JavaScript

- JavaScript is a scripting language as well as programming language which is readable, analyzable & executable by browser itself.
- It is also programming language supports OOPs concept as well.
- It is used to adds interactivity & automate the task to the web site rather than building full scale application.
- JavaScript is used as Front-End as well as back-end.
- In Front-End:- react js, ~~Vue js~~, Angular js, Vue js.
- In back-end:- node js, Express js

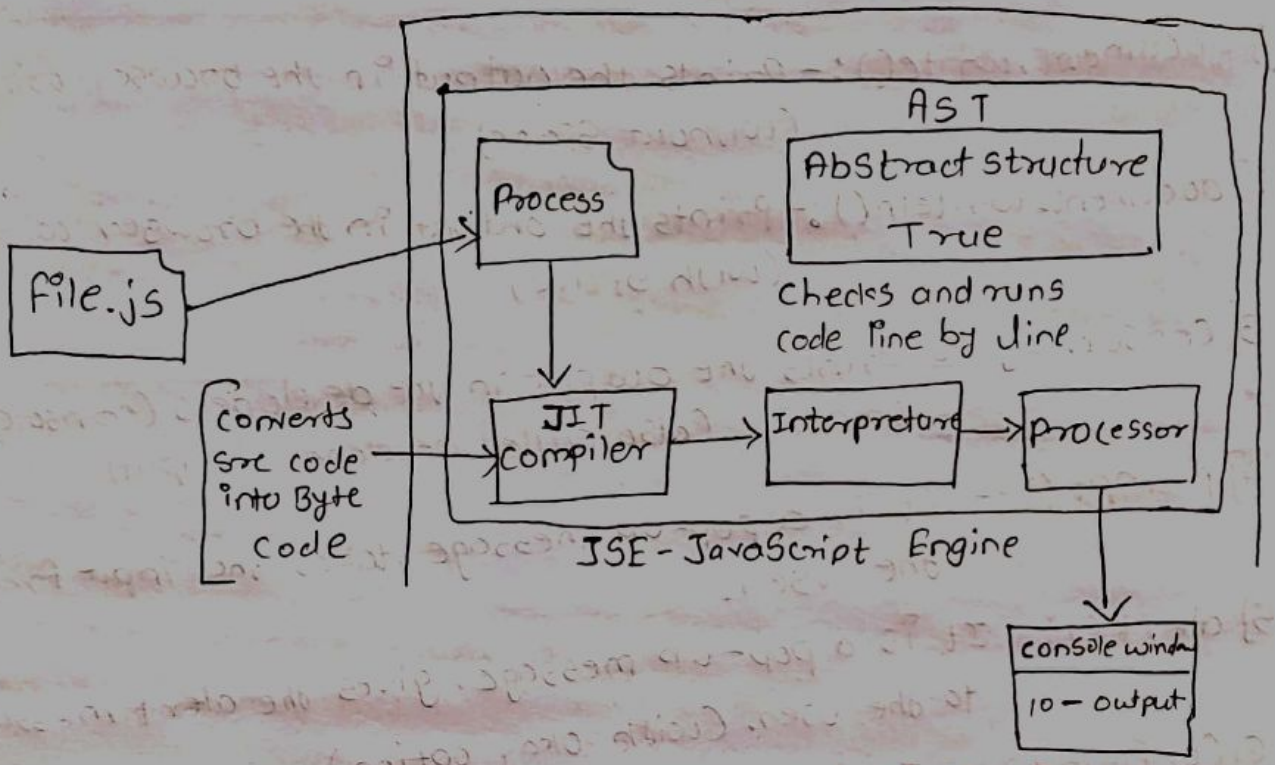
* History of JavaScript:-

- A browser company Nedscape hired the developer called Brenden Eich ~~for~~ to develop a such scripting language for their browser.
- Brenden Eich develop a scripting language called as Mocha in 1995 with in 15 days.
- After it is named as live script but as a marketing startergy the company named it as java script.
- Later Nedscape company handled java script to ECMA (Europian Computer Manufacturing Association).
- ECMA company releases the new version of java script every year in the month of june.
- The trade mark of java script is owned by ~~Ora~~ oracle.

* Features of JavaScript:-

1] It is Scripting language designed for websites.
2] It is client side scripting language.
3] It is loosely or weakly typed language.
4] It is Dynamically typed language.
5] It is interpreted language.
6] It is directly Embeded to HTML File.

*How Javascript engine works?



*How Javascript File sent to Javascript Engine?

- When JS file executed on the Browser, the browser send JS file to JS engine.
- In this process, the file JS is first sent to the PARSER and the original file structure trees (Abstract structure trees).
- The PARSER divides the JS code into parts and send one-by-one to the compiler.
- The JIT compiler shows syntactical errors, then converts source code into Byte code and sent to Interpreter.
- The Interpreter checks the Byte code line-to-line and executes each line at that time itself by sending it to the processor.
- And after that processor will send that code into console window or document for output.

* Output methods of JavaScript

1] document.write() :- Prints the output in the browser window (without space)

2] document.writeln() :- Prints the output in the browser window (with space)

3] console.log() :- Prints the output in the developers (console) window (also called developer output)

4] prompt() :- It is a pop-up message, takes the input from the user.

5] alert() :- It is a pop-up message, gives the alert message to the user. (with okay option).

6] Confirm() :- It is a pop-up message, gives the confirm message (with okay and cancle option)

* JavaScript Advantages :-

1] It is easy to learn and implement

2] It is very popular programming as well as scripting language.

3] It reduces server load.

4] It is ~~versitile~~ in nature.
versatile

5] JavaScript Create reach interfaces.

* Uses of JavaScript :-

1] Web Applications.

2] Web development.

3] Mobile Application.

4] Games development.

5] Servar Applications.

- **Tokens :-** Tokens are the smallest unit of any programming language.

Keywords
- Pre-defined / reserved words
- Should be lowercase

Example: var, let const, if, while

Identifiers
- Can't starts with numbers (but it can have numbers)
- Except ($) and (_) no other special character is allowed.
- keywords can not be used as identifiers.

Example:- name, age, skills

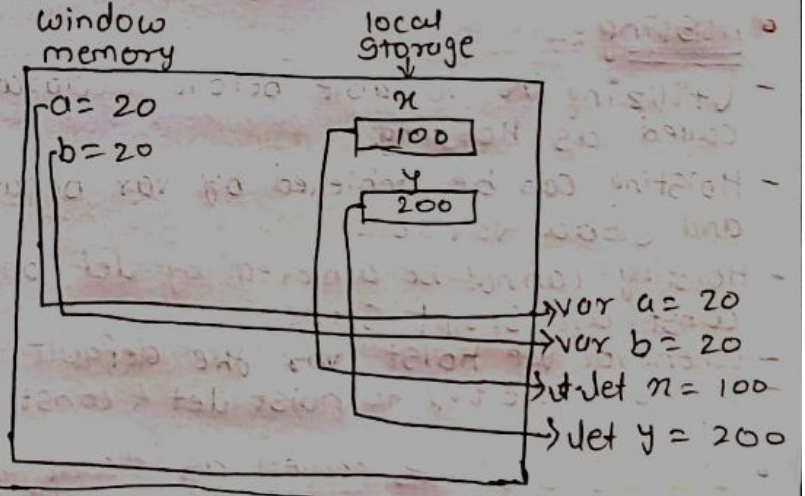
Literals
The data which is used in JS Program
- Number
- Boolean
- Null
- Undefined
- Object
- BigInt
- Symbol

Example:
```
name = "vaibhav";
age = 23
```

- **Variables :-**

- Variables are the container where we can store data
- In Javascript you can store any kind of datatype inside variables.
- There are only three ways to declare variables :
  (i) var (ii) let (iii) const.

window object → Supermost object or variable in JS



var	let	const
• var is a global scope also a function scope.	let is a script scopes also a block scope	const is a script scope also a block scope.
• we can declare multiple variable with same name (The most recently created variable will be used.	we can not declare two variables with the same name with in a block	we can not declare two variable with the same name with in a block.

• The value of the variable can be modified
• var a = 10;
a = 20;
console.log(a)
	The value of the variable can be modified
	let a = 10;
	a = 20;
	console.log(a)
		The value of the variable can not be modified
		const a = 10;
		a = 20;
		console.log(a)

var	let	const
• we can declare var without initialization eg. var x;	we can declare let without initialization eg:- let y;	we can not declare const without initialization eg:- const 2;
• The variable declared using var. Belongs to global scopes (window), we can access them using window object Ex: var a=10; console.log(window.a); //10	The variable declare using let does not belongs scope we can not use them with the help of window. (script / block) Ex:- let b=20; (console.log(window.b); //undefined	The variable declare using const does not belongs to global scope.

- **Window object:-**
  - When a JS file is given to browser by default a global window object is created and the reference is stored in window variable. The global object consist of pre-defined members (Functions & variables) which belongs to browser window.
  - Any member we declare var in JS file is added inside global window object by JS engine. so we can use member with the help of window.
  - var a=10; // var is added into the global window object.
  - console.log(window a); // 10
- **Hoisting:-**
  - Utilizing the variable before declaration & initialization is called as Hoisting.
  - Hoisting can be achieved by var because var is a global scope and global variable.
  - Hoisting cannot be achieved by let and const because let and const are script scope.
  - whenever we hoist var the default is undefined.
  - whenever we try to hoist let & const the default us uncaught reference:
- **Literals also called as datatypes**
  (Datatypes in Javascript)
  - we have 8 datatypes in JS those are string, number, Boolean, null, undefind, object, bigInt, symbol.

1. Strings :- In string we store characters values like name, place.
2. Number:- In number we store number values like, 1, 1.1, 20
3. Boolean:- This type of datatypes returns the true or false value.
4. Undefined:- A variable in Javascript that is without any value has a value of undefined
5. Null:- In Javascript null is "nothing". It is suppose to be something that doesn't exist. In Javascript the data type of null is an object.

6. BigInt :- With BigInt you can safely store operate on large integers beyond the integer limit for numbers
Ex:- BigInt ("123113432432 4");
7. Symbol:- In represents a unique identifiers as be used in various ways
Symbol are used to create object properties for, when you want to assign a unique identifiers to an object.
8. Object:- It is an non-primitive datatype that consist of unordered key-value pair which object may include any mix of these fundamental data types and reference data types, an object are more complicated.

* Operators:- Operators are the symbols which performs specific operations on the operands.

- Types of operators:-
  - Arithmetic operator
  - Assignment operator
  - Relational operator
  - Logical operator
  - Ternary / conditional operator.

- Arithmetic operator :- + - * / %
- Assignment operator :- = += -= *= /= %=
- Relational operator :- < > <= >= == !=
- Logical operator :- && ||
- Ternary / conditional operator :- (condition) ? expression1 : expressio

* Javascript Functions:-
- Functions are the block of statements which will used to perform some specific task.
- Functions get executed whenever it is called or invoked.
- Function names can contain letters, digits, underscores and dollar signs (same rules as identifiers)
- The ~~para~~ parentheses may include parameter names, separated by commas: (Parameter1, parameter2)
- Function arguments are the values retained by the function when it is invoked.
- The code to be executed by the function is placed inside curly brackets: { }.

- Anonymous Function:- The functions which does not have function name is called as Anonymous function.

-Syntax :-
```
function() {
    statements;
};
```

- Named Function:- The function which has function name is defined as named function.

-Syntax :-
```
function function_name() {
    statements; }
function_name();
```

• First class functions:-

- The function which is assigned as a value to be variable then the expression is called as function with expression.
- Syntax :-

```
var variable_name = function (){
        Statements;
    }
    variable_name();
```

- The function which is assigned as a value to a variable then that function is called as first class function.
- First class function is also called as first citizen whatever the function ~~is~~ it may be, if it is assigned to a variable then it is first class function.

• Arrow function:-

- A function having a fat arrows is called as Arrows functions To reduce the function syntax we will go for Arrow function.
- Syntax :-

```
variable_name = () => {
        statements;
    } variable_name();
```

• Nested function:-

- A function inside another function is called as nested function

Note:- One ~~fn~~ inside another function not many inside one function.

```
Function function_name(){
    Statements;
    function function_name2(){
        Statements;
        function function_name3(){
            Statements;
        }
        function_name3();
    }
    function_name2();
}
function_name3();
```

• Immediate Invoke Function Execution:-

- IIFE is a function executes a functions which is enclosed within the parathensis ().
- Syntax:-

```
(function function_name(){
    Statements; })
();
```

- IIFE should be called immediately right after the declaration
- IIFE should be reused, it is only one time usable function.
- IIFE can have function name but it can not be called with function name.
- Anonymous function, Named function, Nested function, Arrow function can be IIFE

Higher Order Functions:-

- The function which takes another function as an argument is called as Higher order function.
- Callback function: The function which is passed as an argument to the function.

Note:- Higher order functions accepts callback function as an argument.
Higher order function calls the call back function.

Javascript Array

- Array is a collection of different elements (data). We can store multiple array elements as well as both Homogenous & Heterogenous data.
- Array index value starts from 0.
- In Javascript we can create array in 3 ways.
  1] By using Array literal.
  2] By using Array constructor. (using new keyword)
  3] By creating a length in constructor array and assigning a value from index.

1] Javascript Array literal

- The Syntax of creating array using array literal is
- var arrayname = [value1, value2, ... value N];
- values are contained inside [] and seperated by , (comma).
- The .length property returns the length of an array

eg.
```
let numbers = [10, 20, 30, 40]
console.log(numbers);
```

2] JavaScript Array constructor (new keyword):-

- Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

eg :- let task = new Array (10, 20, 30)
console.log (task);

3] By creating a length in constructor ~~and~~ array and assigning a value from index.

- let ageOfEmployee = new Array (3)

```
ageOfEmployee[0] = 10;
ageOfEmployee[1] = 20;
ageOfEmployee[2] = 30;
Console.log(ageOfEmployee);
```

- Here, in this way of creating an array you will provide on length of an array in Parunthesis ().
- which means you can enter how many array elements you want, more ~~tho~~ than the give size also.

* JavaScript Array Methods.

Push() :- It will insert an element of an array at the end.
trainers.push("Array data");

Unshift() :- It will insert an element of an array at the first.
trainers.unshift("Array value");

Pop() :- It will remove elements of array from the end.
trainers.pop();

Shift() :- It will remove elements of array from the start.
trainers.shift();

indexOf() :- It will return a index of particular element.

```
console.log(trainers.indexOf("array data");
```

includes() :- It will check whether the particular element is present in array or not.

```
console.log(trainers.includes("array data");
```

Slice() :- It will give slice of an array and it will not affect the original array.

```
console.log(trainers.slice("index value," "length of array");
```

Splice() :- It is used to add or remove elements from an array and splice will affect the original array.

```
console.log(trainers.splice("index value," " length of array", "adding data"
```

Join() :- It is used to join all elements of an array to the String.

```
console.log(trainers.join());
```

Concat() :- It is used to join/concat two or more arrays

```
let num1 = [10, 20, 30];
let num2 = [40, 50, 60];
let res = num1.concat(num2);
console.log(res);
```

Sort() :- It is used to sort array elements in alphabetic order

```
console.log(trainers.sort());
```

Reverse() :- It is used to reverse an array.

```
console.log(trainers.reverse());
```

* Filter () :- Filter method in Java Script is used to create new array elements with test pass condition provided by function and it filters the new Array elements from original array.

- It does not modify its original array instead of this it returns new array with filter elements.

1] eg:- let values = [10, 20, 40, 63, 53, 67]
let newvalues = values.filter((n) => {
return n % 2 == 0
})
//O/P :- [10, 20, 40]
console.log(new values);

* ~~Map ()~~ :-

2] let values = ["dnyanu", "mayuri", "rohini", "meghana"]
let console.log(value);
let newnames = value.filter((n) => {
return (n.includes('m'))
})
o/p :- ["mayuri", "meghana"]

* map() :- map() is used to create new array by applying a function to every element in existing array compatsary. If does not modify original array instead it returns new array with output

eg :- 1] let values1 = [10, 20, 30, 40]
let new value2 = values1.map((n) => {
return n * 10
})
O/P :- [100, 200, 300, 400]
console.log(newvalue2)

2] let names = ["dnyanu", "mayuri", "rohini", "meghana"]
console.log(names);
let newnames = value.map(m(n) => {
return (n.includes('m'))
})
o/p :- [" false, true, false, true]

Reduce():- The reduce() method in javascript is used to apply a function to an accumulator and each element in an array, in order to reduce it to a single value. It iterates over each element in the array and accumulates a value based on the provided function.

eg:-
```
let task1 = [1, 2, 3, 4, 5, 6]
let task2 = task1.reduce(first, second) => {
    return first * second;
}, 10);
console.log(task2);
```

Objects In Javascript

- ~~Creati~~ A javascript object is an entity having state and behaviour (properties and methods).
  for eg:- car, pen, bike, chair etc.
- Javascript is an object-based language. Everything is an object in javascript.
- In objects, we can store the data in the form of key value pair.

Creating Objects in javascript:-

There are 3 ways to create objects.

1) By object literal.
2) By create Instance of Object directly (using new keyword)
3) By using constructor function.

1) Javascript object by object literal.

The Syntax of creating object using object literal is given below.

```
<script>
    let Student = {
                    stdname : "vaibhav",
                    stdId : 101;
                    Subject : "web Tech"
                };
</script>
```

2] By creating instance of object with new constructor-

- The syntax of creating object directly is given below:

var objectname = new object();

Here, new keyword is used to create object.

Example of creating object directly.

```
eg. let person = new object()
    Person.name = "vaibhav";
    Person.eid = 123
    console.log(Person);
```

3] By using an constructor function

- Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
- The this keyword refers to the current object.

The example of creating object by object constructor :-

```
eg function Employee (ename, eid, sal, dept){
        this.ename = ename;
        this.eid = eid;
        this.sal = sal;
        this.dept = dept;
   }
   let E1 = new Employee('Avinash', 123, 2000, 'Java')
   let E2 = new Employee('Harshal', 124, 3000, 'Selenium')
   colsole consolet
      Console.log(E1);
      Console.log(E2);
```

* How to update property value on object

```
eg. let student = {
        Stdname : "vaibhav",
        stdid : 121,
        institute : 'jspiders',
        Subject : 'webtech',
    };
    Student.stdname = 'ABCD';
```

* How to delete the property in object ~~of~~ with delete keyword

- let Student={
    Stdname: 'vaibhav',
    Stdid: 121
  };
  delete Student.Stdname

* objects can also have methods
* Methods are actions that can be performed on object.

* Object Methods:-

1. keys:- It will return array of keys.
   console.log(object.keys(employee));
2. values:- It will return array of values.
   console.log(object.values(employee));
3. Entries: It will return array of keys and values in objects like array format.
   console.log(object.entries(employee));
4. Seal: We can only update the properties we can't delete.
   console.log(object.seal(employee));
5. isSealed: It is used to check whether the particular object is sealed or not.
   console.log(object.issealed(employee));
6. Freeze: ~~It is~~ we cannot do any modification in an object like updation, addition & deletion
   console.log(object.freeze(employee));
7. isfrozen:- It is used to check whether the particular object is frozen or not.
   console.log(object.isfrozen(employee));

* Call():-

- call() is used to invoke a specified function with this values and a list of arguments. It allows you to call a function as it is method of specific object. even if the function is not originally a method of that object.

Syntax:- function.call(Targeted object name, Arguments);

eg. let person1 = {
    name = 'vaibhav',
    Surname = 'chaudhari',
}
let person2 = {
    name = 'Collin',
    Surname = 'Goves',
    Fullname : function(age, salary){
    console.log(this.name + " " + this.Surname + " His age is " + age +
        " And their salary is " + salary);
    }
}
Person2.Fullname(27, 1000);
Person2.Fullname.call(Person1, 22, 1000);

* apply():-

- apply() will work similar to call method but it accept argument in array.

Syntax:- function.apply(Targeted object name, [arg1, arg2]);

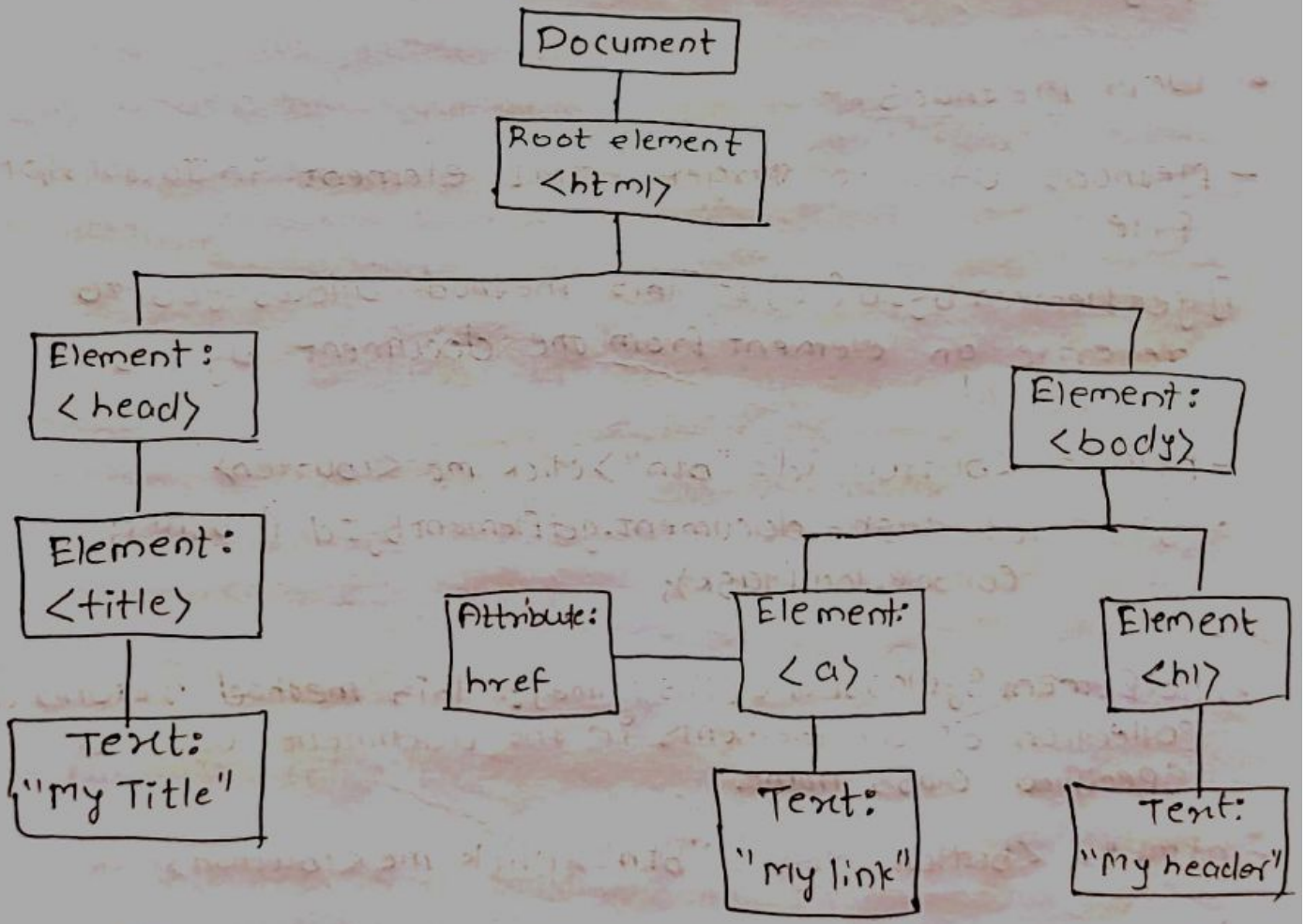
eg.
eg. person.Fullname(30, 1000);
    person.fullname.apply(person1, [24, 1000]);
    Person.Fullname.apply[person2, [22, 1000]];

* Data Object :-

- When a data object is created, a numbers of methods allows you to operats on it.
- ~~me~~ Data methods allows you to get and set the year, month, day hour, minute, second and milisecond of data objects, using with local time are ~~UI~~ UTC Time.
- Data objects are created with the new Data() constructor.
- In Javascript, the Data objects is used for working with date and times. It allows you to ~~op~~ create, manipulate, and formate date and times.
- const date = new Date();
- getfullyear() :- Returns the year(4 digits) of the year.
- ~~get~~ date.getMonth() :- Returns the month (0-11) of the year.
- date.getDate() :- Returns the day of the month (1-31) of the date.
- getDay() :- Returns the Day of the week (0-6) of the date (0 is sunday, 6 is Saturday)
- getHours() :- Returns the hours (0-23) of the time.
- getMinute() :- Returns the minutes (0-59) of the time.
- getSeconds() :- Returns the Seconds (0-59) of the time.
- getMilliSecords() :- Returns the milliseconds (0-999) of the time.
- getTime() :- Returns the number of milliseconds since January 1, 1970 (Unix timestamp)

* DOM (Document object Model)

- In JavaScript, the DOM (Document object Model) is a programming interface for web documents. It represents the structure of a document as a tree-like model. where each node is an object representing a part of the document, such as elements attributes and text.
- When a web page is loaded, the browser creade a Document Object model of the page
- The HTML Dom model is constructed as a tree of objects.



✱ Dom Manipulation

- JavaScript can change all the HTML elements in the page.
- JavaScript can change all the HTML attributes in the page
- Javascript can change all the CSS styles in the page.
- Javascript can remove existing HTML elements and attributes.
- Javascript can add new HTML elements and attributes
- Javascript can react to all existing HTML elements in the page.
- Javascript can create new HTML elements in the page.

• Dom methods :-

- Methods used to target H.TML element in JavaScript file.

1] getElementById(id) :- This method allows you to retrive an element from the document by its unique id.

- html :- <button id = "btn"> click me </button>
- js :- let task = document.getElementById ("btn");
  console.log (task);

2] getElementByClassName(className) :- This method returns a collection of all elements in the document with a specified class name.

- html :- <button class = "btn"> Click me </button>;
- js :-
  let task = document.getElement ByClassName ("btn");
  console.log (task);

3] getElementByTagName(tagName):- Returns collection of elements with the specified tag name.

- html :- <button class="btn">click me</button>;
- js :- let task = document.getElementByTagName("button");
  console.log(task);

4] querySelector(selector):-

- Returns the first element the matches a specified css selector.

- html:- <button class="btn">click me</button>;
- js:- ~~let task = document.g~~
  let task = document.querySelector(".btn");

5] querySelectorAll(selector):-

- Returns a Node List of all elements that match a specified CSS selector.

- html:- <button class="btn">click me</button>;
  <button class="btn">Click me</button>
  <button class="btn">Click me</button>
- js:- let task = doucment.querySelectorAll(".btn");
  console.log(task);

• What is HTML Collection?

- An HTML collection is an array-like collection (list) of HTML elements.
- The elements is a collection can be accessed by index (starts at 0).
- The length property returns the number of elements in the collection.

- What is Nodelist?

- They are a collection of nodes or elements, just like the books by the same author are a collection.
- In ~~the~~ technical terms, a NodeList is a group of nodes extracted from the Document object Model (DOM).
- And we can access these elements by forEach loop which you cannot do in HTML collection

- What is Prototype in javascript?

- In JavaScript, a prototype is an object that serves as a blueprint for other objects.
- It provides a mechanism for objects to inherit properties and methods from one another.

* The main difference between a for loop and a foreach loop is that.

- a for loop provides more control and flexibility, while a foreach loop is simpler and more specialized.

Syntax of for loop

```
for(let index=0; index<array.length; index++){
        const elements = array[index];
}
```

Syntax of forEach loop

```
array.forEach(element =>{
        });
```

- Changing HTML Elements
  - Changing inner text with innerHTML

html :- `<h1 id="hello">hello</h1>`

```
let task = document.getElementById("hello")
task.innerHTML = "hi im vaibhav";
```

// how to create text inside an element

```
btn.innerHTML = "Submit";
btn.innerText = "Reset";
```

~~bet~~ best way to create text inside an element.

```
let btnName = document.createTextNode("Buy Now");
btn.appendchild(btnName);
```

- innerTest :-

- returns all text contained by an element and all its child elements.
- innerHTML returns all text, including html tags, that is contained by an element.

- Changing styling with style and property

html :-

```
<h1 id = "hello" > hello</h1>
```

js :-

```
let task = document.getElementById("hello");
task.style.backgroundColor = "red";
```

- Changing attribute with setAttribute()

html :-

```
<h1 id = "h1" > hello</h1>
```

~~tes~~ let task = document.getElementById("h1");

attribute name and value

```
task.setAttribute("class", "heading");
console.log(task);
```

- Creating html elements by Dom in js

- First we have to create one element with createElement method and that we have to append that element on body with appendChild()

eg.
```
let ele= document.createElement("section");
document.body.appendChild(ele);
console.log(ele);
ele.innerHTML="im section";
```

- createElement():-
  - This method in Javascript is used to create a new HTML element with a specified tag name.
  - The methods takes a single arguments, which is the tag name of the element to be created.
    For example:- document.getElement("div") creates a new div elements

- appendChild():-
  - The appendchild method in Javascript is used to add a new child node to an existing parent node.
  - means the created html element we will add to body of our web page document.

- Deleting elements :-

```
let task= document.getElementById("list");
task.remove(task);
```

- remov():-
  the method removes a specified child node from the current node.

Tip:- The removed child node can be inserted later into any element in the same document.

• DOM Event Listener :-

- The addEventListener() method attaches an event handler to the specified element.
- The first parameter is the type of the event (like "click" or "mouseover" or any other HTML DOM Event.)
- The second parameter is the function we want to call when the event occurs.

ex.
```
element.addEventListener("click", function(){
    alert("Hello World!");
});
```

• DOM Events :-

- Event :- An action performed by the user on the webpage is known as an event.
- A Javascript can be executed when an event occurs, like when a user clicks on an HTML element.
- In the web development process, events refer to user actions, such as clicking a button, minimizing the browser window, or submitting a form. ~~Event~~
- Event handling in Javascript and Javascript frameworks allows us to respond to user actions and interactions and create dynamic and interactive websites.
- The addEventListener() method attaches an event handler to an element without overwritting existing event handlers. You can add many event handlers to one element. You can add many event handlers of the same type to one element.
  - mouseover
  - mouseout
  - click
  - Dblclick

  and so on.

• Form Events :-

1. Submit :- Triggered when a form is submitted. It's often used for form validation before data is sent to the server.

2. Input :- Triggered immediately when the value of an input element changes, allowing for real-time handling of user input.

3. Change :- Triggered when the value of a form element (input, select, textarea) changes. Commonly used for user input validation or dynamic updates.

4. Focus :- Triggered when an element receives focus, such as when a user clicks or tabs into an input field. Useful for providing feedback or enhancing the user experience.

5. Blur :- Triggered when an element loses focus, such as when a user clicks outside an input field or tabs away. Useful for validation or updates triggered by loss of focus

- The method is used to prevent the browser from executing the default action of the selected element. It can prevent the user from processing the request by clicking the link

Syntax :- event.preventDefault().
Parameters :- It does not accept any paramert.

* Target propert :-
Target is a property of an event object that refers to the element that triggered the event. This property can be useful for identifying which element an event originated from, and for getting or setting the value of the element.

* Callback hell in js
Callback hell, also known as the "pyramid of doom", is a phenomenon in Javascript (and other languges) that occurs when you have multiple nested callback functions. This makes your code hard to read understand, and maintain.

Example of call back hell

```
function getData(id, nextData)
{
    setTimeout(() => {
        console.log("Emp ID " + id);
        if(nextData){
            nextData();
        }
    }, 2000);
}
```

Calling multiple callback functions inside one function

```
getData(1234, () => {
    getData(5678, () => {
        getData(9090, () => {
            getData(6767);
        });
    });
});
```

JavaScript Promises

- JavaScript Promise are easy to manage when dealing with multiple asynchronous operations.
- The promise constructor takes only one argument which is a callback function.
- The callback function takes two arguments, resolve and reject
- Perform Operations inside the callback function and if everything went well then call resolve.
- If desired operations do not go well then call reject.

```
let task = new Promise((resolve, reject) => {
    if (CBike = "Triumph")){
        resolve();
    } else{
        reject();
    }
});
task.then(() => {
    console.log("Best Bike");
});
task.catch(() => {
    console.log("Dabba Bikes");
});
```

- When promise get full filled or resolved. Then mentiond will get executed.
- When promise get rejected. Catch method will get executed

• Promise Chaining :-

- Chaining is a straightforward ideal that allows us to initialize a second promice inside of our .then() method and then execute our results as a consequence. The value returned by the earlier promice is then caputured by the function within
- Whenever you will try to write promise in function it return the promise

```
function getData(id) {
    return new Promise((res, rej)
    => {
        setTimeOut(()=>{
            console.log("Emp ID"+id);
            res();
        }, 2000);
    });
}
```

```
getData(123).then(()=>{
getData(456).then(()=>{
    getData(789);
});
});
```

#Time delays in JS:-

- SetTimeOut() and setInterval() are two functions in js allows to schedule tasks to be executed at later time. Used for animations, polling data from server and other asynchronous tasks.

SetTimeOut():- Runs the code with time delay.

Syntax:- window.setTimeOut(function, delay);

- function:- The function you want to execute.
- delay:- The amount of time(in milliseconds) before the function is executed.

setInterval():- The function similar to setTimeOut, but is schedules a function to be executed repeatedly at specified interval.

Syntax:-window.setInterval(function, delay);

JSON (Java Script object Notation)

- It is format of storing and transporting data.
- It is used when the data is ~~send~~ sent from server to web page.
- It is a popular data format for developers because of its human readable text, light weight which requires less coding and faster process.
- The JSON format is syntactically similar to the code for creating Javascript objects. Because of this, a Javascript program can easily convert JSON data into Javascript objects.
- Javascript has a built in function for converting JSON strings into Javascript objects.

Syntax of JSON

Data is in key value pairs

Data is separated by ','

curly braces holds objects

[ ] -- holds array

- In Javascript, there are two main methods used to work with JSON (Javascript object Notation):

1. JSON.Stringify(): used to convert normal object to JSON format.

2. JSON.parse(): used to convert JSON format of data to normal object.

* Advantages of JSON:-

1. Human readable format
2. Support all popular programming languages
3. Easy to organize and access
4. It is light weight.

* Async and await:-

- async and await make promises easier to write.
- async makes a function return a Promise.
- await makes a function wait for a promise.

* Async:- The async keyword is used to define a function that returns a promise.

* Await:- The await keyword is used inside an async function to wait for a promise to settle (either resolve or reject). It can only be used inside an async function.