JavaScript is one of the core technologies of the web, alongside HTML & CSS. It is a high-level interpreted programming language used to create interactive and dynamic website experiences.

Interpreted means executed line by line rather than being compiled into machine code first.

JavaScript is used for :-

DOM (Document Object Model) Manipulation

Event Handling

Asynchronous Requests

Animations & Effects

Data Manipulation (Sorting, filtering, etc)

Storing Data(Cookies, LocalStorage, etc)

Single Page Applications (SPA)

Creating APIs & Wweb Services (Node.js, Deno)

Ways to declare a variable in javascript:-

Var, let, const

Using let we can declare the variable and we can reassign its value.

Using const we can’t declare the variable rather we should initialize and its value cannot be changed.

Variables names must be only letters, numbers, underscores and dollar signs. Cannot start with a number.

camelCase – firstName

PascalCase - PascalCase

JavaScript have two types of datatypes

1.Primitive DataTypes

Undefined – A variable that has not yet been defined/ assigned.

Symbol – Built-in object whose constructor returns a unique symbol.

Number – Integers as well as floating-point numbers.

BigInt – Numbers that are greater than the “Number” type can handle.

2.Reference or Objects or Non-primitive DataTypes

Object literals, arrays and functions are all reference types.

JavaScript is a dynamically typed language – it means we do not explicitly define the types for our variables.

Const name = “Lakshman”;

Console.log(name, typeof name) - Lakshman string

Null – Object

Primitive types stored directly in the stack

Reference types stored in the heap

TypeConversion:-

Let amount = “100”;

Amount = parseInt(amount); // Number

Amount = +amount; //Number

Amount = Number(amount); //Number

If we use toString(), javascript temporarily creates the wrapper.

Console.log(Math.sqrt(-1)); // NaN

Console.log(1+NaN); // NaN

Console.log(Undefined + Undefined); // NaN

Console.log(“foo”/3); // NaN

X = 2==2 // true

X = 2==’2’ // true == tests only value not the type

X = 2===2 // true

X = 2===’2’ // false === evaluates the values as well as type.

Type Coercion – javaScript implicitly changes the type

X = 5 + ‘5’ = 55 string

X = 5 \* ‘5’ = 25 Number

Let x;

Const name = “Lakshman”;

Const age = 23;

X = “hello my name is “+name+ “ and I am “+age+” years old”;

Console.log(x);

Template literals also called as template strings. Backticks are used for template literals.

X = `hello my name is ${name} and I am ${age} years old`;

X = “ hello world”;

X = X.trim() - hello world // removes the space

X = 5.4227;

X = X.toFixed(2); - 5.42 // added the two decimal points

X = X.toPrecision(2); - 5.4

X = X.toLocaleString(“en-US”); // for changing into the Locale.

X = Number.MAX\_VALUE // max value of Number type

X = Number.MIN\_VALUE // min value of Number type

Basically anything that is not a primitive type is going to be an object.

Let d = new Date(2021, 0, 10, 12, 30, 0); -- 2021 Jan 10 & 12:30

Let d = new Date(“2021-07-10T12:30:00”); -- July 10 2021 12:30:00

Let d = new Date(“2021-07-10”) - this will give us the back date. This must use along with time.

Let d = new Date(“07-10-2022”) - this will give us correct output July 10 2022

Timestamp:-

D = Date.now();

D = new Date();

D = D.getTime(); or D = D.valueOf();

TimeStamp in seconds -- D = Math.floor(Date.now()/1000);

X = D.getFullYear();

X = D.getMonth()+1;

X = D.getDate();

X = D.getDay();

X = D.getHours();

X = D.toLocaleString(“default”, {month : “long”}); -- May

X = Intl.DateTimeFormat(“en-US”).format(D);

X = D.toLocaleString(“default”, {weekday:”long”, year:”numeric”, month:”long”, day:”numeric”, hour:”numeric”, minute:”numeric”, second:”numeric”, timeZone:”America/New\_York”,});

Const arr = [34, 55, 95, 20, 15];

Arr.push(100); - will add 100 at the end of the array

Arr.pop(); - will take off the element whatever at the end

Arr.unshift(99) - will add 99 at the 0 index.

Arr.shift() - will takeit off from the beginning.

Arr.reverse() - will reverse the array

X = Arr.include(20) - will give true because 20 is present in the array

X = arr.indexOf(15) - will return the index if it is there, else return -1

Const arr = [34, 55, 95, 20, 15];

Slice does not change the original array x = arr.slice(1,4) 4-1 = 3 elements ---x = [ 55, 95, 20] and original remains [34, 55, 95, 20, 15];

Splice will change the original array. x = arr.splice(1,4) = 4 elements will be deleted from original

Const arr = [34, 55, 95, 20, 15];

X = arr.splice(3,1) -- x = 20 original array = [34, 55, 95, 15];

We can use concat to add the two arrays as one array.

**Spread operator (…)** will add the properties from the particular thing.

function sum(x, y, z) {

return x + y + z;

}

const numbers = [1, 2, 3];

console.log(sum(...numbers));

// Expected output: 6

console.log(sum.apply(null, numbers));

// Expected output: 6

X = […fruits, …berries]; same like concat of array

X = arr.flat(); --- used for flatten the array by taking the nested arrays and making it into single array.

Static Methods of Array Objects

X = Array.isArray(fruits); // to check the fruits is an array or not and returns true if it is.

X = Array.from(“12345”); // will convert each character into array object [‘1’, ‘2’, ‘3’, ‘4’, ‘5’];

Const a = 1;

Const b = 2;

Const c = 3;

X = Array.of(a,b,c); // will convert a,b,c into array

Object literal are the key value pairs

Const obj4 = Object.assign({}, obj1, obj2); // assign will works same like spread operator

Object.keys(todo);

Object.values(todo);

Object.entries(todo);

We can’t apply length on the objects instead we can get the keys out by Object.keys(todo); so it will return as String and then we can perform length in it by Object.keys(todo).length;

Todo.hasOwnProperty(“name”); // returns true if this property have in the objects

Const todo = {Id:1, Tite:”take out”, user:{name:”Lakshman”,},};

Const {Id, Title, user:{name}} = todo;

Destructuring Arrays:-

Const numbers = [23, 67, 33, 49];

Const [first, second, …rest] = numbers;

Console.log(first, second, rest); // 23 67 [33, 49]

JSON – JavaScript Object Notation and it is a lightweight data interchange format.

So it is essentially a way of storing the data in a simple, human readable format.

const post = {

    id:1,

    title:"Lakshman",

    body:"empty body",

};

//covert to JSON string

const str = JSON.stringify(post);

console.log(str);

JSON.stringify() will convert it into Object to JSON.

const obj = JSON.parse(str);

will covert JSON into Object

In Json format we can’t access the properties whereas we can through by converting it into object form and then access.

Random - Const randomIndex = Math.floor(Math.random()\*arr.length);

If local variable is overriding the global variable then it is called as shadowing….

**Var is not block scoped whereas const and let are block scoped……………. But Var is function scoped**

**If we add var as global object then it will add it into the window also…**

**Function Declaration**

Function addDollar(value)

{

Return ‘$’+value;

}

Console.log(addDollar(100)); // $100…..

**Function Expression**

Const addplus = function(value)

{

Return ‘+’+value;

};

Console.log(addplus(200)); // +200….

In JavaScript we don’t need to provide the semicolon at the end of the function declaration but we should provide for function expressions

Function calling can be at starting also for function declaration.

But for function expression we must define the expression and then calling next…

**Arrow Function**

Const add = (a, b) =>{

Return a+b;

};

Console.log(add(1,2)); // 3

Implicit return of arrow function

Const sub = (a,b) => a-b;

Const sub = a=> a;

Const sub = () => ({ Name:”Lakshman”; });

**Foreach():-**

Const num = [1,2,3,4,5];

Num.forEach(function(n))

{

Console.log(n); // 1 2 3 4 5

};

**Arrow function in a call back**

Num.forEach((n) =>console.log(n));

**IIFE (Immediately Invoked Function Expressions)** – In javascript that lets us to create a function or declare a function and invoke it at the same time and that’s using something called an Immediately invoked function expression.

(function(){

Const user = “Lakshman”;

Console.log(user);

})();

When we run any javascript a special environment is created to handle the transformation & execution of code .This is called the execution context.

JavaScript is single threaded and it is synchronous so it is executed line by line.

When we run a function we know that it opens up a new function execution context.It also gets added to something called the call stack.

In Boolean empty string, 0, null, NaN and undefined values are falsy conditions.

[], {}, function() is also truthy.

Loose Equality:-

Console.log(false==0); // true

Console.log(‘’==0); // true

Console.log(null==undefined); // true

B = 10||20 // returns first true value as 10

B = 0||20 // returns first true value as 20

B = false||20 // returns first true value as 20

B = true||20 // returns first true value as true

B = 0||null||’’||undefined // returns last false value as undefined

**Nullish coalescing operator ??** It returns right side value if left side value is null or undefined, else returns left side

C = 10??20 // returns 10 since left side is not null or undefined.

C = null??20 // returns 20 since left side value is null.

C = undefined??20 // returns 20 since left side value is undefined.

C = 0??20 // returns 0 since left side is not null or undefined.

C = ‘’??20 // returns since left side is not null or undefined.

**Ternary Operator:-**

Let auth = true;

Const redirect = auth?(alert(“welcome to dashboard”), “/dashboard”) : (alert(“Access Denied”), “/login”);

**Shortand Operator** we use this if we don’t have else statements.

Auth&&console.log(“welcome to the dashboard”); // here we don’t have else statements

**For Of Loop**

Const items = [1,2,3,4,5];

For(const item of items)

{

Console.log(item); }

Const map = new Map();

Map.set(“name”, “Lakshman”);

Map.set(“age”, 23);

For(const [key, value] of map)

{

Console.log(key, value);

}

let language = "JavaScript";  
  
let text = "";  
for (let x of language) {  
text += x;  
}

J  
a  
v  
a  
S  
c  
r  
i  
p  
t

The JavaScript for of statement loops through the values of an iterable object.

It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

**For in loop**

const person = {fname:"John", lname:"Doe", age:25};  
  
let text = "";  
for (let x in person) {  
  text += person[x];  
}

John Doe 25

* The **for in** loop iterates over a **person** object
* Each iteration returns a **key** (x)
* The key is used to access the **value** of the key
* The value of the key is **person[x]**

For each loop:-

const numbers = [45, 4, 9, 16, 25];  
  
let txt = "";  
numbers.forEach(myFunction);  
  
function myFunction(value, index, array) {  
  txt += value;  
}

45  
4  
9  
16  
25

Note that the function takes 3 arguments:

* The item value
* The item index
* The array itself

**DOM:-**

Document.head;

Document.body;

Document.head.children;

document .body.children;

document.doctype;

document.all;

document.all[11];

document.all.length;

document.documentElement;

document.domain;

document.URL;

document.characterSet;

document.contentType;

document.forms;

document.forms[0];

document.forms[0].id;

document.forms[0].action;

document.forms[0].id = “new-id”;

document.links;

document.links[0];

document.links[0].href;

document.links[0].id = “google”;

document.links[0].href = “https://facebook.com”;

document.links[0].className = “google-class”;

document.links[0].classList;

document.images;

document.images[0];

document.images[0].src;

// Get/change content

const title = document.getElementById(“app-title”);

console.log(title.textContent);

title.textContent = “hello world”;

title.innerText = “hello Again”;

title.innerHTML = “<strong>Shopping List</strong>;

//change styles

Title.style.color = “red”;

Title.style.backgroundColor = “black”; // don’t use background-color.

Title.style.padding = “10px”;

Title.style.borderRadius = “10px”;

//document.querySelector();

For getElementById whatever we have selected we had to have an Id on and in some cases we didn’t have full control of HTML. So that’s where jQuery came in and gave us a big advantage because we could select anything.

Console.log(document.querySelector(“h1”)); // it selects only the first h1.

Console.log(document.querySelector(“#app-title”)); // it selects only that particular id.

Console.log(document.querySelector(“.conatiner”)); // it selects only that particular class.

Console.log(document.querySelector(“input[type=”text”]”));

Console.log(document.querySelector(“li”)); // it will give only first item in that list

Console.log(document.querySelector(“li:nth-child(2)”).innerText); // it selects only the second item in the list.

Const second = Console.log(document.querySelector(“li:nth-child(2)”));

Second.innerText = “Apple Juice”;

Second.style.color = “red”;

Const list = document.querySelector(“ul”);

Console.log(list);

Const first = list.querySelector(“li”);

First.style.color = “blue”;

Const item2 = document.getElementsByClassName(“item”);

Console.log(item2[2].innerText);

Const item2Array = Array.from(item2);

Item2Array.forEach(I => console.log(i.innerText));

Const item3 = document.getElementsByTagName(“li”);

Const listItems = document.querySelectorAll(“.item”);

Console.log(listItems[1].innerText);

**Traversing the DOM:-**

Const parent = document.querySelector(“.parent”);

Let output = parent.children;

Let output = parent.children[0];

Let output = parent.children[1];

Let output = parent.children[2];

Let output = parent.children[0].innerText;

Let output = parent.children[0].className;

Let output = parent.children[0].nodeName;

parent.children[0],innerText = “child two;

parent.children[0].style.color = “red”;

parent.firstElementChild.innerText = “child one”;

parent.lastElementChild.innerText = “child one”;

// get parents from a child

const child = document.querySelector(“.child”);

output = child.parentElement;

child.parentElement.style.border = “1px solid #ccc”;

child.parentElement.style.padding = “10px”;

// Sibiling element

Const secondItem = document.querySelector(“.child:nth-child(2)”);

Const output = secondItem;

Const output = secondItem.nextElementSibiling;

secondItem.nextElementSibiling.style.color = “red”;

secondItem.previousElementSibiling.style.color = “orange”;

**Traversing the DOM nodes:-**

let output;

const parent = document.querySelector(“.parent”);

output = parent.childNodes;

output = parent.childNodes[0];

output = parent.childNodes[0].textContent;

output = parent.childNodes[0].nodeName;

output = parent.childNodes[3].outerHTML; // child 1

output = parent.childNodes[5].style.color = “red”; // child 2

output = parent.firstChild;

output = parent.lastChild;

parent.lastChild.textContent = “hello”;

//parent node

Cosnt child = document.querySelector(“.child”);

Output = child.parentNode;

Output = child.parentElement;

Child.parentNode.style.backgroundColor = “#ccc”;

Child.parentNode.style.padding = “10px”;

// sibilings

Const second = document.querySelector(“.child:nth-child(2)”);

Output = second.nextSibling;

Output = second.previousSibiling;

//creating elements

Const div = document.createElement(“div”);

div.className = “my-element”;

div.id = “my-element”;

div.setAttribute(“title”, “my element”);

div.innerText = “hello world”;

const text = document.createTextNode(“hello world”);

div.appendChild(text);

document.body.appendChild(div); // will add the div at the end of the body tag

// innerHTML vs createElement

//quick and dirty

Function createListItem(item)

{

Const li = document.createElement(“li”);

li.innerHTML = `${item} <button class = “remove-item btn-link text-red”><I class = “fa-solid fa-xmark”></i></button>`;

Document.querySelector(“.items”).appendChild(li);

}

createListItem(“Eggs”);

//clean and performant

Function createNewItem(item)

{

Const li = document.createElement(“li”);

li.appendChild(document.createTextNode(“item”));

Const button = document.createElement(“button”);

Button.className = “remove-item btn-link text-red”;

Const icon = document.createElement(“I”);

icon.className = “fa-solid fa-xmark”;

button.appendChild(icon);

li.appendChild(button);

Console.log(li.innerHTML);

}

createNewItem(“Cheese”);

**InnerHTML vs CreateElement():-**

**//Quick & Dirty**

Function createListItem(item)

{

Const li = document.createElement(`li`);

li.innerHTML = `${item} <button class = “remove-item btn-link text-red”>

<I class = “fa-solid fa-xmark”></i>

</button>`;

Document.querySelector(“.items”).appendChild(li);

}

createListItem(“Eggs”);

//Clean & Performant

Function createNewItem(item)

{

Const li = document.createElement(“li”);

li.appendChild(document.createTextNode(item));

Const button = document.createElement(“button”);

Button.className = “remove-item btn-link text-red”;

Const icon = document.createElement(“i”);

icon.className = “fa-solid fa-xmark”;

//const button = createButton(“remove-item btn-link text-red”);

button.appendChild(icon);

li.appendChild(button);

document.querySelector(“.items”).appendChild(li);

}

createNewItem(“Cheese ”);

**//Refactor to the multiple functions:-**

Function createButton(classes)

{

Const button = document.createElement(“button”);

Button.className = classes;

Const icon = createIcon(“fa-solid fa-xmark”);

button.appendChild(icon);

Return button;

}

Function createIcon(classes)

{

Const icon = document.createElement(“I”);

Icon.className = classes;

Return icon;

}

//insert elements, text and HTML

Function insertElement()

{

Const filter = document.querySelector(“.filter”);

Const h1 = document.createElement(“h1”);

H1.textContent = “insertAdjacentElement”;

Filter.insertAdjacentElement(“beforebegin”, h1);

}

insertElement();

function insertText()

{  
const item = document.querySelector(“li:first-child”);

Item.insertAdjacentText(“afterend”, “insertAdjacentText”);

}

insertText();

function insertHTML()

{

Const clearBtn = document.querySelector(“#clear”);

clearBtn.insertAdjacentHTML(“afterend”, “<h2>insertAdjacentHTML</h2>”);

}

insertHTML();

function insertBeforeItem()

{

Const ul = document.querySelector(“ul”);

Const li = document.createElement(“li”);

li.textContent = “insertBefore”;

const thirdItem = document.querySelector(“li:nth-child(3)”);

ul.insertBefore(li, thirdItem);

}

insertBeforeItem();

//insertAfter

Function insertAfter(newEl, existingEl)

{

ExistingEl.parentElement.insertBefore(newEl, existingEl.nextSibling);

}

Const li = document.createElement(“li”);

li.textContent = “insert Me After!”;

const firstitem = document.querySelecot(“li:first-child”);

insertAfter(li, firstItem);

//Replace Elements

Function replaceFirstItem()

{

Const firstItem = document.querySelector(“li:first-child”);

Const li = document.createElement(“li”);

li.textContent = “Replaced First”;

firstItem.replaceWith(li);

}

replacedFirstItem();

function replaceSecondItem()

{

Const secondItem = document.querySelector(“li:nth-child(2)”);

secondItem.outerHTML = “<li>Replaced Second</li>”;

}

replaceSecondItem();

function replaceAllItems()

{

Const lis = document.querySelectorAll(“li”);

//querySelectorAll return the nodeList

Lis.forEach((item, index) => {

//Item.outerHTML = “<li>Replace All</li>;

//Item.innerHTML = “ReplaceAll”;

If(index===1)

{

Item.innerHTML = “Second Item”;

}

Else

{

Item.innerHTML = “replace All”;

}

});

// lis.forEach((item, index) => (item.outerHTML = index===1 ? “<li>Second //Item</li>”:”<li>Item</li>”));

//}

}

replaceAllItems();

function replaceChildHeading()

{

Const header = document.querySelector(“header”);

Const h1 = document.querySelector(“header h1”);

Const h2 = document.querySelector(“h2”);

H2.id = “app-title”;

H2.textContent = “Shopping List”

Header.replaceChild(h2,h1);

}

replaceChildHeading();

// removing DOM elements

Remove is going to be called on the element we want to remove.

RemoveChild is called on the parent element and then we pass in the item we want to remove.

Function removeClearButton()

{

Const clearBtn = document.querySelecor(“#clear”);

clearBtn.remove();

}

removeClearButton();

function removeFirstItem()

{

Const ul = document.querySelector(“ul”);

Const li = document.querySelector(“li:first-child”);

Ul.removeChild(li);

}

removeFirstItem();

function removeItem(number)

{

Const ul = document.querySelector(“ul”);

Const li = document.querySelector(`li:nth-child(${number})`);

ul.removeChild(li);

}

removeItem(2);

function removeItem2(number)

{

Const ul = document.querySelector(“ul”);

Const li = document.querySelectorAll(“li”)[number-1];

ul.removeChild(li);

}

removeItem2(1);

function removeItem3(number)

{

Const li = document.querySelectorAll(“li”);

Li[number-1].remove();

}

removeItem3(3);

const removeItem4 = number => document.querySelectorAll(“li”)[number-1].remove();

Any interaction that the user has with our web page all the way down to the actual loading of the page is an event.

**Event listner:-**

Const clearbtn = document.querySelector(“#clear”);

Clearbtn.addEventListener(“click”, function () {});

1st parameter – type of event;

2nd parameter – fallback function

setTimeout(() => clearbtn.click(), 5000); // creates the event after 5sec

clearBtn.addEventListener(“click”, onClear);

setTimeout(() => clearBtn.removeEventListener(“click”, onClear),5000); // removes the event

function onClear()

{

Const itemList = document.querySelector(“ul”);

Const items = itemList.querySelectorAll(“ul”);

//Items.forEach(item => item.remove());

While(itemList.firstChild)

{

itemList.removeChild(itemList.firstChild);

}

}

//Mouse events:-

Const logo = document.querySelector(“img”);

Const onClick = () =>console.log(“click event”);

Const onDoubleClick = () =>console.log(“doubleclick event”);

Const onRightClick = () => console.log(“right click event”);

Const onMouseDown = () =>console.log(“mouse down event”);

Const onMouseUp = () =>console.log(“mouse up event”);

Const onMouseWheel = () =>console.log(“mouse wheel event”);

Const onMouseOver = () =>console.log(“mouse over event”);

Const onMouseOut =() =>console.log(“mouse out event”);

Const onDragStart = () =>console.log(“drag start event”);

Const onDrag = () =>console.log(“drag event”);

Const onDragEnd = () =>console.log(“drag end event”);

Logo.addEventListener(“click”, onClick)

Logo.addEventListener(“dblclick”, onDoubleClick);

Logo.addEventListener(“contextmenu”, onRightClick);

Logo.addEventListener(“mousedown”, onMouseDown);

Logo.addEventListener(“mouseup”, onMouseUp);

Logo.addEventListener(“wheel”, onMouseWheel);

Logo.addEvetnListener(“mouseover”, onMouseOver);

Logo.addEventListener(“mouseout”, onMouseOut);

Logo.addEventListener(“dragstart”, onDragStart);

Logo.addEventListener(“drag”, onDrag);

Logo.addEventListener(“dragend”, onDragEnd);

The Event Object:-

Const logo = document.querySelector(“img”);

//Logo.addEventListener(“click”, finction €

//{

//Console.log(e);

//});

/\*

Function onClick(e)

{

Console.log(e);

}

Logo.addEventListener(“click”, onClick);

\*/

Function onClick(e)

{

Console.log(e.target);

Console.log(e.currentTarget);

}

Logo.addEventListener(“click”, onClick);

Document.body.addEventListener(“click”, function (e)

{

Console.log(e.target);

Console.log(e.currentTarget);

}

e.timeStamp – will give us the time

e.clientX – will give the x-axis dimensions

e.clientY – will give the y-axis dimensions

document.querySelector(“a”).addEventListener(“click”, function (e)

{

e.preventDefault();

console.log(“link was clicked”);

});

Function onDrag(e)

{

Document.querySelector(“h1”).textContent = `X ${e.clientX) Y ${e.clientY}`;

}

Logo.addEventListener(“drag”, onDrag);

Keyboard Events and key properties:-

Const item = document.getElementById(“item-input”);

Const onKeyPress = e =>console.log(“keypress”);};

Item.addEventListener(“keypress”, onKeyPress);

Const onKeyPress = e=>console.log(“keyup”);};

Item.addEventListener(“keyup”, onKeyUp);

Const onKeyDown = e=>{

If(e.key===”Enter”){ // (e.keyCode===13)

Alert(“you pressed enter”);

}

};

Item.addEventListener(“keydown”, onKeyDown);

If(e.code===”Digit1”)

If(e.repeat){

Console.log(“you are holding down the “+e.key);

Console.log(“Shift: “+e.shiftKey);

Console.log(“Control: “+e.ctrlKey);

Console.log(“Alt:”+e.altKey);

}

// e.key ===’k’

Const item = document.getElementById(“item-input”);

Const priority = document.getElementById(“priority-input”);

Const checkbox = document.getElementById(“checkbox”);

Const heading = document.querySelector(“h1”);

Function onInput(e)

{

Console.log(e.target.value);

}

Item.addEventListener(“input”, onInput);