**Java Script Notes**

* Java Script- By Netscape
* We can make following things by JS:
  + Web App, Mobile App, CLI Tools, Games, Network etc.
* To run JavaScript, we need JS Engine. (Chrome has V8 JS Engine)
* Earlier JS was used to run on browser only that is client side.
* Now JS can execute outside the browsers also, JS combines with C++ known as Nodes on server.
* console.log(“Hello World”) – To Print on console windows
* Semi colon is used to terminate the line.
* Best Practice is to add the script after the other contents of the body.
* This is done to enhance the user experience. Done because all the contents should be parsed before the parsing of the script. HTML and CSS Should be loaded before the JS. So that if JS loads something for long time we need not to wait for JS.
* // These are used for the comments.
* Try to place all the JS code in new file called index.js . Need not to use script tag.
* Link the JS file in index.html file.
* You can also run as on terminal – node index.js

HW – Java V/s JavaScript

**Variables**

* Keyword let is used to create the variables.
  + let a=5, let name=”Harshit”
  + let is used within block scope in a function or in the { blocks}.
* Keyword var is also used to declare the variables.
  + Used within the functions scope. Acts as a global Variable.
  + var a=6
* Keyword const is used to declare constant variable which can not be change further.
  + Const a=6

**Primitive Types**

* String, Number, Boolean, Undefined (In C++ Garbage Value), Null

**Dynamically Typed**

* It is a dynamically typed language. Need not to declare the data types of variables. Same variable can be assigned to different data types further.
  + For eg.
  + let lastName=’Rai,’ lastName=6

**Reference Types**

* Objects, Arrays, Functions
* Arrays
  + let names=[1,2,3]
  + names[1] -> 2, names[0]=1

**Operators** – Arithmetic ( \*\* for power), Assignment Operator(Pre/Post/Incr/Decre),

Comparison Operator (=== - Strict Equality), (!== - Strict Not Equality)

* Loose Equality :- (==) Value should be same, Data may or may not be same.
* Strict Equality:- (===) Value and Data both should be same.
  + Eg. Let num=1, let str=’1’
  + Num===str -> This will return false (Different Data and same value)
  + Num==str -> This will return true (Different Data and same value)

Ternary Operator: Condition?True:False  
 let age=17, let status =(age>=18)?’I can vote’: ’I can’t vote’   
Logical Operator (AND, OR, NOT)  
(false || “Harshit”) -> O/P: Harshit

* Short Circuiting Concept

Bitwise Operators (AND,OR)  
Operator Precedence

**Control Statements**

* if, else if, else
* switch case
  + switch(num){
    - case 1:console.log(‘A’)
    - break;
    - default: console.log(‘B’)
  + }

**Loops:**

* for loop:
  + for(let i=0; i<5;i++){
    - console.log(‘Hello);
  + }
* While loop:
* Do while loop: Will Execute at least one time.
  + do{
  + }while(condition)
* For in Loop: For-in loop in JavaScript is used to iterate over the properties of an object.
  + //declaring a object employee
  + const courses = {
  + firstCourse: 'JavaScript',
  + secondCourse: 'React',
  + thirdCourse: 'Angular'
  + };
  + let value = '';
  + //using for in loop
  + for (let item in courses) {
  + value += courses[item] + " ";
  + }
  + console.log(value);
  + O/P: JavaScript React Angular

**Objects:**

let rectangle = {

    length:1,

    breadth:2,

    draw: function(){

        console.log('draw rectangle');

    }

};

* rectangle.draw(), rectangle.length()

**Object Creation – How?**

* Factory Function
* Constructor Function
* // Factory Function
* function createRectangle(){
* return rectangle = {
* length:1,
* breadth:2,
* draw: function(){
* console.log('draw rectangle');
* }
* };
* // return rectangle;
* }
* // Creating the Object
* let rectnagleObj1=createRectangle();
* rectnagleObj1.draw()
* rectnagleObj1.length
* Passing the parameters:-

function createRectangle(len ,bre){

    return rectangle = {

        length:len,

        breadth:bre,

        draw: function(){

            console.log('draw rectangle');

        }

    };

    // return rectangle;

}

// Creating the Object

let rectnagleObj1=createRectangle(4,5);

rectnagleObj1.draw()

rectnagleObj1.length

rectnagleObj1.breadth

* **Constructor Function:** Follows Pascal Notation- First Character is always capital.
* // Constructor Function
* function Rectangle(){
* this.length=1;
* this.breadth=2;
* this.draw=function(){
* console.log("drawing the rectangle");
* }
* }
* // object creation using constructor
* let rectangleObject=new Rectangle();
* rectangleObject.draw()
* By Passing Parameter
* // Constructor Function
* function Rectangle(len,bre){
* this.length=len;
* this.breadth=bre;
* this.draw=function(){
* console.log("drawing the rectangle");
* }
* }
* // object creation using constructor
* let rectangleObject=new Rectangle(4,5);
* rectangleObject.draw()

**Dynamic Nature of Objects**

rectangleObject.color="red";

console.log(rectangleObject)

delete rectangleObject.color;

console.log(rectangleObject)

* **Constructor Property**
* console.log(rectangleObject.constructor)
* console.log(Rectangle.constructor)

let Rectangle1=new Function(

    `length`,`breadth;`,

    `this.length=length;

     this.breadth=breadth;

     this.draw=function(){

        console.log('drawing');

     }`

);

let rect=new Rectangle1(2,3);

console.log(rect);

* **Functions are object.**
* Rectangle is also a function and an object also, because it has also some properties. Like Rectangle has also some property.

**Primitive are copied by their values. (Pass by value)  
References are copied by their address or references. (Pass by reference)  
Objects are references they are not primitive so values are changed.**

* Example
* // Primitive
* let a=10;
* function inc(a){
* a++;
* }
* inc(a);
* console.log(a) // 10
* // Reference
* let b={value:10};
* function inc1(b){
* b.value++;
* }
* inc1(b);
* console.log(b.value) //11

**For in Loop**

// For in Loop

let rectangle2={

    length:2,

    breadth:3

};

for(let key in rectangle2){

    // keys are reflected by key variable

    // values are reflected by Rectangle2[key]

    console.log(key,rectangle2[key]);

}

**For of Loop**

// For of Loops :works on iterables

for(let key of Object.keys(rectangle)){

    console.log(key);

}

for(let key of Object.entries(rectangle)){

    console.log(key);

}

**To check weather given properties are present or not**

if('color' in rectangle){

    console.log('Present');

}

else{

    console.log('Absent');

}

**Object Cloning: How to copy the object:**

* **Iteration**
* **Assign**
* **Spread**
* let src={
* a:10,
* b:20,
* c:30
* };
* // Iteration
* let des={}
* for(let key in src){
* des[key]=src[key]
* }

// Assign

let d1=object.assign({},src);

// Spread

let d2={...src};

console.log(src);

console.log(des);

console.log(d2);

**Garbage Collections:** Done by Garbage Collectors, Automatically deallocates the memory which is not being used by the variables.