYY nice is bigger”);*

*} else*

*{*

*console.log(“You have a short one”);*

*}*

**\*\*\*Syntax is pretty much the grammar of programming.**

**Summary so far**

**Confirm and prompt:**

*confirm(“type anything”);*

*prompt(“type anything”);*

**Data types:**

*Numbers (e.g. 14.3, 256)*

*Strings (e.g. “Dogs go woof”, “Marian is a weirdo”)*

*Boolean (e.g. true, 5 > 4)*

**Conditionals**

If the first condition is met, execute the first code block. If it is not met, execute the code in the else block. See the code on the right for another example.

**Modulo**

When *%*is placed between two numbers, the computer will divide the first number by the second, and then return the **remainder** of that division.

So if we do 23 % 10, we divide 23 by 10 which equals 2 with 3 left over. So 23 % 10evaluates to 3.

Example of **Modulo** and **If else** used together.

*if(16 % 8 == 0 ) {*

*console.log("The first number is even");*

*} else {*

*console.log("The first number is odd");*

*}*

**Substring**

It is a way of manipulating string, sometimes users don’t want to display the whole string, so a way to display only a ***part*** of the string, and this is called a substring.

Example of writing a substring:

*“Hello”.substring(1,4); This displays “ello”*

*“Niggabitch”.substring(5,10); This displays “bitch”*

*Console.log(“bitchass”.substring(5,8)); This displays “ass” in the console*

**Variables**

It is a way to save values from the code.

To declare a variable type:

*var thisVariable = 35; this saves a* ***number*** *value*

*var thisVariable = “You nigga”; this saves a* ***string*** *value*

*var thisVariable = true; this saves a* ***Boolean*** *variable* ***true***

*console.log(thisVariable.length); this* ***prints out*** *the* ***length*** *of the variable in the* ***console.***

**Change variable values**

**Below I’m declaring a variable called myName with my name “Levi”**

*var myName = "Levi";*

**Below I’m printing out the myName variable on the console**

*console.log(myName);*

**Below I’m changing the value of myName to be just the first 2 letters of my name**

*myName = myName.substring(0,2);*

**Below I’m printing out myName variable on the console**

*console.log(myName);*

**Function**

A function takes in inputs, does something with them, and produces an output.

**Example:**

*var sayHello = function(name){*

*console.log(‘Hello’ + name);*

*}*

1. First we declare a function using **var**, and then give it a name *sayHello*. The name should begin with a lowercase letter and the convention is to use *lowerCamelCase* where each word (except the first) begins with a capital letter.
2. Then we use the function keyword to tell the computer that you are making a function
3. The code in the parentheses is called a **parameter**. It's a placeholder word that we give a specific value when we call the function

**Another example of function:**

*var greeting = function (name) {*

*console.log("Great to see you," + " " + name);*

*};*

*greeting("Levi");*

This function outputs ***“Great to see you, Levi”***

**\*\*\*Important**

Remember that typing the keyword ***function*** tells the computer that ***functionName*** is a function and not something else. **Parameters** go in the parentheses. The computer will look out for it in the code block. The code block is the reusable code that is between the *curly brackets* **{ }.** Each line of code inside **{ }** must end with a semi-colon. **The entire function ends with a semi-colon**. To use the function, we **call** the function by just typing the function's name, and putting a parameter value inside parentheses after it. The computer will run the reusable code with the specific parameter value substituted into the code.

**Return Keyword**

The *return* keyword simply gives the programmer back the value that comes out of the function. So the function runs, and when the*return*keyword is used, the function will immediately stop running and *return*the value.

***Example:*** *It simply makes a function called timesTwo that takes in a number and returns the number multiplied by* ***two****.*

var timesTwo = function(number) {

return number \* 2;

};

var newNumber = timesTwo(5);

console.log(newNumber);

**Functions, return and if / else**

var quarter = function (number){

return number / 4;

}

;

if (quarter(12) % 3 === 0 ) {

console.log("The statement is true");

} else {

console.log("The statement is false");

}

*Above example simply combines* ***functions****,* ***return*** *and* ***if /else*** *in one code.*

**Functions with TWO parameters**

To call a function with more than one parameter, just enter a value for each parameter in the parentheses. For example, areaBox(3,9); would return the area of a box with a length of 3 and a width of 9.

**Example**: Below example will return a value of **32**

var areaBox = function (length, width){

return length + length + width + width;

}

areaBox(6,10);

**Global and Local Variables**

Scope can be Global or Local. Variables defined **outside** a function are accessible anywhere once they have been declared. They are called **global variables** and their scope is **global**. AND variables defined **inside** a function are **local variables**. They cannot be accessed outside of that function.

Example of a **Global Variable**:

var globalVar = "hello";

var foo = function() {

console.log(globalVar); *// prints "hello"*

}

Example of a **Local Variable:**

var bar = function() {

var localVar = "howdy";

}

console.log(localVar); *// error*

**\*\*\*\* Important:** The var keyword creates a new variable **in the current scope**. That means if var is used outside a function, that variable has a global scope. If var is used inside a function, that variable has a local scope.

Using a function without the var *keyword* refers to the **global** variable that has already been declared outside the function. However, if you use the var *keyword* inside a function, it declares a **NEW local** variable that only exists **WITHIN** that function.

**For loop**

*Below is a general syntax for a for loop:*

for (var i = 1; i < 11; i = i + 1) {

*/\* your code here \*/*;

}

Every for loop makes use of a counting variable. Here, our variable is called i *(but it can have any name).* The variable has many roles. The first part of the for loop tells the computer to start with a value of 1 for i. It does this by declaring the variable called i and giving it a value of 1.

**\*\*\*\*\*Important:** Make sure to use ***i += 12,*** and not ***i + 12***. The latter will crash your browser, because it will never be updated and the loop will go on forever!

**Arrays**

Arrays store lists of data, it can store different data types at the same time, and are ordered so the position of each piece of data is fixed.

To write an array:

var names = ["Mao","Gandhi","Mandela"];

Small complication: the position (or the index) of each bit of data is counted starting from 0, not 1. The position of things in arrays is fixed.