**Js code for print values :**

First import file to html like :

<script src="index.js"></script>

**Then for print values :**

Syntax : alert();

alert("Js excuted");

var firstnum = 78,secondnum=45;

var div=firstnum/secondnum;

alert(div);

the alert command lets you print your message which popup on your browser when you refresh your browser moreover you can also use window.alert which works same as alert does.

You can also check other functions apart from alert by saying windows. (dot) by saying this list of other usable commands are popup in your vs code.

**NODE JS CLI WORK :**

to run your code in cli (command line interface) rather than on browser you need node js for that which run your code on serverside.for that install node js first and then follow below steps to execute your code in Command prompt or cli.

Syntax : console.log();

* 1.write in your vs code rather than alert write : console.log(“your input”); and save it.
* Open cmd where your folder of project exist
* 2.open your folder where your project is placed then on command prompt write : node -v
* 3.to output your code write : node space yourfilename.js
* 4.output printed

console.log("node working");

you can also view result of console.log on browser by going inpect and then there you can see output.

**Document.write**

It is used to display message into your browser not as popup or in cli in node.which is not used professionally but have a look at it.

Syntax : document.write();

document.write("Hey there\n");

document.write("\nsum is ->"+(50+25));

**Undefined and null :**

Undefined means variable is declared but value is not initialize.

Null means value is initialized memory is allocated but there is no value in variable.

Var value=null; (null form)

Var varlue; (undefined)

**Dynamic datatype :**

Since we don’t need to point a variable about a datatype either we are storing string or number its interpreter/engine automatically detects what datatype you assign and initialize.but professionally you would.

Other datatypes include const , let.

**Typeof operator :**

Let you know the datatype of variable you wanna know. Write typeof in consol or else where with your variable / identifier name.

var string="values";

var floated=31.22;

var booll=true;

console.log(typeof string);

console.log(typeof floated);

console.log(typeof booll);

var booll=32;

console.log(typeof booll);

**Prompt :**

Takes user input

Syntax : var identifirier= prompt(“massage”);

And display your userinnput through alert or consol.log or document.write etc.

var firstname=prompt("What is your first name");

var lastname=prompt("What is your first name");

alert(firstname+ " "+lastname);

**conditions :**

Boolean used or others are used in conditions which is used as == or === or < or > != etc

== it is used to match value given in input with the value already intitialized

var ask=prompt("Enter which is the samallest city in the world");

var score=null;

var userIQ;

if(ask == "Vetican"){

    score++;

    userIQ="Genius";

    alert("Correct!"+" "+userIQ+" "+score);

}

=== it works same as == but checks datatype as well should be same as input and intialiazed value.

var vari =prompt("where do you live ");

if(vari==="pakistan")

{

    alert("Correct!");

}

else{

    alert("no correct!");

}

**Expression :**

Combination of values variable functions.those things concatenate or combine and give a single result in output called expression,

Ie. String1+string2 or divide = 3/3; etc.

**Variable legal and illegal names :**

Variable cant contain space.

Variable’s first character cant be number or special characters second character would be.

Underscore can be used if want space between the name of variable.

Example :

Var %age , var \_age , var na$me; , var age1; , var first\_name; (all are allowed mentioned on left)

Var your name , var 2name (this are not allowed)

Comments :

Comments used to document your code or for explanation to the viewer of your code not concerned with the actual code or machine.There are two types that you can comment.

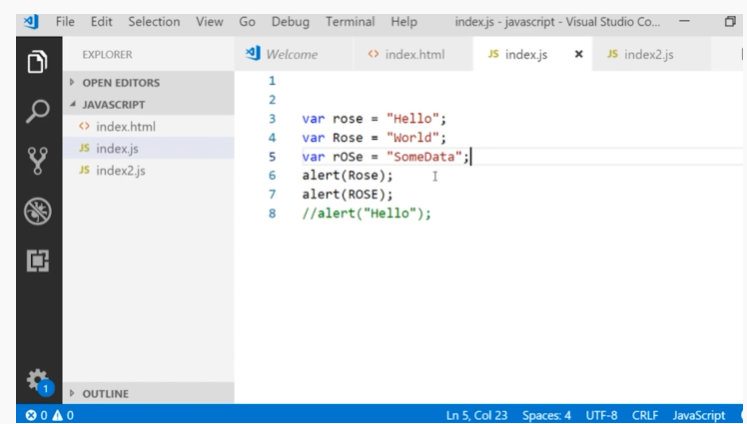
Single line comment : //abc

Multiline comment : /\* var r=a +b

C= r; \*/

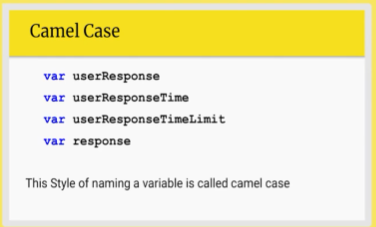
**Case sensitive:**

Means there is a difference between capital Karachi or small letter karachi both are different.



**Camel case :**

Means you have same name of variable in multiple times you can use it as var second have a same variable as original one but have some more characters in second variable with one capital letter and so on as you want this is called camel case.



**convert string into decimal :**

when we input the value if it is some number and you want it to add with the already predefined number then it will concatenate both value because it reads your input as strings.

There are some methods to solve a problem of arithmetic operation to be applicable rather than concatenation ultimately which is not our intention:

Use : **parseint(value)**

this function convert string into number which will then be used to apply arithmetic operations such as + - / \* etc.

for fractional value we use :

**parsefloat(value)**

which convert your user input fractional string into fractional number to apply arithmetic operations.

We have an alternative of above two function which is :

**number(value);** used for both integer and float values.

Lets see examples now :

var age = prompt("enter your age ");

var per= prompt("enter percentage ");

var num1=4;

var  sum =parseInt(age) + num1;

var num2=8

var sub=parseFloat(per) - num2;

alert("     "+sum);

alert(sub);

another example :

var age = prompt("enter your age ");

var per= prompt("enter percentage ");

var num1=4;

var  sum =Number(age) + num1;

var num2=8

var sub=Number(per) - num2;

alert("     "+sum);

alert(sub);

***Stack:***

It is a form of data structure follows last in first out approach.

To add element in stack we call it push and to retrieve / remove element from array called pop.

Example :

var values =[];

values.push("car");

values.push("bike");

values.push("truck");

values.push("tanker");

values.push("airplane");

console.log(values);   // print all values in stack

console.log(values.length);  // length 5

var v1=values.pop();   // airplane removed

var v2=values.pop();   // tanker removed

console.log(v1);  //printing poped value

console.log(v2);  //printing poped value

console.log(values);   // print all values in stack

console.log(values.length);  // length 3

***Queue:***

A linear data structure follows first in first out concept.

To add element in stack we call it push and to retrieve / remove element from array called shift.

Example :

// queue

var values1 =[];

values1.push("sam");

values1.push("mike");

values1.push("zack");

values1.push("finn");

values1.push("frade");

console.log(values1);   // print all values in stack

console.log(values1.length);  // length 5

var v3=values1.shift();   // sam removed

var v4=values1.shift();   // mike removed

console.log(v3);  //printing poped value

console.log(v4);  //printing poped value

console.log(values1);   // print all values in stack

console.log(values1.length);  // length 3

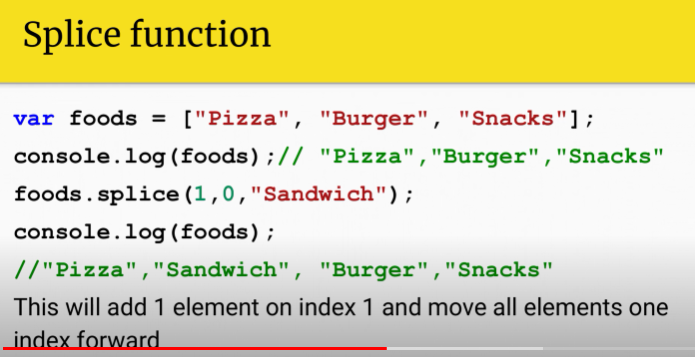
unshift function:

add element at the top of the stack

values1.unshift("jenny"); // added at the top of the stack

console.log(values1);

console.log(values1[0]);  //showed here



Example :

var cities=["karachi","peshawar","islamabad","Nawabshah"];

console.log(cities);  // karachi  peshawar islamamabad nawabshah

cities.splice(1,0,"sukkur");

console.log(cities);   // karachi sukkur peshawar islamamabad nawabshah

cities.splice(2,2,"Gilgit","hyderabad");  // adding gilgit at index 3 and removing peshawar and islamabad from list

console.log(cities); // karachi sukkur gilgit hydrabad nawabshah

var arr=["same","casey","Dewayne","Rexha"];

for(var i=0;i<arr.length;i++){

    console.log(arr[i]);  // printing array values one by one

}

var arr2=[45,44,45,96,32,1,2,47,58,56,22];

for(var i2=0;i2<arr2.length;i2++){

    arr2[i2]=arr2[i2]\*2;

}

for(var j=0;j<arr2.length;j++){

    console.log(arr2[j]);  // printing array values one by one

}

var arr2=[45,44,45,96,32,1,2,47,58,56,22];

for(var i2=0;i2<arr2.length;i2++){

    arr2[i2]=arr2[i2]\*2;

}

for(var j=0;j<arr2.length;j++){

    console.log(arr2[j]);  // printing array values one by one

}

***Lower case and upper case:***

 var cityToCheck = prompt("Enter your city");

 var cityToCheck = prompt("Enter your city");

 var cityToCheck = prompt("Enter your city");

 cityToCheck = cityToCheck.toLowerCase();

 //cityToCheck = cityToCheck.toUpperCase();

 //var cleanestCities = ["cheyenne", "santa fe", "tucson", "great falls", "honolulu"];

 for (var i = 0; i < cityToCheck.length; i++) {

 console.log(cityToCheck[i]);

}

***Mathmatical function calculations :***

Round :

Graphical user interface, text, application, email

Description automatically generated

var num=(54+33+1-11)/21;

console.log(num); //3.6666666666666665

var rounded=Math.round(num);

console.log(rounded);  //4

Random function :

Generates number under 0 .to generate above number multiply it by another number you want to have number grater than 0

var random1=Math.random();

var another=(random1\*6); //here we found a random number grater than 0

var another2=Math.floor(another); //round of the number

var ask=parseInt(prompt("Enter a number to guess for a dice role "));

if(ask===another2){

    console.log("you had won a prize bond !");

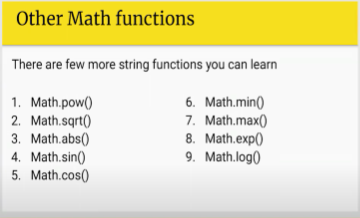
}

else{

    console.log("better luck next time !");

}

Other match functions :



var powerr=Math.sqrt(rounded);

console.log(powerr);  //2

var logg=Math.log(rounded);

console.log(logg);  //1.33

var expp=Math.exp(rounded);

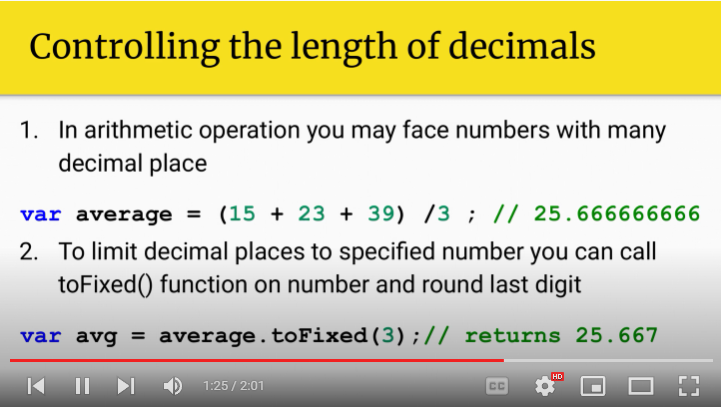
console.log(expp);  // 54.59 =

var num2=[45,55,57,99,63,5];

var found=Math.max(num2);

console.log(found);

***to fix function :***

var round3=num.toFixed(2);

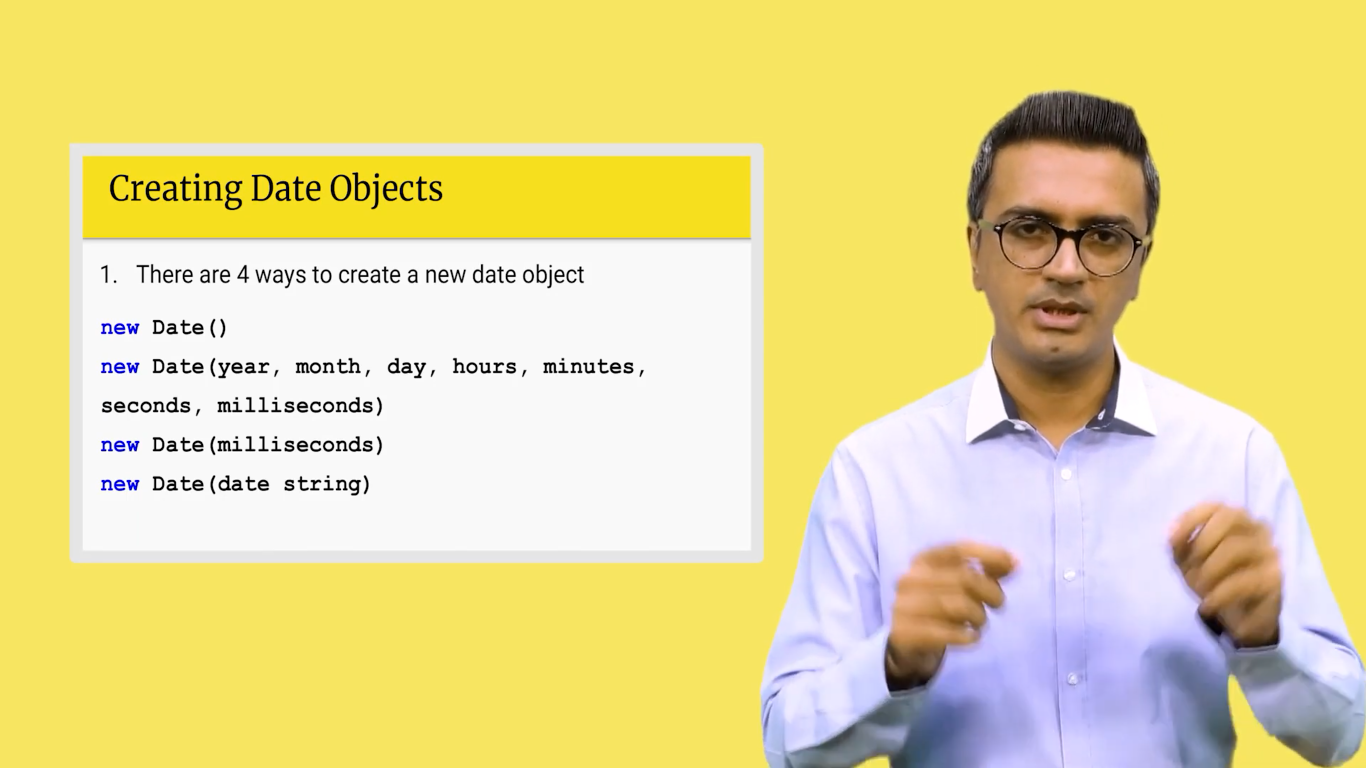
console.log(round3);  //3/67

***date function:***

if on system browser return system’s current date.if on server then where server deployed it return there’s date. example :

var date=new Date(); // constructor

console.log(date);  //Mon Aug 08 2022 23:13:05 GMT+0500 (Pakistan Standard Time)



Examples :

var anotherdate=new Date("2020/7/5");

console.log(anotherdate);//Sun Jul 05 2020 00:00:00 GMT+0500 (Pakistan Standard Time)

var anotherdate=new Date("2020/7/5 10:13:44:120");

console.log(anotherdate); //Sun Jul 05 2020 10:13:44 GMT+0500 (Pakistan Standard Time)

**unix time :**

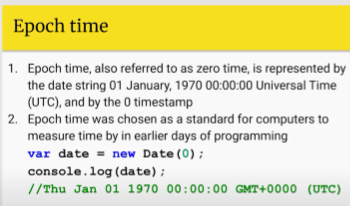
starting from 1st january 1970 since today it gives you how many milliseconds were passed.

Graphical user interface, text, application, email

Description automatically generated

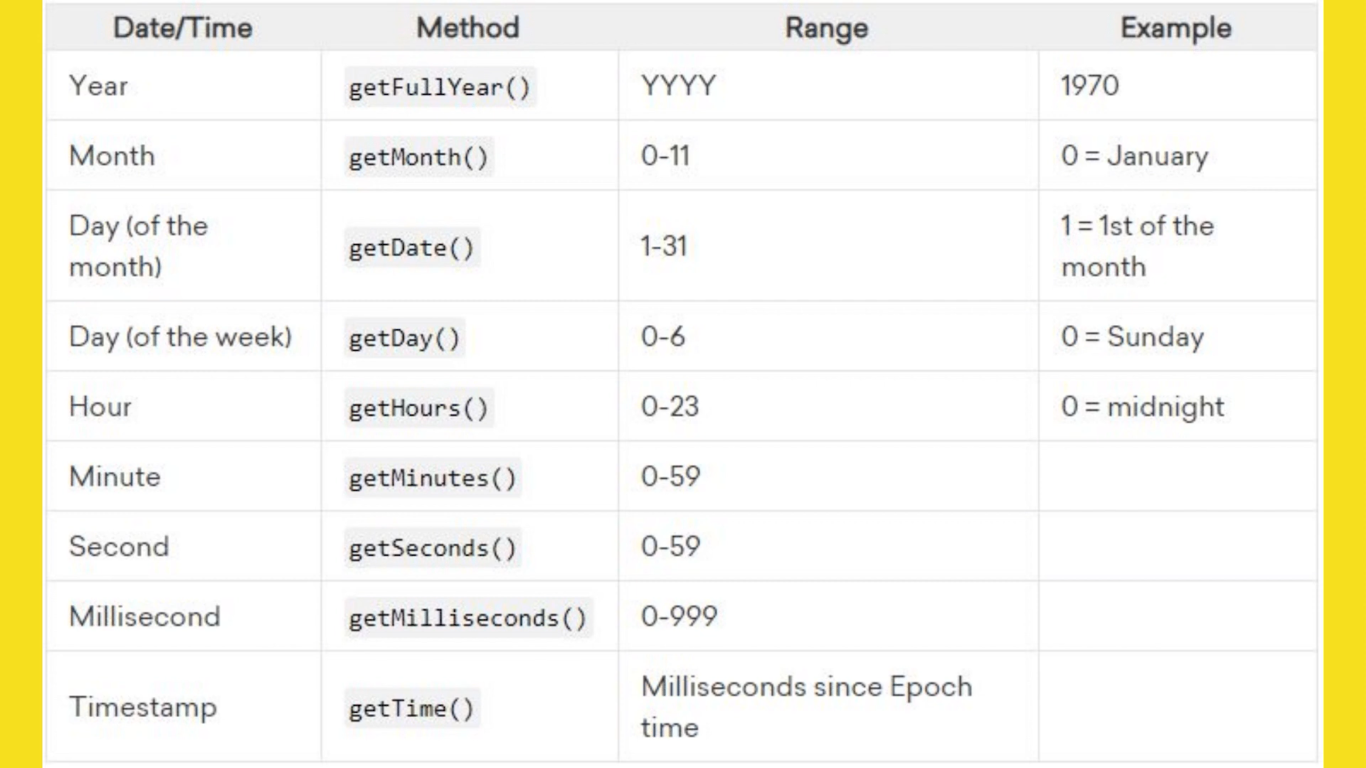
**Epoch time :**

The time settled in javascript which is starting from first jan 1970



**Accessing particular component of the date :**

Like just want hour or month or year or day.



Text, table

Description automatically generated

Examples :

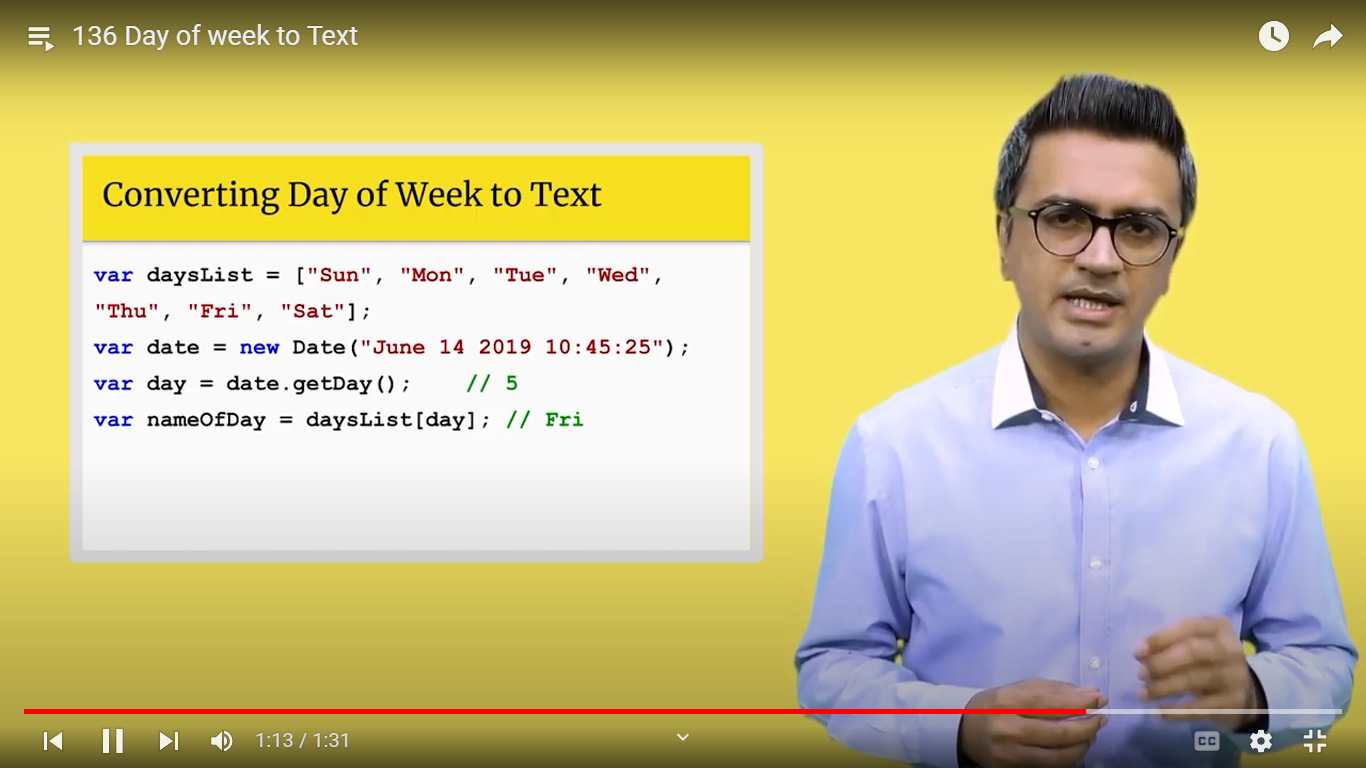
console.log(date.getTime()); // 1660132270289 mili seconds passed sice 1st jan 1970 at 4:51 , 8/10/2002

console.log(date.getDay());     // day of week since this is second week's third day from 7 to 10 its 3rd day

console.log(date.getFullYear());

console.log(date.getHours());

console.log(date.getMonth()); // indexing starts from 0 so month +1 becomes 8th month

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

// time diffrence calculation to know how many hours remain from due data of something

var date1=new Date("2020/7/5 10:13:44:120");

var date2=new Date("2020/7/15 10:13:44:120");

var diff=date2.getTime()- date1.getTime();

var secondsinaday=60\*60\*24\*1000;

var daydiff=diff/secondsinaday;

console.log(daydiff);  // 10 days

***Placing scripts :***

In html code you can scatter your pieces of codes anywhere In html file In order to perform operations as you want no need to have a separate javascript file and deliver its source to the html code to see results.

But coders prefers to work separately as many javascript files as you want and drop out their source anywhere you want it in html code.

 <script>

            function sayHi() {

            alert("Hello world!");

            }

           function sayBye() {

            alert("Buh-bye!");

            }

            sayHi();  // calling funtion

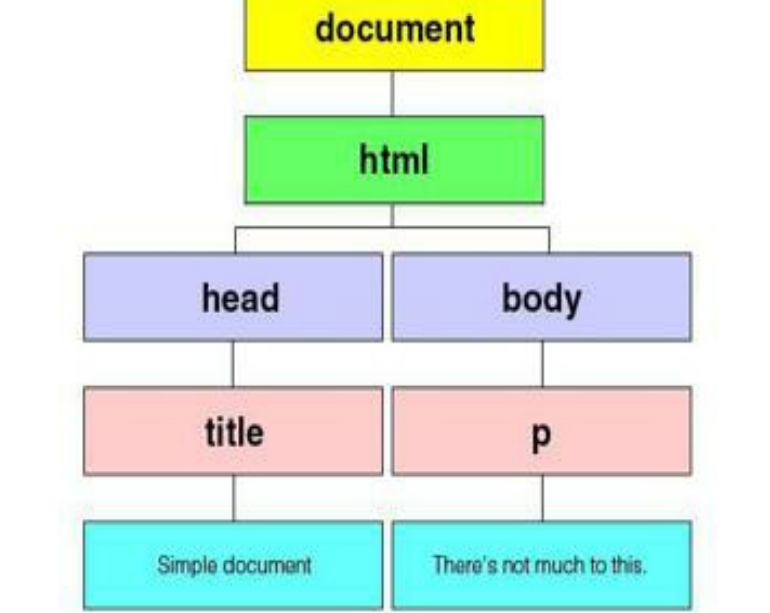
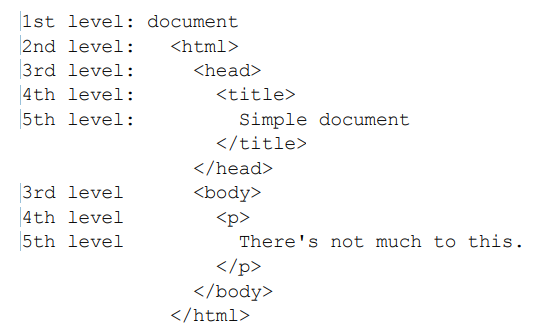
            sayBye(); // calling function

           </script>

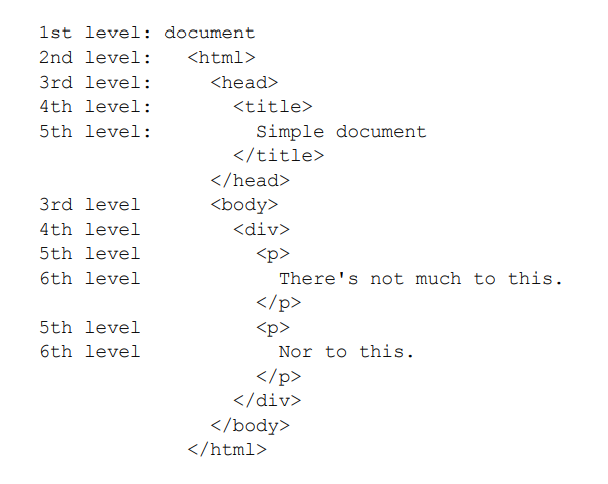
***Data object Manipulation:***

The DOM is an organization chart, created automatically by the browser when your web page loads, for the whole web page. All the things on your web page—the tags, the text blocks, the images, the links, the tables, the style attributes, and more—have spots on this organization chart.

Here's a simplified web page. I've indented the different levels in the hierarchy. The three top levels of the DOM are always the same for a standard web page.



***Child and parents in DOM:***



Except for the document node, each node is enclosed within another node. The and nodes are enclosed within the node. The node is enclosed within the node. Two nodes are enclosed within the node. And a text node is enclosed within each of the nodes.and so on

***Global time and date format (unix time , jwt) :***

const unixTimestamp = 1691926371;

const date = new Date(unixTimestamp \* 1000); // Convert to milliseconds

// Adjust for Pakistan's local time (UTC+5)

//date.setUTCMinutes(date.getUTCMinutes() + 5 \* 60); // Add 5 hours

// Format the date and time

const options = {

hour: "numeric",

minute: "numeric",

second: "numeric",

timeZoneName: "short",

};

const localTime = date.toLocaleTimeString("en-PK", options);

console.log(localTime);

***converting values into array:***

The JavaScript **Array from()** method returns an Array object from any object with a length property or an iterable object.

**Syntax :**

Array.from(object, mapFunction, thisValue)

**Example:**

 var myInt = 235345;

    // Getting the string as a parameter

    // and typecasting it into an integer

    let myFunc = num => Number(num);

    var intArr = Array.from(String(myInt), myFunc);

    // Print the result array

    console.log(intArr);

**Output:**

[2, 3, 5, 3, 4, 5 ]

**Example:**

const numbersArray = Array.from(

{ length: 5 },

(\_, index) => index + 1

);

console.log(numbersArray)

Output:

[ 1, 2, 3, 4, 5 ]

**Alternate of last example :**

const numbersArray = [];

for (let i = 1; i <= pageNumber; i++) {

numbersArray.push(i);

}

Output:

[ 1, 2, 3, 4, 5 ]