**Javascript Handbook**

**What is JavaScript**

* High-level programming language, interpreted language that doesn’t require a compiler to run a program.
* ECMA specification
* Multi paradigm, meaning there are multiple ways to write a code (oop, prototype, classes)
* Typically runs on browser for client-side application; node.js for server-side application

**Syntax -** JavaScript syntax is a set of rules for how Java programs are written.

* JavaScript is case sensitive, meaning Apple and apple are two different variables. (Commonly constructor use capital-case letter and variables or functions will use lower-case letter.)
* JavaScript ignores whitespaces, spaces and line breaks can be added to keep the code clean.
* Semicolons are not required to end a statement in Javascript. But can be used optionally to keep the code clean.
* Single line comment can be used with // at the front of the line, multiple lines comment uses /\* at the start and \*/ at the end of the line.
* ${...} is used to interpolate variables and expressions into strings.

<script>...</script> tag is used for implementing javascript into a HTML webpage.

<html>

<body>

<script>

document.write(‘Hello’)

<script>

</body>

<html>

This will print out ‘Hello’ in the HTML page. Alternatively, to keep the code clean, javascript codes are kept in a separate file and the file will be attached to the HTML using the src attribute.

<script src=”path / or script.js ”</script>

**Datatypes -** In Javascript, there are three basic, or primitive, types of data.

* Numbers - 1, 123, 5.61, etc.
* String - ‘Word’, ‘One’, ‘This is a string’, etc.
* Boolean - True or False.

There are also two more types of data which are null and undefined. Null is an assignment value. It can be assigned to a variable as a representation of no value. Undefined means a variable has been declared but has not yet been assigned a value.

**Variables** **-** A variable is a value assigned to an identifier, so you can reference and use it later in the program. We use three different keywords to declare three different variables. **‘var’**, **‘let’**, and **‘const’**.

‘var’ was used before ES6 came out and was replaced with ‘let’ because ‘var’ has scope constraints. ‘let’ is constrained to whichever scope it is declared in. Its declaration and assignment are similar to var. ‘let’ was introduced to mitigate issues posed by variable scope which developers face during development.

So, in ES6 we use ‘const’ and ‘let’ to define variables. ‘const’ defines a constant reference to a value meaning it cannot be changed and a new value cannot be assigned. On the other hand, ‘let’ allows a new value to be assigned to it even after declaration.

For example,

To declare variables with ‘let’,

let x = 6;

To declare variables with ‘const’

const x = 6;

Difference between ‘let’ and ‘const’ is that using let, we can change the value of x with this:

let x = 6;

x = 7;

Whereas if ‘const’ was used, doing this:

const x = 6;

x = 7;

TypeError: Assignment to constant variable will be prompted.

‘const’ should always be used when you are sure that the value of the variable won't be changed; use ‘let’ when you know a new value will be reassigned to the variable.

**Operators -** allow you to get two simple expressions and combine them to form a more complex expression.

* = to assign a value to a variable
* + addition
* - subtraction
* / division
* % remainder
* \* multiplication
* \*\* exponentiation
* < less than
* <= less than or equal to
* > more than
* >= more than or equal to
* != not equal to
* === check for equality

**Conditionals** – if, else conditionals. An if else statement is used to make the program take a route, or another, depending on the result of an expression evaluation.

For example,

let i = 4;

if (i > 3) {

document.write(‘This number is greater than 3’);

} else {

document.write(‘This number is smaller than 3’);

The conditional checks the expression for a true or false value, in this case, if i > 3, it will print ‘This number is greater than 3’; else it will print ‘This number is smaller than 3’.

**Array** - is a collection of elements which holds more than one value. The value can be different types.

let animal = [‘dog’, ‘cat’, ‘monkey’];

This will create an array with ‘dog’, ‘cat’, ‘monkey’ with 3 elements in it.

Different ways to interact with an array.

* .length - count the number of elements in an array
* indexOf() - find out the index number of a specific element in an array
* push() - to add element to the end of array
* pop() - to remove element from the end of array
* unshift() - to add element to the front of array
* shift() - to remove first element from an array
* splice() - remove a specific element from an array
* concat() - to join two arrays

**Loops -** Loops are one of the main control structures of JavaScript. Loop automates and repeats a block of code for a determined number of times or infinitely.

**While loop -**  loops through a block of code while a specified condition is true

const list = ['a', 'b', 'c']

let i = 0

while (i < list.length) {

console.log(list[i])

console.log(i)

i = i + 1

}

This block of code loops through the array and outputs the item and the item number until it reaches the final item in the array.

**Do while loop -** loops through a block of code while a specified condition is true, condition is evaluated after code block is executed

const list = ['a', 'b', 'c']

let i = 0

do {

console.log(list[i]) //value

console.log(i) //index

i = i + 1

} while (i < list.length)

This does the same as the above but the condition is evaluated after the code block is executed.

**For loop -** loops through a block of code a number of times

const list = ['a', 'b', 'c']

for (let i = 0; i < list.length; i++) {

console.log(list[i]) //value

console.log(i) //index

}

This directly gives a set of 3 instructions which are the initialization, the condition, and the increments.

**Switch -** assign variables based on condition

const x = 11;

const color = x > 10 ?(if) ‘red’ :(else) ‘blue’;

switch(color) {

Case ‘red’:

//do something

Break;

Case ‘blue’:

//do something

Break;

Default:

//do something;

}

Switch can assign different variables based on the condition. In this case, x is 11, which is greater than 10, therefore it meets the ‘red’ condition.

**Functions -** block of code design to perform a particular task

Function sum (num1, num2){

Return num1 + num2;

}

Arrow function

Let sum = (num1, num2) => num1+ num2;

Let result = sum(2, 3);

5

**Objects – value that is not of a primitive type is an object**

Const person1 = new Person(‘John’, ‘Doe’, ‘1-2-1990’);

Const person = new Person()

**Constructor Function**

Function Person(firstName, lastName, dob){

this.firstName = firstName;

this.lastName = lastName;

this.dob = new Date(dob);

this.getFullName = function(){

Return ‘${this.firstName}’ ${this.lastName}’;

}

getFullName());

**Prototype**

Person.prototype.getFullName = function() {

Return ‘${this.firstName} ${this.lastName}’;

{

**Classes**

class Person {

constructor(firstName, lastName, dob){

this.firstName = firstName;

this.lastName = lastName;

this.dob = new Date(dob);

}

getFullName(){

return’$[this.firstName ${this.lastName}’;

}