# Javascript Basics:

Alert(“text”); : creates a pop up message with the text in the parameters.

document.body.internalHTML=”text”; : It changes the whole webpage html code and displays the given text.

# Math functions in js:

Basics of any programming lang.

The math library is to be learned from java.

# String functions in js:

Can create strings with both “ ” and ‘ ‘. No difference is imposed by the system.

+ operator is used to concatenate the strings.

[typeof() function is used to know the data type of any variable]

“string” + int results in a “stringint” [“hello” +3 = “hello3”]

# HTML CSS review, console.log

In side the html body use a <script></script> to include js;

onclick is used to make actions on clicking on specific elements in a webpage. Syntax: onclick= “ code ”;

Example:

onclick=”

alert(“hey there”);

“

It produces a pop of message saying the provided text.

Console.log: is used to display whatever normal operations t=you do with a variable ono the website console.

Syntax: console.log( operations );

Example: console.log( 2+2); this displays 4 on the console.

Console.log(`hey there buddy` ${var\_name}); used to concatenate string and variables.

# Variables

let : used to declare a variable

syntax: let variable\_name= data;

example: let var\_name=5;

after that to make any further alterations to the variable let keyword isn’t required.

Example:

Let a=4;

a = 6;

const: used to initialize a variable but the value cant be changed later.

Syntax: const var\_name = value;

Example: const a=10;

var : same as let , older convention of js. It doesn’t follow the rules of scope.

Syntax: var var\_name=value;

Example: var a =7;

# Variable Naming Conventions:

Camel case = ashutoshPurushottam

Pastel case= AshutoshPurushottam

Kebab case = ashutosh-purushottam

Snake case= ashutosh\_purushottam

# Booleans

true : used to showcase true value

false : used to showcase false value

values which equal to false are:

0, false, null , ‘‘ , undefined, NaN

[NaN ( not a number) : is the output js gives us when we perform invalid math operations, such as dividing a string by a number]

[undefined: when a declared variable is not yet initialized its value is undefined]

## Comparison Operators:

< , > , <= , >= , == , === , !==

== compares two values by converting them into same data types. Example:

If we write ( 5== “5”) . it is true as they are converted into a similar data type.

But if we write ( 5===”5”) it is false

And !== is used always irrespective of any case.

## Logical Operators:

And operator syntax: && [ also called guard operator, if prior parameter is false, second parameter isn’t approached and thus doesn’t come into function.]

Or operator syntax: || [ does the same thing if the first value is true]

Not Operator syntax: !

# If -Else statements:

Syntax:

if ( Boolean ){

Code;

}

else if ( Boolean ) {

Code;

}

else {

code;

}

## Ternary operator [ shortcut to if else] :

Syntax: boolean ? val1 : val2;

can be nested like : Boolean ? val 1 : Boolean ? val2: val3;

# Functions:

Definition of functions is made inside in the script tag.

Syntax:

Function function\_name(parameters){

Code…;

return val/var\_name;

}

Return statement isn’t mandatory in a function.

# Object:

Objects in JS are merely like pointers of c.

The name of the object is just a reference to the memory location where the desired data is stored.

Thus, if we make a new object = old object, it doesn’t make 2 copies of the same data, but rather makes both the object names refer to the same data inside the memory. This results in a much more efficient way of storage.

Syntax:

let/const obj\_name ={

variables : value,

variables : value

};

[similar to python dictionary]

To access a variable inside of an obj syntax : obj\_name.variable\_name;

Also we can use syntax: obj\_name[ ‘variable name’ ]; to access the data. [// bracket notation]

We can add new variables inside an already declared object.

Example:

obj.new\_var\_name=value;;

this creates a new variable inside the object along with the value.

To use variable names that are not allowed , we can use bracket notation:

Obj\_name [“name-variable”]=value;

To delete already present variables inside of an object, syntax:

Delete obj\_name.variable\_name;

## Functions inside objects:

Syntax:

Obj\_name ={

Var1: val,

fun\_name : function( parameters){

code;

return ;

}

}

To call a function from a object:

Obj\_name.functionname();

Another way to define a function inside an object: //shortcut

Obj1 ={

Fun\_name( parameters ){

Code;

}

};

Auto-boxing :

We can use string methods and other data type methods in js.

Example :

‘hello’.length(); ‘hello’.toUpperCase(); 0.5.toString(); true.toString();

In these cases, the js converts the string ‘hello’ into an object and then operates the methods.

This feature of converting data into an object is called auto-boxing.

NOTE: auto-boxing doesn’t work with null and unefined.

## Shortcuts of objects in JS:

If we try to store the data from inside a object with the same variable name outside of an object.

Example:

let message = obj.message;

We can instead use:

let {message} = obj; //bcuz the variable name is the same as inside the object.

This feature is called destructuring.

We can also use them for multiple variables. Example:

let {name, age, height} = obj;

The same can be used in the opposite manner i.e., to store a data from a variable into a object:

let obj = { name : name};

we can instead use:

let obj ={ name};

# JSON object and methods:

JSON format is easily comprehended by mostly every other compiler/ programming language. Thus, for transfer of data in or out of a js environment, we convert js and JSON objects vice versa.

The JSON build in object facilitates us to covert java script objects into JSON and vice versa.

JSON.stringify():

Used to convert a JS object into JSON string.

Syntax:

JSON.stringify( object\_name);

Return type : string [ in this scenario called as a JSON string]

NOTE: Any function stored in the provided object will be lost , as a JSON string doesn’t contain functions.

Example:

obj = { v1: ”name” , v2 : 10 , meth : function () { alert(“hey there” );} };

JSON.stringify( obj);

Now the output of the stringify method will contain v1 , v2 but not meth as it’s a function.

## JSON.parse():

Syntax:

JSON.parse( json\_string);

Return type: JS object.

Example:

obj = { v1: ”name” , v2 : 10 , meth : function () { alert(“hey there” );} };

let json\_str= JSON.stringify( obj);

alert(JSON.parse(json\_str)); // pops up the js object converted back from a JSON string.

# Local Storage:

An already present build in object used to store values in local storage rather than just the current page which loses all values when refreshed.

localStorage.setItem():

Used to save the data into local storage.

syntax:

localStorage.setItem( ‘ variable\_name\_to\_store\_string ’, ‘string\_to be stored’);

example: localStorage.setItem( ‘message’ , ‘hello’); //stores ‘hello’ in a variable message

localStorage.getItem() :

Used to fetch stored data from local storage.

Syntax:

localStorage.getItem( ‘variable\_name’);

returns the value stored in the variable.

localStorage.removeItem():

Used to remove a string from local storage.

Syntax:

localStorage.removeItem(‘variable\_name’);

## Storing objects in local storage:

If we need to store an obj into local storage directly, its not possible as the setItem method only stores strings. To overcome this issue we can convert our obj into a JSON string using the stringify methods and store that in turn into local storage.

strval=JSON.stringify( obj); //convert JS obj to JSON string

localStorage.setItem( ‘ obj ‘ , strval); // store the string into local storage

now to get the item back as an obj:

strval = localStorage.getItem( ‘obj’); //fetch item back as a string

obj=JSON.parse(strval); //convert back the string into an obj

# Document Object Model (DOM):

The DOM combines the HTMl code into the javascript and thus gives you access to make alterations onto the webpage via javascript.

It treats every element in the HTML document as an object and thus the name : Document Object Model.

Example:

document.title= ‘ new title’; //changes the current title of webpage into ‘new title’

document.body.innerHTML= ‘hello’ //changes the whole HTML code inside body into just ‘hello’

The DOM returns the whole element as an object for the JS to perform functions on.

document.body selects the entire body element from the html.

The innerHTML attribute is used to replace the HTML code of the selected object.

The document.selectQuery(‘element\_name’) selects the first element of the given name, which can then be altered.

We can store the returned object into a object in js ad operate further functionalities

We can differentiate between different elements of same kind by using classes/id.

NOTE : prefer using either classes or id , differently for CSS and JS.

The classList method for an HTML object can be used to modify class to an selected element.

HTMLobj.classList.add(“new\_class\_name”); the add() method is used to add a class to the given HTMLobj.

HTMLobj.classList.remove(“class\_name”); the remove() method removes the given class name.

Preference of usage:

Store the required html element as an object : objHTML=document.querySelector(‘class/id’);

Then use the objHTML variable for further functions.

## Input element as an object in js:

The input element doesn’t have a .innnerHTML method, but rather has .value that outputs the input value given in the element.

The default output of the value function in a string , to convert any input value into an number the Number(var) function is used.

### Onkeydown :

The html has a onkeydown=”code” attribute which can specify behaviour for when any key from keyboard is pressed for the specified element.

These kind of attributes are called **EVENT LISTENERS**.

More such event listeners:

* onclick
* onscroll
* onmouseenter
* onmouseleave
* etc,.

the event listeners all have an event object which further has event variables.

One such variable is the event.key which contains what data was entered in the input.

**STUDY MORE ABOUT EVENT VARIABLES**

# CSS with JS:

We can add and remove classes for required elements and have class wise CSS declared which will come into force when a class for a particular element is added.