**JAVASCRIPT**

JavaScript has only one call stack, meaning it can run one piece of code at a time — line by line.It does not execute multiple pieces of JavaScript code simultaneously

**Operator**

**\*\***

3\*\*2 =3^2=9

**Datatypes:**

**Basic Datatypes**

* var-global scope
* let.const-block scope within curly braces

**Main Types**

* Primitive datatype(Eg:number,String)
* Non-primitive datatype(Array,Object)

**Eg:1**

console.log(a)

var a=6;

o/p:undefined

**Eg:2**

console.log(a)

let a=6;

o/p:reference error

**Primitive datatype(Call by Value)**

//value-(primitive-oru ethulaa update anaa ella idthalayum agathu)

Eg:

let a=10;

let b=a;

a=20;

console.log(b);

**Output:**

10

**Non-primitive datatype(Call by Reference)**

let object1={age:33};

let object2=object1;

object1.age=22;

console.log(object2);

**Output:**

{age:22}

**Functions:**

function rep()

{

let i;

for(i=1;i<=10;i++)

{

  console.log(i);

}

}

const print=()=>

{

    let i;

for(i=1;i<=10;i++)

{

  console.log(i);

}

}

rep();

rep();

rep();

print();

**DOM Manipulation**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style></style>

</head>

<body>

    <p id="demo"></p>

    <button onclick="call()">Click Me!</button>

    <script>

        function call()

        {

            document.getElementById("demo").innerHTML="Subika The Great Legend";

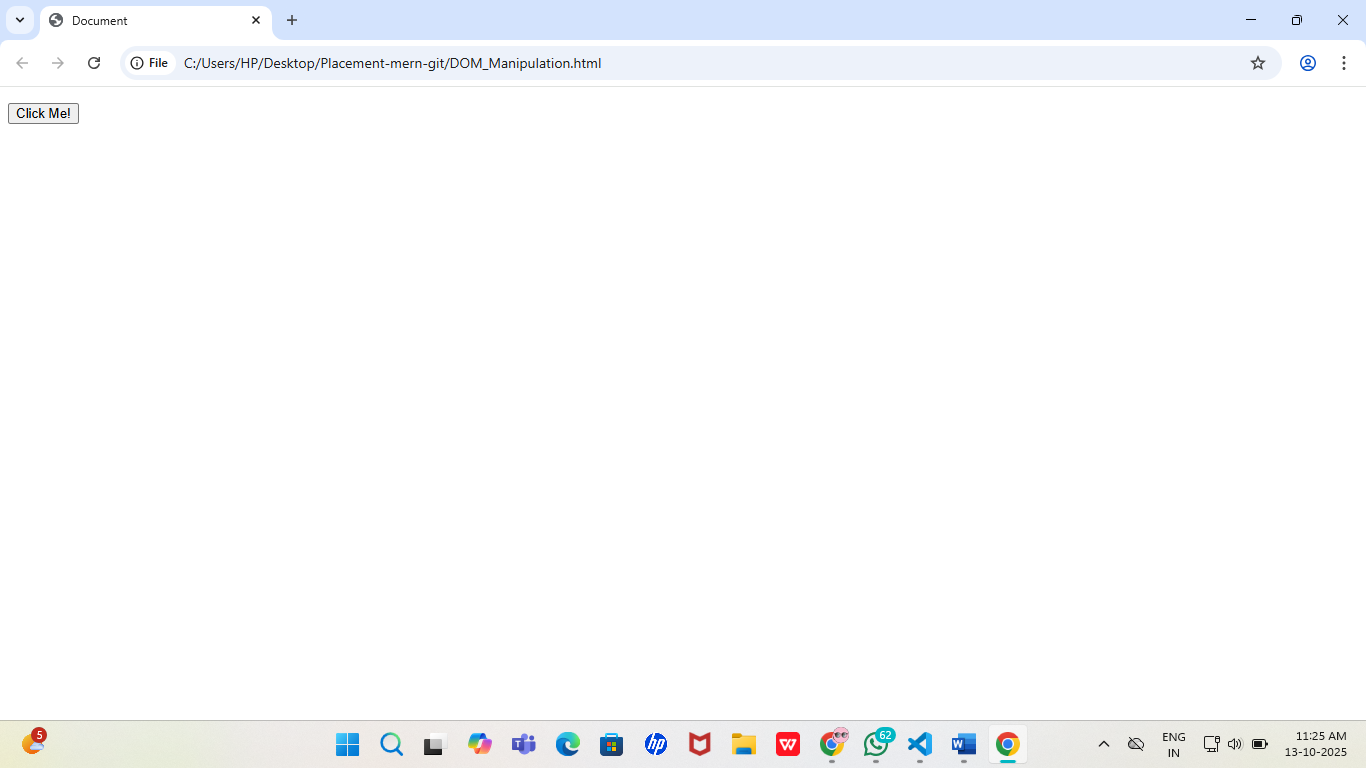
        }

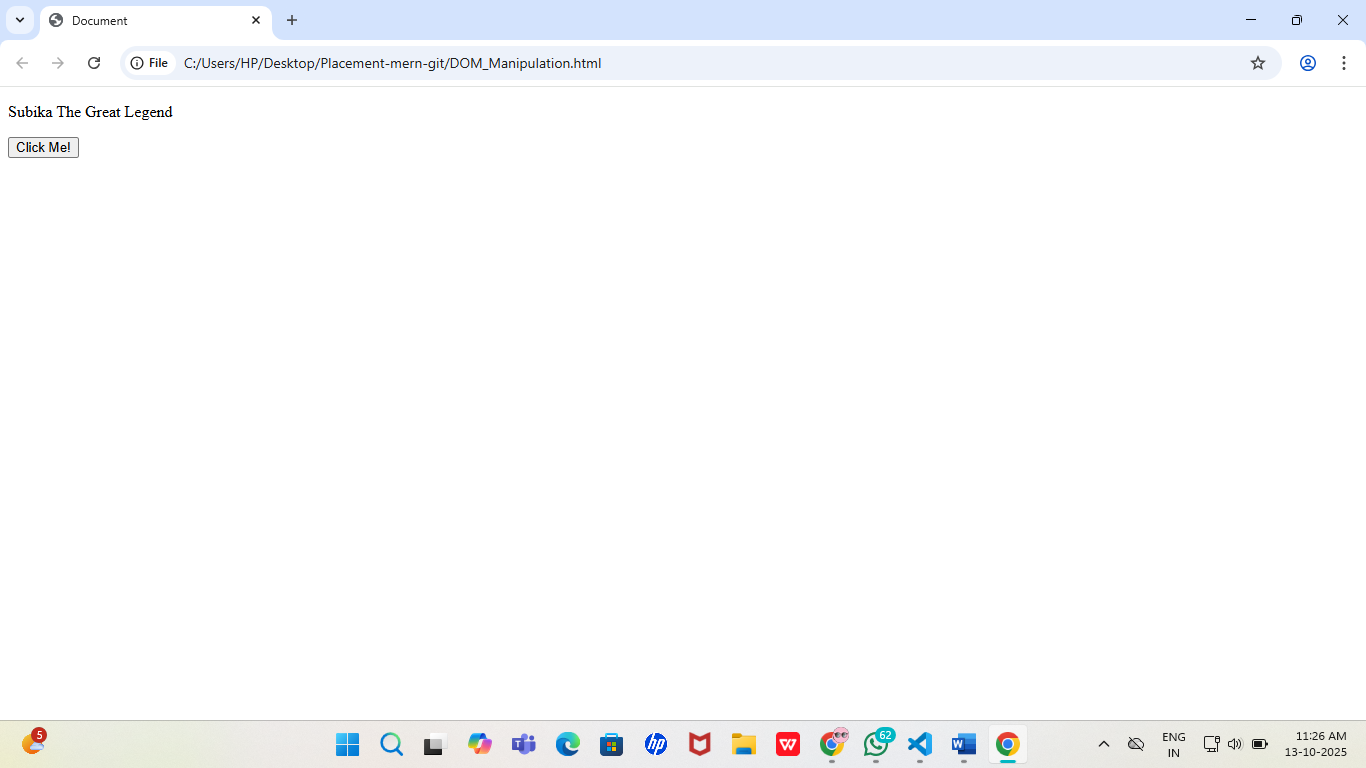
    </script>

</body>

</html>

**Output:**





**Array Functions:**

1. map()-applies mathematical logic to each element of an array and returns modified array(eg:calculate multipleof2,multipleof3)
2. filter()-condition checking.(eg:to find odd,even,multipleof 5 etc).
3. reduce()-operation perform panni ,single valuaa return pannum(eg:sum,multiply etc).

**1.map()=function passed as parameter(callback)**

let arr=[1,2,3,4,5];

//applies functions and returns as array

let doubledarray=arr.map((n)=>2\*n);

console.log(arr);

console.log(doubledarray);

**Output:**

[ 1, 2, 3, 4, 5 ]

[ 2, 4, 6, 8, 10 ]

**2.filter()**

**Eg:1 – Odd Elements**

let arr1=[1,2,3,4,5];

//odd will be added ,even will not be added,and it returns odd array

let odd=arr.filter((n)=>{return n%2!=0});

console.log(arr);

console.log(odd);

**Output:**

[ 1, 2, 3, 4, 5 ]

[ 1, 3, 5 ]

**Eg:2-Multiples of 5**

let arr3=[13,15,20,35,27];

let multiplesof5=arr3.filter((n)=>{return n%5==0;});

console.log(arr3);

console.log(multiplesof5);

**Output:**

[ 13, 15, 20, 35, 27 ]

[ 15, 20, 35 ]

**3.reduce()**

**Eg:1**

//1st parameter is array iterating element

//2nd parameter is sum value stored place while iterating

//the one present after closed set bracket is initial value of sum

let arr4=[1,2,3,4,5,6];

let sumofarray=arr4.reduce((n,sum)=>{

return n\*sum;},1);

console.log(sumofarray);

**Output:**

720

**Eg:2**

//arr5-stores actual arr value

let arr5=[1,2,3,4,5];

//arr6-stores the array value of arr5,where each element is multiplied by 3

let arr6=arr5.map((n)=>n\*3);

//arr7-stores the number which is even

let arr7=arr6.filter((n)=>{return n%2==0});

//sumofarray1-stores sum of all elements

let sumofarray1=arr7.reduce((sum1,n)=>{return sum1+n;},0);

console.log(arr5);

console.log(arr6);

console.log(arr7);

console.log(`Sum:${sumofarray1}`);

/\*

let arr6=arr5.map((n)=>n\*3)

             .filter((n)=>n%2==0)

             .reduce((sum,n)=>{return sum\*n;},0);

console.log(arr6);

\*/  
  
**Output:**

[ 1, 2, 3, 4, 5 ]

[ 3, 6, 9, 12, 15 ]

[ 6, 12 ]

Sum:18

**Destructuring-assigning array elements to variables in single line**

**Eg:1**

//Destructuring-assigning variables to some other variables in one line

let b=[1,2,3,4,5];

const [one,two,three,four,five]=b;

console.log(three);

**Eg:2 – Rest Operator**

//In first variable 1 number will be stored,and in second variable rest of the elements are assigned.

let b=[1,2,3,4,5];

const [one,...two]=b;

console.log(two);

**Eg:3-Spread Operator**

//adding extra elements to an array using spread operator

let b=[1,2,3,4,5];

let num=[0,...b,6,7];

console.log(num);

**Eg:4-Spread Operator-merging arrays and objects**

**Eg:**

const obj1={x:3,y:4};

const obj2={y:5,z:6};

//while merging one object will be added first and then second object will be addec ,so the second object

const spread={...obj1,...obj2};

console.log(spread);

**Tasks**

* Microtasks-first executes in a program(eg:console.log)
* Macrotasks-executes after micro tasks(eg:setTimeout)

**Macrotasks**

Eg:setTimeOut()

**Microtasks**

Eg:Promises

**3 states**

* pending
* success(content print from resolve block)
* failure(content print from reject block).

**Function Types:**

* Synchronous Functions
* Asynchronous Functions

**Synchronous Functions**

Runs one after another, blocks the next line

**Eg:**

console.log(), math operations

**Asynchronous Functions**

Runs later without blocking main thread

**Eg:**

setTimeout(), fetch(), Promises

**Example:**

const promise=new Promise((resolve,reject)=>{

    setTimeout(()=>{

    const success=true;

    if(success){

        resolve("Resolved Successfully");

    }

    else{

        reject('rejected');

    }

    },2000);

});

    console.log("Before Promise Execution");

    promise.then((result)=>console.log(result))

           .catch(err=>console.log(err))

           .finally(()=>{console.log("Promises completed");})

    console.log("After promises execution");

**API Fetch**

* Calling an API (fetch(...))
* Waiting for a response
* Converting it to JSON
* Printing the result
* Handling errors if anything goes wrong

🌤️ Example:

When you request:

https://api.openweathermap.org/data/2.5/weather?q=Chennai&appid=YOUR\_API\_KEY

Here’s what happens behind the scenes:

1. The API receives your request.
2. It goes to the OpenWeather database, which constantly updates data from satellites and sensors.
3. It fetches the data for Chennai’s weather.
4. It sends that data back to your app as a response.

💡 So, in simple words:

* The database/server 🗄️ = *where the information lives*
* The API 🔗 = *the messenger or bridge that brings it to you*
* Your app 📱 = *the one that requests and displays it*

So yes:

The API doesn’t hold the information,  
it’s the link between your app and the place (server) where the information is stored.

   //API Fetch

    fetch("https://jsonplaceholder.typicode.com/users")

      .then((response)=>{return response.json()})

      .then((data)=>console.log(data))

      .catch((err)=>console.log(err))

Output:

[

  {

    id: 1,

    name: 'Leanne Graham',

    username: 'Bret',

    email: 'Sincere@april.biz',

    address: {

      street: 'Kulas Light',

      suite: 'Apt. 556',

      city: 'Gwenborough',

      zipcode: '92998-3874',

      geo: [Object]

    },

    phone: '1-770-736-8031 x56442',

    website: 'hildegard.org',

    company: {

      name: 'Romaguera-Crona',

      catchPhrase: 'Multi-layered client-server neural-net',

      bs: 'harness real-time e-markets'

    }

  },

  {

    id: 2,

    name: 'Ervin Howell',

    username: 'Antonette',

    email: 'Shanna@melissa.tv',

    address: {

      street: 'Victor Plains',

      suite: 'Suite 879',

      city: 'Wisokyburgh',

      zipcode: '90566-7771',

      geo: [Object]

    },

    phone: '010-692-6593 x09125',

    website: 'anastasia.net',

    company: {

      name: 'Deckow-Crist',

      catchPhrase: 'Proactive didactic contingency',

      bs: 'synergize scalable supply-chains'

    }

  },

  {

    id: 3,

    name: 'Clementine Bauch',

    username: 'Samantha',

    email: 'Nathan@yesenia.net',

    address: {

      street: 'Douglas Extension',

      suite: 'Suite 847',

      city: 'McKenziehaven',

      zipcode: '59590-4157',

      geo: [Object]

    },

    phone: '1-463-123-4447',

    website: 'ramiro.info',

    company: {

      name: 'Romaguera-Jacobson',

      catchPhrase: 'Face to face bifurcated interface',

      bs: 'e-enable strategic applications'

    }

  },

  {

    id: 4,

    name: 'Patricia Lebsack',

    username: 'Karianne',

    email: 'Julianne.OConner@kory.org',

    address: {

      street: 'Hoeger Mall',

      suite: 'Apt. 692',

      city: 'South Elvis',

      zipcode: '53919-4257',

      geo: [Object]

    },

    phone: '493-170-9623 x156',

    website: 'kale.biz',

    company: {

      name: 'Robel-Corkery',

      catchPhrase: 'Multi-tiered zero tolerance productivity',

      bs: 'transition cutting-edge web services'

    }

  },

  {

    id: 5,

    name: 'Chelsey Dietrich',

    username: 'Kamren',

    email: 'Lucio\_Hettinger@annie.ca',

    address: {

      street: 'Skiles Walks',

      suite: 'Suite 351',

      city: 'Roscoeview',

      zipcode: '33263',

      geo: [Object]

    },

    phone: '(254)954-1289',

    website: 'demarco.info',

    company: {

      name: 'Keebler LLC',

      catchPhrase: 'User-centric fault-tolerant solution',

      bs: 'revolutionize end-to-end systems'

    }

  },

  {

    id: 6,

    name: 'Mrs. Dennis Schulist',

    username: 'Leopoldo\_Corkery',

    email: 'Karley\_Dach@jasper.info',

    address: {

      street: 'Norberto Crossing',

      suite: 'Apt. 950',

      city: 'South Christy',

      zipcode: '23505-1337',

      geo: [Object]

    },

    phone: '1-477-935-8478 x6430',

    website: 'ola.org',

    company: {

      name: 'Considine-Lockman',

      catchPhrase: 'Synchronised bottom-line interface',

      bs: 'e-enable innovative applications'

    }

  },

  {

    id: 7,

    name: 'Kurtis Weissnat',

    username: 'Elwyn.Skiles',

    email: 'Telly.Hoeger@billy.biz',

    address: {

      street: 'Rex Trail',

      suite: 'Suite 280',

      city: 'Howemouth',

      zipcode: '58804-1099',

      geo: [Object]

    },

    phone: '210.067.6132',

    website: 'elvis.io',

    company: {

      name: 'Johns Group',

      catchPhrase: 'Configurable multimedia task-force',

      bs: 'generate enterprise e-tailers'

    }

  },

  {

    id: 8,

    name: 'Nicholas Runolfsdottir V',

    username: 'Maxime\_Nienow',

    email: 'Sherwood@rosamond.me',

    address: {

      street: 'Ellsworth Summit',

      suite: 'Suite 729',

      city: 'Aliyaview',

      zipcode: '45169',

      geo: [Object]

    },

    phone: '586.493.6943 x140',

    website: 'jacynthe.com',

    company: {

      name: 'Abernathy Group',

      catchPhrase: 'Implemented secondary concept',

      bs: 'e-enable extensible e-tailers'

    }

  },

  {

    id: 9,

    name: 'Glenna Reichert',

    username: 'Delphine',

    email: 'Chaim\_McDermott@dana.io',

    address: {

      street: 'Dayna Park',

      suite: 'Suite 449',

      city: 'Bartholomebury',

      zipcode: '76495-3109',

      geo: [Object]

    },

    phone: '(775)976-6794 x41206',

    website: 'conrad.com',

    company: {

      name: 'Yost and Sons',

      catchPhrase: 'Switchable contextually-based project',

      bs: 'aggregate real-time technologies'

    }

  },

  {

    id: 10,

    name: 'Clementina DuBuque',

    username: 'Moriah.Stanton',

    email: 'Rey.Padberg@karina.biz',

    address: {

      street: 'Kattie Turnpike',

      suite: 'Suite 198',

      city: 'Lebsackbury',

      zipcode: '31428-2261',

      geo: [Object]

    },

    phone: '024-648-3804',

    website: 'ambrose.net',

    company: {

      name: 'Hoeger LLC',

      catchPhrase: 'Centralized empowering task-force',

      bs: 'target end-to-end models'

    }

  }

]

//API Fetch-Fetch only name from json

fetch("https://jsonplaceholder.typicode.com/users")

      .then((response)=>{return response.json()})

      .then((data)=>data.map((user)=>{console.log(user.name);}))

      .catch((err)=>{console.log(err);})

**Output:**

Leanne Graham

Ervin Howell

Clementine Bauch

Patricia Lebsack

Chelsey Dietrich

Mrs. Dennis Schulist

Kurtis Weissnat

Nicholas Runolfsdottir V

Glenna Reichert

Clementina DuBuque

**Example2:**

async function fetchAPI()

    {

        try{

            const response =await fetch(

                "https://jsonplaceholder.typicode.com/users"

            )

            const data=await response.json();

            data.map((user)=>{

                console.log(user.name);

            })

        }

        catch(error)

        {

            console.log(error);

        }   }

    fetchAPI();

**Output:**

Leanne Graham

Ervin Howell

Clementine Bauch

Patricia Lebsack

Chelsey Dietrich

Mrs. Dennis Schulist

Kurtis Weissnat

Nicholas Runolfsdottir V

Glenna Reichert

Clementina DuBuque

**String Literal**

let name=”Subika”

let age=20;

console.log(`I am ${name} and my age is ${age}`);

**REACT**

**Commands to be typed after creating new folder**

PS C:\Users\HP\Desktop\Placement-mern-git\React> npm create vite@latest

> npx

> create-vite

◇ Project name:

vite\_react\_project1

◇ Select a framework:

React

◇ Select a variant:

JavaScript

◇ Use rolldown-vite (Experimental)?:

No

◇ Install with npm and start now?

Yes

◇ Scaffolding project in C:\Users\HP\Desktop\Placement-mern-git\React\vite\_react\_project1...

◇ Installing dependencies with npm...

added 153 packages, and audited 154 packages in 27s

32 packages are looking for funding

run `npm fund` for details

found 0 vulnerabilities

◇ Starting dev server...

> vite\_react\_project1@0.0.0 dev

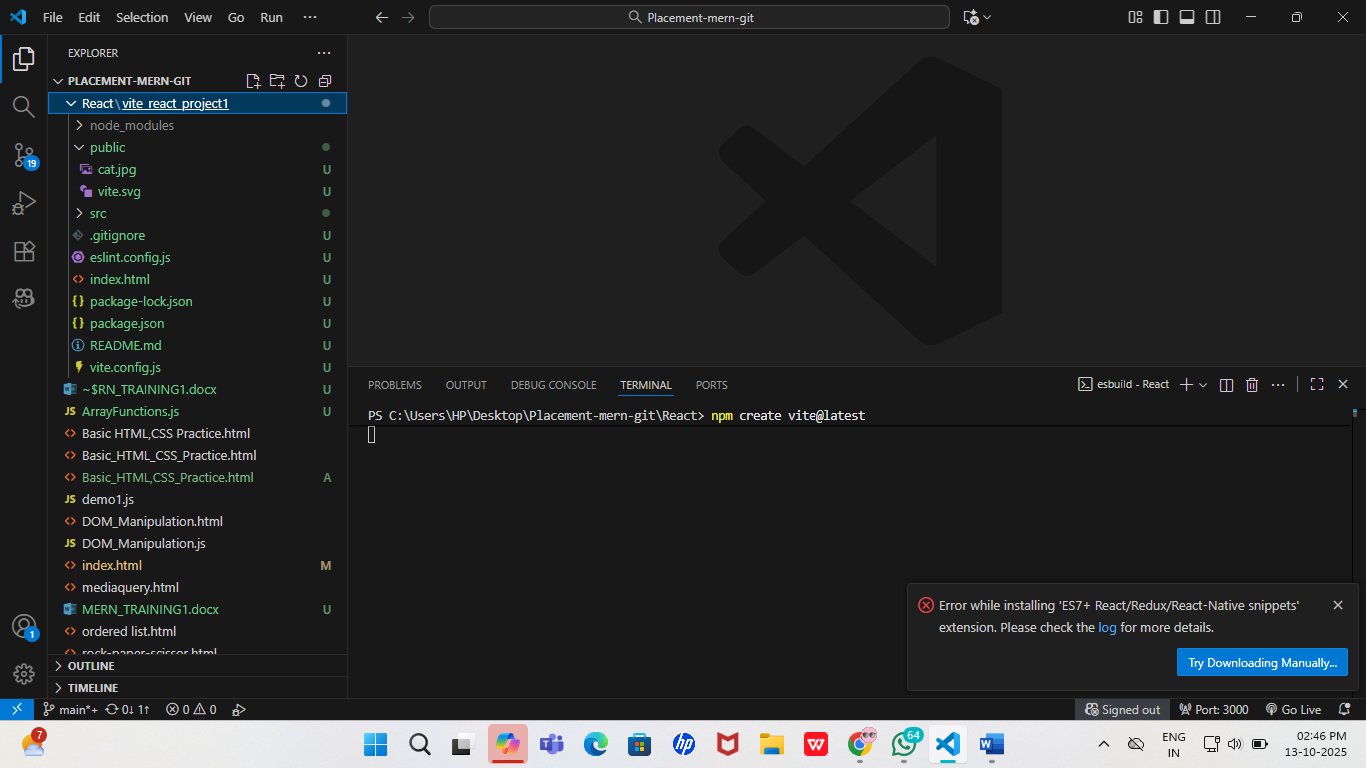
> vite

VITE v7.1.9 ready in 532 ms

➜ Local: http://localhost:5173/

➜ Network: use --host to expose

➜ press h + enter to show help



**Folders**

1. **node\_modules/**
   * Contains all the libraries and dependencies installed via npm (like React, Vite, etc.).
   * You normally don’t touch these files manually.
2. **public/**
   * Contains static files like index.html, images, icons, etc.
   * Files here are served **as-is** by the development server.
3. **src/**
   * This is where all your **React app code** lives.
   * Typical files here:
     + App.jsx → The main React component.
     + main.jsx → Entry point of the app; renders <App /> to the DOM.
     + App.css / index.css → CSS files for styling your app.
     + assets/ → Images, icons, fonts, etc. used in your components.
     + ArrayFunctions.js, DOM\_Manipulation.js → Your custom JavaScript files.

**Files in the root**

1. **.gitignore**
   * Lists files/folders that Git should **ignore** (like node\_modules/ or .env).
2. **package.json**
   * Defines your project, dependencies, and scripts (like npm start, npm build).
3. **package-lock.json**
   * Keeps a **record of exact versions** of all installed dependencies to ensure consistency.
4. **README.md**
   * Documentation for your project.
5. **vite.config.js**
   * Vite configuration file.
   * You currently have:

import { defineConfig } from 'vite'

import react from '@vitejs/plugin-react'

export default defineConfig({

plugins: [react()]

})

* + It tells Vite to use the React plugin for building your app.

1. **.eslintconfig.js**
   * ESLint configuration file for enforcing JavaScript/React coding standards.

**Other miscellaneous files**

* index.html → The HTML page that loads your React app.
* .docx or practice HTML files → Your personal notes or practice files.

**✅ Key Points**

* The **main app flow**: index.html → main.jsx → App.jsx → other components.
* **src/** is where most of your coding happens.
* **public/** is for static assets.
* vite.config.js tells Vite how to bundle your project.

**🗂 Inside src/ Folder**

**1️⃣ main.jsx**

📍 **This is the entry point of your React app.**

import React from 'react'

import ReactDOM from 'react-dom/client'

import App from './App.jsx'

import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>,

)

**Explanation:**

* main.jsx connects your React app to the **HTML file (index.html)**.
* ReactDOM.createRoot() mounts your main React component (App) into the HTML <div id="root">.
* App.jsx is the top-level component that holds all other components.
* It also imports global CSS (index.css).

**In short:**  
🧠 main.jsx = “Start point of your app” (connects React → HTML page)

**2️⃣ App.jsx**

📍 **Main React component (root UI).**

import './App.css'

function App() {

return (

<>

<h1>Hello React + Vite!</h1>

<p>This is your main app component.</p>

</>

)

}

export default App

**Explanation:**

* This is your **main React component** — the app’s root layout.
* Inside it, you can call other components (like <Navbar />, <Footer />, etc.).
* It’s styled by App.css.

**In short:**  
🎨 App.jsx = “Your main UI container”

**3️⃣ App.css**

📍 **Contains styles specific to App.jsx.**

Example:

h1 {

color: blue;

text-align: center;

}

**Explanation:**

* You can style components here directly.
* React automatically imports this when you use import './App.css'.

**In short:**  
💅 App.css = “Styling for App.jsx”

**4️⃣ index.css**

📍 **Global stylesheet for your entire app.**

Example:

body {

margin: 0;

font-family: Arial, sans-serif;

background-color: #f2f2f2;

}

**Explanation:**

* Used for global CSS rules that apply to the entire app (like body font, background, etc.).
* Imported in main.jsx.

**In short:**  
🌎 index.css = “Global app styles”

**5️⃣ assets/**

📍 **Folder for images, icons, etc.**

Example:

assets/

├── logo.png

├── background.jpg

**Explanation:**

* You store media here and import them inside components like this:
* import logo from './assets/logo.png';
* <img src={logo} alt="App logo" />

**In short:**  
🖼️ assets/ = “All your images and static files”

**6️⃣ ArrayFunctions.js, DOM\_Manipulation.js, etc.**

📍 **Your personal JS practice files**

Example:

let arr = [1, 2, 3];

let doubled = arr.map(n => n \* 2);

console.log(doubled);

**Explanation:**

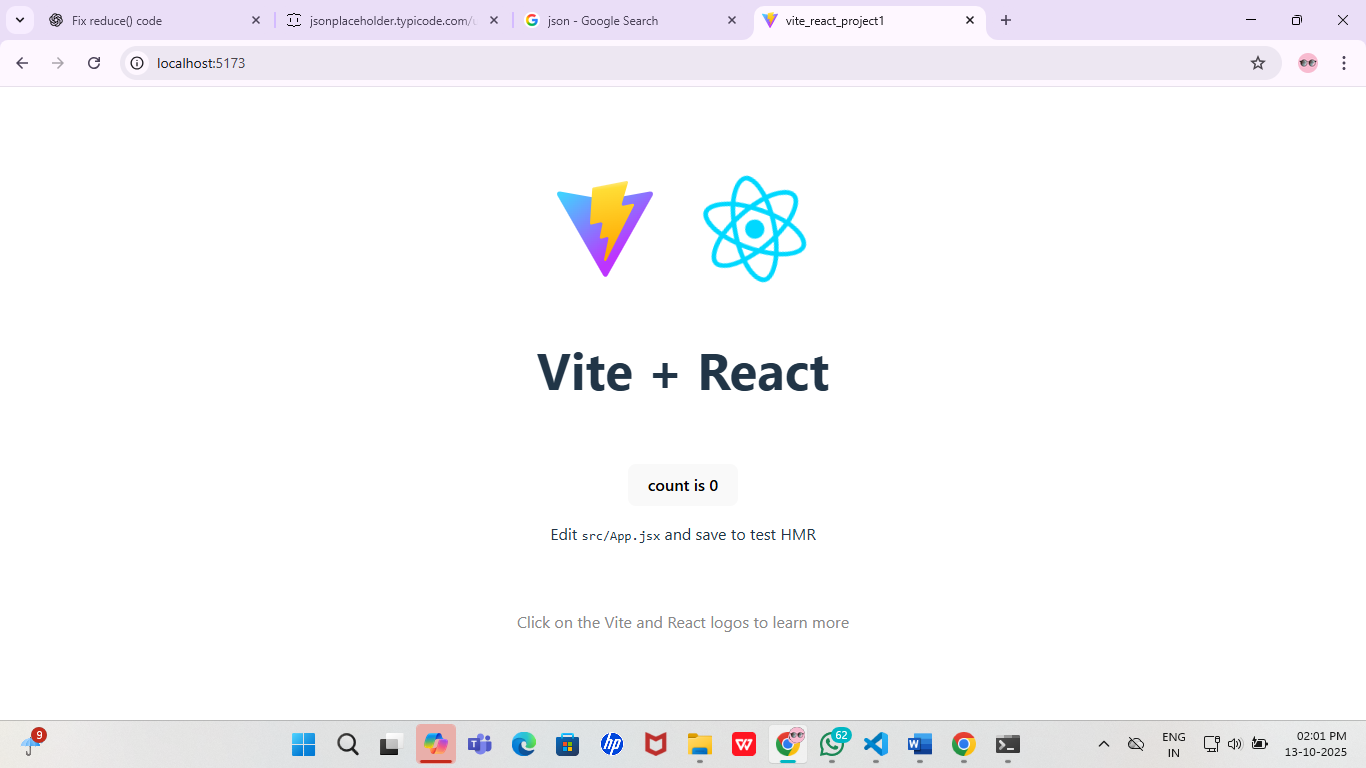
* These are **independent JS files** you can use for learning or testing.
* They aren’t part of your React app unless you explicitly import them into components.

**In short:**  
🧩 Your practice scripts for JS logic.

**✅ Summary — How They Work Together**

| **File / Folder** | **Purpose** |
| --- | --- |
| main.jsx | Starts the React app, mounts <App /> |
| App.jsx | Main component holding all UI |
| App.css | Styles for App.jsx |
| index.css | Global styles for the whole app |
| assets/ | Images, icons, and other static files |
| ArrayFunctions.js, DOM\_Manipulation.js | Your JS practice or helper code |

Type <http://localhost:5173/> in browser



* Download ES7+React/Redux/React-Native snippets in from extensions
* Remove everything from App.jsx
* type rafce in app.jsx and click enter
* the output of code present in app.jsx will be shown in that react page
* remove everything from index.css

Eg:

import React from 'react'

const App = () => {

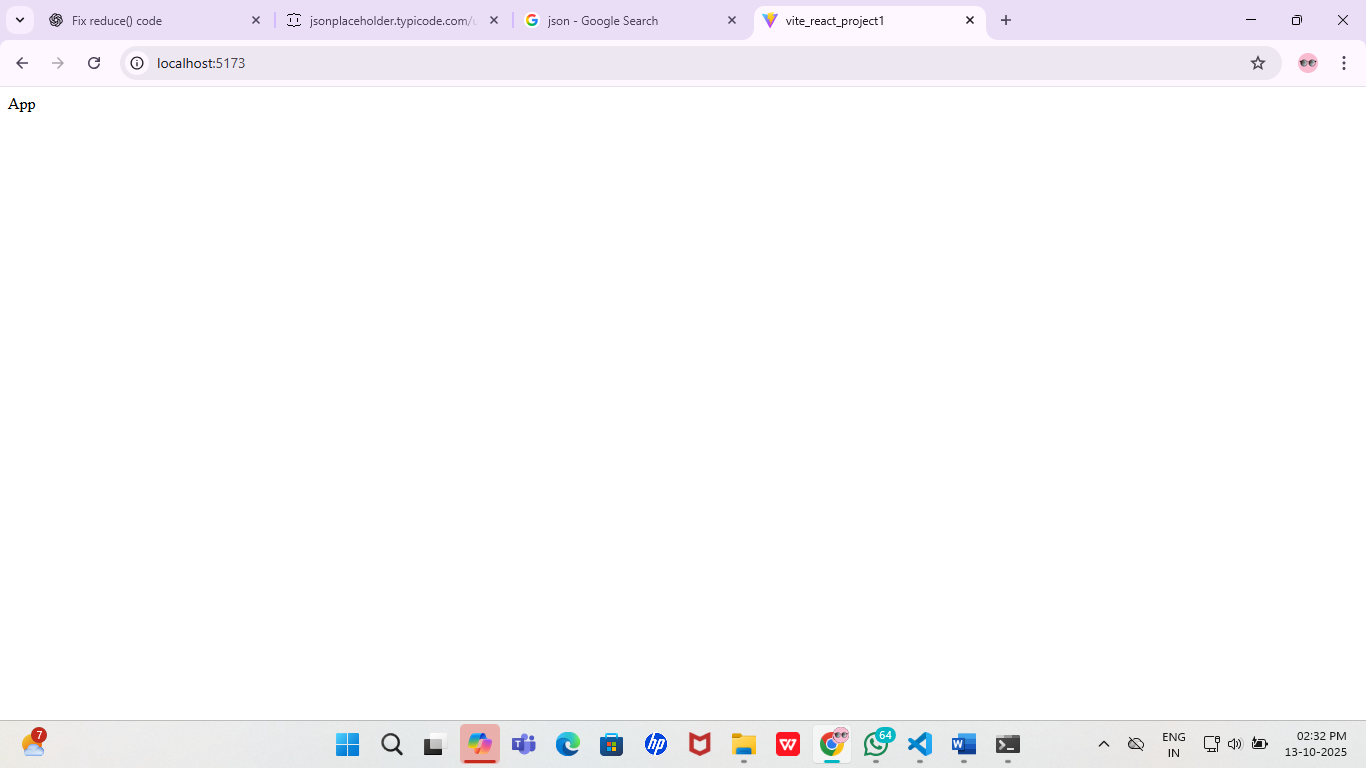
  return (

    <div>App</div>

  )

}

export default App



**Notes:**

* Functions gets stored in stack

Eg:

const mul=(a,b)=>{

    return a\*b;

}

const square=(n)=>{

    return mul(n,n);

}

const print=(n)=>{

    let s=square(n);

    console.log(s);

}

print(4);

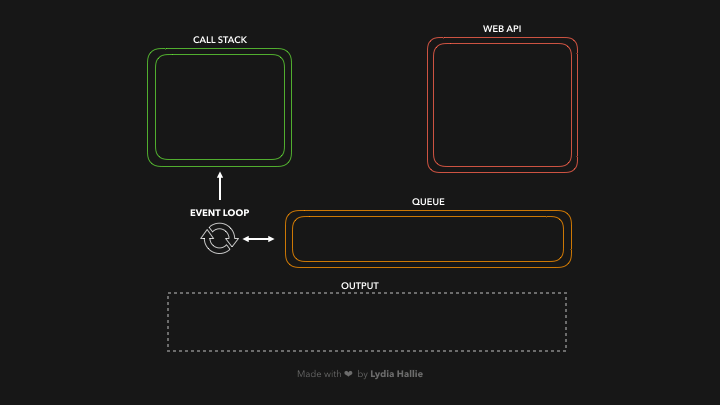
* Objects will be stored in Heap
* Javascript is generally synchronous
* We make Javascript work as asynchronous using event looping

**Event Loop**

* Queue
* Event loop
* Call Stack
* Web API

**Eg:**

**const foo = () => console.log("First");  
const bar = () => setTimeout(() => console.log("Second"), 500);  
const baz = () => console.log("Third");  
bar();  
foo();  
baz();Output:First  
Third   
Second**



**Eg:**

console.log("first");

Promise.resolve().then(console.log("From promises"));

setTimeout(()=>{

    console.log("Inside Timeout");

},2000)

console.log("End");

**Output:**

first

From promises

End

Inside Timeout

**Portfolio Model**

<https://youtu.be/zJE-ze4TfXc?si=GCvcpZGH0KyhEb3L>

**Used to create animations**

React Bits

GSAP

Framer

Spline