JAVASCRIPT NOTES

**Var:** lets you define variables globally

**Let:** lets you define variables locally or in a defined scope.

* Variables declared with **const** keyword are constant, hence can’t be changed or reassigned.
* Strings are immutable, Once declared can’t be changed or reassigned.
* Arrays declared with **const** can be changed but can’t be reassigned.

**Ex.**

const s = [5, 6, 7];

s = [1, 2, 3]; // throws error, trying to assign a const

s[2] = 45; // works just as it would with an array declared with var or let

console.log(s); // returns [5, 6, 45].

* **Object Freezing/Object Mutation:** Once the object is frozen, you can no longer add, update, or delete properties from it. Any attempt at changing the object will be rejected without an error.

**Ex.**

let obj = {

name:"FreeCodeCamp",

review:"Awesome"

};

Object.freeze(obj);

obj.review = "bad"; //will be ignored. Mutation not allowed

obj.newProp = "Test"; // will be ignored. Mutation not allowed

console.log(obj);

// { name: "FreeCodeCamp", review:"Awesome"}

* **Higher Order Functions:** Functions which take input as functions and also return output as functions.

**Ex:** map(), reduce(), filter().

* **Map:** The map() method Iterates the over the whole array and creates a new array with the results of calling a function for every array element.

**Note:** this method does not change the original array but can change the objects inside the array passed.

**Ex.**

var team = [{

name:"Divyajeet",

age:20

},{

name:"Himanshu",

age:23

},{

name:"Akshay",

age:22

}

]

team.map(teamMember=> teamMember.age+=10)

console.log(team)

// 0: {name: "Divyajeet", age: 30}

// 1: {name: "Himanshu", age: 33}

// 2: {name: "Akshay", age: 32}

// length: 3

* **Filter:** The filter() method creates an array filled with all array elements that pass a test (provided as a function).

**Note:** filter() does not change the original array.

**Ex.**

var team = [{

name:"Divyajeet",

age:20

},{

name:"Himanshu",

age:23

},{

name:"Akshay",

age:22

}

]

var a = team.filter(teamMember=> teamMember.age<23)

console.log(a)

// 0: {name: "Divyajeet", age: 20}

// 1: {name: "Akshay", age: 22}

// length: 2

Difference Between Map and Filter

They both return a new array. map returns a new array of elements where you have applied some function on the element so that it changes the original element (typically). filter returns a new array of the elements of the original array (with no change to the original values). filter will only return elements where the function you specify returns a value of true for each element passed to the function. So map returns the same number of elements as the original, but the element value will be transformed in some way and filter will return the same or less number of elements than the original but not change the original elements’ values.

* **Reduce:** It takes 2 arguments in the call-back function 1) Accumulator 2) Current Value. It returns a single variable out of array depending upon the requirements passed into the function.
* **.forEach(a,b,c):** forEach takes up three arguments, a=>iterable element, b=>Index of **a**, c=>the whole array on which forEach is iterating.
* **Objects:** JavaScript Objects are mutable (can be changed), Object in JS are assigned with reference not with value.

**Ex.**

var person = {name:"divyajet",age:20}

var x = person;

console.log(person)

//{name: "divyajet", age: 20}

x.age=21

console.log(x)

//{name: "divyajet", age: 21}

console.log(person)

// {name: "divyajet", age: 21}

* **Object destructing:** Destruction makes the assignment of the values of an array or object to the new variable easier**.**

**Ex.**

//Old way of doing it.

const person={

name:"Divyajeet",

age:21,

Designation:"Software Consultant"

}

const Fullname = person.name;

const Age = person.age;

const Position = person.Designation;

console.log(`Hello, My name is ${Fullname}, I am ${Age} years old and work as a ${Position}`)

//New way of doing it.

const person={

name:"Divyajeet",

age:21,

Designation:"Software Consultant"

}

const{ name:Fullname,age:Age,Designation:Position }= person;

console.log(`Hello, My name is ${Fullname}, I am ${Age} years old and work as a ${Position}`)

// Hello, My name is Divyajeet, I am 21 years old and work as a Software Consultant.

* **Parameter Collector:** whenever we pass arguments into a JS function, a keyword aka “arguments” contains all the parameters passed into a function. We can use that “arguments” keyword to toggle through the function. “arguments” store parameters in an array like format.

function myConcat(separator,hola) {

let result = ''; // initialize list

let i;

// loop through arguments

for (i = 1; i < arguments.length; i++) {

result += arguments[i] + separator;

}

return result;

}

console.log(myConcat(', ', 'fred', 'wilma', 'barney', 'betty'))

//output -> fred, wilma, barney, betty,