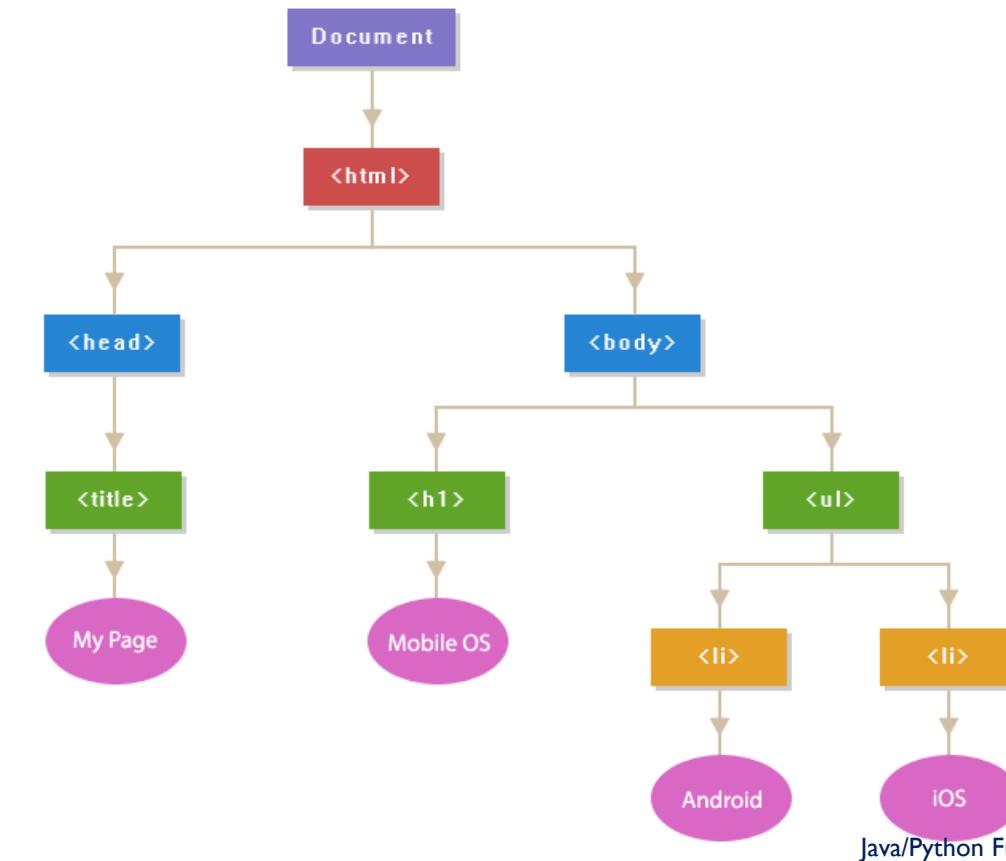


ReactJS

What is React

- ReactJS is an open-source JavaScript library Created and Maintained by Facebook.
- React is a flexible and efficient JS library capable of rendering front-ends of simple, scalable, and fast web applications
- React is a component-based framework & It is a popular choice for building SPA (single-page applications).
- Features of ReactJS
 - Virtual DOM

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>My Page</title>
5  </head>
6  <body>
7      <h1>Mobile OS</h1>
8      <ul>
9          <li>Android</li>
10         <li>iOS</li>
11     </ul>
12 </body>
13 </html>
```



- Virtual DOM is an abstraction of the Real DOM, created and maintained by JavaScript libraries such as React.
- The Virtual DOM is a lightweight copy of the Real DOM, which allows for faster updates and improved performance.
- When a user interacts with a web page, React updates the Virtual DOM, compares it with the previous version, and only updates the Real DOM with the necessary changes

JSX (JavaScript Syntax Extension)

- JSX, is a combination of HTML and JavaScript
- It describes the user interface appearance of an app and is used to create React components

Extensions

- React supports various extensions used in React application. Examples., server-side rendering, Redux, React Hooks

One-way Data Binding

- One-way binding means that React uses unidirectional data flow, i.e., data is transferred in one direction, which is from top to bottom or from parent to child components.

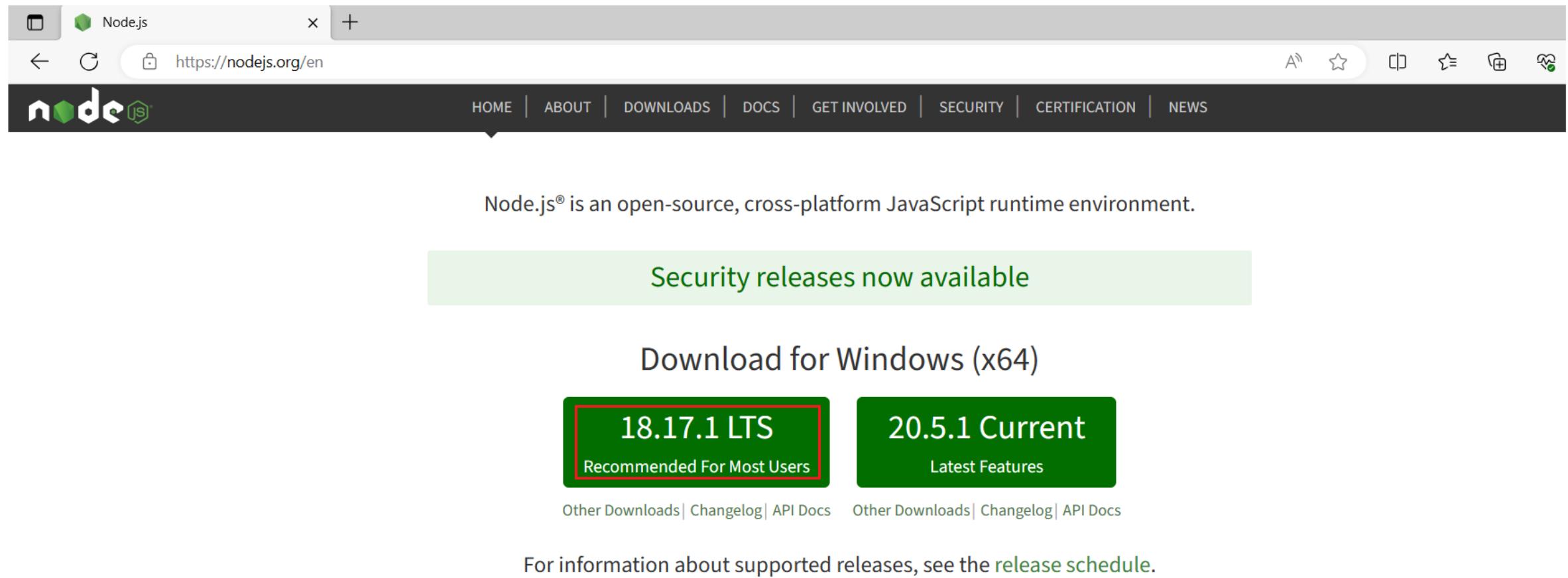
Component-based Architecture

- ReactJS divides the application into multiple components.

Software Needed

- Bowser
- VS Code
- Node JS
- React JS

Installing NodeJS



The screenshot shows the official Node.js website at <https://nodejs.org/en>. The page header includes a back button, a refresh button, a search icon, and a URL bar showing the site's address. Below the header is a dark navigation bar with links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. The main content area features a green banner with the text "Security releases now available". Below this, there are two prominent download buttons: a red-bordered button for "18.17.1 LTS" labeled "Recommended For Most Users" and a green button for "20.5.1 Current" labeled "Latest Features". At the bottom of the page, there are links for "Other Downloads", "Changelog", and "API Docs" for both release versions.

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Security releases now available

Download for Windows (x64)

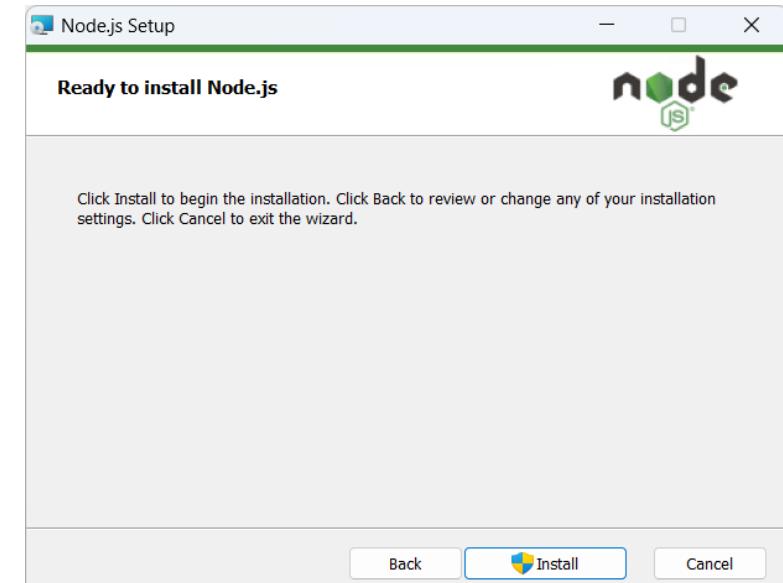
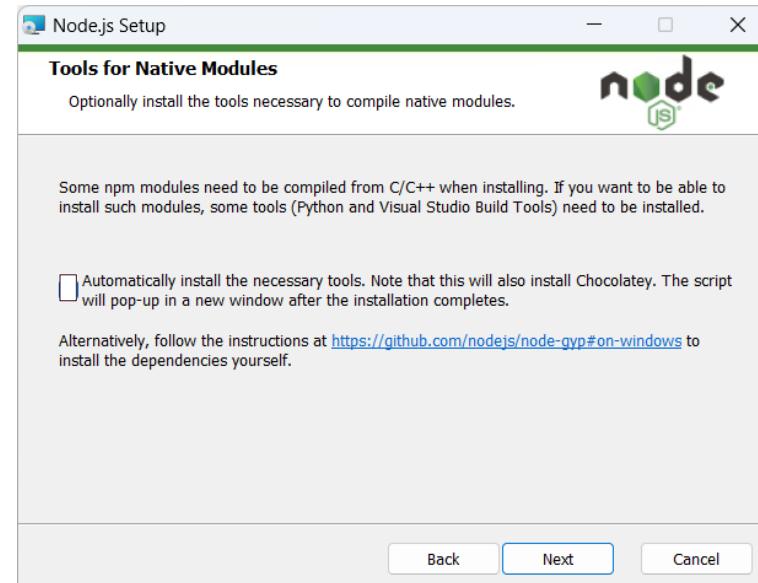
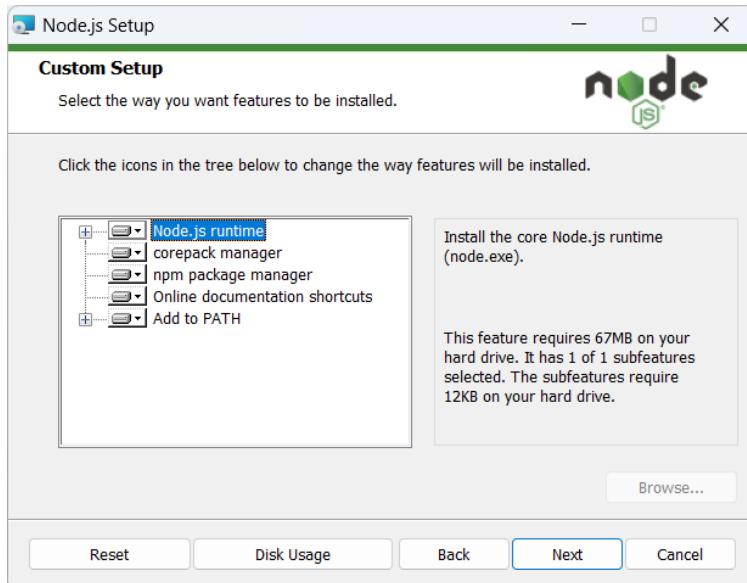
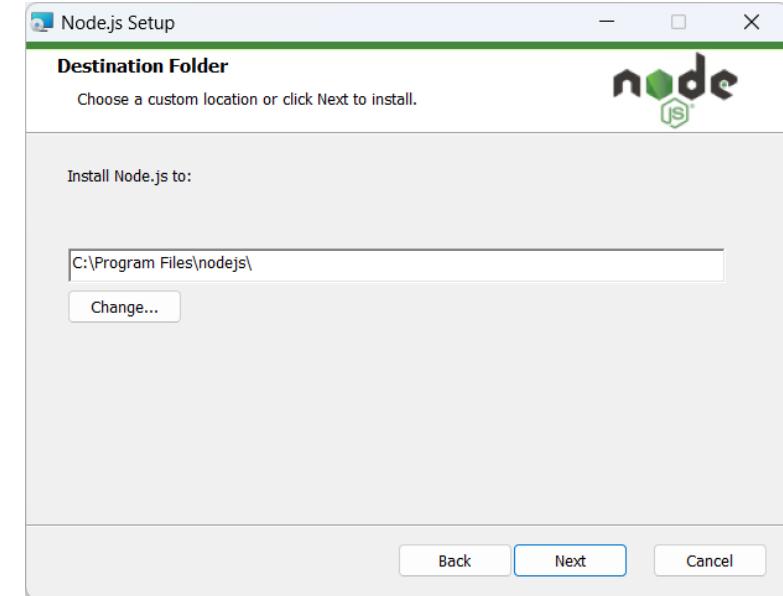
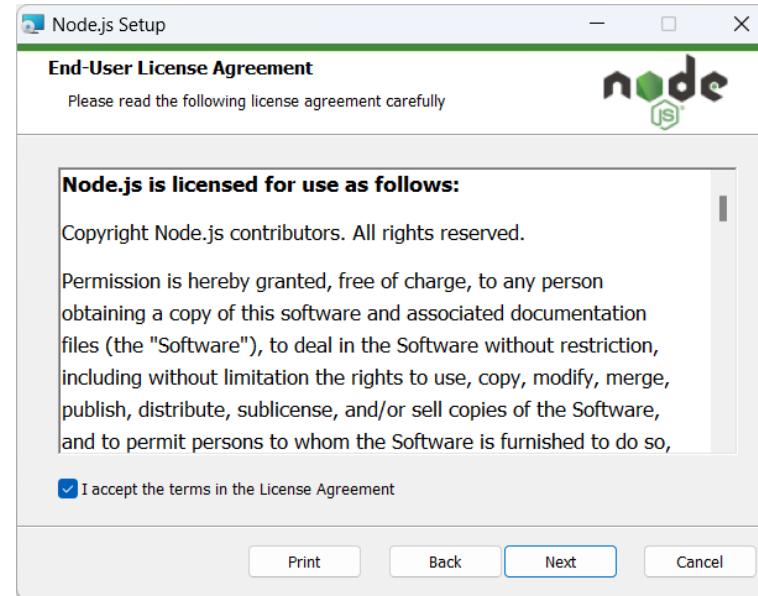
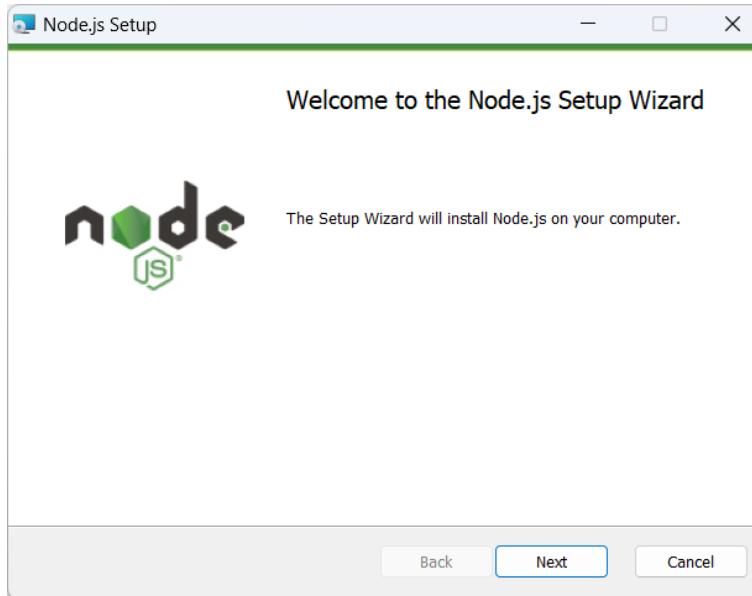
18.17.1 LTS
Recommended For Most Users

20.5.1 Current
Latest Features

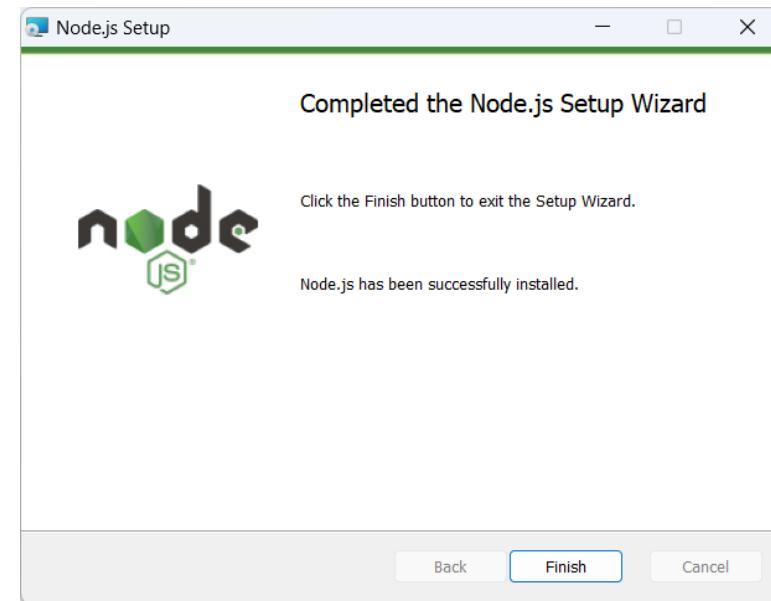
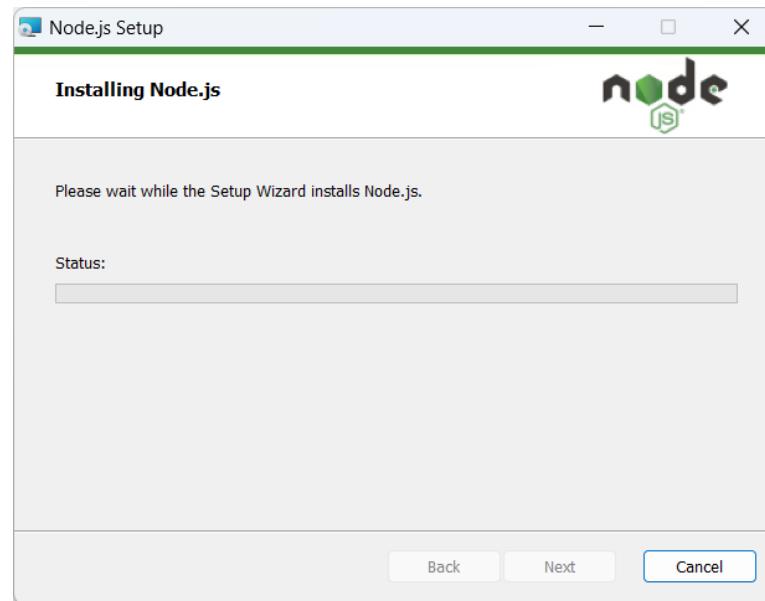
Other Downloads | Changelog | API Docs Other Downloads | Changelog | API Docs

For information about supported releases, see the [release schedule](#).

Installing NodeJS...



Installing NodeJS...



A screenshot of a Windows Command Prompt window titled "Command Prompt". The title bar shows "Microsoft Windows [Version 10.0.22621.2134]" and "(c) Microsoft Corporation. All rights reserved.". The command line shows two commands run in sequence:

```
C:\Users\Mohammed Ali>node --version  
v18.17.1
```



```
C:\Users\Mohammed Ali>npm --version  
9.6.7
```

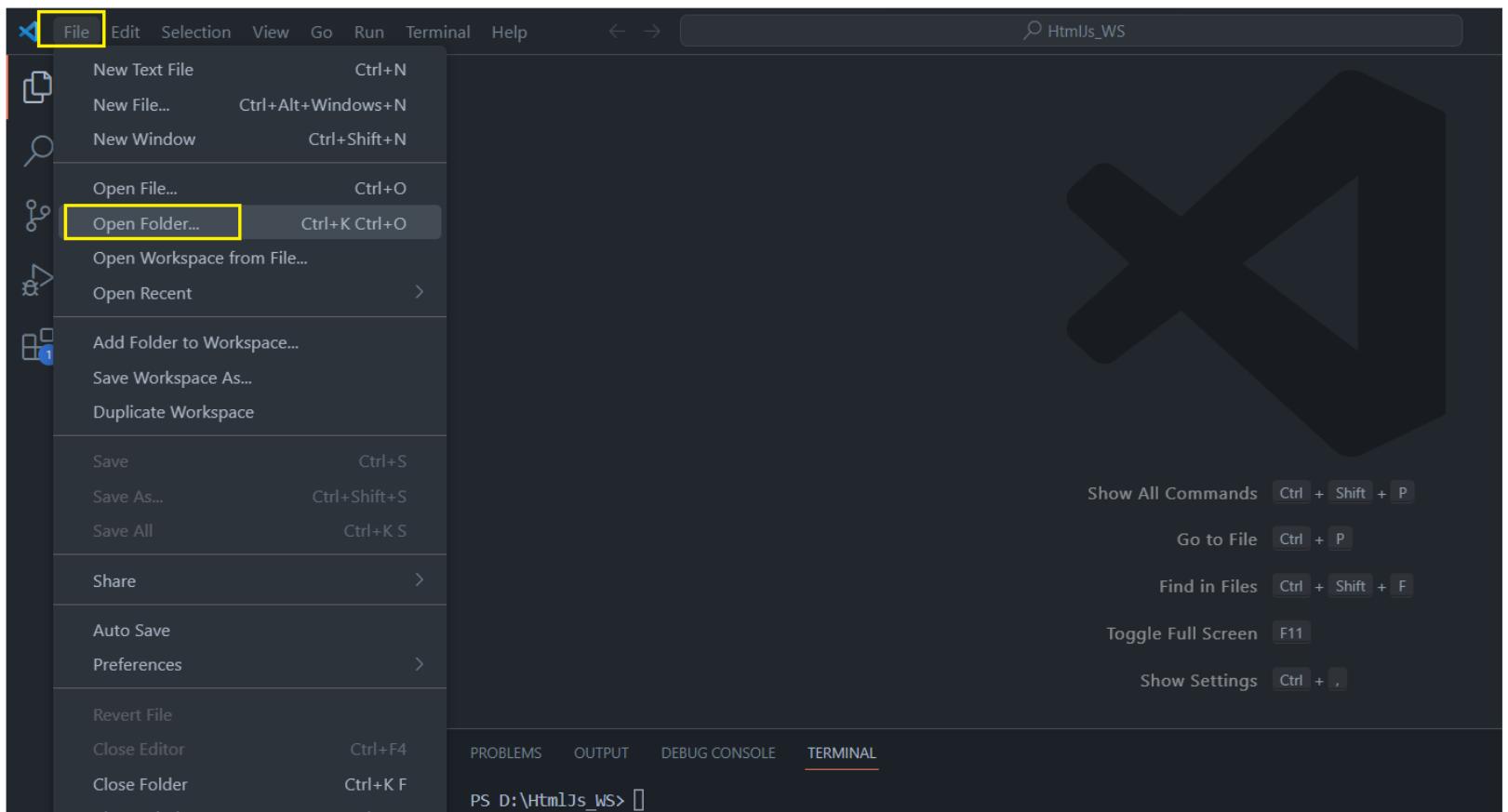
The first command's output is highlighted with an orange border, and the second command's output is highlighted with a green border.

Create ReactJS Application

- STEP-1: Create "react_workspace" folder in C or D drive
- STEP-2: cd react_workspace & run below command
D:\react_workspace>npx create-react-app emp-reactapp
- STEP-3: cd emp-reactapp & run below command
D:\react_workspace\emp-reactapp>npm start

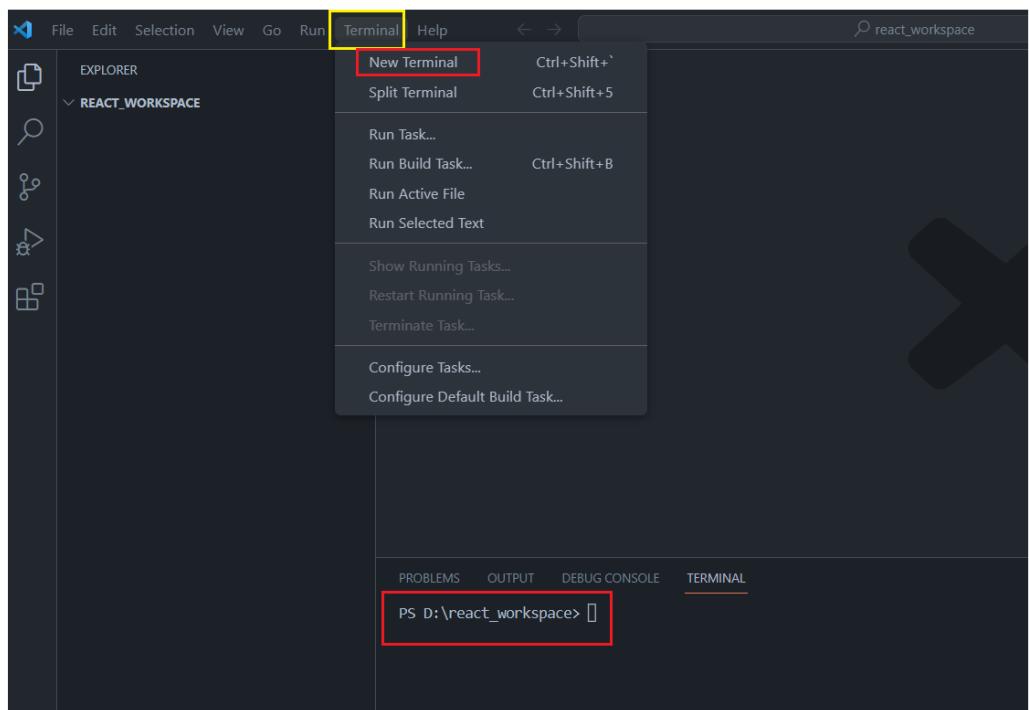
STEP - I

STEP-I: Create
"react_workspace"
folder in C or D
drive

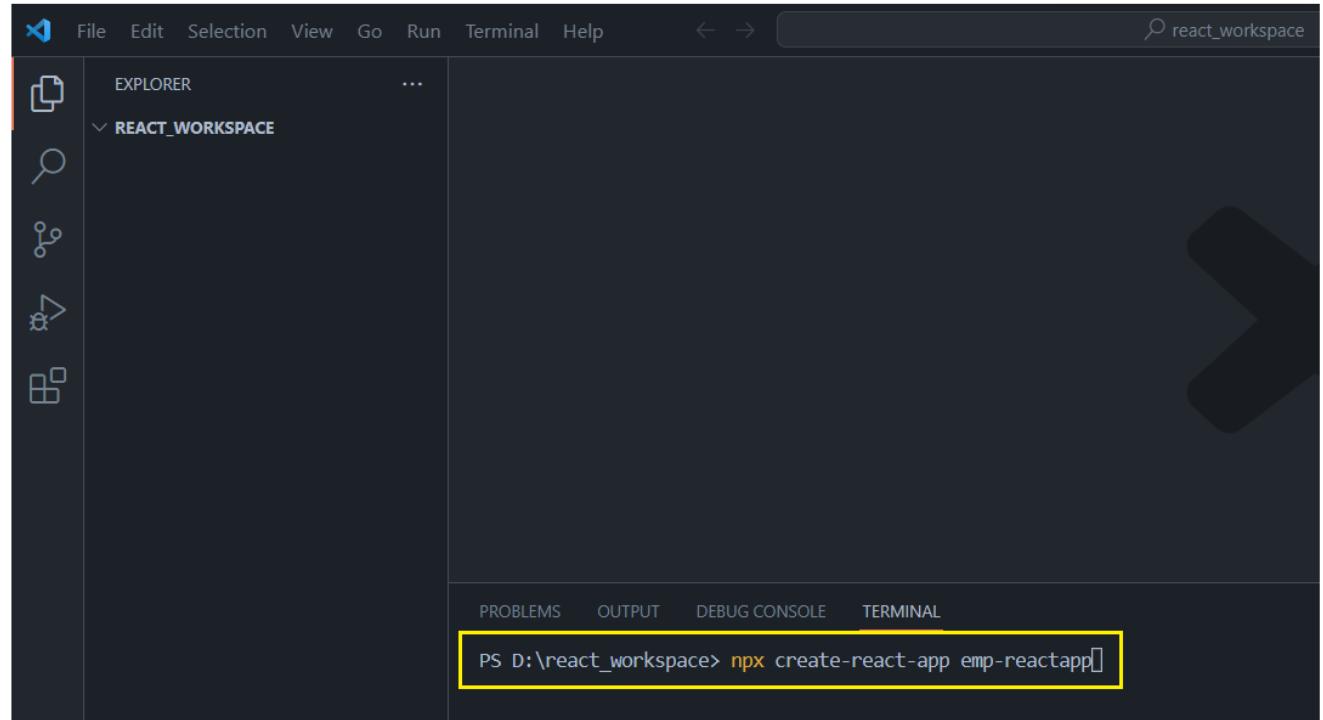


STEP - 2

run `create-react-app` command



A screenshot of the Visual Studio Code interface. The title bar shows "react_workspace". The left sidebar has icons for Explorer, Search, Problems, and Terminal. The main area shows a dark workspace with a large "X" icon. The bottom navigation bar includes PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL tabs, with TERMINAL selected. A terminal window is open with the prompt "PS D:\react_workspace> []". A yellow box highlights the "Terminal" tab in the top menu bar, and another yellow box highlights the terminal input field.



A screenshot of the Visual Studio Code interface, similar to the first one but with a different terminal command. The title bar shows "react_workspace". The left sidebar is visible. The main area shows a dark workspace with a large "S" icon. The bottom navigation bar includes PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL tabs, with TERMINAL selected. A terminal window is open with the prompt "PS D:\react_workspace> npx create-react-app emp-reactapp []". A yellow box highlights the terminal input field.

STEP – 2...

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command `npx create-react-app emp-reactapp` being run in the directory `D:\react_workspace`. The output shows the creation of a new React app named `emp-reactapp`, including the installation of packages and configuration steps. A red box highlights the command and its initial output. Another red box highlights the success message and the suggested commands for the new app.

```
File Edit Selection View Go Run Terminal Help ⏪ react_workspace  
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
PS D:\react_workspace> npx create-react-app emp-reactapp  
Need to install the following packages:  
  create-react-app@5.0.1  
  ok to proceed? (y) y  
npm WARN deprecated tar@2.2.2: This version of tar is no longer supported, and will not receive security updates. Please upgrade asap.  
Creating a new React app in D:\react_workspace\emp-reactapp.  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts with cra-template...  
added 1441 packages in 4m  
241 packages are looking for funding  
  run `npm fund` for details  
Initialized a git repository.  
Installing template dependencies using npm...  
added 69 packages, and changed 1 package in 40s  
245 packages are looking for funding  
  run `npm fund` for details  
Removing template package using npm...  
removed 1 package, and audited 1510 packages in 3s  
245 packages are looking for funding  
  run `npm fund` for details  
6 high severity vulnerabilities  
To address all issues (including breaking changes), run:  
  npm audit fix --force  
Run `npm audit` for details.  
Success! Created emp-reactapp at D:\react_workspace\emp-reactapp  
Inside that directory, you can run several commands:  
  npm start  
    Starts the development server.  
  npm run build  
    Bundles the app into static files for production.  
  npm test  
    Starts the test runner.  
  npm run eject  
    Removes this tool and copies build dependencies, configuration files  
    and scripts into the app directory. If you do this, you can't go back!  
We suggest that you begin by typing:  
  cd emp-reactapp  
  npm start  
Happy hacking!  
npm notice  
npm notice New major version of npm available! 9.6.7 -> 10.0.0  
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.0.0  
npm notice Run npm install -g npm@10.0.0 to update!  
npm notice  
PS D:\react_workspace>
```

index.html < ==> index.js < ==> App.js

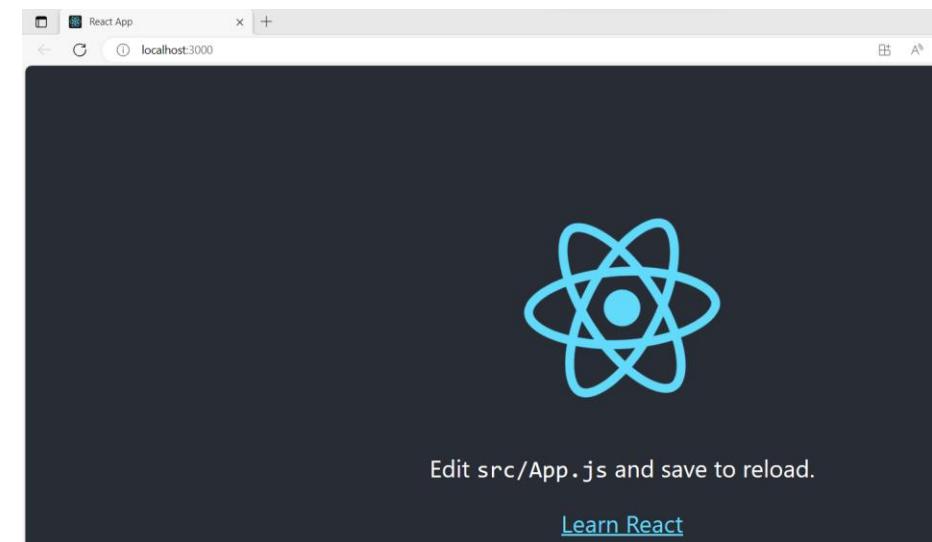
STEP-3: Run React Application

Go to React Project folder & Run below command

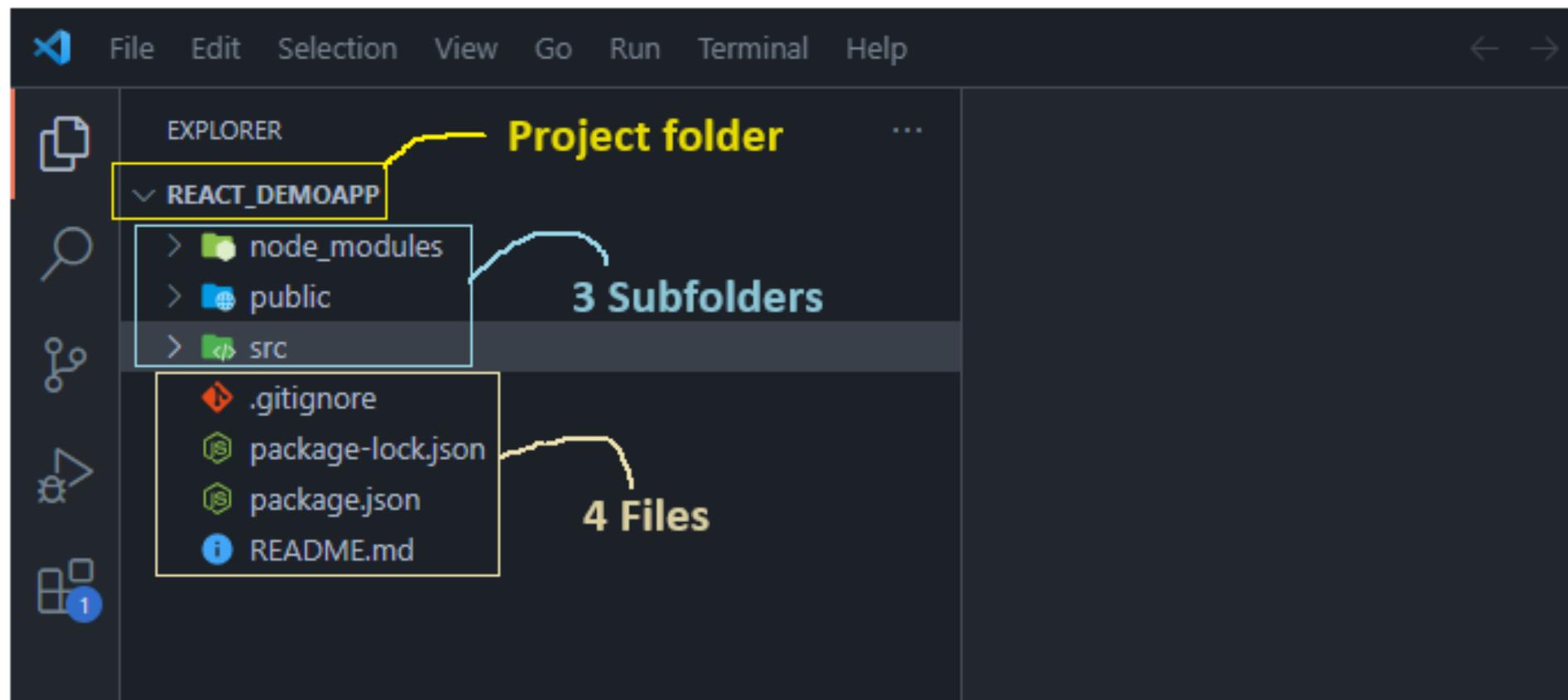
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
PS D:\react_workspace> cd emp-reactapp  
PS D:\react_workspace\emp-reactapp> 
```

D:\react_workspace\emp-reactapp>npm start

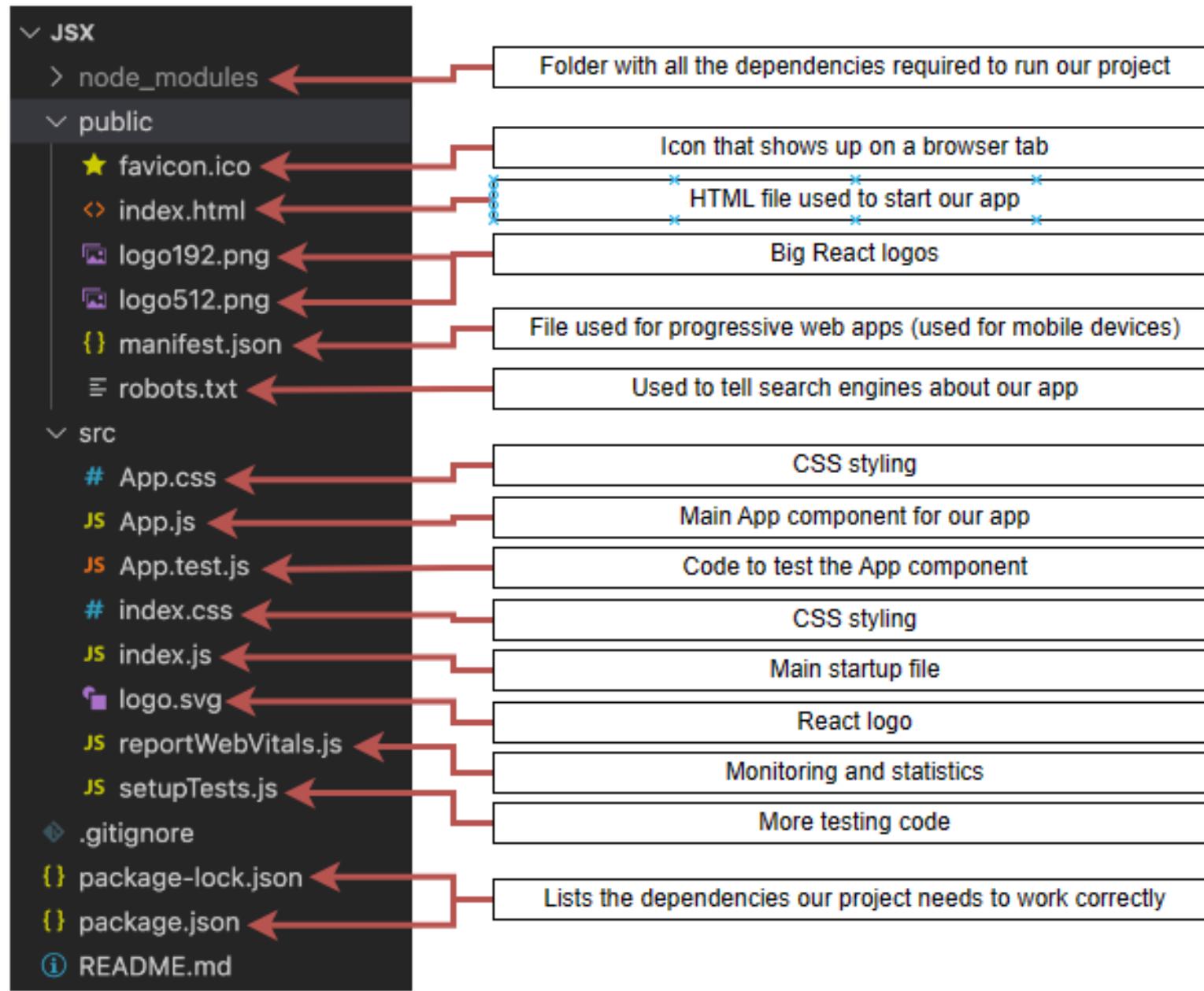
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
Compiled successfully!  
You can now view emp-reactapp in the browser.  
Local: http://localhost:3000  
On Your Network: http://192.168.0.104:3000  
Note that the development build is not optimized.  
To create a production build, use npm run build.  
webpack compiled successfully
```



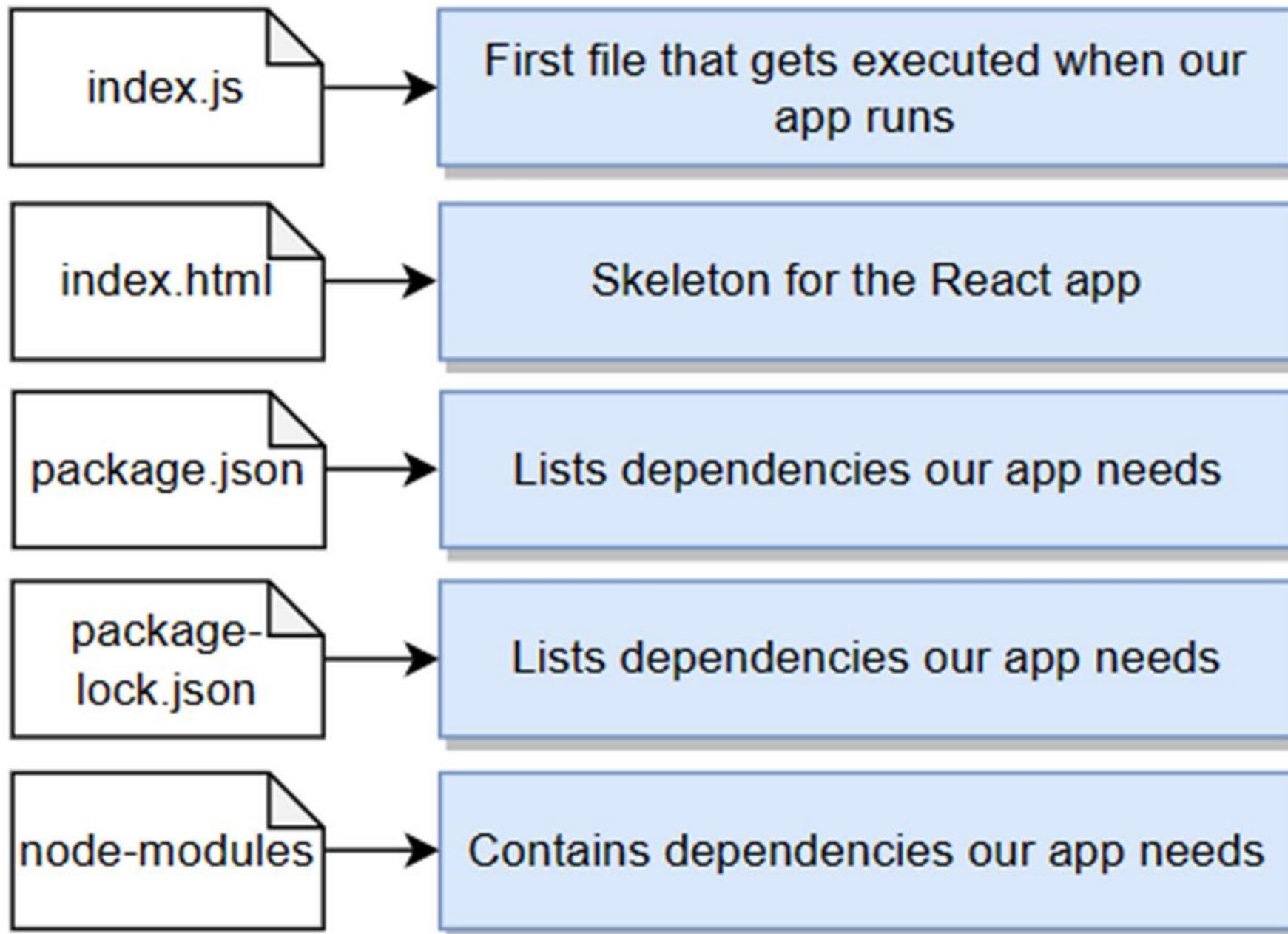
React Project Folder Structure



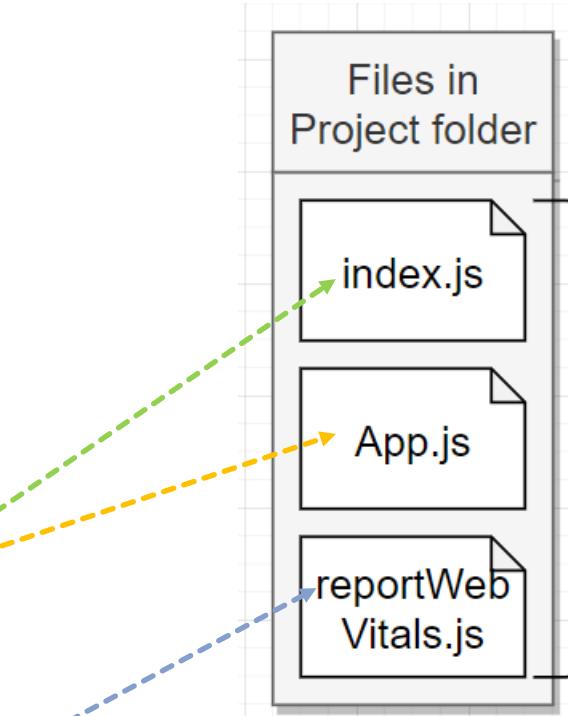
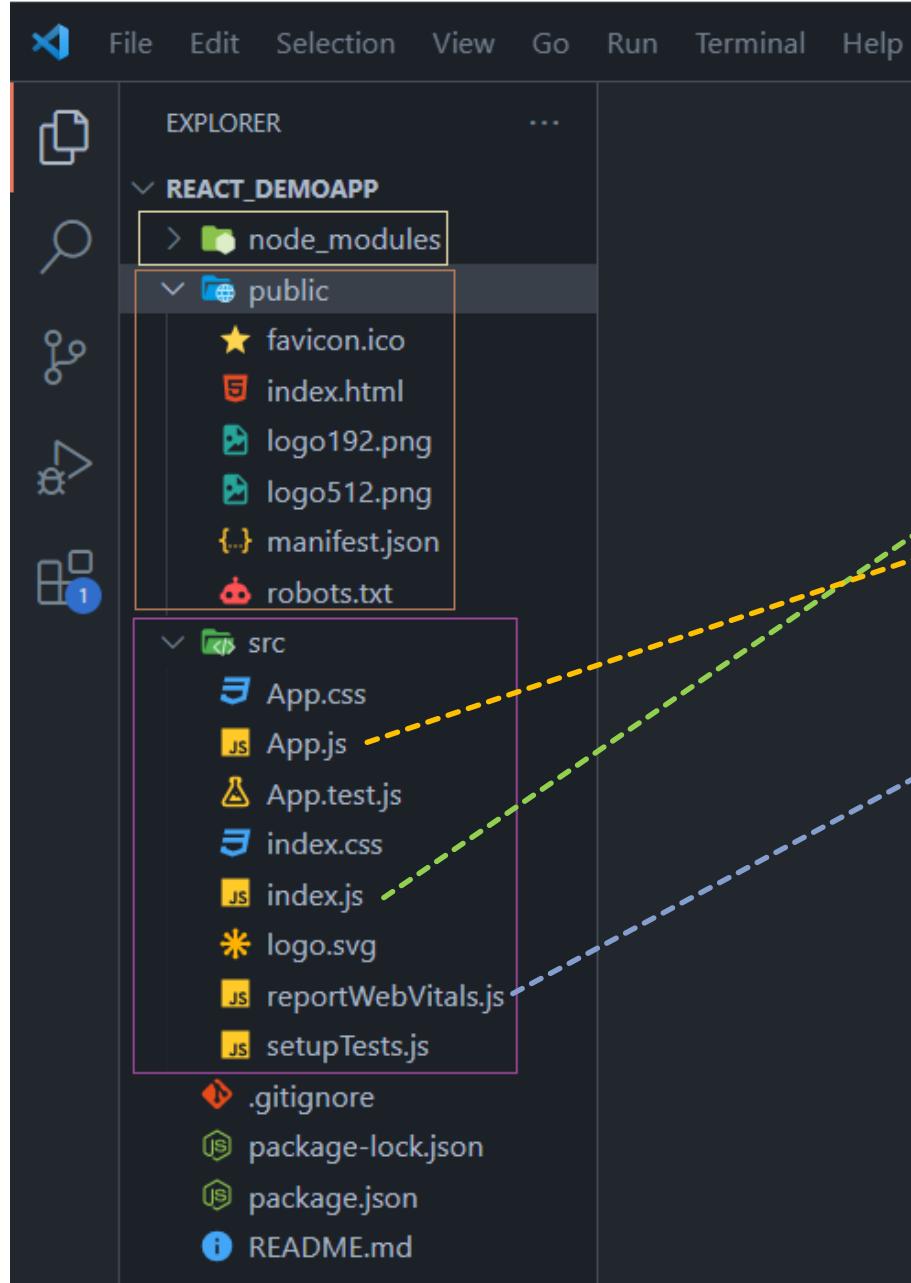
React Project – Different files



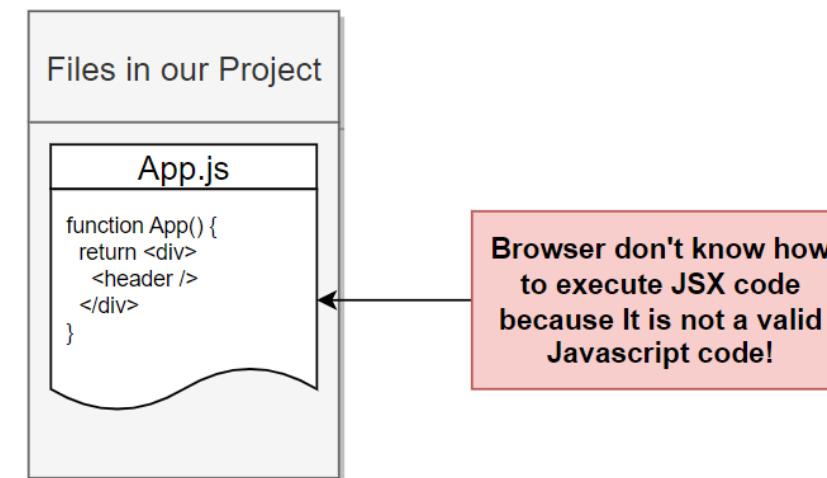
React Project – Some Important files



Need of Transpiler



These files contain JSX code & browser doesn't know how to execute JSX code



Solution is, We need to transpile the code into browser understandable format

How does React Application Works?

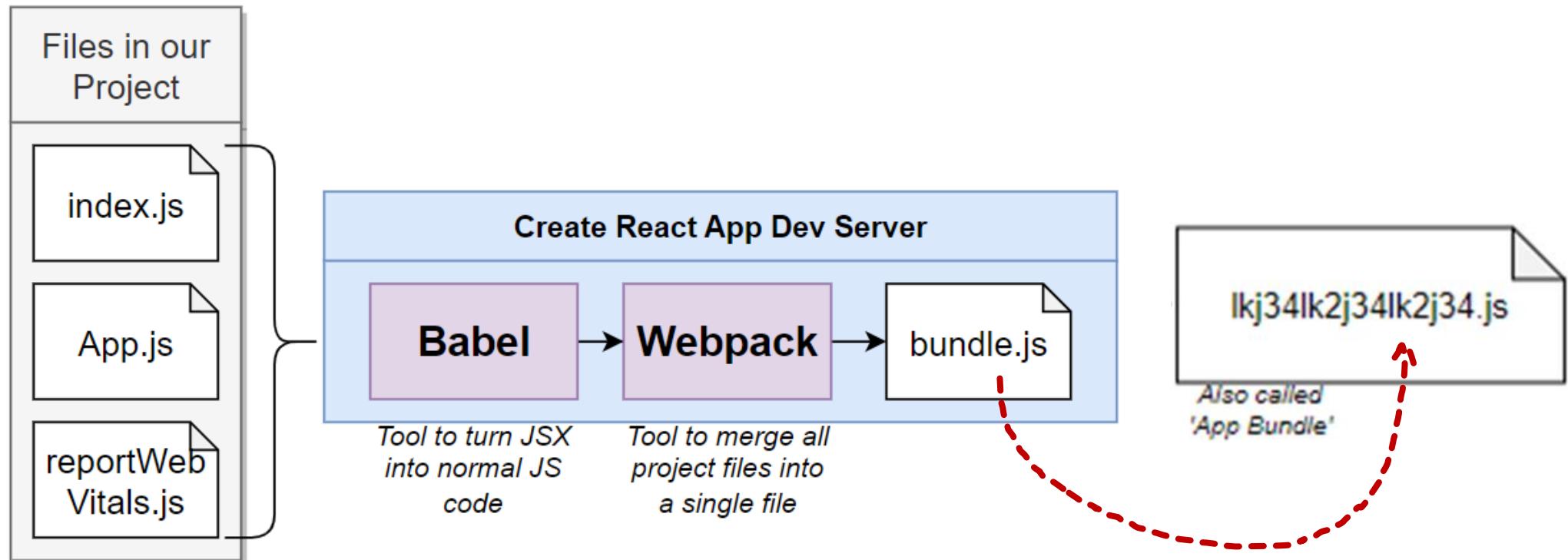
How does a React app works?

- 1 All of our project's JS files are 'bundled' together into a single file called `bundle.js`, then `bundle.js` will be placed onto a server
- 2 User makes a request to the server and gets an HTML file + the bundle
- 3 User's browser executes your code

How does React Application Works...?

1

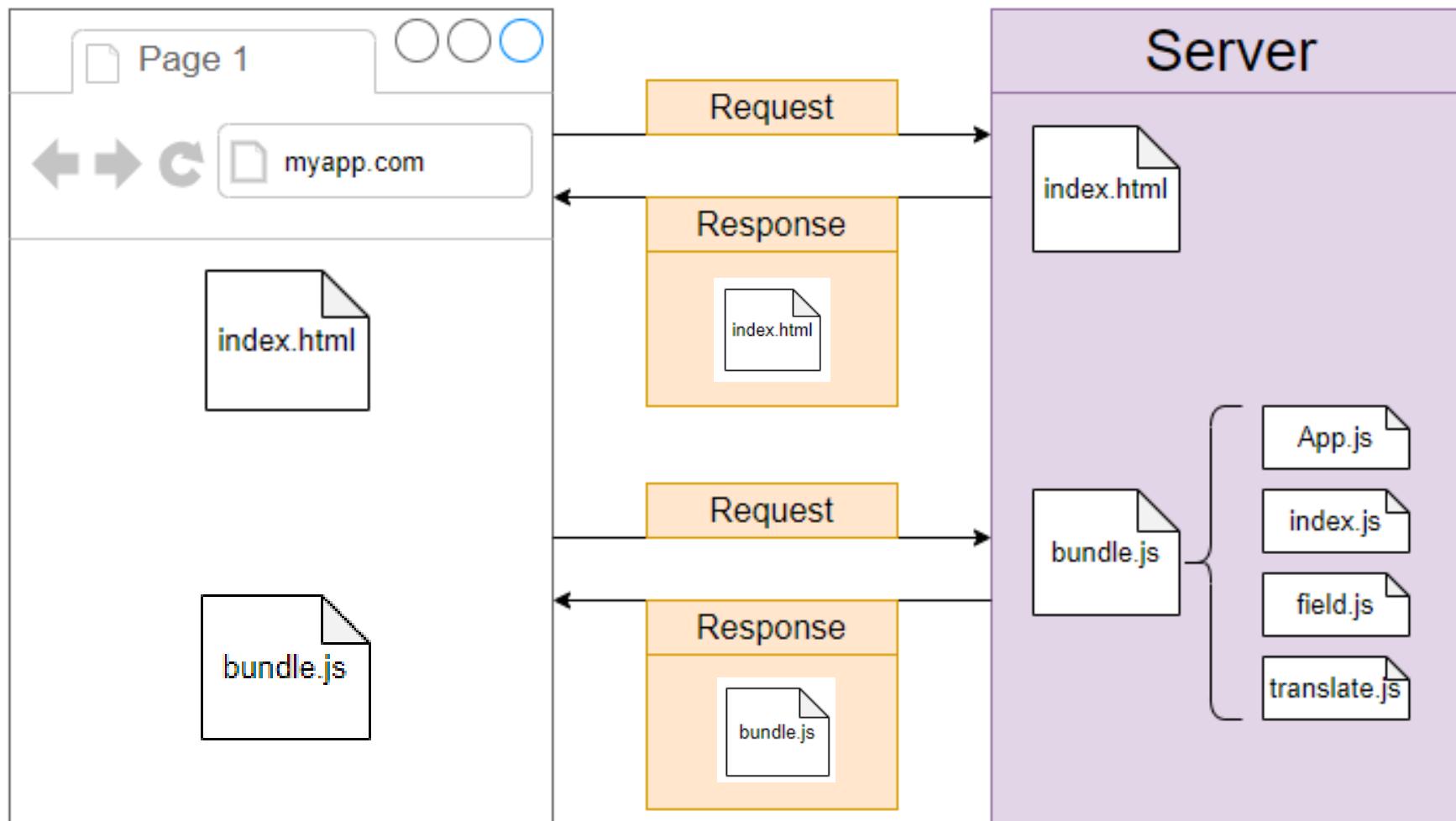
All of our project's JS files are 'bundled' together into a single file, then placed onto a server



How does React Application Works ... ?

2

User makes a request to the server and gets an HTML file + the bundle



How does React Application Works... ?

3

User's browser executes your code

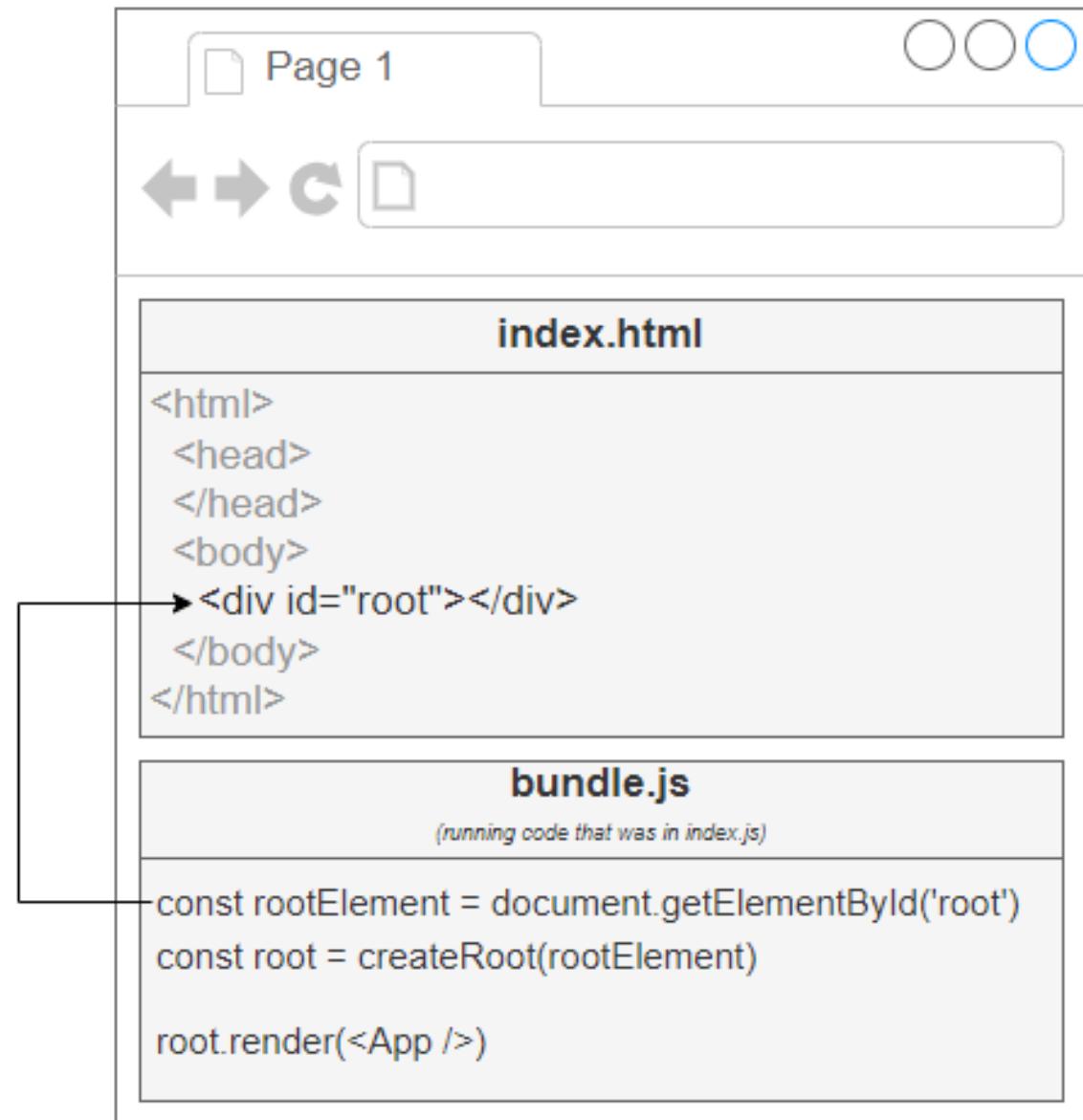
React Startup Process

All inside the `index.js` file

1 Find the div with id of 'root' in the DOM

2 Tell React to take control of that element

3 Tell React to get JSX from the App component, turn it into HTML, and show it in the root

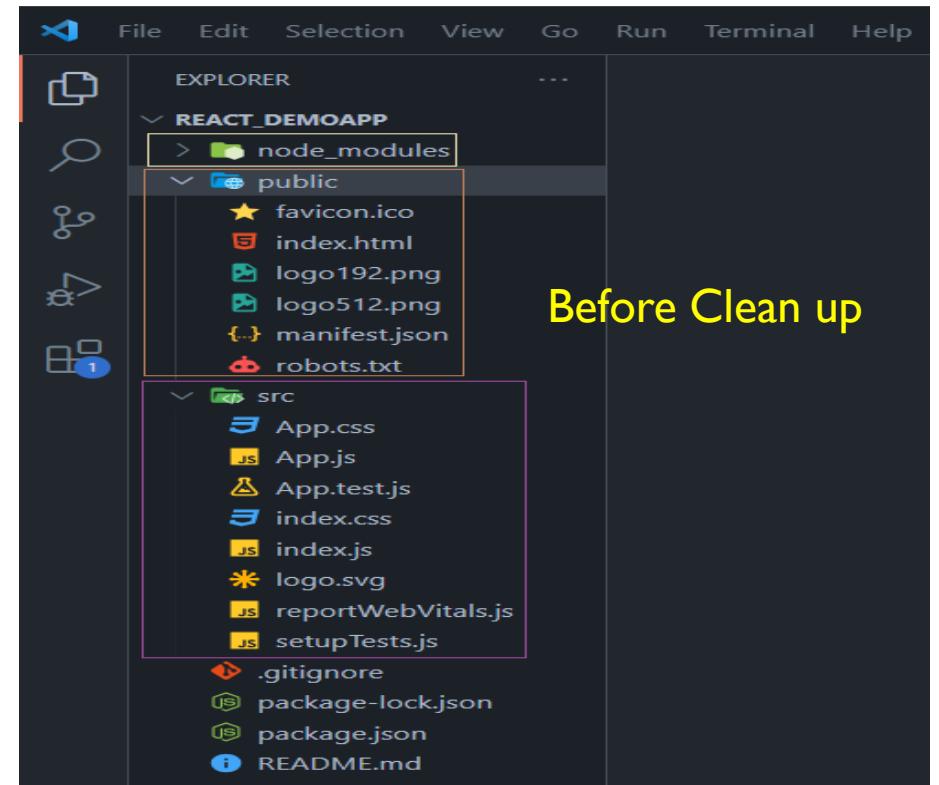


Creating a Plain React Application

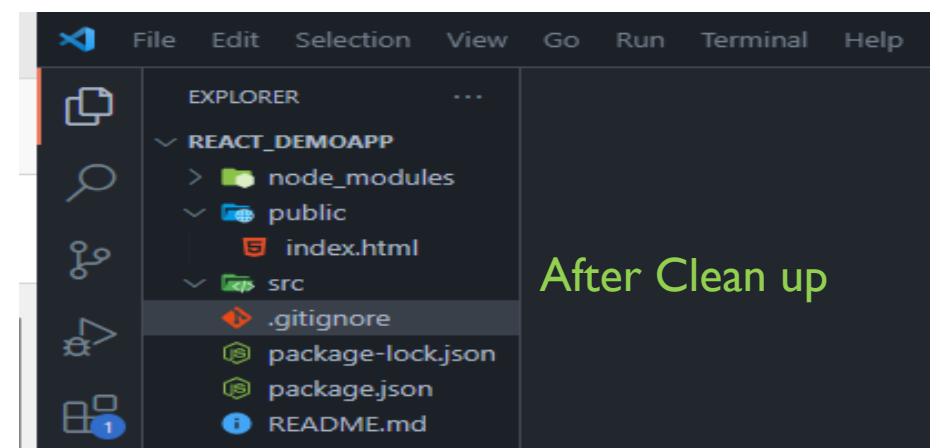


Starting Plain React Application

- Remove all un-necessary files from the project folder
 - ✓ From public folder remove all files except index.html
 - ✓ From src folder delete all the files
- Create index.js file inside src folder.
 - ✓ index.js is the first file which gets executed when our application runs inside the browser
 - ✓ Inside index.js file we will have a code for Component which will be show on the screen
 - ✓ Creating a component is a FIVE Steps process



Before Clean up



After Clean up

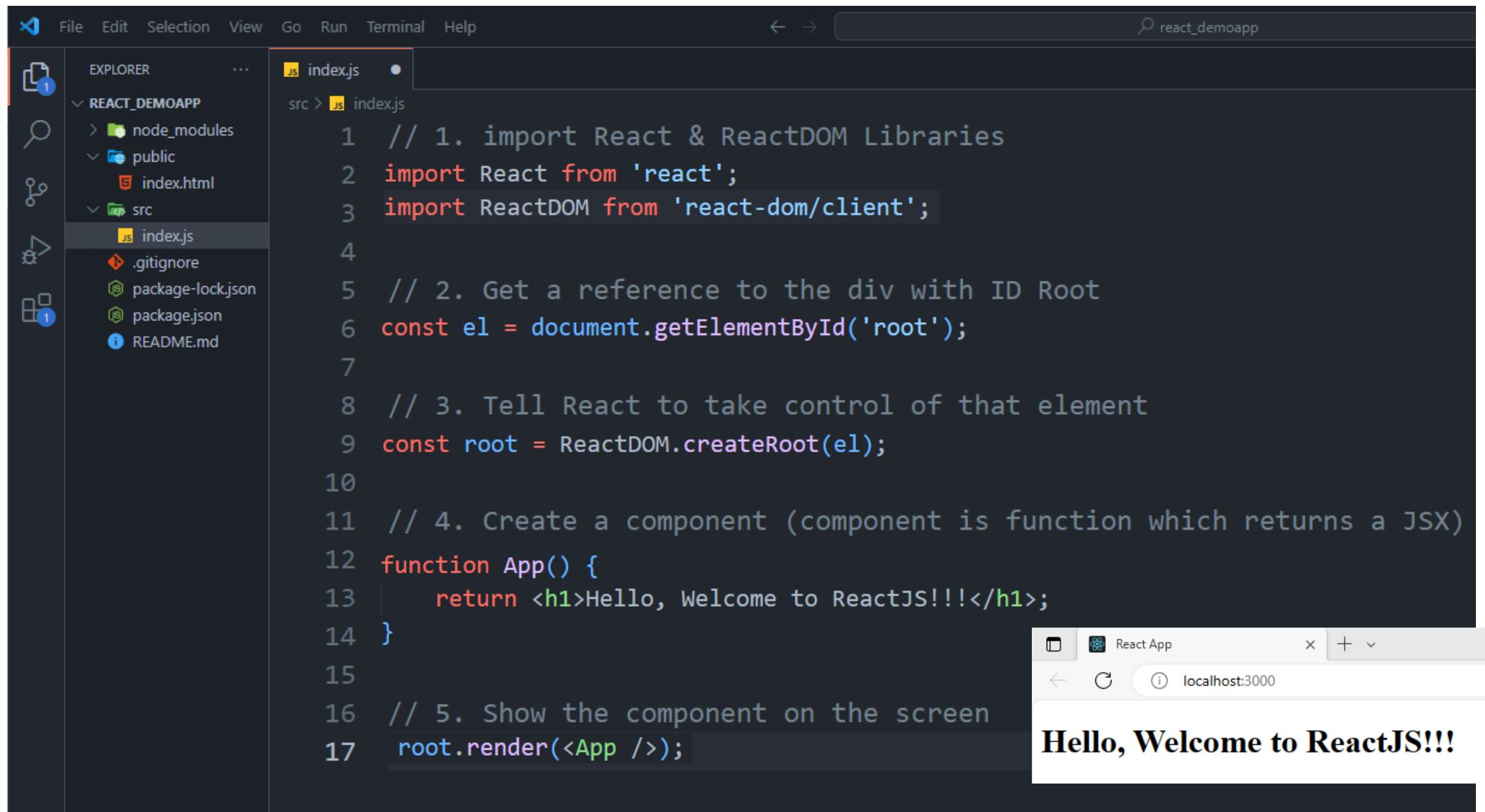
public/index.html

The screenshot shows the VS Code interface with the file `index.html` open in the editor. The code is a template for a React application. A yellow box highlights the line `<div id="root"></div>`, which is the root element where the React application will be rendered.

```
File Edit Selection View Go Run Terminal Help
EXPLORER index.js index.html
REACT_DEMOAPP
  node_modules
  public
    index.html
  src
    index.js
    .gitignore
    package-lock.json
    package.json
  README.md
public > index.html > html > body > div#root
WORK CORRECTLY DOWN WITH CLIENT-SIDE ROUTING AND A NON-ROOT PUBLIC URL.
24      WORK CORRECTLY DOWN WITH CLIENT-SIDE ROUTING AND A NON-ROOT PUBLIC URL.
25      Learn how to configure a non-root public URL by running `npm run build`.
26      -->
27      <title>React App</title>
28      </head>
29      <body>
30      <noscript>You need to enable JavaScript to run this app.</noscript>
31      <div id="root"></div>
32      <!--
33      This HTML file is a template.
34      If you open it directly in the browser, you will see an empty page.
35
36      You can add webfonts, meta tags, or analytics to this file.
37      The build step will place the bundled scripts into the <body> tag.
38
39      To begin the development, run `npm start` or `yarn start`.
40      To create a production bundle, use `npm run build` or `yarn build`.
41      -->
42      </body>
43      </html>
44
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

src/index.js



The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar displays the project structure under 'REACT_DEMOAPP': node_modules, public (with index.html), and src (containing index.js, .gitignore, package-lock.json, package.json, and README.md). The 'index.js' file in the src folder is open in the main editor area. The code is as follows:

```
// 1. import React & ReactDOM Libraries
import React from 'react';
import ReactDOM from 'react-dom/client';

// 2. Get a reference to the div with ID Root
const el = document.getElementById('root');

// 3. Tell React to take control of that element
const root = ReactDOM.createRoot(el);

// 4. Create a component (component is function which returns a JSX)
function App() {
  return <h1>Hello, Welcome to ReactJS!!!</h1>;
}

// 5. Show the component on the screen
root.render(<App />);
```

In the bottom right corner, a browser window titled 'React App' shows the rendered output: 'Hello, Welcome to ReactJS!!!'. The browser's address bar indicates the URL is 'localhost:3000'.

What is JSX

React JSX

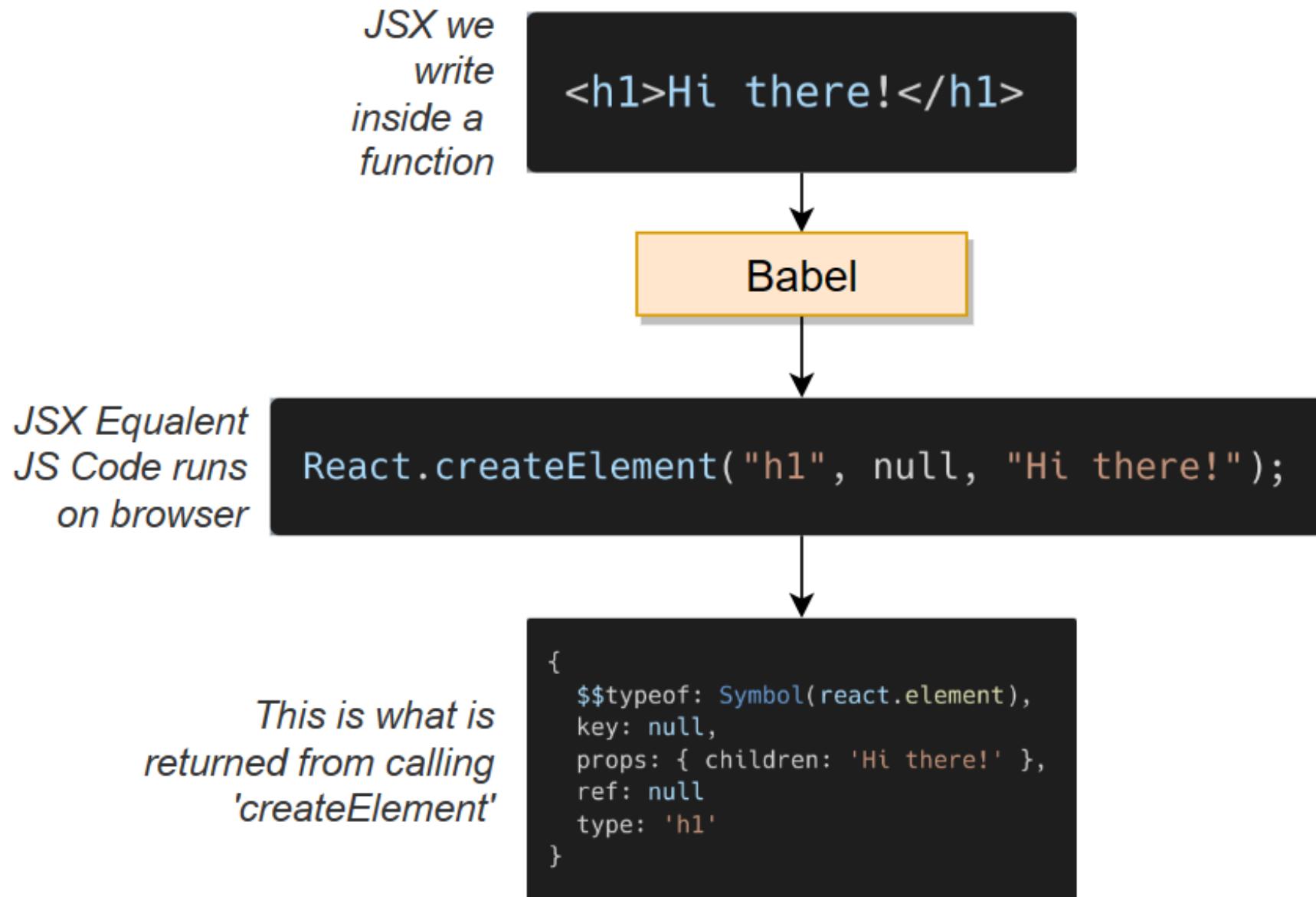
- JSX stands for JavaScript XML.
- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement()
- JSX converts HTML tags into react elements.
- JSX makes it easier to write and add HTML in React.

JSX:

```
const myElement = <h1>I Love JSX!</h1>;
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Why we use JSX



babeljs.io/repl

BABEL

```
1 <div>
2   <h1>Hello, Welcome to JSX</h1>
3   <ul>
4     <li>Ali</li>
5     <li>Malli</li>
6     <li>Ravi</li>
7   </ul>
8 </div>
```

JSX Code

```
1 import { jsx as _jsx } from "react/jsx-runtime";
2 import { jsxs as _jsxs } from "react/jsx-runtime";
3 /*#__PURE__*/_jsxs("div", {
4   children: [/*#__PURE__*/_jsx("h1", {
5     children: "Hello, Welcome to JSX"
6   }), /*#__PURE__*/_jsxs("ul", {
7     children: [/*#__PURE__*/_jsx("li", {
8       children: "Ali"
9     }), /*#__PURE__*/_jsx("li", {
10       children: "Malli"
11     }), /*#__PURE__*/_jsx("li", {
12       children: "Ravi"
13     })
14   })
15 });
```

Java Script Code runs on browser

Important Note

```
<h1>Hi there!</h1>
```

Writing this doesn't make anything show up in the browser automatically

This creates an **instruction** for React, telling it to make an element

We have to **return** it from a component for React to use it

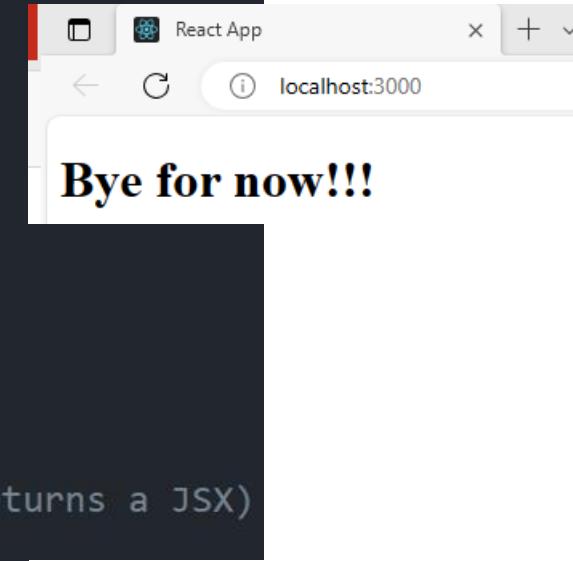
```
function App() {  
  return(  
    <div>  
      <h1>Hello, This is my First React Component</h1>;  
      <h1>Hello, This is my First React Component12345</h1>;  
    </div>  
  );  
}
```

NOTE, in below function since we are not returning first **<h1>** Tag content it will not show up on the browser

```
function App() {  
  <h1>This is extra text from COMPONENT</h1>  
  return <h1>Hello, Welcome to ReactJS!!!..</h1>;  
}
```

JavaScript Variables in JSX

```
index.js x
src > index.js > App
1 // 1. import React & ReactDOM Libraries
2 import React from 'react';
3 import ReactDOM from 'react-dom/client';
4
5 // 2. Get a reference to the div with ID Root
6 const el = document.getElementById('root');
7
8 // 3. Tell React to take control of that element
9 const root = ReactDOM.createRoot(el);
10
11 // 4. Create a component (component is function which returns a JSX)
12 function App() {
13   let message = 'Bye for now!!!';
14
15   if(Math.random() > 0.5) {
16     message = 'Hello, Guest';
17   }
18
19   return <h1>{message}</h1>;
20 }
21
22 // 5. Show the component on the screen
23 root.render(<App />);
```



Bye for now!!!

Curly braces mean...referencing a JS variable or expression

```
function App() {
  const message = 'Hi there!';

  return <h1>{message}</h1>
}
```

JavaScript Variables in JSX...

- Most of the times we use curly braces to show String OR Numbers

```
function App() {  
  let message = 'Hello, Guest';  
  
  return <h1>{message}</h1>;  
}
```

```
function App() {  
  let sum = 10 + 12;  
  
  return <h1>{sum}</h1>;  
}
```

Some values might not display as we expect

```
function App() {  
  const value = true;  
  
  return <h1>{value}</h1>  
}
```

This does not show anything on browser

- ✓ true
- ✓ false
- ✓ null
- ✓ Undefined
- ✓ [1, 2, 3]
- ✓ {} – Object, Error

JavaScript Variables in JSX...

Good!

JSX on same line as return

```
function App() {  
  return <h1>Hi!</h1>  
}
```

Bad!

JSX on next line

```
function App() {  
  return;  
  <h1>Hi!</h1>;  
}
```

Good!

on same line as return

```
function App() {  
  return (  
    <h1>Hi!</h1>  
  );  
}
```

Bad!

on next line

```
function App() {  
  return  
  (<h1>Hi!</h1>);  
}
```

JSX Expression – Expressions are any code that results in a value

```
function App() {  
  const date = new Date();  
  const time = date.toLocaleDateString();  
  
  return <h1>{time}</h1>;  
}
```

```
function App() {  
  return <h1>{new Date().toLocaleDateString()}</h1>  
}
```

Good

```
function App() {  
  return <h1>{1 + 1}</h1>  
}
```

Bad

```
function App() {  
  return <h1>{const message = 'hi there'}</h1>  
}
```

What would be the output?

```
function App() {  
    const employee = {name: 'Md Ali'}  
  
    return <h1>{employee}</h1>  
}
```

```
12 function App() {  
13     const name = 'Md Ali'  
14  
15     return (  
16         <div>  
17             | My Name is:  
18             |     <h1>{name}</h1>  
19         </div>  
20     );  
21 }  
22  
23 export default App;  
24
```

```
12 function App() {  
13     const name = 'Md Ali'  
14     const age = 21;  
15  
16     return (  
17         <div>  
18             | My Name is: {name} and age is: {age}  
19         </div>  
20     );  
21 }  
22  
23 export default App;
```

Component Layout

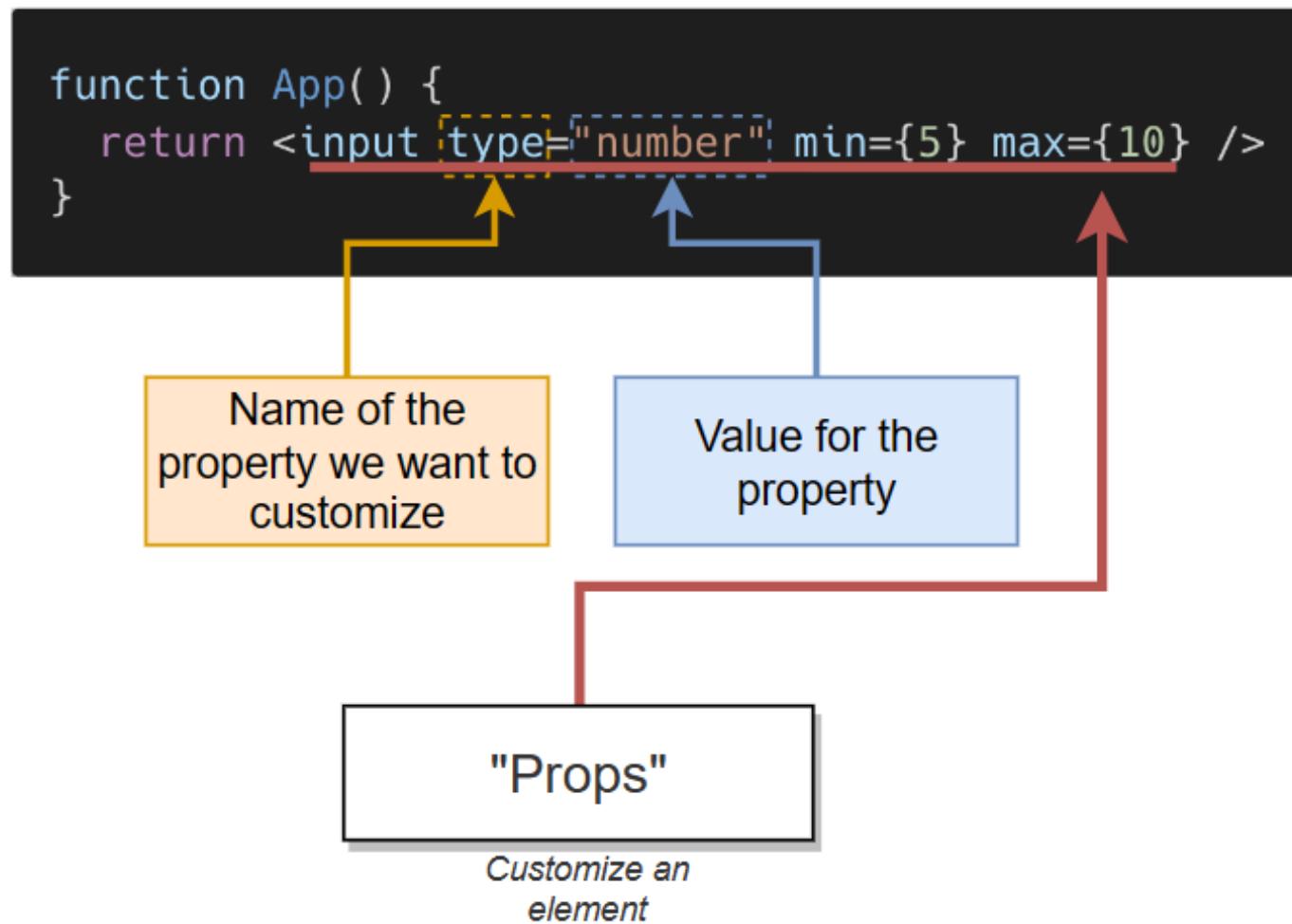
Code to compute values
we want to show in our
JSX

Content we want this
component to show

```
function App() {  
  const message = 'Hello';  
  const sum = 1 + 1;  
  
  return (  
    <div>  
      <div>Message is: {message}</div>  
      <div>Sum is: {sum}</div>  
    </div>  
  );  
}
```

Element with Props

- Define a HTML field for entering a number with minimum value is 5 and max value is 10
 - ✓ <input type="number" min="5" max="10">



More on Props

```
function App() {  
  const inputType = "number"  
  const minValue = 5;  
  
  return (  
    <input  
      type={inputType}  
      min={minValue}  
    />  
  );  
}
```

Props can refer to a variable using the same curly braces syntax

```
function App( ) {  
  return (  
    <input  
      type="number"  
      min={5}  
    />  
  );  
}
```

Props don't have to be defined as variables

Strings - Wrap with double quotes

Numbers - Wrap with curly braces

Invalid Props

- ✓ Props which are not provided in Curly braces OR with in Double quotes are invalid

```
function App() {  
  return (  
    <input  
      id=number-input  
      max=10  
    />  
  );  
}
```

Error! Should have double quotes

Error! Should have curly braces

Array, Object as Props

```
function App() {  
  const message = 'Enter age';  
  
  return (  
    <input  
      type="number"  
      min={5}  
      max={10}  
      list={[1,2,3]}  
      style={{ color: 'red' }}  
      alt={message}  
    />  
  );  
}
```

This is VALID

Arrays - Wrap with curly braces

Objects - Wrap with curly braces

Variables - Wrap with curly braces

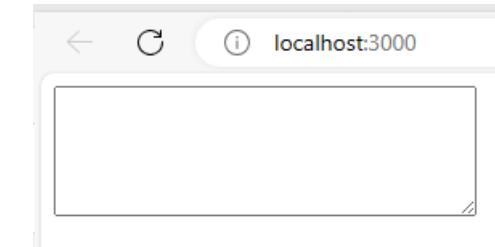
```
function App() {  
  const config = { color: 'red' }  
  return (  
    <div>  
      <h1>{config}</h1>  
      <input abc={config} />  
    </div>  
  );  
}
```

Trying to **display** an object. Doesn't work!

Trying to provide an object as a **prop**. OK!

Converting HTML to JSX

```
function App() {  
  return <textarea rows={5} cols={33} autofocus/>;  
}
```



Not Auto Focused

- ✓ Names/values of attributes that we provide to elements in HTML are slightly different when writing JSX

Rules for Converting HTML to JSX

1

All prop names follow camelCase

2

Number attributes use curly braces

3

Boolean 'true' can be written with just the
property name. 'False' should be written with
curly braces

4

The 'class' attribute is written as 'className'

5

In-line styles are provided as objects

Rules for Converting HTML to JSX...

Rule #1

In JSX, all prop names follow camelCase

| HTML | Equivalent JSX |
|-------------------------|------------------------------------|
| <input maxlength="5" /> | <input maxLength ={5} /> |
| <form autocapitalize /> | <form autoCapitalization /> |
| <form novalidate /> | <form noValidate /> |

Rules for Converting HTML to JSX...

Rule #2

In JSX, attributes meant to be numbers should be provided as numbers with curly braces

HTML

```
<input maxlength="5" />
```

Equivalent JSX

```
<input maxLength={5} />
```

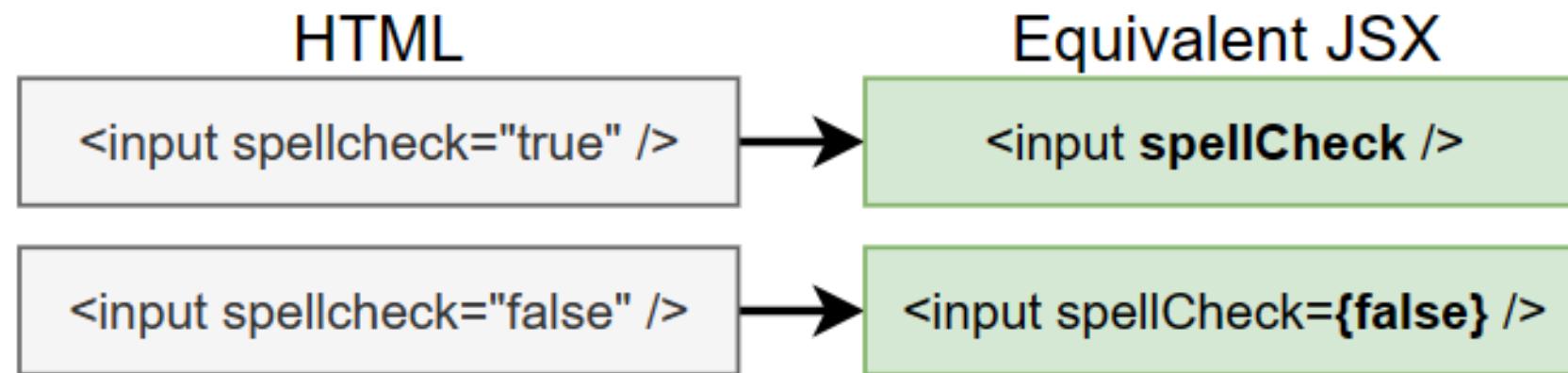
```
<meter optimum="50" />
```

```
<meter optimum={50} />
```

Rules for Converting HTML to JSX...

Rule #3

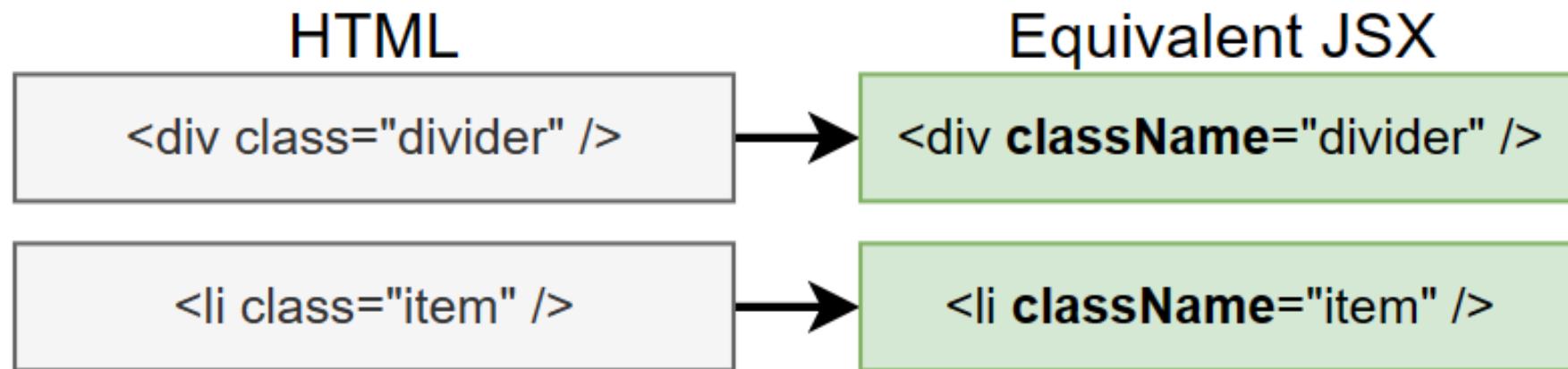
In JSX, boolean 'true' can be written with just the property name. 'False' should be written with curly braces



Rules for Converting HTML to JSX...

Rule #4

In JSX, the 'class' attribute is written as 'className'



Rules for Converting HTML to JSX...

Rule #5

In JSX, in-line styles are provided as objects

HTML

```
<a  
  style="text-decoration: 'none'; padding-top: '5px';"  
/>
```

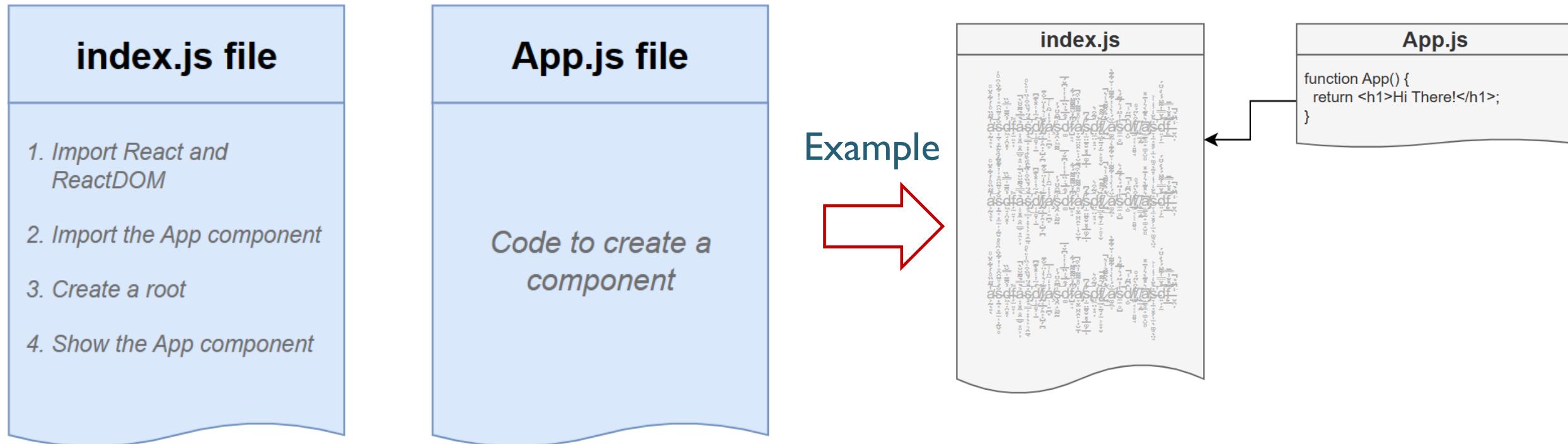
Equivalent JSX

```
<div  
  style={{ textDecoration: 'none', paddingTop: '5px' }}  
/>
```

```
function App() {  
  
    return (  
        <div className="wrapper">  
            <textarea  
                readOnly  
                maxLength={5}  
                cols={33}  
                spellCheck  
                autoFocus  
                style={{ backgroundColor: 'gray' }}  
            />  
        </div>  
    );  
}
```

Creating components in a separate file

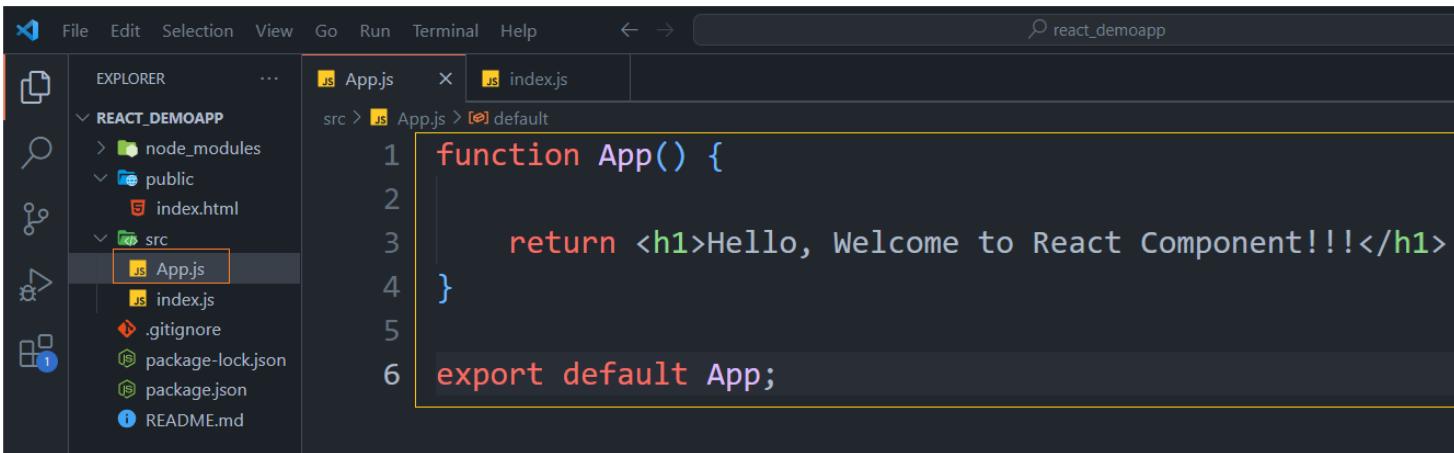
- ✓ The index.js file can get messy quickly when we add lots of components into it.
 - ✓ Create components in separate files for Easier to organize



Steps to Create a Components in a separate file

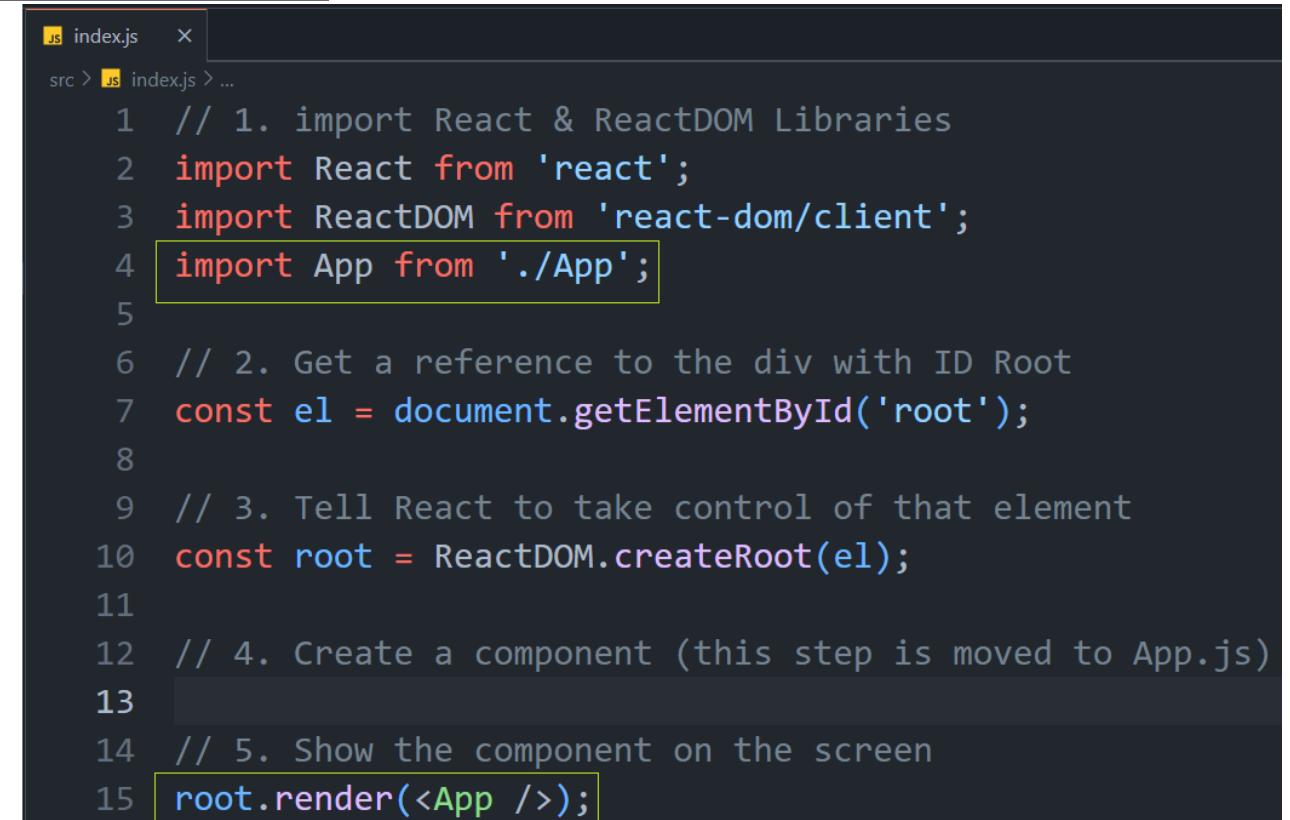
- 1** Create a new file.
(By convention) File should start with a capital letter → **JS App.js**
- 2** Make your component. Should be a function that returns JSX. → **function App() { return <hi>Hi</hi> }**
- 3** Export the component at the bottom of the file → **export default App;**
- 4** Import the component into another file → **import App from './App';**
- 5** Use the component → **<App />**

Steps to Create a Components in a separate file - IMPL



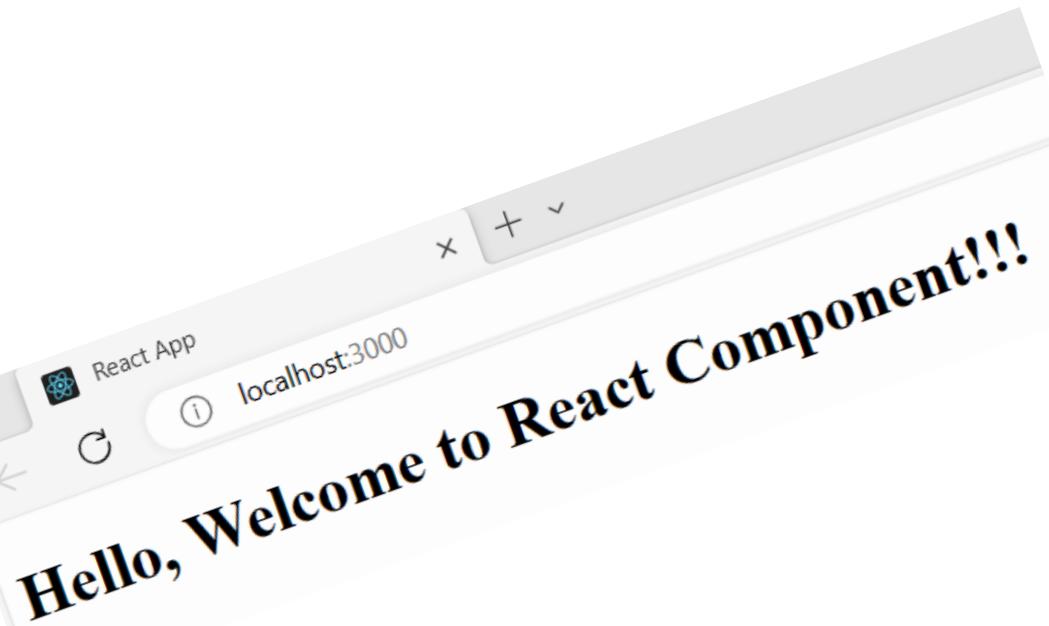
VS Code interface showing the file structure of a React application named "REACT_DEMOAPP". The "src" folder contains "App.js" and "index.js". The "App.js" file contains the following code:

```
function App() {  
  return <h1>Hello, Welcome to React Component!!!</h1>  
}  
  
export default App;
```



VS Code interface showing the "index.js" file content:

```
// 1. import React & ReactDOM Libraries  
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';  
  
// 2. Get a reference to the div with ID Root  
const el = document.getElementById('root');  
  
// 3. Tell React to take control of that element  
const root = ReactDOM.createRoot(el);  
  
// 4. Create a component (this step is moved to App.js)  
  
// 5. Show the component on the screen  
root.render(<App />);
```

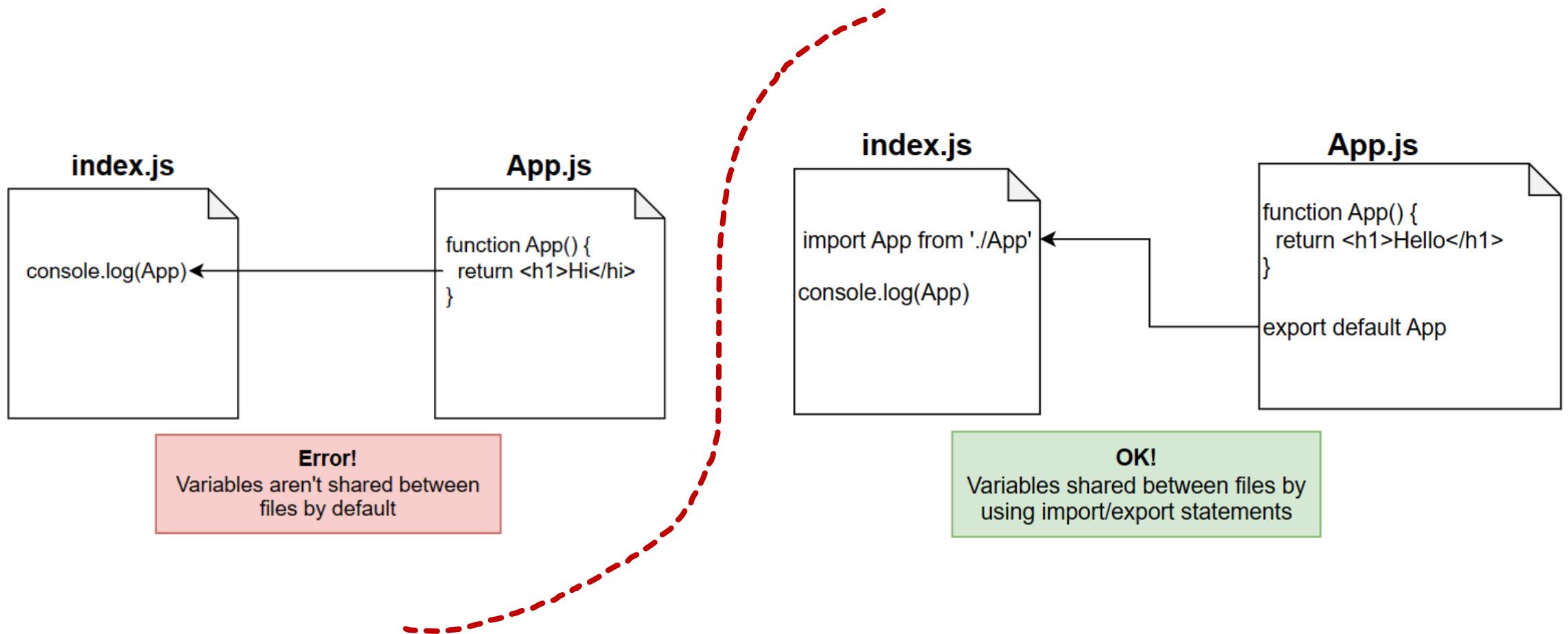


Module System (OR)

Import/Exports Statements

Import/Exports Statements

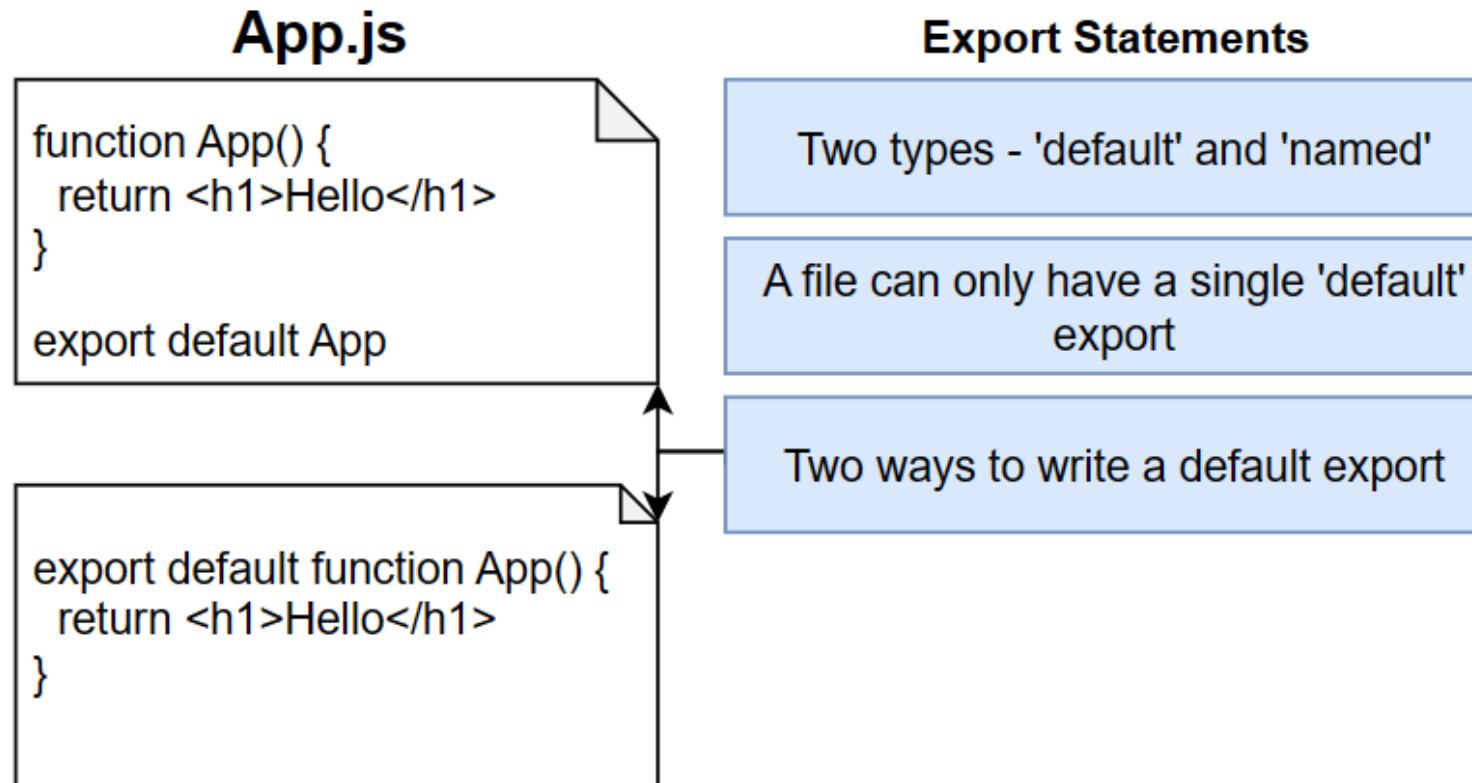
- We can not share the variables OR Functions from one file to another file. If we do so we get a error
- We can share the variables or function in different files using the module system.



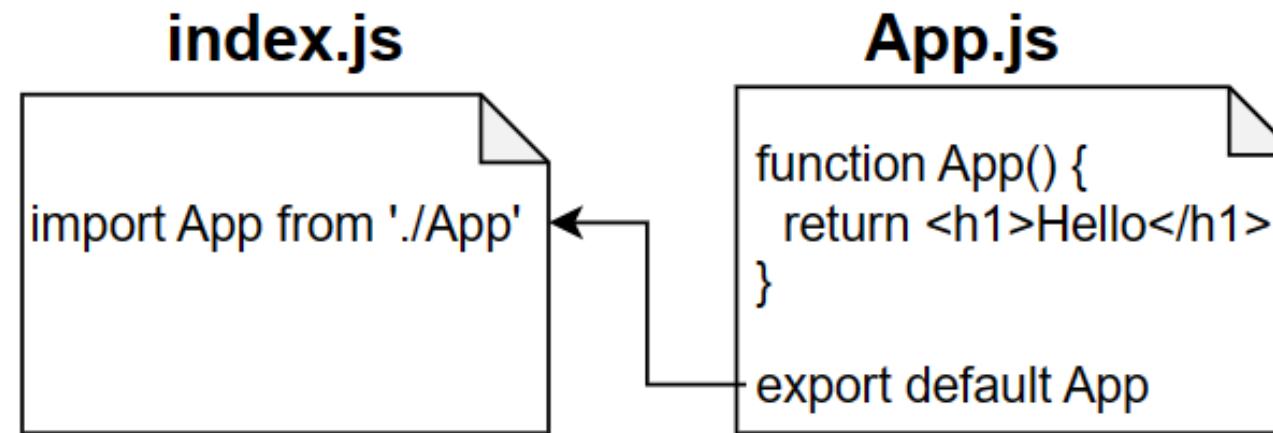
Rules - Exports Statements

- Two kinds of Export statements available

- ✓ default
- ✓ named



Module System - Import Statements



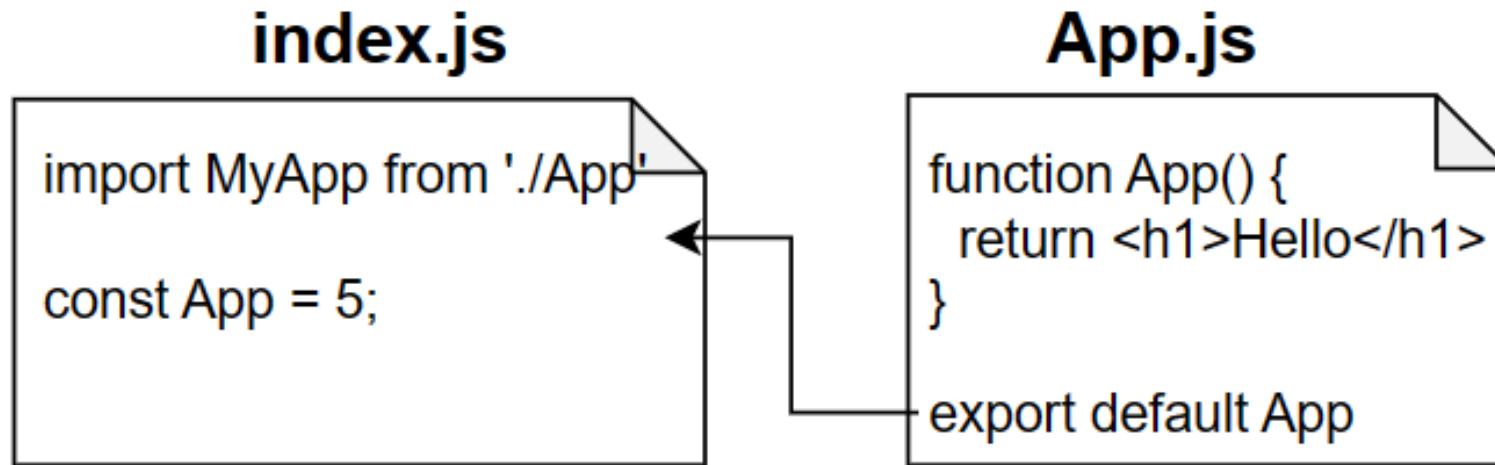
Import Statements - Behind the Scenes

1 Declare a variable called App

2 Find the default export from App.js

3 Assign the default export to App variable

Default export can be Re-Named



Confusing Thing!
Default exports can be renamed in the *importing* file!!!

- 1 Declare a variable called `MyApp`
- 2 Find the default export from `App.js`
- 3 Assign the default export to `MyApp` variable

NOTE: This case is valid scenario where when there are two functions from two files and function names are same then there might be conflict to avoid conflict we can re-name the variable or function names.

Named exports Statement

App.js

```
function App() {  
  return <h1>Hello</h1>  
}  
export default App  
  
const message = 'hi'  
export { message }
```

```
export default function App() {  
  return <h1>Hello</h1>  
}  
  
export const message = 'hi'
```

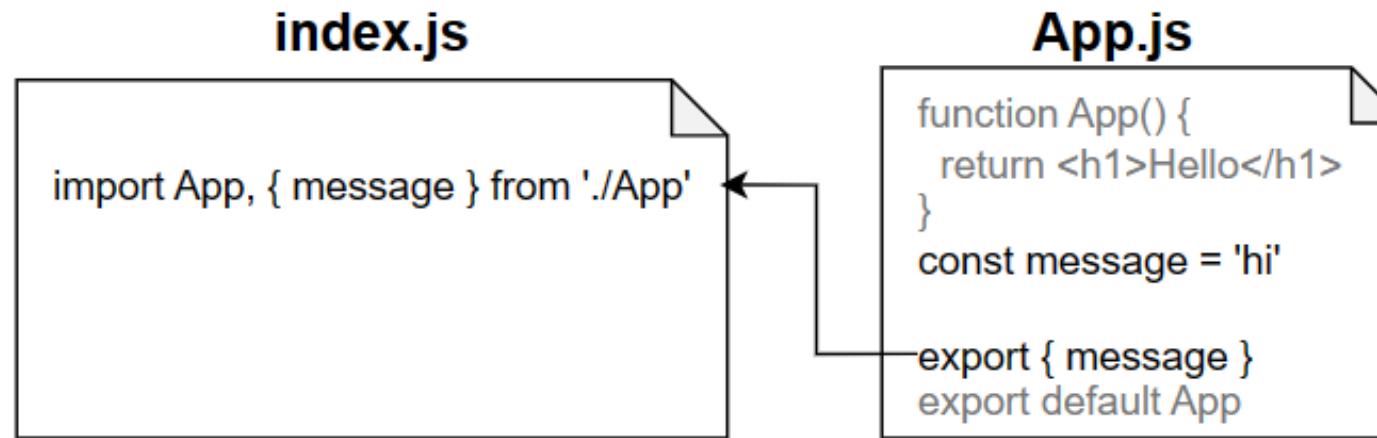
Named Export Statements

Use when exporting multiple variables

Can have as many named exports as we want

Two ways to write a named export

Importing - Named exports



Import Statements

Curly braces indicate we want a 'named' export

Single import statement can get both default +
named exports

Named exports cannot be renamed!!!

```
import App from './App'
```

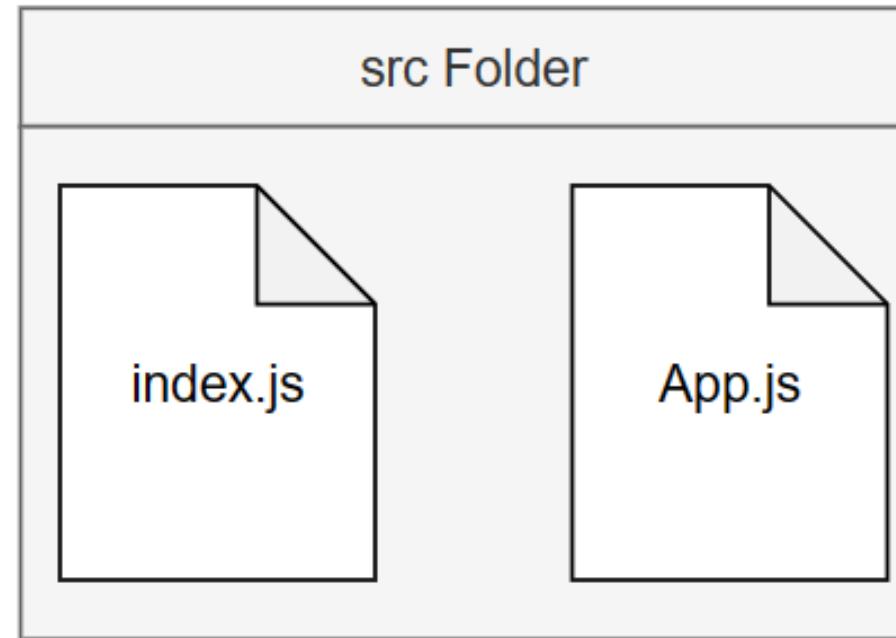
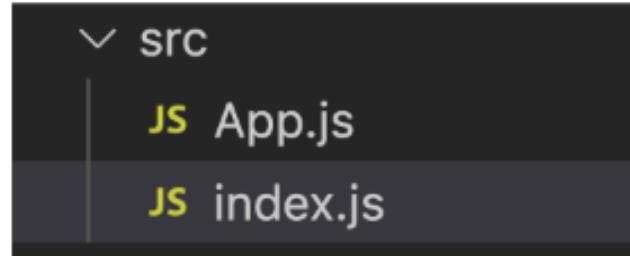
```
import React from 'react'
```

```
import ReactDOM from 'react-dom/client'
```

'./' or '../' means we are *importing a file that we created*

No './' or '../' means we are *importing a package*

Import Demo - I

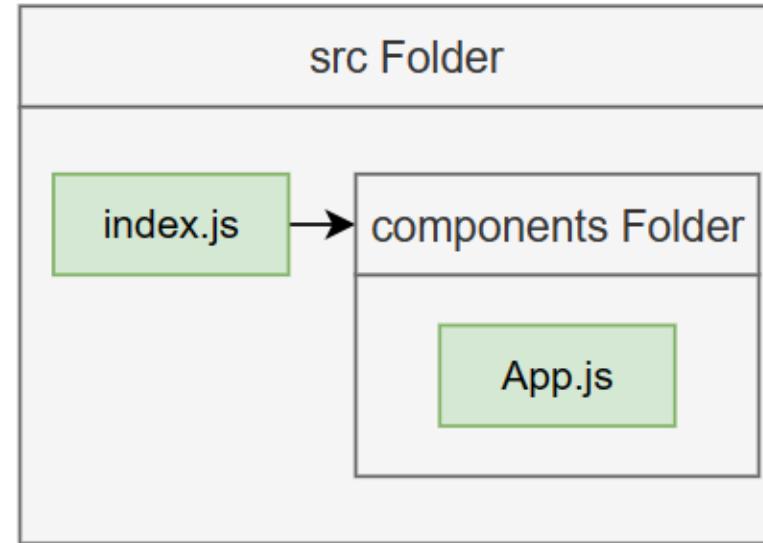
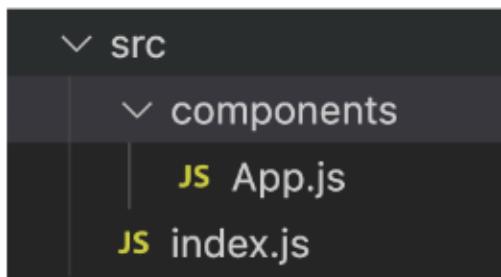


If we are in the index.js file and want to import from App.js....

```
import App from './App';
```

NOTE: ./ means same folder/directory

Import Demo - 2

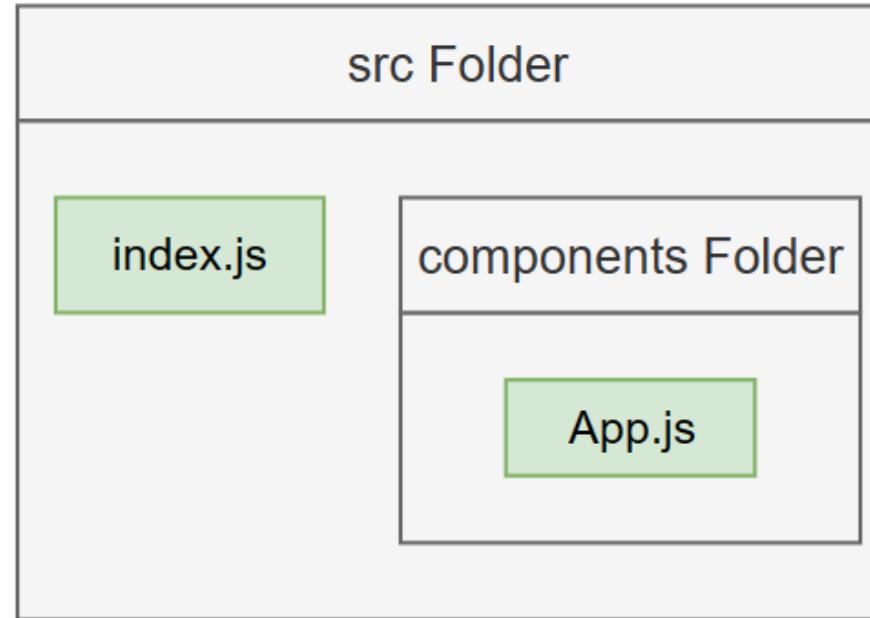


If we are in the index.js file and want to import from App.js....

import App from './components/App';

Import index.js in App.js

```
src
  components
    App.js
    index.js
```



../ means Up one directory

If we are in the App.js file and want to import from index.js....

import index from ‘.../index’

`./`

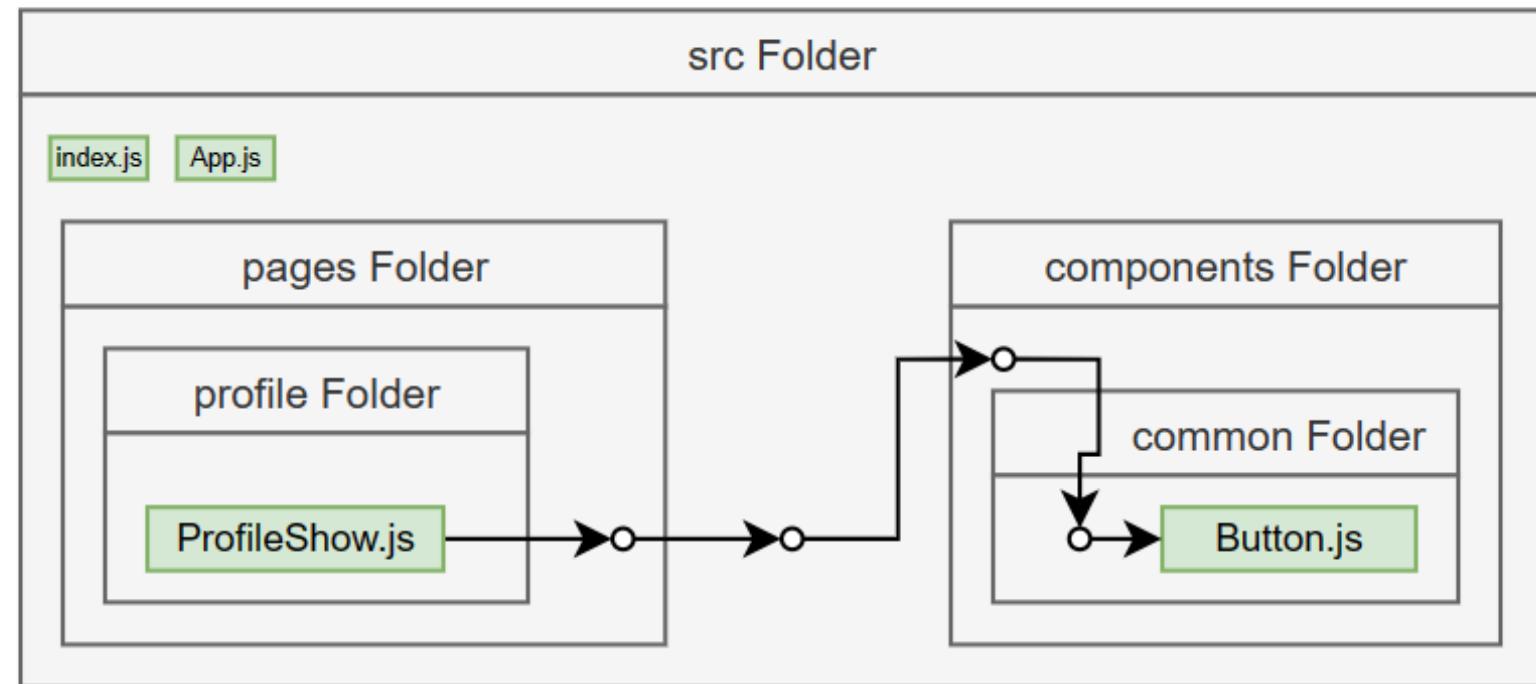
Same directory

`../`

Up one directory

Import Demo - 5

```
src
  components
    common
      Button.js
    pages
      profile
        ProfileShow.js
      App.js
      index.js
```



If we are *in the ProfileShow.js file* and want to import from *Button.js*....

import asdf from '../..//components/common/Button'

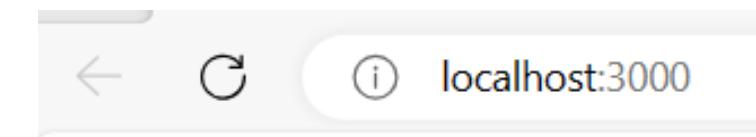
Import Demo - 6

```
1 function App() {  
2  
3     return <h1>Hello, Welcome to React Component!!!</h1>  
4 }  
5  
6 const username = 'mdali';  
7 const email = 'mdali@gmail.com';  
8  
9 export { username, email };  
10  
11 export default App;
```

```
1 // 1. import React & ReactDOM Libraries  
2 import React from 'react';  
3 import ReactDOM from 'react-dom/client';  
4 import App from './App';  
5 import {username, email} from './App';  
6
```

Import Demo - 7

```
1 const App = () => {
2     const count = 10;
3     const name = 'Md Ali';
4     const booleanTrue = true;
5     const booleanFalse = false;
6     const nullValue = null;
7     const undefinedValue = undefined;
8
9     return (
10         <ul>
11             <li>Number: {count}</li>
12             <li>String: {name}</li>
13             <li>True: {booleanTrue}</li>
14             <li>False: {booleanFalse}</li>
15             <li>Null: {nullValue}</li>
16             <li>Undefined: {undefinedValue}</li>
17         </ul>
18     );
19 };
20
21 export default App;
```



- Number: 10
- String: Md Ali
- True:
- False:
- Null:
- Undefined:

Vite

Creating React Application

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\React_7AM_WS\testapp> cd ..  
PS C:\React_7AM_WS> npm create vite@latest  
Need to install the following packages:  
  create-vite@5.2.3  
Ok to proceed? (y)y  
✓ Project name: ... first-vite-proj  
✓ Select a framework: » React  
✓ Select a variant: » JavaScript
```

Scaffolding project in C:\React_7AM_WS\first-vite-proj...

Done. Now run:

```
cd first-vite-proj  
npm install  
npm run dev
```

```
PS C:\React_7AM_WS> 
```

```
PS C:\React_7AM_WS> cd first-vite-proj  
PS C:\React_7AM_WS\first-vite-proj> npm install
```

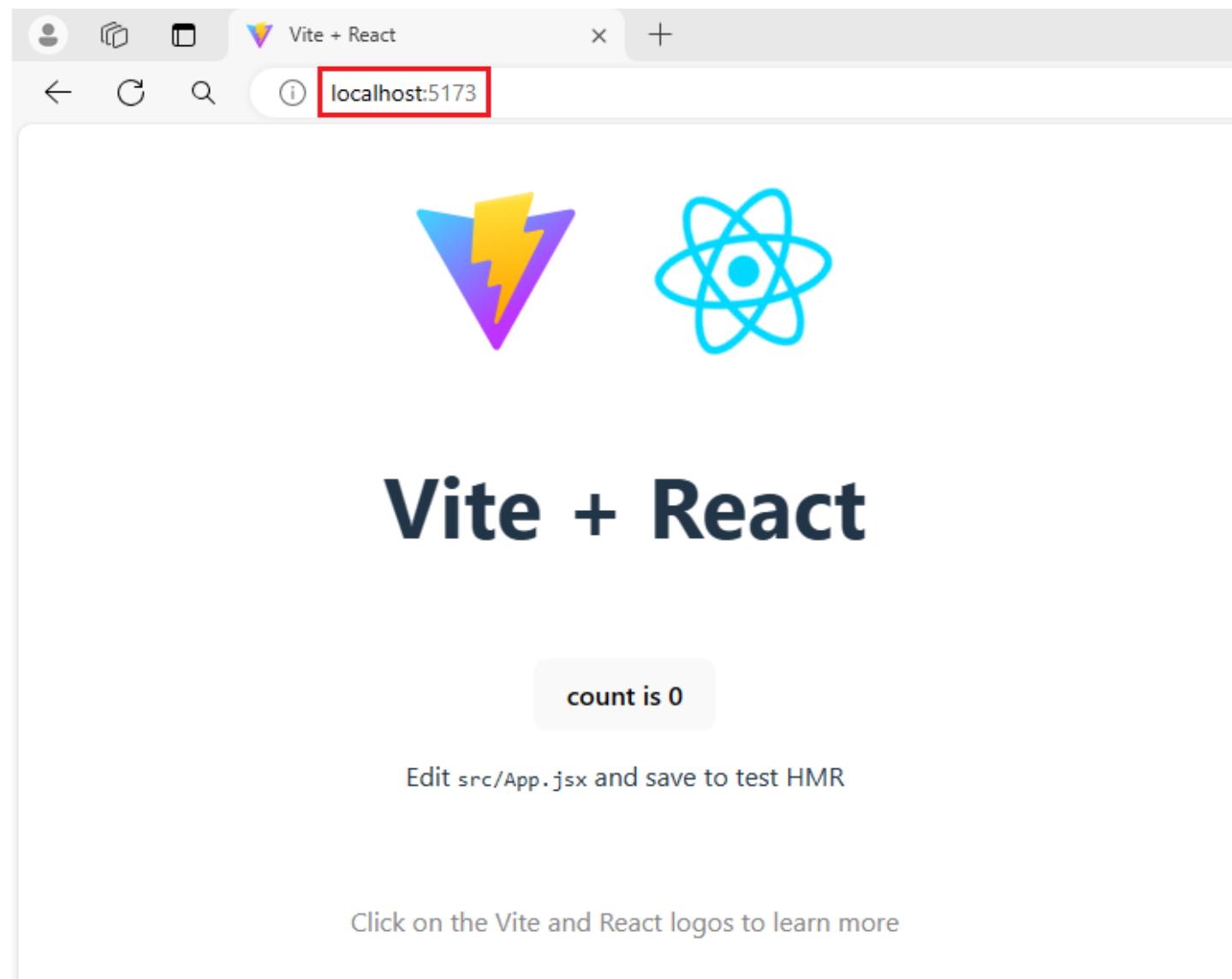
```
added 278 packages, and audited 279 packages in 32s  
103 packages are looking for funding  
  run `npm fund` for details  
found 0 vulnerabilities  
PS C:\React_7AM_WS\first-vite-proj> 
```

npm run dev

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

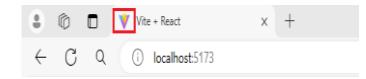
```
VITE v5.2.8 ready in 215 ms
```

```
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```



Understanding Vite-React Application Templated

- **node_module** - is the folder where all our installed packages and library present
- **public** - Contains static assets which will be directly served to the client
- **src**
 - **Assets** - we put all our images, icon, fonts used in React Components
 - **App.css** - Used for styling used in App.jsx
 - **App.jsx** - Root component of our application means App.jsx is the starting point of Component Chain. We will call it as “root component”. It is similar to JS
 - **index.css** – Styling used in main.jsx file.
 - **main.jsx** - Most important file of React. main.jsx is the file which connects our components with index.html file which is our main html file
- **index.html** – This is main html file which runs on the browser. div with id root is the main tag in which all our react components will display on browser
- **.gitignore** - We specify which files or folders not uploaded on GIT
- **package.json** - This file contains all information about our application like name, version, package versions etc



Basic Concepts Of React

- Setup new React Project Using Vite
- Building Components
- JSX & Babel
- Adding Elements
- Adding JSX Expression
- Attributes
- Events
- State
- useState Hook
- Handling Inputs
- Mapping Lists
- Implement one Project using above Concepts

Create React Project Using Vite

A screenshot of a terminal window within a code editor interface. The terminal tab is highlighted with a red border. The command `npm create vite@latest` is entered, followed by a series of configuration steps: Project name (react-vite-proj), framework (React), and variant (JavaScript). The process then moves on to scaffolding the project in the specified directory. Finally, instructions are given to run the project using `npm run dev`.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\React_7AM_WS> npm create vite@latest
✓ Project name: ... react-vite-proj
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in C:\React_7AM_WS\react-vite-proj...

Done. Now run:

cd react-vite-proj
npm install
npm run dev

PS C:\React_7AM_WS>
```

A screenshot of a terminal window showing the execution of `npm install` and `npm run dev`. The output shows the installation of 278 packages, the detection of 103 packages looking for funding, and 0 vulnerabilities found. After running `npm run dev`, the Vite development server starts at `http://localhost:5173/`.

```
PS C:\React_7AM_WS> cd react-vite-proj
PS C:\React_7AM_WS\react-vite-proj> npm install
added 278 packages, and audited 279 packages in 20s
103 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\React_7AM_WS\react-vite-proj> npm run dev

> react-vite-proj@0.0.0 dev
> vite

VITE v5.2.8 ready in 203 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

Building First React Component

- Component is a reusable piece of code that is used to define certain part of user interface using JS & JSX
- Create a folder called “components” inside “src” folder
- Inside “components” create a jsx file called MyComponent.jsx
- React components are two types Functional Components(Easy) & Class Components(Complex)
- We can export the component in two ways...default export, named export
- Import component in main.jsx
- `{/* */}` – Comments in JSX

Add React Snippet extension

File Edit Selection View Go Run Terminal Help ← → react_workspace_8am ⌂ ⌃ ⌄ ⌅

EXTENSIONS: MARKETPLACE

- react sn**
- Simple React Snippets** 3.4M ★ 5
Dead simple React snippets you will love.
Burke Holland [Install](#)
- ES7+ React/Redux/...** 10M ★ 4.5
Extensions for React, React-Native ...
dsznajder [Install](#)
- React-Native/React/...** 1.3M ★ 5
Code snippets for React-Native/React...
EQuiper [Install](#)
- ES7 React/Redux/Gra...** 1M ★ 4
Simple extensions for React, Redux...
rodrigovallades [Install](#)
- TypeScript React co...** 584K ★ 4.5
Code snippets for react in typescript
infeng [Install](#)
- React Native Snippet** 511K ★ 5
React Native, Typescript React Native...
jundat95 [Install](#)
- React/Redux/react-r...** 223K ★ 5
React Ecosystem code snippets all ...
discoutry [Install](#)
- Fullstack React/Rea...** 219K ★ 4.5
Code snippets for React/React Native...
Walter Ribeiro [Install](#)

Extension: ES7 React/Redux/GraphQL/React-Native snippets

 ES7 React/Redux/GraphQL/React-Native snippets v1.9.3

rodrigovallades | 1,056,502 | ★★★★☆(5)

Simple extensions for React, Redux and GraphQL in JS/TS with ES7 syntax (forked from dsznajder)

[Install](#) [Settings](#)

DETAILS FEATURE CONTRIBUTIONS CHANGELOG

VS Code ES7 React/Redux/React-Native/JS snippets

Version

This extension provides you JavaScript and React/Redux snippets in ES7 with Babel plugin features for VS Code

Here is direct link to marketplace [ES7 React/Redux/React-Native/JS Snippets](#)

Supported languages (file extensions)

- JavaScript (.js)
- JavaScript React (.jsx)
- TypeScript (.ts)
- TypeScript React (.tsx)

Snippets info

Every space inside { } and () means that this is pushed into next line :) \$ represent each step after tab.

Categories: [Snippets](#)

Extension Resources:

- Marketplace
- Repository
- rodrigovallades

More Info:

- Published: 2018-12-14, 01:35:24
- Last released: 2018-12-14, 01:35:24
- Identifier: rodrigovallades.es7-react-js-snippets

ES7 React/Redux snippets only functional component and hooks

The screenshot shows the VS Code Marketplace interface. On the left, there's a sidebar with icons for file operations like Open, Save, Find, and Run. The main area has a search bar at the top with the text "react_workspace_8am". Below it, a list of extensions is displayed:

- react hooks snippets** (selected): "Simple extensions for React, Redux in JS/TS with ES7 syntax" by p1e7r0. It has 9,649 installs. A green "Install" button is visible.
- react hooks/js**: "非常实用的代码片段" (Very useful code snippets) by ccl. It has 46 installs.
- Code snippets - React / React Native / React Hooks / Styling**: "A group of code snippets to improve your coding speed" by Eduardo Liberato Sotero. It has 3K installs.
- Mobx/Rematch React Snippets and taro react hooks**: "a snippets at mobx react, rematch react, taro react hooks" by qingchong. It has 2K installs.
- TypeScript snippets - React / React Native / React Hooks...**: "TypeScript snippets for React, React Native, and React Hooks" by jv[. It has 444 installs.

On the right, a detailed view of the selected extension is shown:

Extension: ES7 React/Redux snippets only functional component and hooks

ES7 React/Redux snippets only functional component and hooks by p1e7r0 | 9,649 | ★★★★☆

Simple extensions for React, Redux in JS/TS with ES7 syntax

Disable | **Uninstall** | ⚙

This extension is enabled globally.

DETAILS FEATURE CONTRIBUTIONS CHANGELOG RUNTIME STATUS

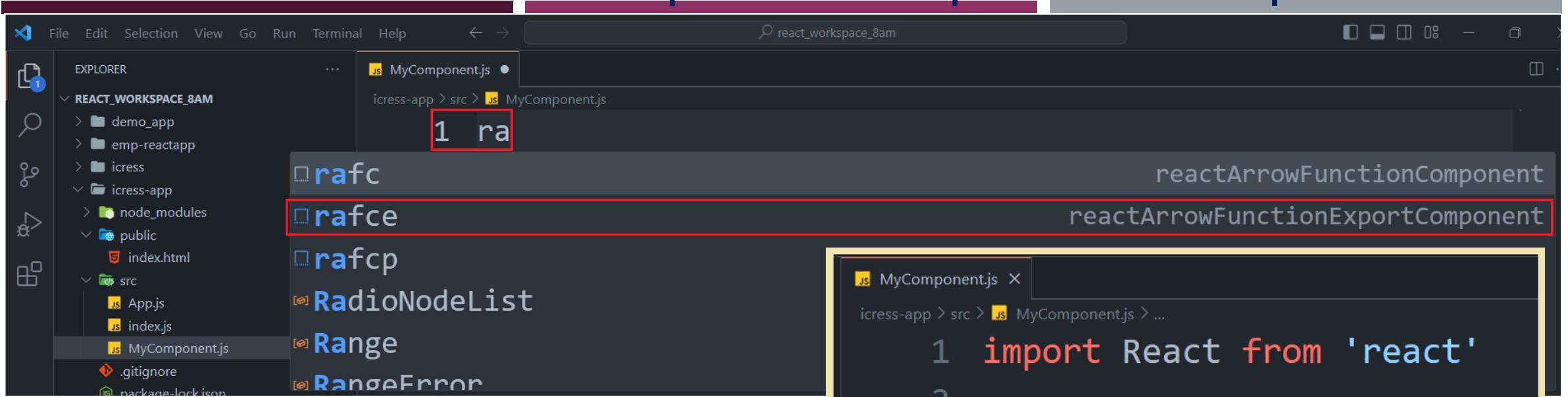
VS Code ES7 React/Redux/React-Native/JS snippets

Version Install Downloads Ratings

Categories: Snippets

Extension Resources: Marketplace

Create Arrow Function Component & Export the Component



The screenshot shows the VS Code interface with the following details:

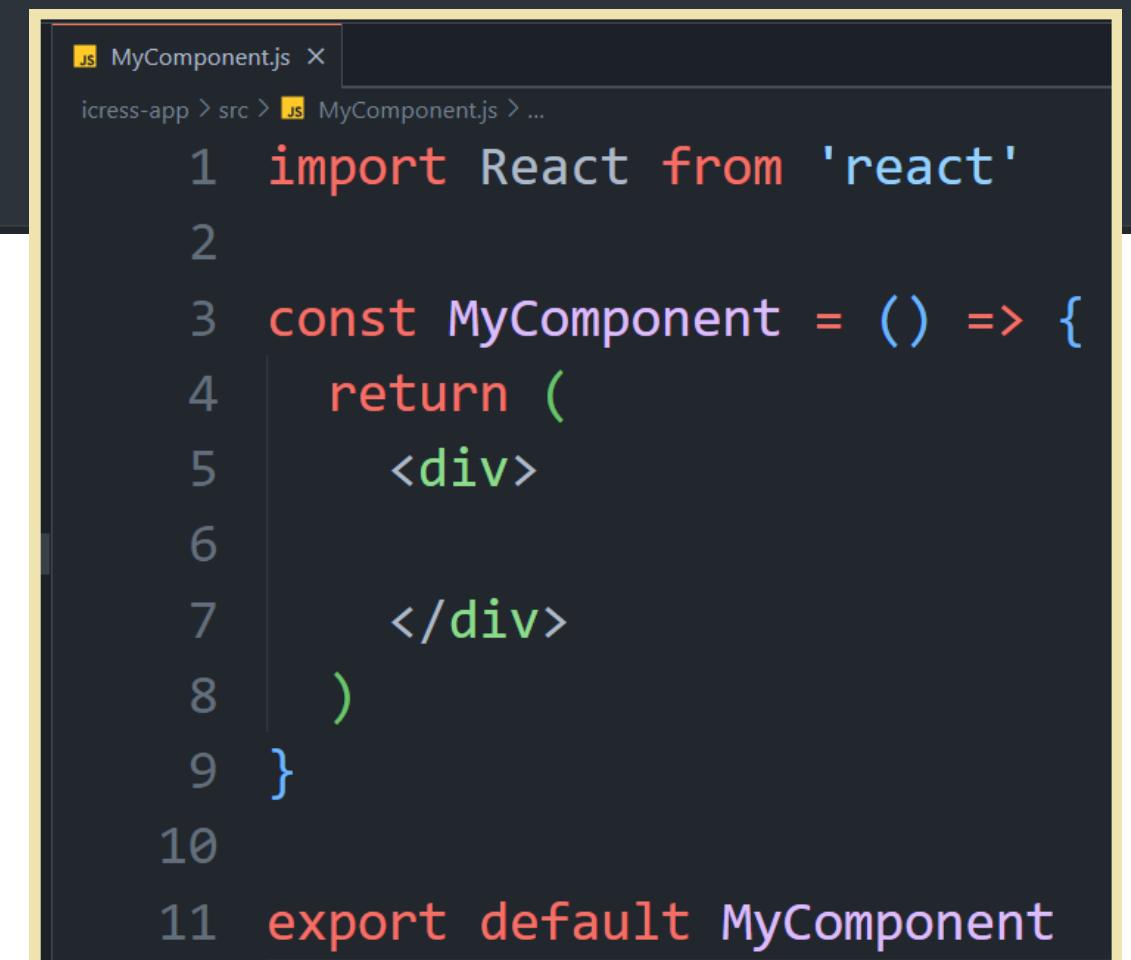
- File Explorer:** Shows a project structure under "REACT_WORKSPACE_8AM".
- Search Bar:** Displays "react_workspace_8am".
- Code Editor:** Shows the file "MyComponent.js".
- Completion Suggestion:** A dropdown menu is open, showing suggestions starting with "ra": "rafce", "rafcp", "RadioNodeList", "Range", and "RangeError". The suggestion "rafce" is highlighted with a red border.

✓ **NOTE** – This is a stateless component.

To use the state in Functional Component we need to use useState from react hook

✓ useState is a Hook(function) that allows us to have state variables in functional components

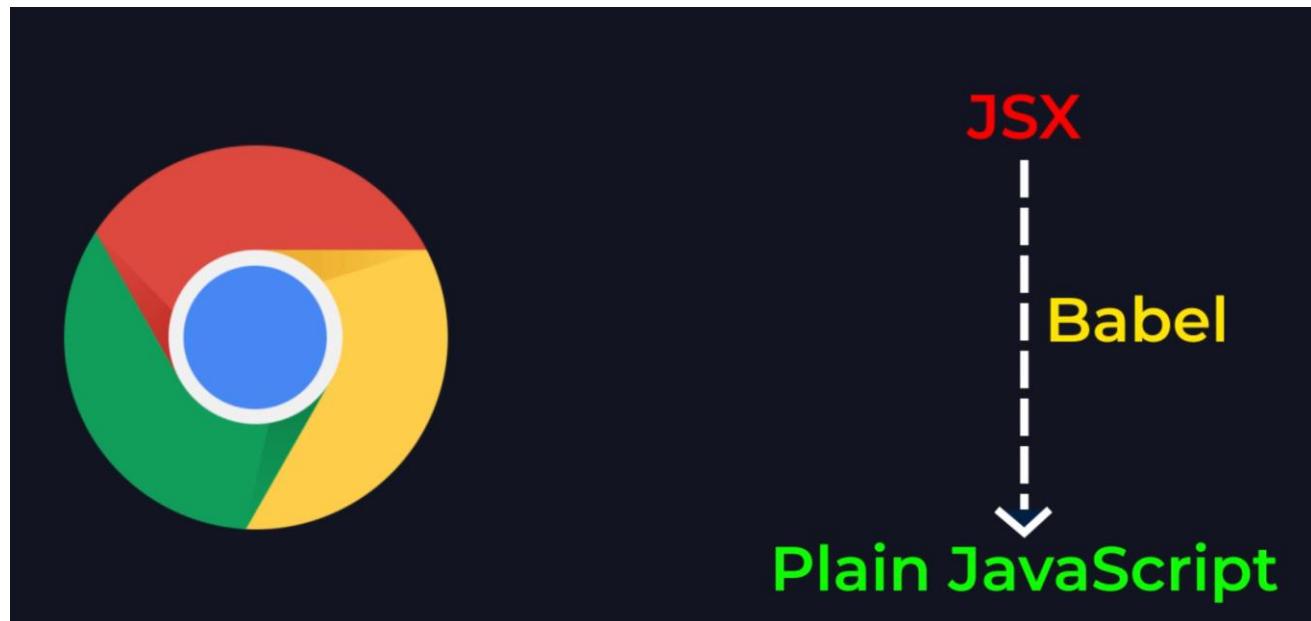
```
const [state, setstate] = useState(initialState)
```



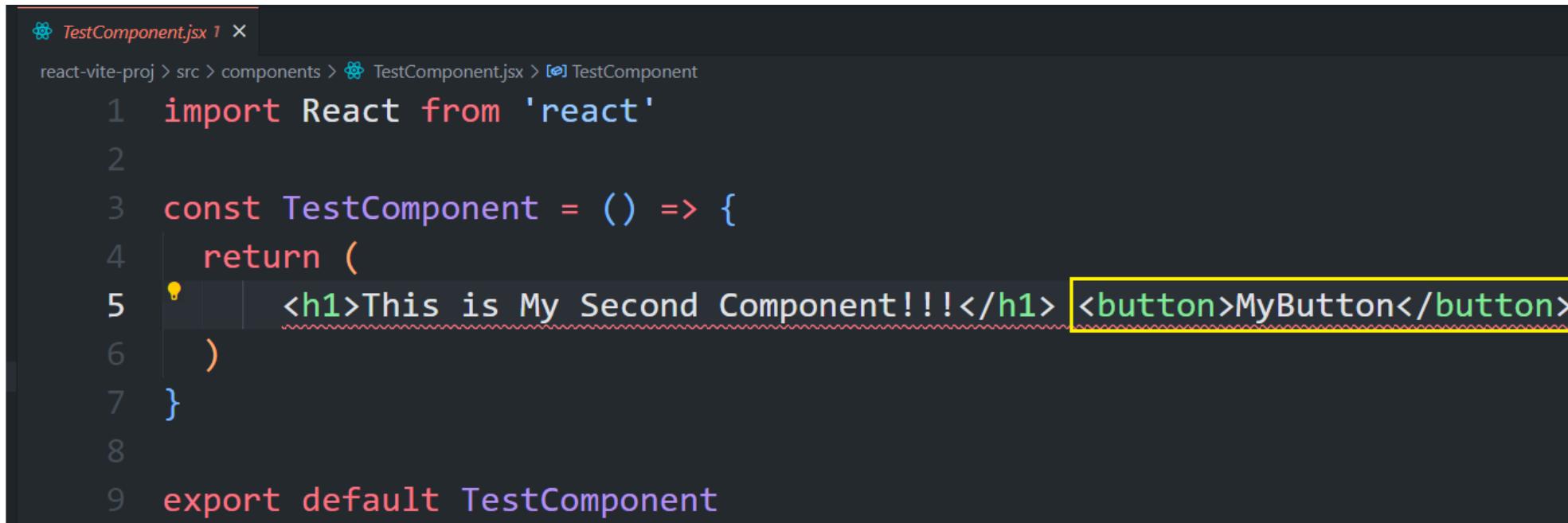
```
1 import React from 'react'
2
3 const MyComponent = () => {
4     return (
5         <div>
6
7             </div>
8     )
9 }
10
11 export default MyComponent
```

JSX vs Babel

- JSX(JavaScript XML) is a modern code for writing HTML and JavaScript together
- Babel is used to compile JSX code into Plain Vennela JavaScript code.



Adding Elements to our Component



```
TestComponent.jsx 1 ×
react-vite-proj > src > components > TestComponent.jsx > TestComponent
1 import React from 'react'
2
3 const TestComponent = () => {
4   return (
5     <h1>This is My Second Component!!!</h1> <button>MyButton</button>
6   )
7 }
8
9 export default TestComponent
```

- **NOTE:** In JSX we add elements under one parent element

```
TestComponent.jsx X
react-vite-proj > src > components > TestComponent.jsx > default
1 import React from 'react'
2
3 const TestComponent = () => {
4   return (
5     <div>
6       <h1>This is My Second Component!!!</h1> <button>MyButton</button>
7     </div>
8   );
9 }
10
11 export default TestComponent
```

```
❖ TestComponent.jsx •  
react-vite-proj > src > components > ❖ TestComponent.jsx > TestComponent  
1 import React from 'react'  
2  
3 const TestComponent = () => {  
4   return (  
5     <React.Fragment>  
6       <h1>This is My Second Component!!!</h1> <button>MyButton</button>  
7     </React.Fragment>  
8   );  
9 };  
10  
11 export default TestComponent
```

TestComponent.jsx x

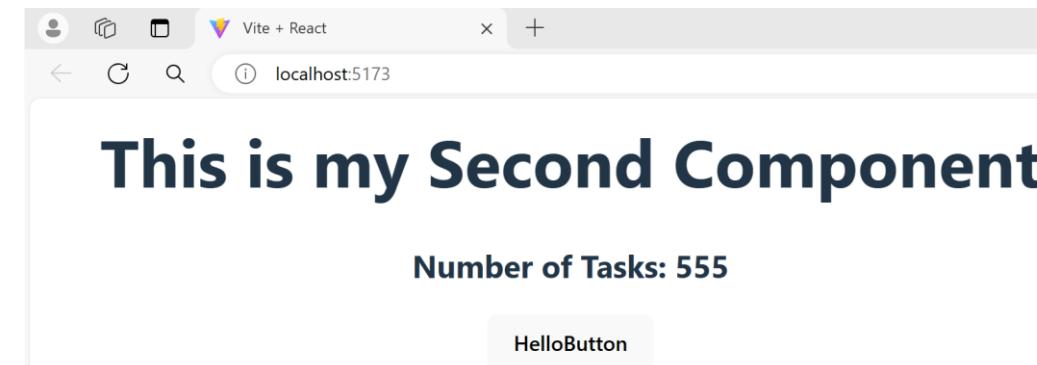
react-vite-proj > src > components > TestComponent.jsx > ...

```
1 import React from 'react'
2
3 const TestComponent = () => {
4     return (
5         <>
6             <h1>This is My Second Component!!!</h1> <button>MyButton</button>
7         </>
8     );
9 }
10
11 export default TestComponent
```

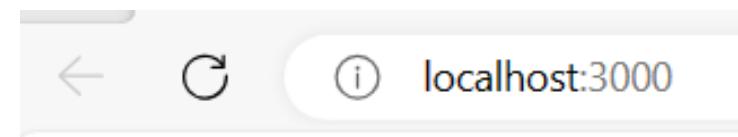
JavaScript expression in JSX

```
TestComponent.jsx ×  
react-vite-proj2 > src > components > TestComponent.jsx > TestComponent  
1 import React from 'react'  
2  
3 const TestComponent = () => {  
4   const tasks = 555;  
5   return (  
6     <>  
7       <h1>This is my Second Component</h1>  
8       <h2>Number of Tasks: {tasks}</h2>  
9       <button>HelloButton</button>  
10    </>  
11  );  
12};  
13  
14 export default TestComponent
```

```
main.jsx ×  
react-vite-proj2 > src > main.jsx  
1 import React from 'react'  
2 import ReactDOM from 'react-dom/client'  
3 import App from './App.jsx'  
4 import './index.css'  
5 import MyComponent from './components/MyComponent.jsx'  
6 import TestComponent from './components/TestComponent.jsx'  
7  
8  
9 ReactDOM.createRoot(document.getElementById('root')).render(  
10   <React.StrictMode>  
11     {/*<App />  
12      <MyComponent />*/*}  
13      <TestComponent />  
14    </React.StrictMode>,  
15 )
```



```
1 const App = () => {
2     const count = 10;
3     const name = 'Md Ali';
4     const booleanTrue = true;
5     const booleanFalse = false;
6     const nullValue = null;
7     const undefinedValue = undefined;
8
9     return (
10        <ul>
11            <li>Number: {count}</li>
12            <li>String: {name}</li>
13            <li>True: {booleanTrue}</li>
14            <li>False: {booleanFalse}</li>
15            <li>Null: {nullValue}</li>
16            <li>Undefined: {undefinedValue}</li>
17        </ul>
18    );
19};
20
21 export default App;
```



- Number: 10
- String: Md Ali
- True:
- False:
- Null:
- Undefined:

Calling function from JSX Expression

TestComponent.jsx X

react-vite-proj2 > src > components > TestComponent.jsx > [o] TestComponent

```
1 import React from 'react'
2
3 const TestComponent = () => {
4   const tasks = 555;
5
6   const countTasks = () => {
7     return tasks + 45;
8   }
9
10  return (
11    <>
12      <h1>This is my Second Component</h1>
13      <h2>Number of Tasks: {tasks}</h2>
14      <h2>countTasks Fun called: {countTasks()}</h2>
15      <button>HelloButton</button>
16    </>
17  );
18};
19
20 export default TestComponent
```

This is my Second Component

Number of Tasks: 555

countTasks Fun called: 600

HelloButton

Calling function from JSX Expression...

```
❖ TestComponent.jsx ×
react-vite-proj2 > src > components > ❖ TestComponent.jsx > [●] TestComponent > [●] countTasks
1 import React from 'react'
2
3 const TestComponent = () => {
4   const tasks = 555;
5
6   const countTasks = () => {
7     if(tasks === 0) {
8       return "Sorry, No tasks available!!!";
9     } else {
10       return "Number of Tasks = " + tasks;
11     }
12   }
13
14   return (
15     <>
16       <h1>This is my Second Component</h1>
17       <h2>Number of Tasks: {tasks}</h2>
18       <h2>countTasks Fun called: {countTasks()}</h2>
19       <button>HelloButton</button>
20     </>
21   );
22 };
23
24 export default TestComponent
```

Calling function from JSX Expression...

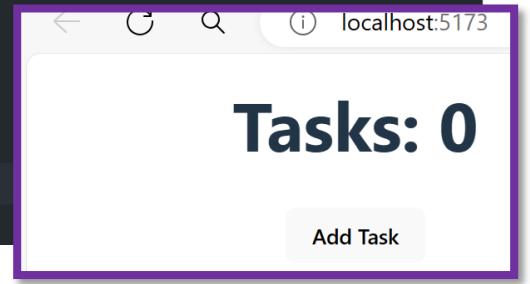
TestComponent.jsx X

```
react-vite-proj2 > src > components > TestComponent.jsx > ...
1 import React from 'react'
2
3 const TestComponent = () => {
4     const tasks = 555;
5
6     const countTasks = () => {
7         return tasks === 0? "Sorry, No tasks available!!!": `Number of Tasks = ${tasks}`;
8     }
9
10    return (
11        <>
12            <h1>This is my Second Component</h1>
13            <h2>Number of Tasks: {tasks}</h2>
14            <h2>countTasks Fun called: {countTasks()}</h2>
15            <button>HelloButton</button>
16        </>
17    );
18};
19
20 export default TestComponent
```

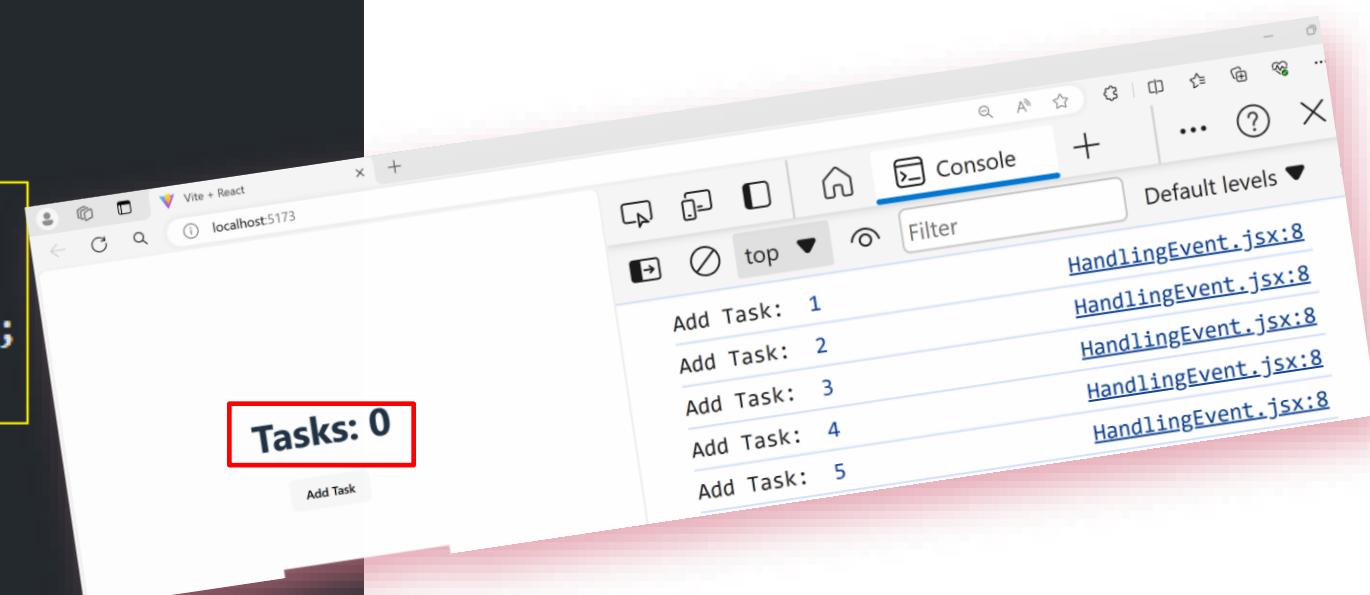
Events In React

Handling events in React is similar to handling events in HTML

```
HandlingEvent.jsx ×  
react-vite-proj2 > src > components > HandlingEvent.jsx > ...  
1 import React from 'react'  
2  
3 const HandlingEvent = () => {  
4     const tasks = 0;  
5  
6     const handleClick = () => {  
7         console.log("Add Task");  
8     }  
9  
10    return (  
11        <div>  
12            <h1>Tasks: {tasks}</h1>  
13            <button onClick={handleClick}>Add Task</button>  
14        </div>  
15    );  
16}  
17  
18 export default HandlingEvent
```

```
main.jsx ×  
react-vite-proj2 > src > main.jsx  
1 import React from 'react'  
2 import ReactDOM from 'react-dom/client'  
3 import App from './App.jsx'  
4 import './index.css'  
5 import MyComponent from './components/MyComponent.jsx'  
6 import TestComponent from './components/TestComponent.jsx'  
7 import HandlingEvent from './components/HandlingEvent.jsx'  
8  
9 ReactDOM.createRoot(document.getElementById('root')).render(  
10    <React.StrictMode>  
11        {/*<App />  
12        <MyComponent />  
13        <TestComponent />}*/}  
14        <HandlingEvent />  
15    </React.StrictMode>,  
16)  
  

```

```
HandlingEvent.jsx ×  
react-vite-proj2 > src > components > HandlingEvent.jsx > [o] default  
1 import React from 'react'  
2  
3 const HandlingEvent = () => {  
4     let tasks = 0;  
5  
6     const handleClick = () => {  
7         tasks = tasks + 1;  
8         console.log("Add Task: ", tasks);  
9     }  
10  
11     return (  
12         <div>  
13             <h1>Tasks: {tasks}</h1>  
14             <button onClick={handleClick}>Add Task</button>  
15         </div>  
16     );  
17 };  
18  
19 export default HandlingEvent
```



NOTE: Value of tasks is not reflecting on Web Page. i.e., Value is not updating on DOM.

What is a State

- State allows us to manage and display the changing data in our application
- When we define a Variables through JavaScript then React will not reflect the changes done to that Normal Variable. To solve this problem we have to define this variable as a State Variable
- State is used to tell React... Watch the variable and if any changes done to that variable then reflect the changes on DOM
- How to define a variable as a state variable in functional component... we use React Hooks

What are hooks

Hooks are functions that make functional components work like class components
All hook names start with use... Example useState hook

```
HandlingEvent.jsx ×  
react-vite-proj2 > src > components > HandlingEvent.jsx > HandlingEvent  
1 import React, {useState} from 'react'  
2  
3 const HandlingEvent = () => {  
4     let tasks = 0;  
5  
6     const countArray = useState(0); //useState(0) returns array  
7     console.log("useState hook = ", countArray);  
8  
9     const handleClick = () => {  
10         tasks = tasks + 1;  
11         console.log("Add Task: ", tasks);  
12     }  
13  
14     return (  
15         <div>  
16             <h1>Tasks: {tasks}</h1>  
17             <button onClick={handleClick}>Add Task</button>  
18         </div>  
19     );  
20 };  
21  
22 export default HandlingEvent
```

UseState hook =
▼ (2) [0, f] i
 ▶ 0: 0
 ▶ 1: f ()
 length: 2
 ▶ [[Prototype]]: Array(0)

HandlingEvent.jsx:7

HandlingEvent.jsx X

react-vite-proj2 > src > components > HandlingEvent.jsx > default

```
1 import React, {useState} from 'react'
2
3 const HandlingEvent = () => {
4     let tasks = 0;
5
6     const countArray = useState(0); //useState(0) returns array
7     console.log("useState hook = ", countArray);
8     const countValue = countArray[0];
9     const setCountValue = countArray[1];//Function used to update countValue
10
11    const handleClick = () => {
12        tasks = tasks + 1;
13        setCountValue(countValue + 1);
14        console.log("Add Task: ", tasks);
15    }
16
17    return (
18        <div>
19            <h1>Tasks: {tasks}</h1>
20            <h1>Tasks using Hooks: {countValue}</h1>
21            <button onClick={handleClick}>Add Task</button>
22        </div>
23    );
24}
25
26 export default HandlingEvent
```

Tasks: 0

Tasks using Hooks: 4

Add Task

Add Task: 1

HandlingEvent.jsx:14

useState hook = ▶ (2) [4, f] HandlingEvent.jsx:7

useState hook = ▶ (2) [4, f] HandlingEvent.jsx:7

HandlingEvent.jsx X

```
react-vite-proj2 > src > components > HandlingEvent.jsx > [HandlingEvent]
1 import React, {useState} from 'react'
2
3 const HandlingEvent = () => {
4
5     //const countArray = useState(0); //useState(0) returns array
6     //const countValue = countArray[0];
7     //const setCountValue = countArray[1];//Function used to update countValue
8     const [countValue, setCountValue] = useState(0); //Array destructuring
9
10    const handleClick = () => {
11        setCountValue(countValue + 1);
12    }
13
14    return (
15        <div>
16            <h1>Tasks using Hooks: {countValue}</h1>
17            <button onClick={handleClick}>Add Task</button>
18        </div>
19    );
20}
21
22 export default HandlingEvent
```

1. State Variable

2. Function to update State Variable

Handling user inputs

```
HandlingEvent.jsx X
react-vite-proj2 > src > components > HandlingEvent.jsx > handleEvent > handleChange

1 import React, {useState} from 'react'
2
3 const HandlingEvent = () => {
4     const [countValue, setCountValue] = useState(0); //Array destructuring
5
6     const handleClick = () => {
7         setCountValue(countValue + 1);
8     };
9
10    const handleChange = (event) => {
11        console.log(event.target.value);
12    };
13
14    return (
15        <div>
16            <h1>Tasks using Hooks: {countValue}</h1>
17            <input type="text" onChange={handleChange} />
18            <button onClick={handleClick}>Add Task</button>
19        </div>
20    );
21};
22
23 export default HandlingEvent
```

Tasks using Hooks: 0

 Add Task

Hello
Hello

```
HandlingEvent.jsx X

react-vite-proj2 > src > components > HandlingEvent.jsx > default

1 import React, {useState} from 'react'
2
3 const HandlingEvent = () => {
4     const [countValue, setCountValue] = useState(0); //Array destructuring
5     const [inputText, setInputText] = useState("");
6
7     const handleClick = () => {
8         setCountValue(countValue + 1);
9     };
10
11    const handleChange = (event) => {
12        //console.log(event.target.value);
13        setInputText(event.target.value);
14    };
15
16    return (
17        <div>
18            <h1>Tasks using Hooks: {countValue}</h1>
19            <input type="text" onChange={handleChange} />
20            <button onClick={handleClick}>Add Task</button>
21
22            <h1>Input Text: {inputText}</h1>
23        </div>
24    );
25};
26
27 export default HandlingEvent
```

Tasks using Hooks: 0

hello

Input Text: hello

Mapping List / Rendering List in JSX

MappingList.jsx X

react-vite-proj2 > src > components > MappingList.jsx > [o] MappingList

```
1 import React from 'react';
2
3 const MappingList = () => {
4     let names = ["Hari", "Giri", "Suri", "Mari", "Siri"];
5
6     return (
7         <div>
8             <ul>
9                 for(let name of names) {
10                     <li>name</li>
11                 }
12             </ul>
13         </div>
14     );
15 };
16
17 export default MappingList;
```

for(let name of names)
 name

Not working

MappingList.jsx X

react-vite-proj2 > src > components > MappingList.jsx > [d] default

```
1 import React from 'react';
2
3 const MappingList = () => {
4     let names = ["Hari", "Giri", "Suri", "Mari", "Siri"];
5
6     return (
7         <div>
8             <ul>
9                 {
10                     JSX Expression
11                     names.map(name => <li key={name}>{name}</li>)
12                 }
13             </ul>
14         </div>
15     );
16
17     export default MappingList;
```

This is Important as Virtual DOM needs this
key to uniquely identify each item

- Hari
- Giri
- Suri
- Mari
- Siri

Task Management Project

Using Concepts learned till now



Task Management Application Overview & Planning

The screenshot shows a web-based task management application interface. At the top, there is a header bar with a file icon labeled "Page 1", a refresh/circular arrow icon, and a URL field containing "http://localhost:5173". To the right of the URL field are three small circular icons. Below the header is a search bar with the placeholder text "Enter your task". Underneath the search bar is a row of colored buttons: Java (pink), Python (green), React (orange), and MongoDB (blue). To the right of these buttons are two more buttons: "To Do ▾" (grey) and "+ Add Task" (blue). Below this row, there are three main sections: "To Do", "In Progress", and "Done". Each section has a title with an icon and a list of tasks. Each task card includes a delete icon.

Page 1

← → C http://localhost:5173

Enter your task

Java Python React MongoDB To Do ▾ + Add Task

To Do

- Task-1 Java React X
- Task-11 Python React X

In Progress

- Task-2 Java React X
- Task-22 Java Python X

Done

- Task-3 Java MongoDB X

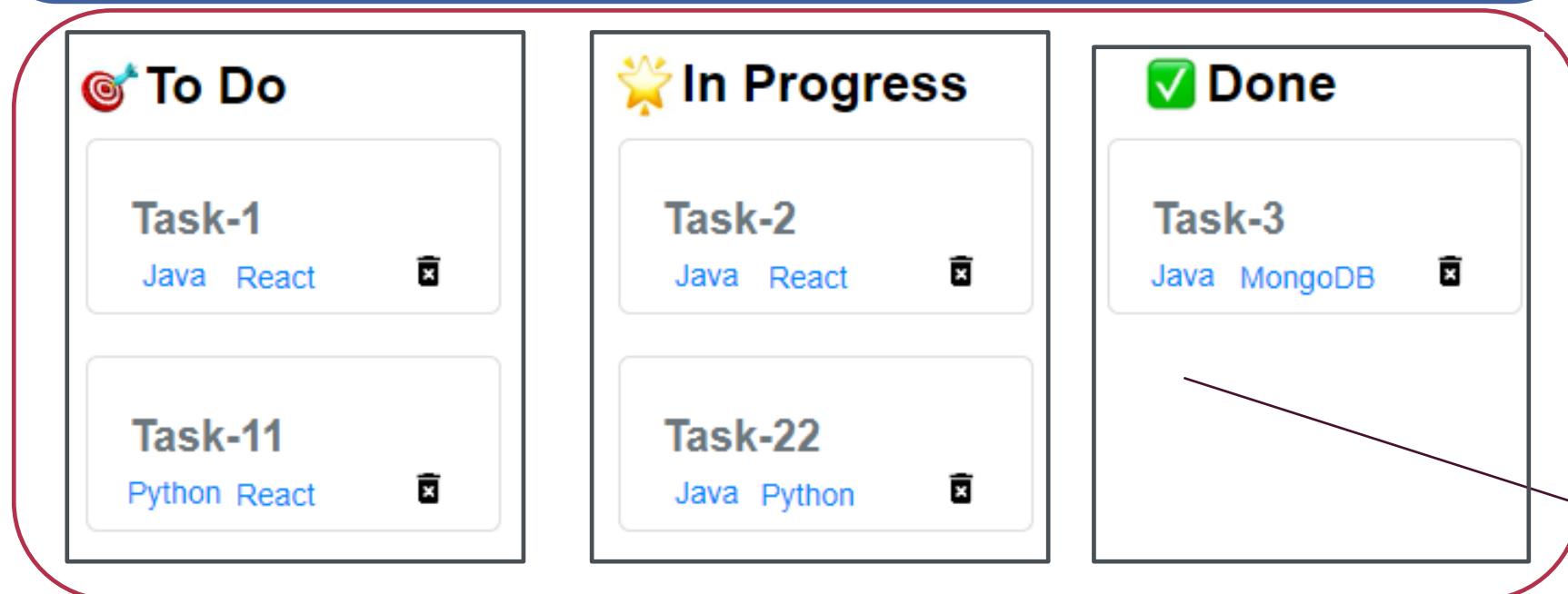
Creating Layout Of Application



Enter your task

Java Python React MongoDB To Do + Add Task

Header



Main Section

Creating Layout

- Step - I: Reset all predefined CSS
 - index.css - Remove existing content & Add new content
 - App.jsx - Remove existing content & Add new content
 - App.css - Remove existing content & Add new content

```
App.jsx  x
react-vite-proj2 > src > App.jsx > App
1 import React from 'react';
2 import './App.css';
3
4 const App = () => {
5   return (
6     <div className='app'>
7       <header className='app_header'>Header Section</header>
8
9       <main className='app_main'>
10         <section className='task_column'>Section - 1</section>
11         <section className='task_column'>Section - 2</section>
12         <section className='task_column'>Section - 3</section>
13       </main>
14
15     </div>
16   );
17 }
18
19 export default App;
```

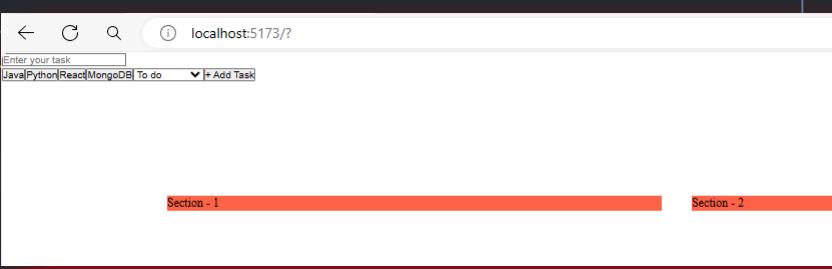
```
App.css  x
react-vite-proj2 > src > App.css > .task_column
1 .app {
2   display: grid;
3   grid-template-rows: 150px auto;
4 }
5
6 .app_main {
7   display: flex;
8   justify-content: space-evenly;
9   padding: 20px 8%;
10 }
11
12 .task_column {
13   width: 33.33%;
14   background-color: tomato;
15   margin: 20px;
16 }
```

```
index.css  x
react-vite-proj2 > src > index.css > *
1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5 }
```

Create New Component – src/TaskManagementForm.jsx

TaskManagementForm.jsx x src/TaskManagementForm.jsx

```
react-vite-proj2 > src > components > TaskManagementForm.jsx > ...
1 import React from 'react';
2
3 const TaskManagementForm = () => {
4   return (
5     <header className='app_header'>
6       <form>
7         <input type='text' className='task_input' placeholder='Enter your task'/>
8         <div className='task_form_bottom_line'>
9           <button className='tag'>Java</button>
10          <button className='tag'>Python</button>
11          <button className='tag'>React</button>
12          <button className='tag'>MongoDB</button>
13
14         <select className='task_status'>
15           <option value='todo'>To do</option>
16           <option value='inprogress'>In progress</option>
17           <option value='done'>Done</option>
18         </select>
19
20         <button type='submit' className='task_submit'>+ Add Task</button>
21       </div>
22     </form>
23   </header>
24 );
25
26 export default TaskManagementForm;
```



localhost:5173/?

Section - 1 Section - 2 Section - 3

App.jsx x

```
react-vite-proj2 > src > App.jsx > default
1 import React from 'react';
2 import './App.css';
3 import TaskManagementForm from './components/TaskManagementForm';
4
5 const App = () => {
6   return (
7     <div className='app'>
8       <TaskManagementForm />
9
10    <main className='app_main'>
11      <section className='task_column'>Section - 1</section>
12      <section className='task_column'>Section - 2</section>
13      <section className='task_column'>Section - 3</section>
14    </main>
15
16  );
17};
18
19 export default App;
```

Adding Styles for TaskManagementForm Component

TaskManagementForm.css X

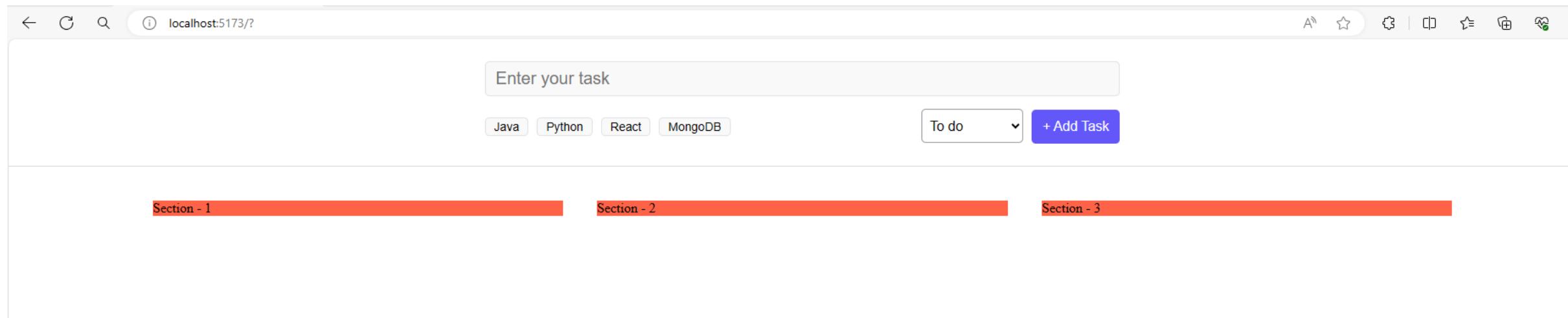
```
react-vite-proj2 > src > components > TaskManagementForm.css >
1 .app_header {
2   display: flex;
3   align-items: center;
4   justify-content: center;
5   border-bottom: 1px solid #dcdcdc;
6 }
7
8 .app_header > form {
9   width: 40%;
10}
11
12 .task_input {
13   font-size: 20px;
14   font-weight: 500;
15   background: #f9f9f9;
16   color: #000;
17   border: 1px solid #dfe3e6;
18   border-radius: 5px;
19   padding: 8px 12px;
20   margin-bottom: 15px;
21   width: 100%;
22 }
23
24 .task_form_bottom_line {
25   display: flex;
26   align-items: center;
27   justify-content: space-between;
28 }
```

```
30 .tag {
31   font-size: 14px;
32   font-weight: 500;
33   background-color: #f9f9f9;
34   border: 1px solid #dfe3e6;
35   border-radius: 5px;
36   padding: 2px 10px;
37   margin-right: 10px;
38   cursor: pointer;
39 }
40
41 .task_status {
42   font-size: 16px;
43   font-weight: 500;
44   border: 1px solid #999;
45   border-radius: 5px;
46   width: 120px;
47   height: 40px;
48   padding: 0 5px;
49 }
50
```

```
51 .task_submit {
52   font-size: 16px;
53   font-weight: 500;
54   background-color: #6457f9;
55   color: #fff;
56   border-radius: 5px;
57   border: 1px solid #dfe3e6;
58   height: 40px;
59   padding: 3px 13px;
60   margin-left: 10px;
61   border: none;
62   cursor: pointer;
63 }
```

NOTE: CSS File Name is TaskManagementForm.css

Final Output of Task Management Application



What is Props

- Props stands for properties
- Props are arguments passed into React Components
- Props is a way to pass data from parent component to child component



```
const TaskManagementForm = () => {
  return (
    <header className='app_header'>
      <form>
        <input type='text' className='task_input' placeholder='Enter your task'/>
        <div className='task_form_bottom_line'>
          <ButtonTag name="Java"/>
          <ButtonTag name="Python"/>
          <ButtonTag name="React"/>
          <ButtonTag name="MongoDB"/>
        </div>
      </form>
    </header>
  );
}

const ButtonTag = (props) => {
  return (
    <div>
      <button className='tag'>{props.name}</button>
    </div>
  );
}

export default TaskManagementForm;
```

NOTE:

We can pass Objects, Arrays, Functions through the props

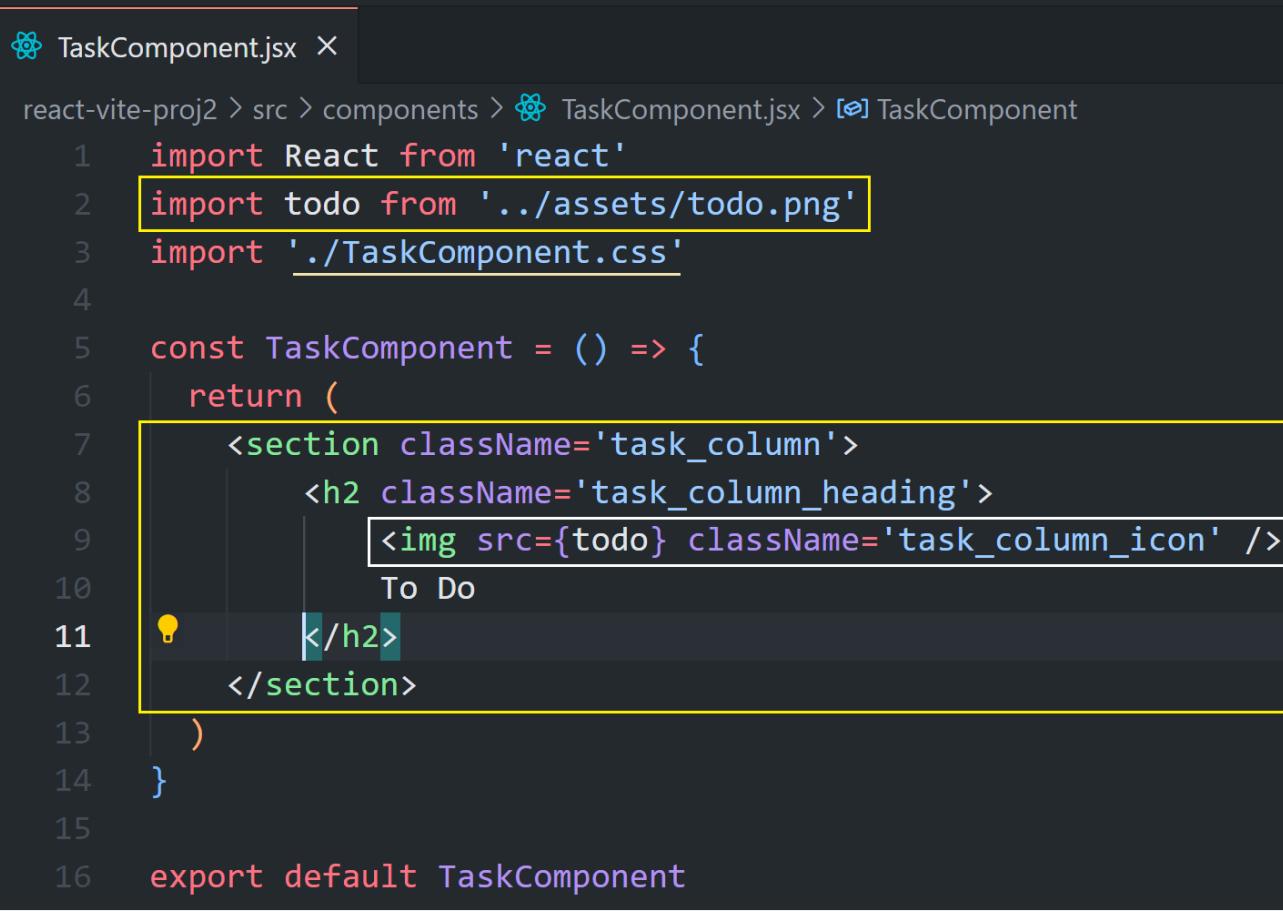
```
3
4  const ButtonTag = (props) => {
5    return (
6      <div>
7        <button className='tag'>{props.name}</button>
8      </div>
9    );
10 };
11
12 export default ButtonTag;
```

Designing Main Section



Working with Images in React

- Copy all images into project_folder/src/assets folder
- Import the image into the component where we are going to use that image
- Use imported variable inside



```
TaskComponent.jsx × react-vite-proj2 > src > components > TaskComponent.jsx > TaskComponent
1 import React from 'react'
2 import todo from '../assets/todo.png'
3 import './TaskComponent.css'
4
5 const TaskComponent = () => {
6   return (
7     <section className='task_column'>
8       <h2 className='task_column_heading'>
9         <img src={todo} className='task_column_icon' />
10        To Do
11        </h2>
12     </section>
13   )
14 }
15
16 export default TaskComponent
```

Designing Task Component

TaskComponent.jsx X

react-vite-proj2 > src > components > TaskComponent.jsx > TaskComponent

```
1 import React from 'react'
2 import todo from '../assets/todo.png'
3 import './TaskComponent.css'
4
5 const TaskComponent = () => {
6   return (
7     <section className='task_column'>
8       <h2 className='task_column_heading'>
9         <img src={todo} className='task_column_icon' />
10        To Do
11        </h2>
12     </section>
13   )
14 }
15
16 export default TaskComponent
```

TaskComponent.css X

```
react-vite-proj2 > src > components > TaskComponent.css
1 .task_column_heading {
2   display: flex;
3   align-items: center;
4 }
5
6 .task_column_icon {
7   width: 30px;
8   margin-right: 5px;
9 }
```

Object De-Structuring

```
TaskComponent.jsx X
react-vite-proj2 > src > components > TaskComponent.jsx > TaskComponent
1  import React from 'react'
2  import './TaskComponent.css'
3
4  const TaskComponent = (props) => {
5      const {title, task_img} = props;
6  return (
7      <section className='task_column'>
8          <h2 className='task_column_heading'>
9              {/*<img src={props.task_img} className='task_column_icon' />
10             {props.title}*//}
11              <img src={task_img} className='task_column_icon' />
12              {title}
13          </h2>
14      </section>
15  )
16}
17
18 export default TaskComponent
```