**Basics**

**Functions**

Console.log(); Basically prints to system.

You can create functions in two different methods.

function name() { }

let name = () => {}

let name = param => condition; also works in one line when you only have one param.

Parameters are attributes of functions that will use its values within the function. Located within ()

let myName = “Bruh Moment”

let name = (myName) => {

console.log(myName);

}

name(); This works because myName is global and can be used from every function.

name(myName); Also works;

can also set a default value

let name = (myName = “Guest”) => {} and will run “Guest” if myName is undefined;

**Return**

By default, even if a function runs correctly, it needs a return value so it will take that as the result of running the function

let myName = “Bruh Moment”

let name = (myName) => {

return myName ? true : false;

}

This will tell the system that the function will give the value of true;

let name = myName => myName ? true: false; also works

**Methods**

.length() takes the value of the string’s length and return a number.

.toUpperCase() changes string to all uppercase letters

.startsWith() checks the value of the first letter and the inputted value to check true or false

.trim() removes whitespace.

[**Math Methods**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math)

Math.random() takes a random value between 0 to 1. Multiply for bigger pool of numbers

Math.floor() rounds the number down to the nearest whole number.

Math.ceil() rounds the number up to the nearest whole number.

**Data Types**

Number: Any numeric value including those with decimals

String: A group of characters surrounded by quotes

Boolean: Data type that has only two values, true or false\

Null: Data type that has a value of null aka absence of a value.

Undefined: Data type that is similar to Null.

Symbol: Unique identifiers

Object: Collections of related data.

Variables: Stored data we created and define them with unique names

const is the static variable when you just want to keep it the same throughout the system.

let is the dynamic variable when you want to change the value through the system.

Don’t use var.

When we want to add a variable to a string to put on console.log, we can either use + or ${} using ``

console.log(“Bruh” + bruh) or console.log(`Bruh ${bruh}.`);

[**Arrays**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array)

Arrays consists of a list of objects in a single variable. Needs to include [] and separate items with ,

let array = [“One”, “Two”, “Three”];

When referencing an element of an array, start with 0 and then add one for each proceeding element.

let array = [“One”, “Two”, “Three”]; Then it means array[2] = Three;

Nested arrays are basically arrays that can be nested.

let array[1, [2, 3]];

array[1] = [2, 3] array[1][1] = 3

**Ops**

.join() will take every element of an array and form a single string statement

let array = [“One”, “Two”, “Three”];

array.join(“ ”);

Will result array reading as “One Two Three”

.push() allows us to add values to existing arrays

let array = [“One”, “Two”, “Three”]; array.push(“Four”);

Results in array = [“One”, “Two”, “Three”, “Four”];

.pop() is the opposite of .push where it would take out the right most object out of the array.

array.pop() will result in array to remove “Four”.

.shift() is similar to .pop, except it takes out the left most object out of the array.

.unshift() is the opposite of .shift, where it would add an object to the left side of the array.

.slice() can take up to two values to shorten the array.

let array = [“One”, “Two”, “Three”]; array.push(“Four”);

array.slice(0, 2) will return an array = [“One”, “Two”];

.splice() takes 3 values and it is used to replace elements in the array in a group

let array = [“One”, “Two”, “Three”];

array.splice(0, 3, “Four”) will result in starting at index 0, removing 3 elements, and replaced

array = [“Four”];

.indexOf() finds the index number of an object in an array

array.indexOf(“Three”) will equal 2.

**Arithmetic operators**

Basic arithmetic includes (+, -, \*, /, and %)

% represents remainder of two numbers dividing.

You can also use = as a += such as w +=1 aka w = w + 1;

When using counters you can use ++ or -- to increment or decrement a value

w++ aka w+=1

**Other operators**

typeof checks inputted value’s data type.

**Conditional statements**

if () is a statement that asks if the condition is occurs. If the condition does happen, then an action will take place.

if (1 < 2) {

console.log(“True”);

}

if else () is a statement that is asked if the if statement is not true.

else is the final conditional statement when the if statement and every other if else statements are false.

Essentially runs when the whole procedure is false.

**“False” terms**

Whenever a value is being checked for either true or false, it will not be false unless it is equal to 0, an empty string, equal to null, equal to undefined, or equal to NaN.

0, “”, null, undefined, NaN

**Comparison Operators**

Helps checks the relation between two objects in a condition.

<, >, <=, >=, ===, and !== are the basic ones.

=== means that it is equal to while !== means it is not equal to

1 === 2 will run false and 1 !== 2 will run true

**Logical Operators**

logical operators are used to check two separate statements in a single condition

&& aka AND checks if both statements are true

|| aka OR checks if either one of the statements are true

! aka NOT will reverse if the statement is true/false

(1 === 1 && 1 > 2) will return false

(1 === 1 || 1 > 2) will return true

(!1 === 1) will return false

**Creating “Default” values**

Assume we are creating finding a username from our database, if the username doesn’t exist, then we treat the person as a “Guest”

This can be achieved by using the || such as the following.

let username = null;

let name = username || “Guest”

In this instance since username is null, it will make name be defined as “Guest”

**Ternary Operator**

This exists when you want to write an if, else statement in one line.

By utilizing ? and : we are able to achieve this in one line.

if (1 > 2) {

console.log(“Impossible”);

}

else {

console.log(“Naw duh”);

}

(1 > 2) ? console.log(“Impossible”) : console.log(“Naw duh”);

**Switch**

switch() is used as an alternative to using if else statements for easier syntax to read.

You would define the condition and then state what happens when a case happens. If all fails, it will be defined in the default section.

let number = 4;

switch (number) {

case 1:

console.log(“This is 1”);

break;

case 2:

console.log(“This is 2”);

break;

case 3:

console.log(“This is 3”);

break;

default:

console.log(“This is an unknown number”);

break;

}