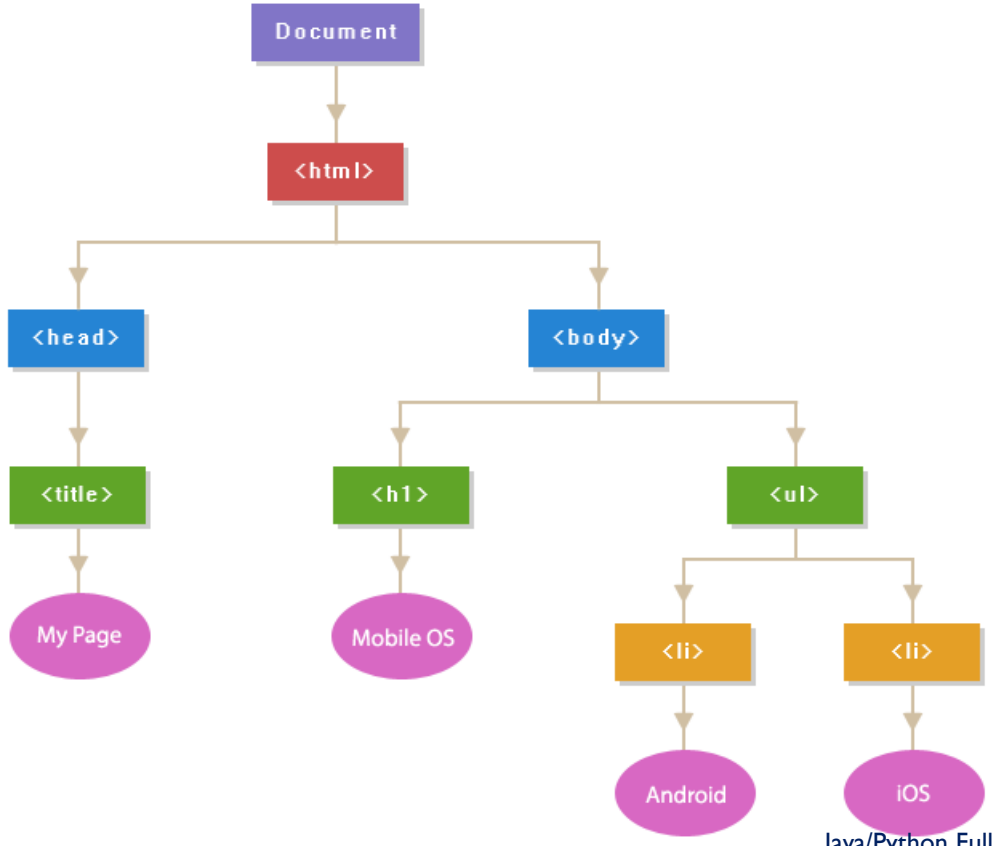
ReactJS

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>
```



Features of ReactJS…

- 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

Features of ReactJS…

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.

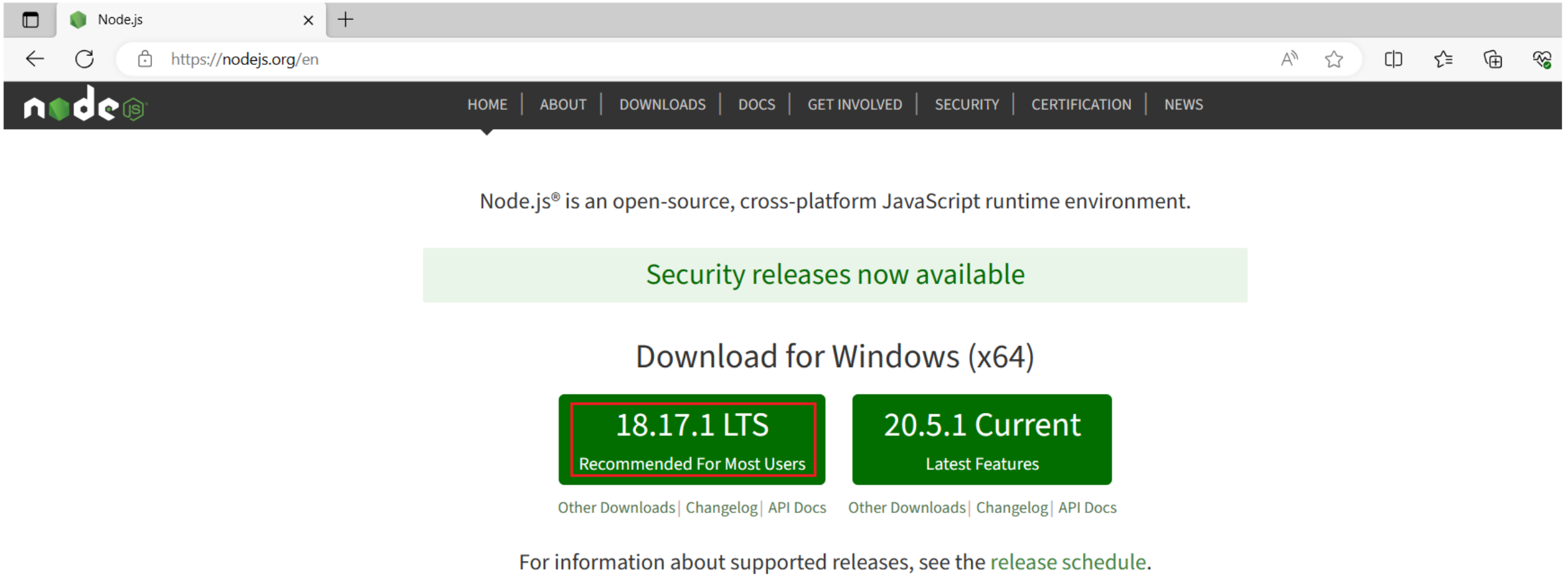
Component-based Architecture

- ReactJS divides the application into multiple components.

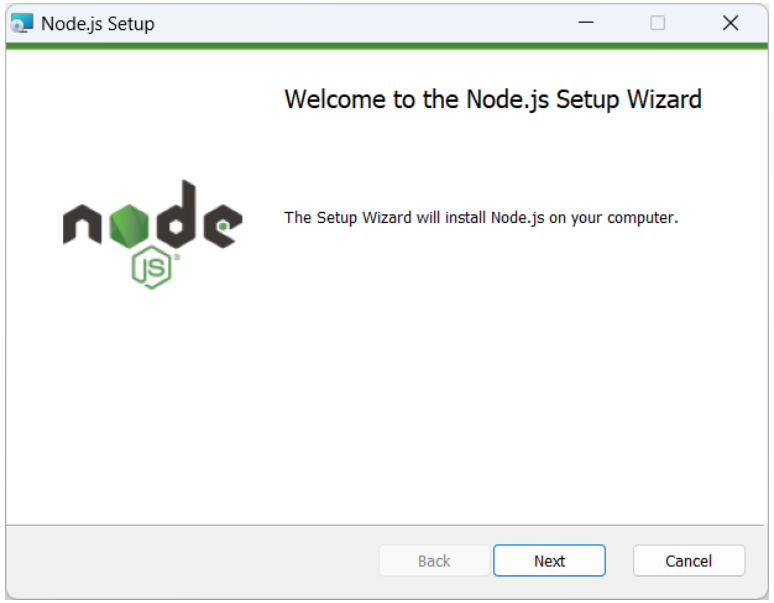
Software Needed

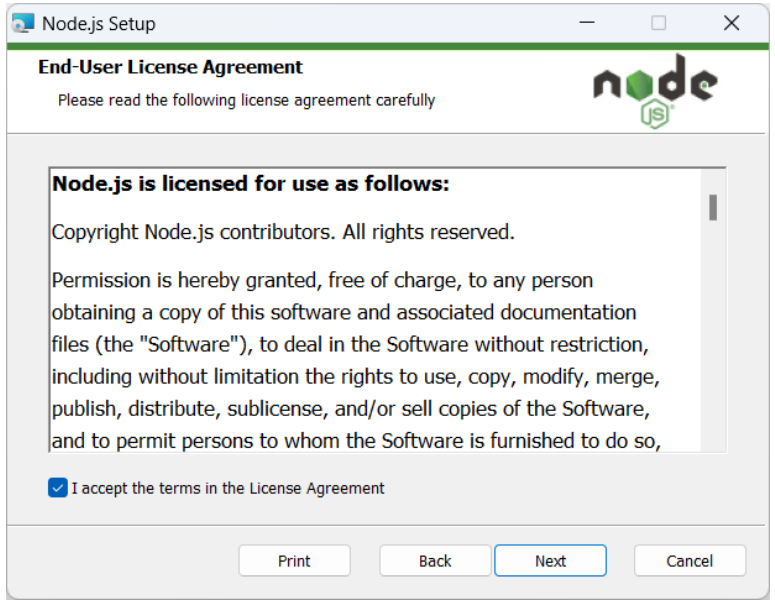
- Bowser
- VS Code
- Node JS
- React JS

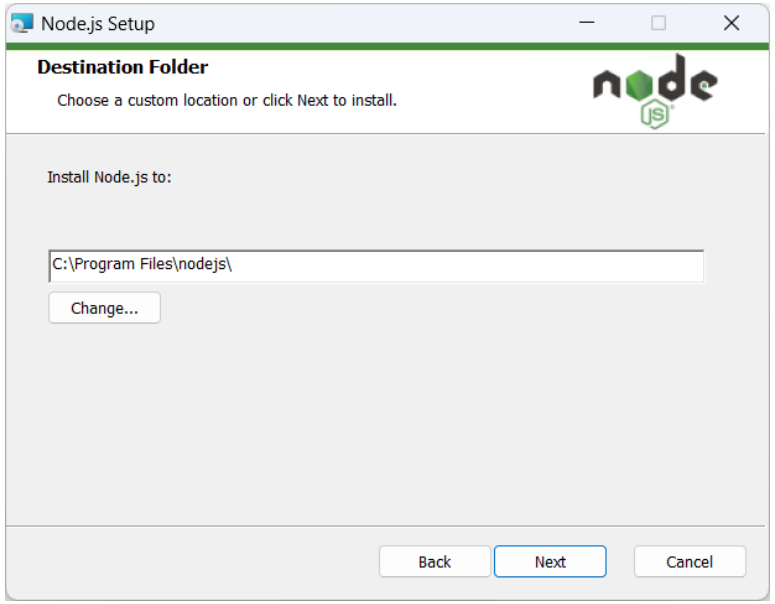
Installing NodeJS

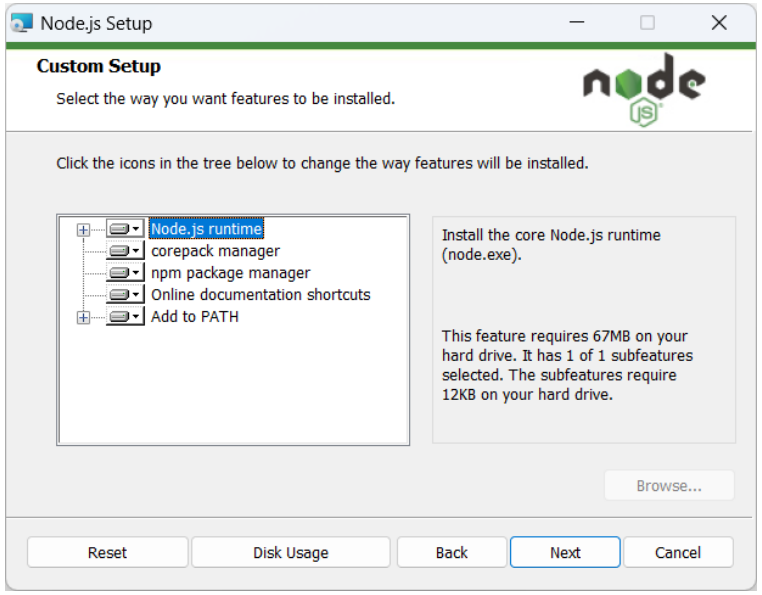


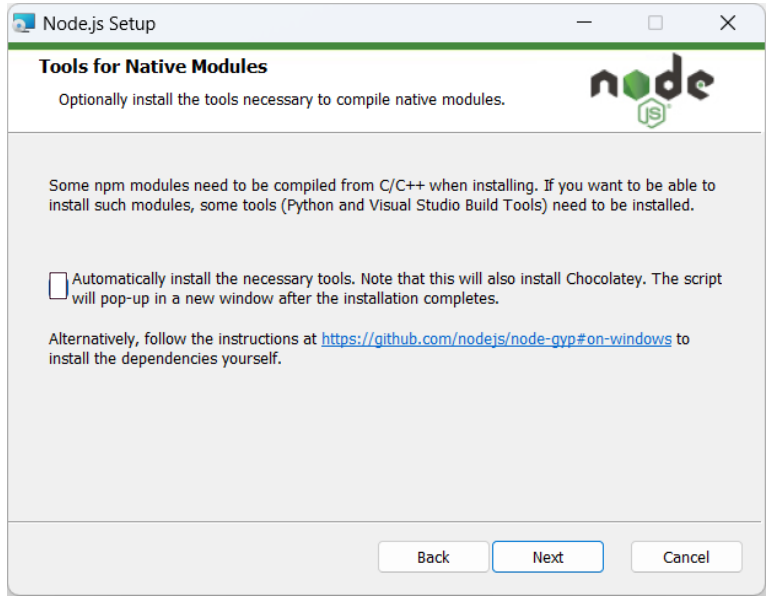
Installing NodeJS…

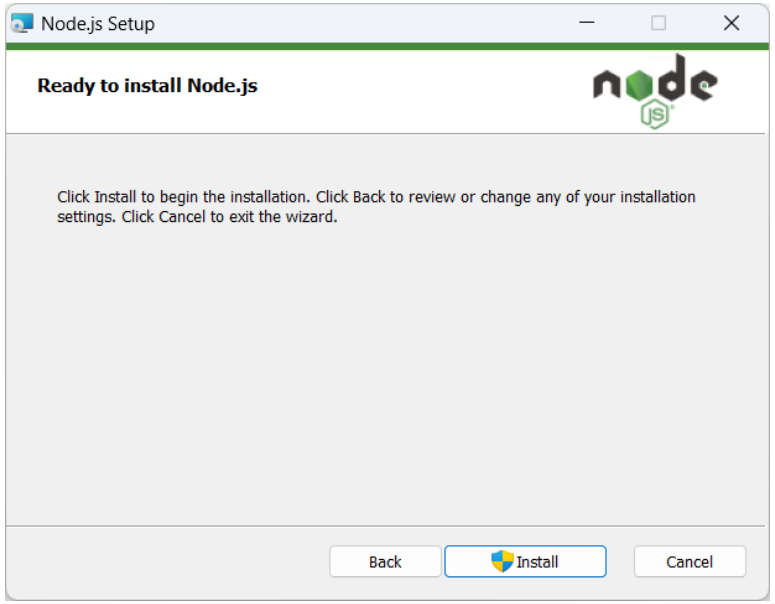




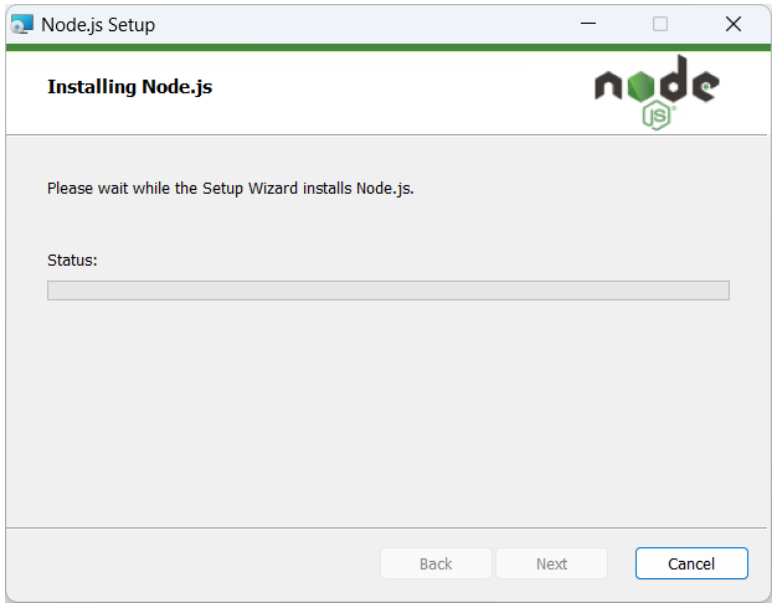


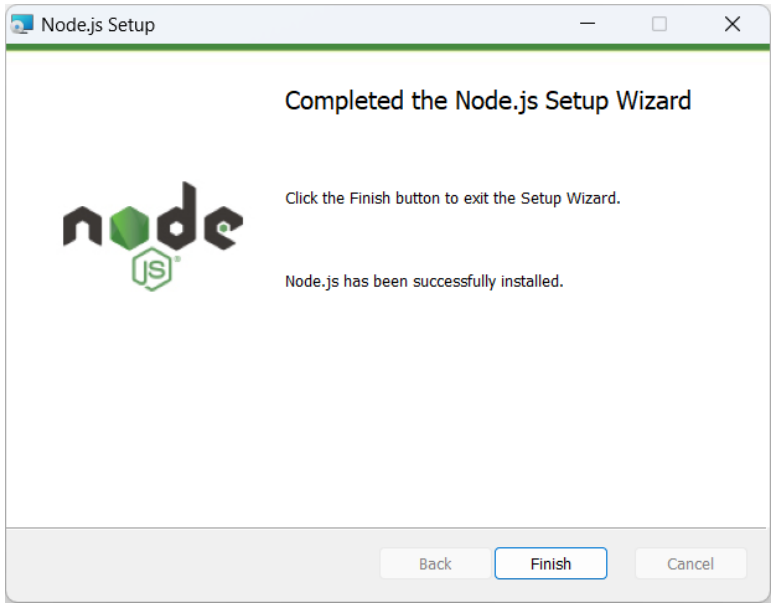


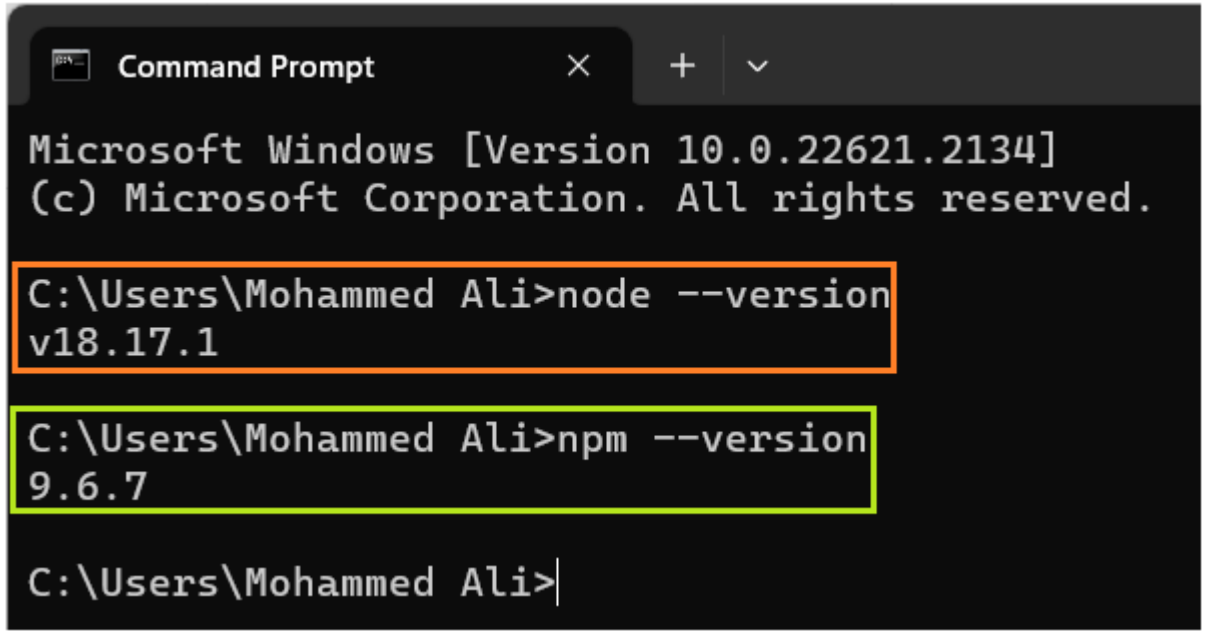




Installing NodeJS…





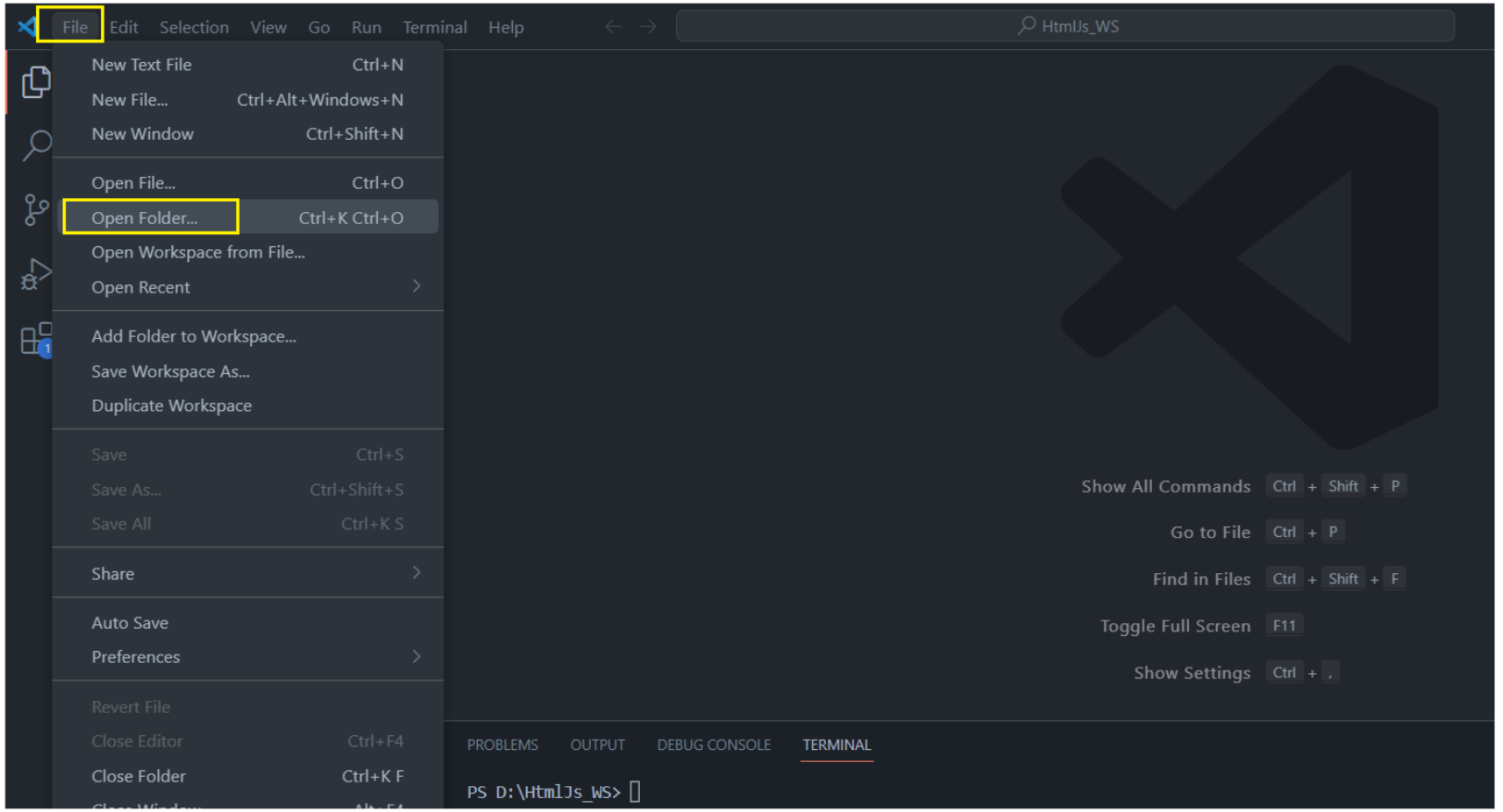


Create ReactJS Application

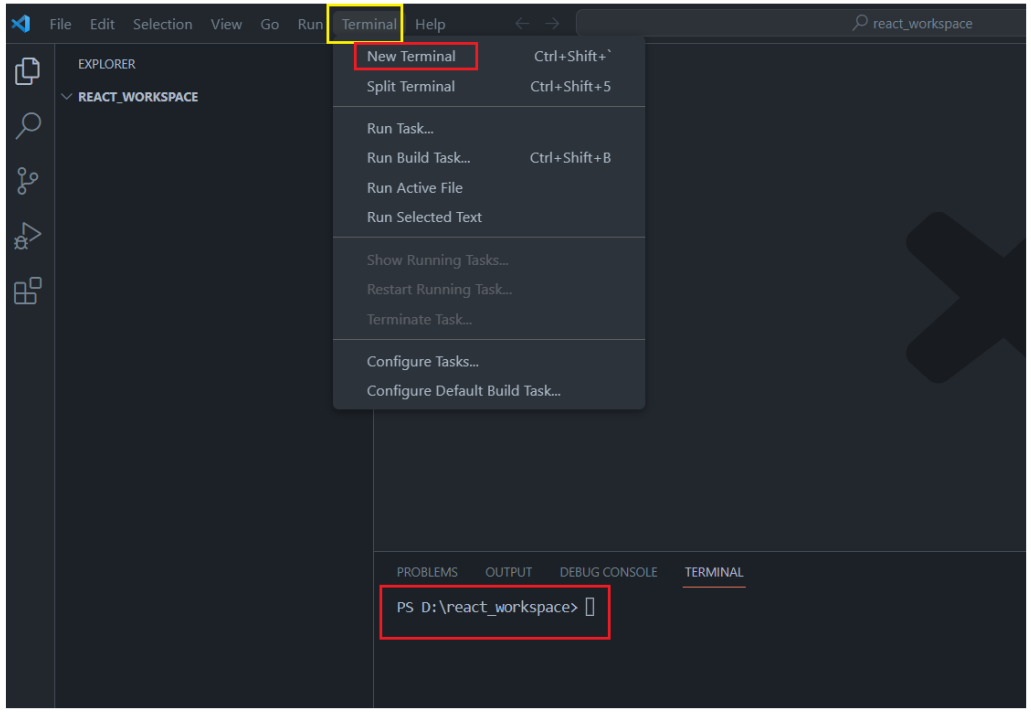
- STEP-1: Create "react_workspace" folder in C or D drive
- STEP-2: cd react_workspace & run below command
  D:\react_workspace>npx create-react-app emp-reactapp
- STEP-3: cd emp-reactapp & run below command
  D:\react_workspace\emp-reactapp>npm start

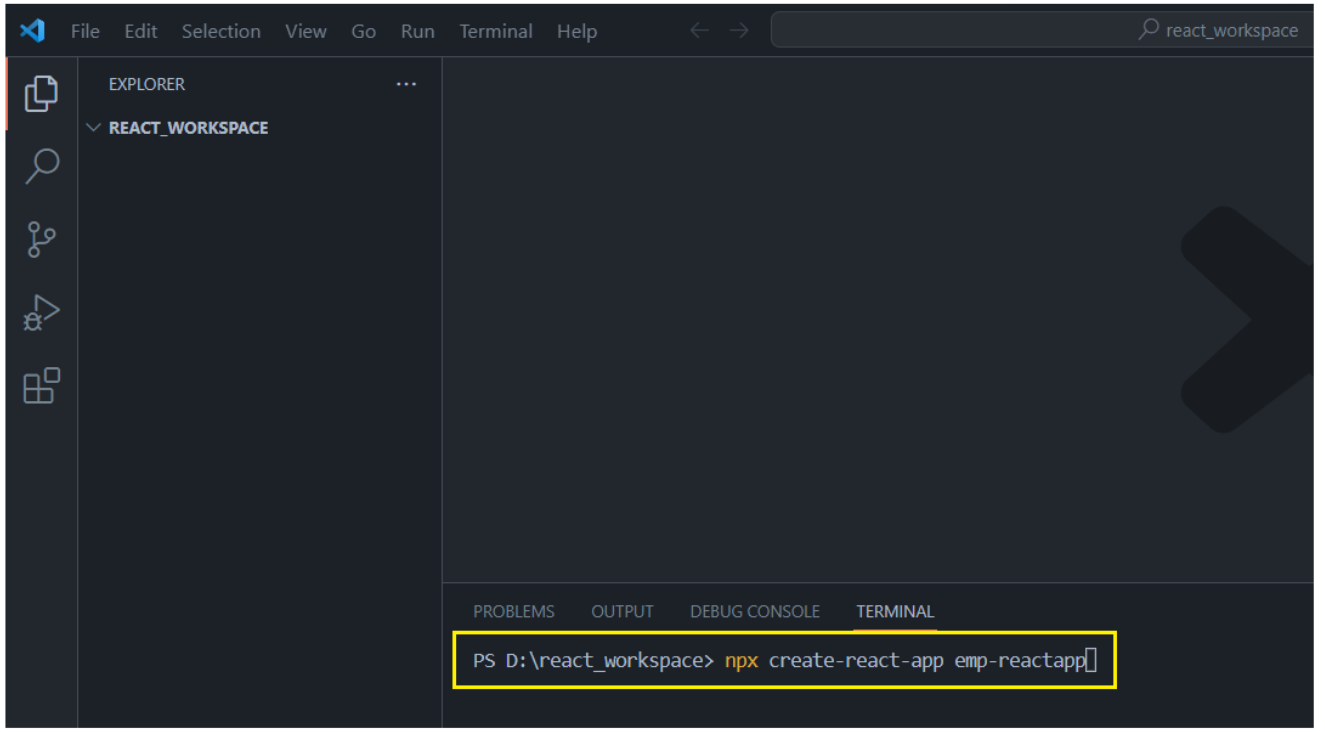
STEP - 1

STEP-1: Create "react_workspace" folder in C or D drive

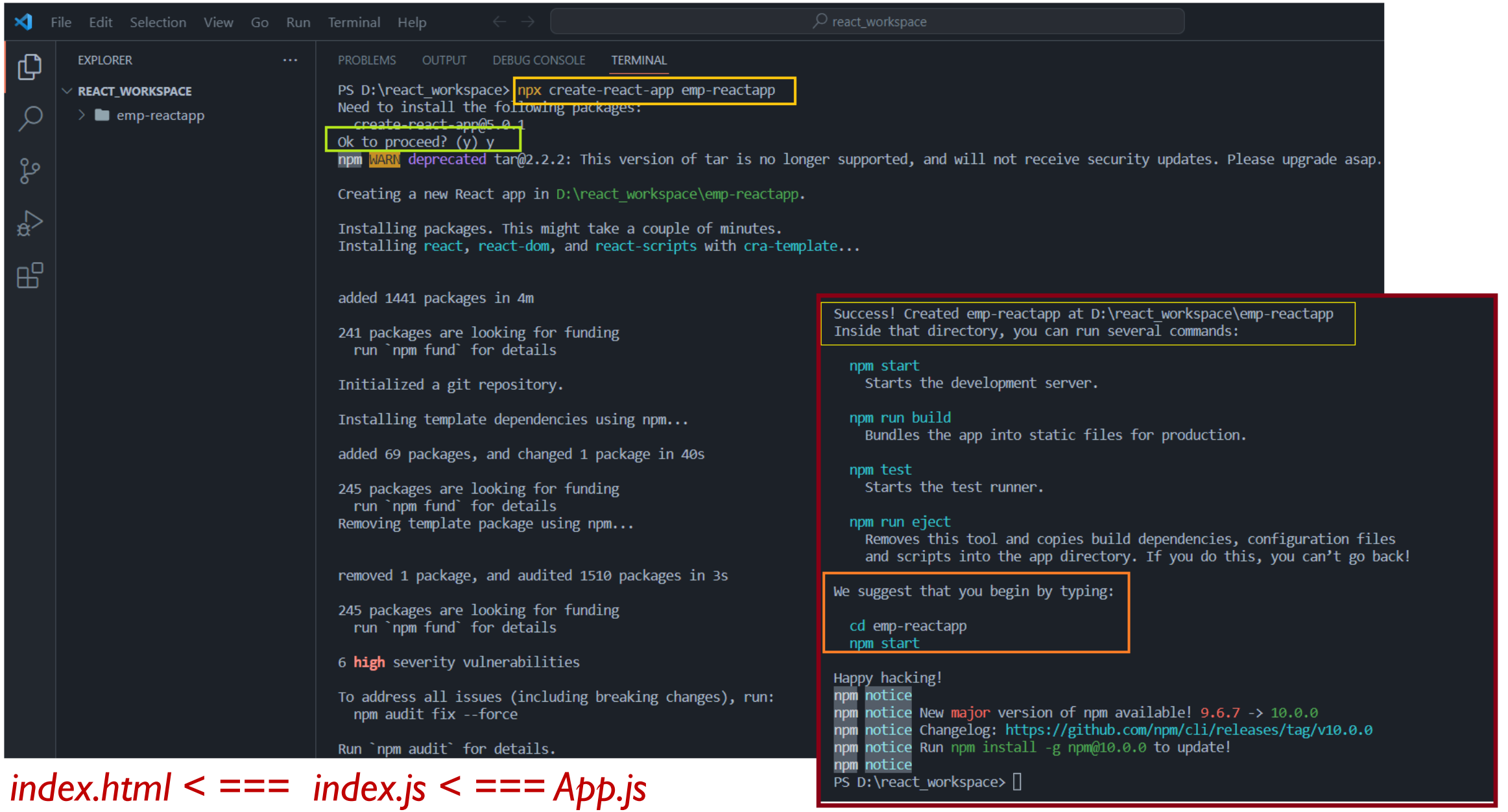


STEP - 2 run **create-react-app** command





STEP – 2…



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

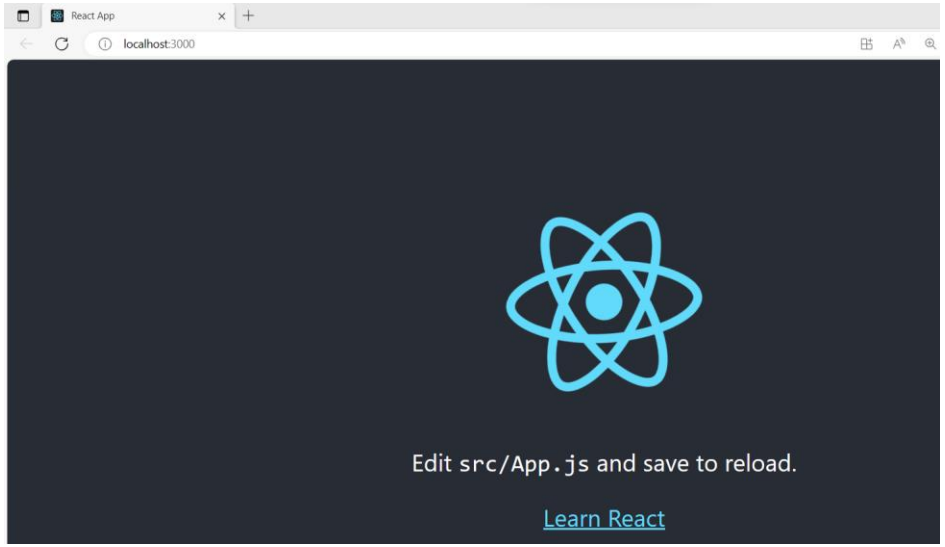
STEP-3: Run React Application

Go to React Project folder & Run below command

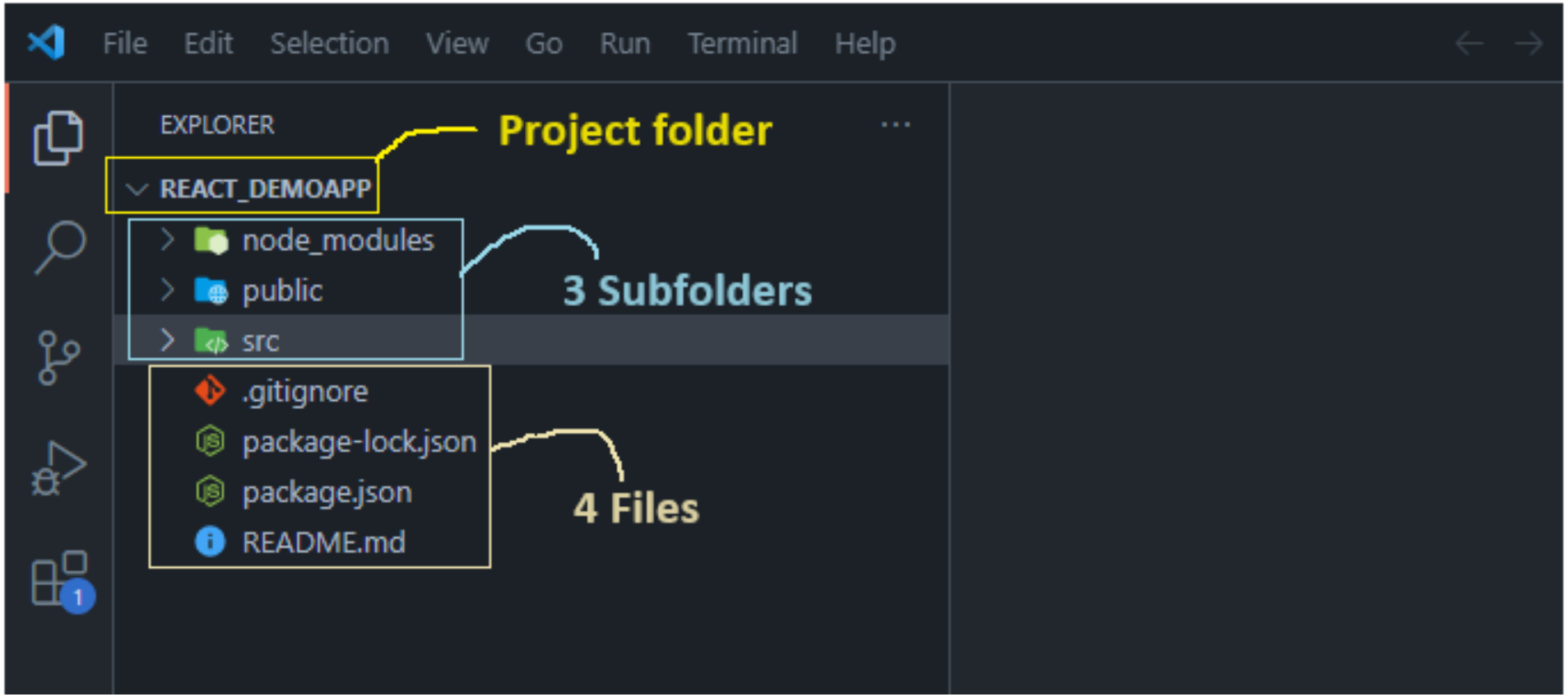
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\react_workspace> cd emp-reactapp
PS D:\react_workspace\emp-reactapp>
```

D:\react_workspace\emp-reactapp>npm start

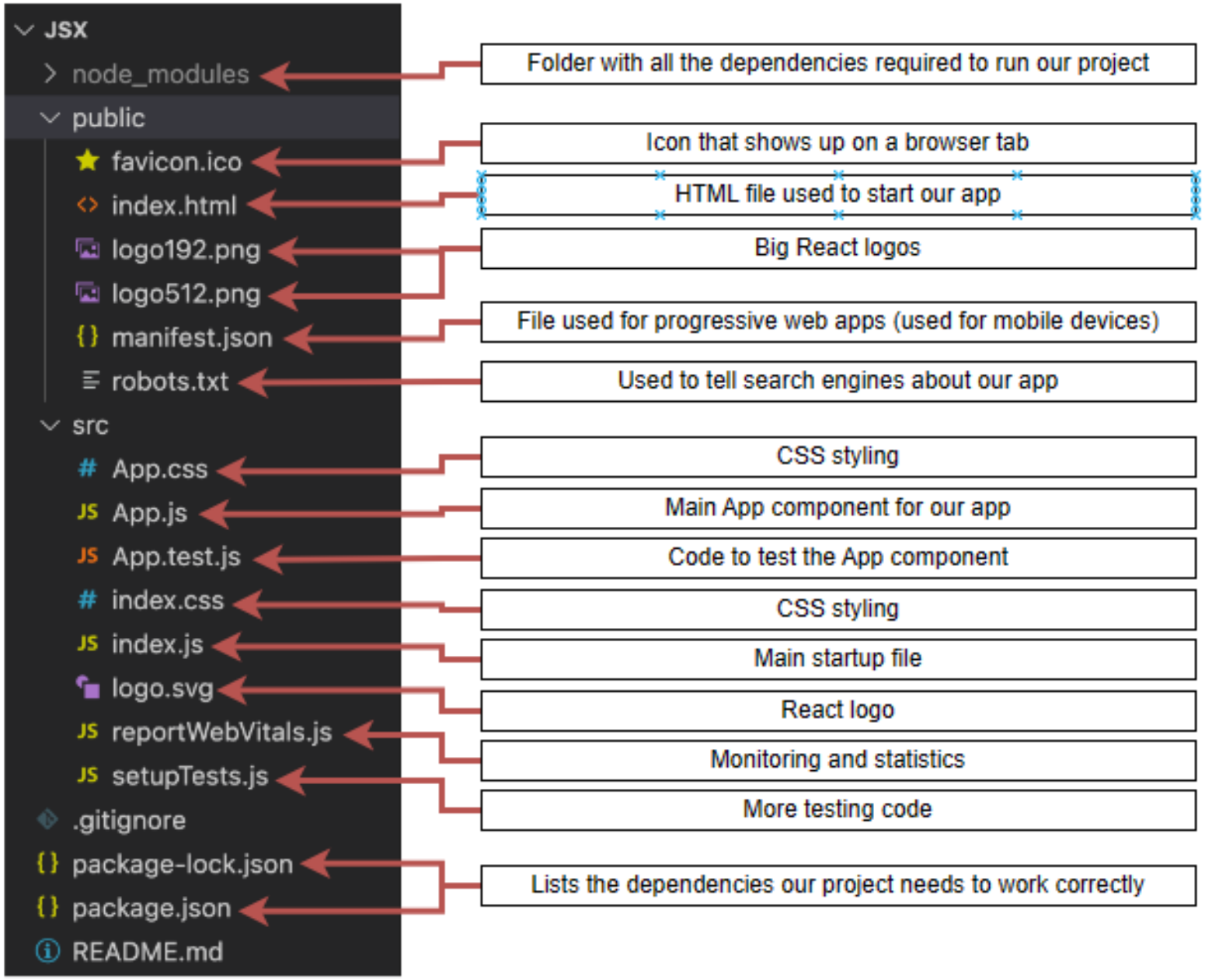
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

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
```



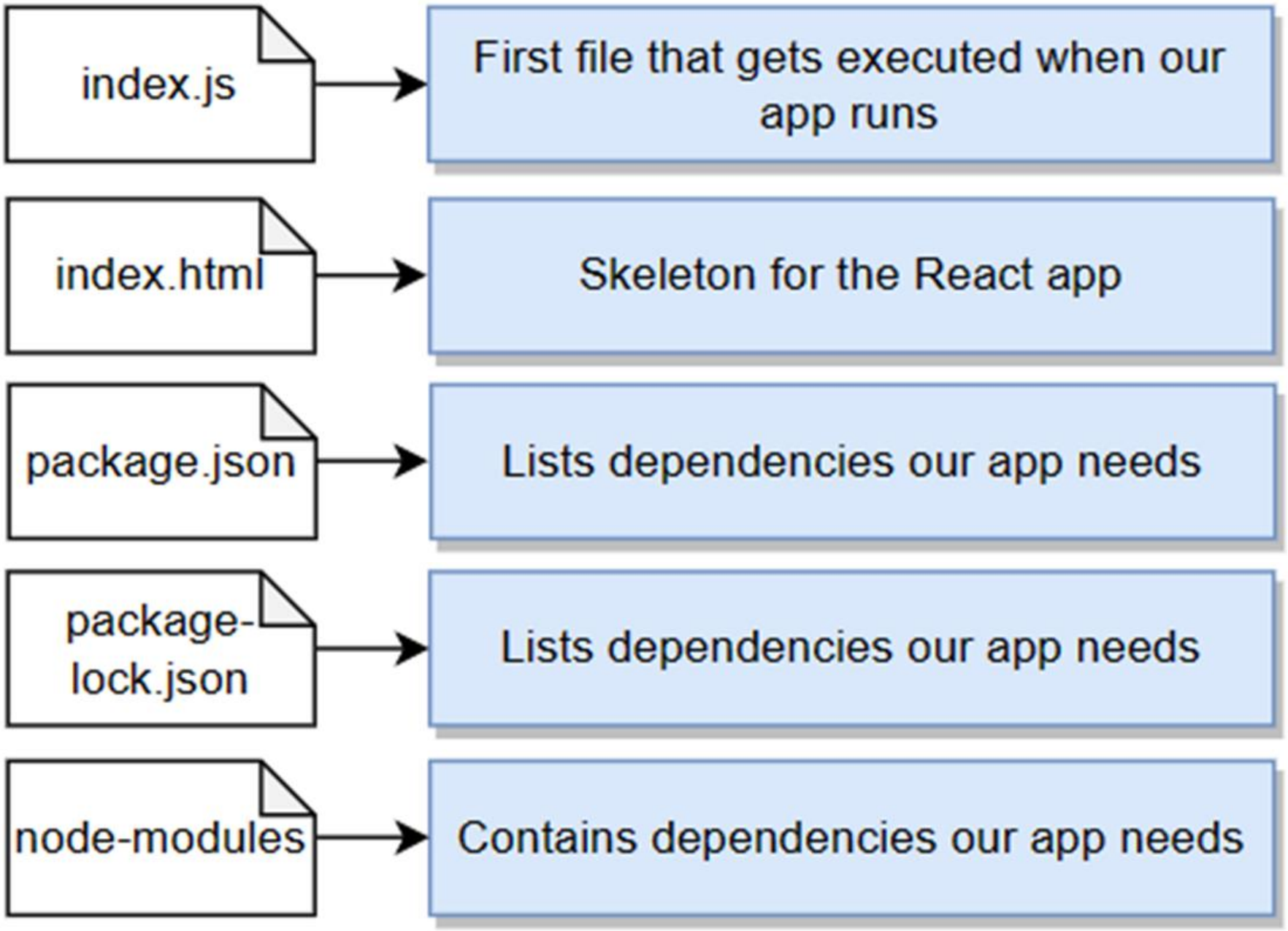
React Project Folder Structure



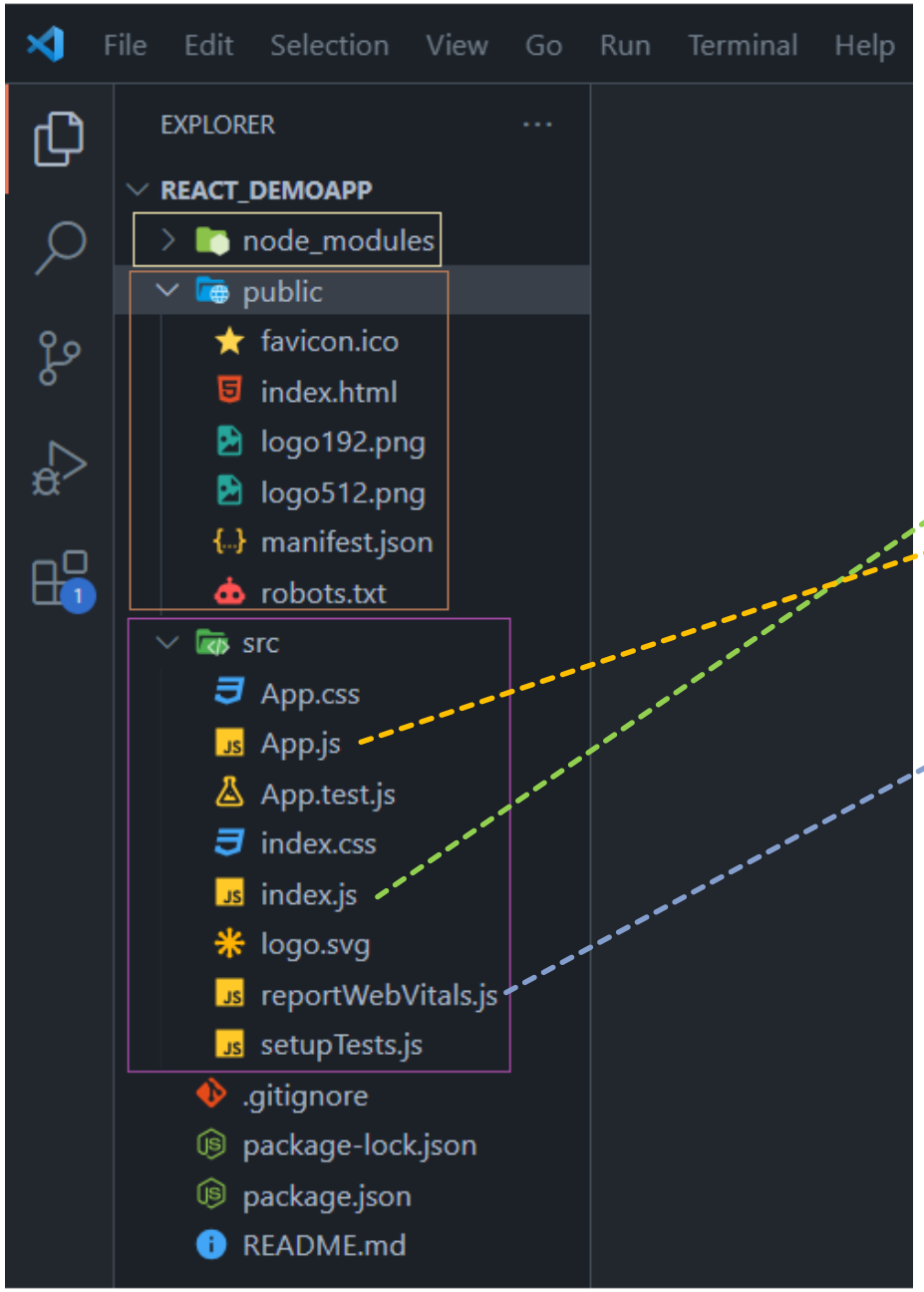
React Project – Different files

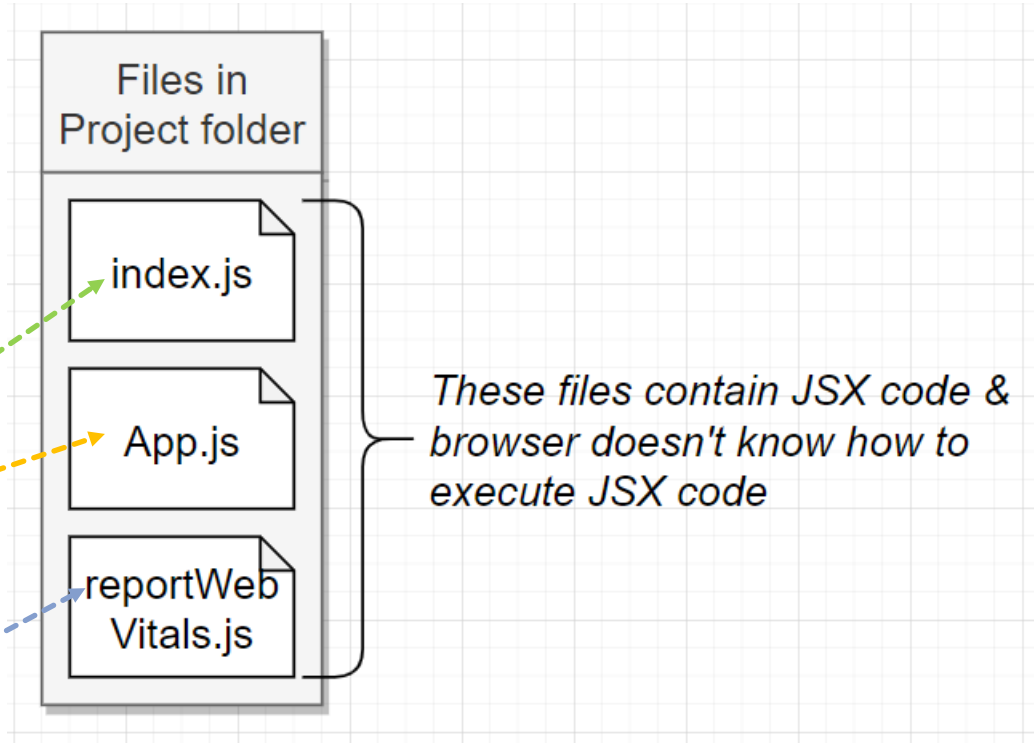


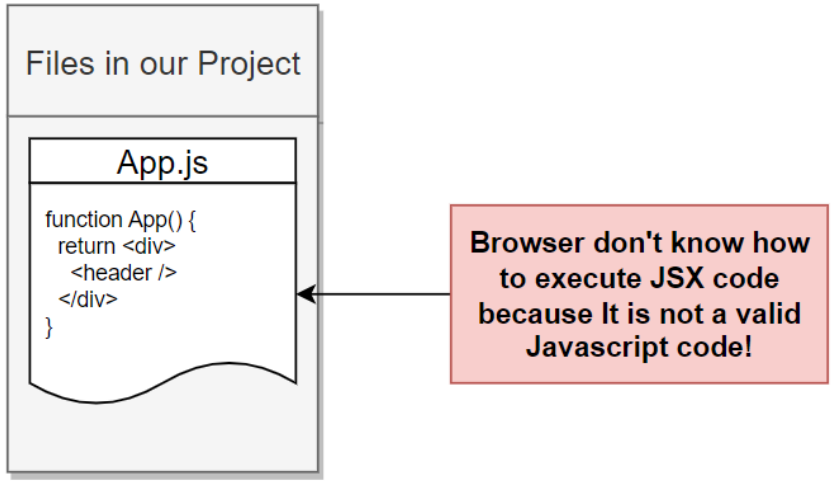
React Project – Some Important files



Need of Transpiler

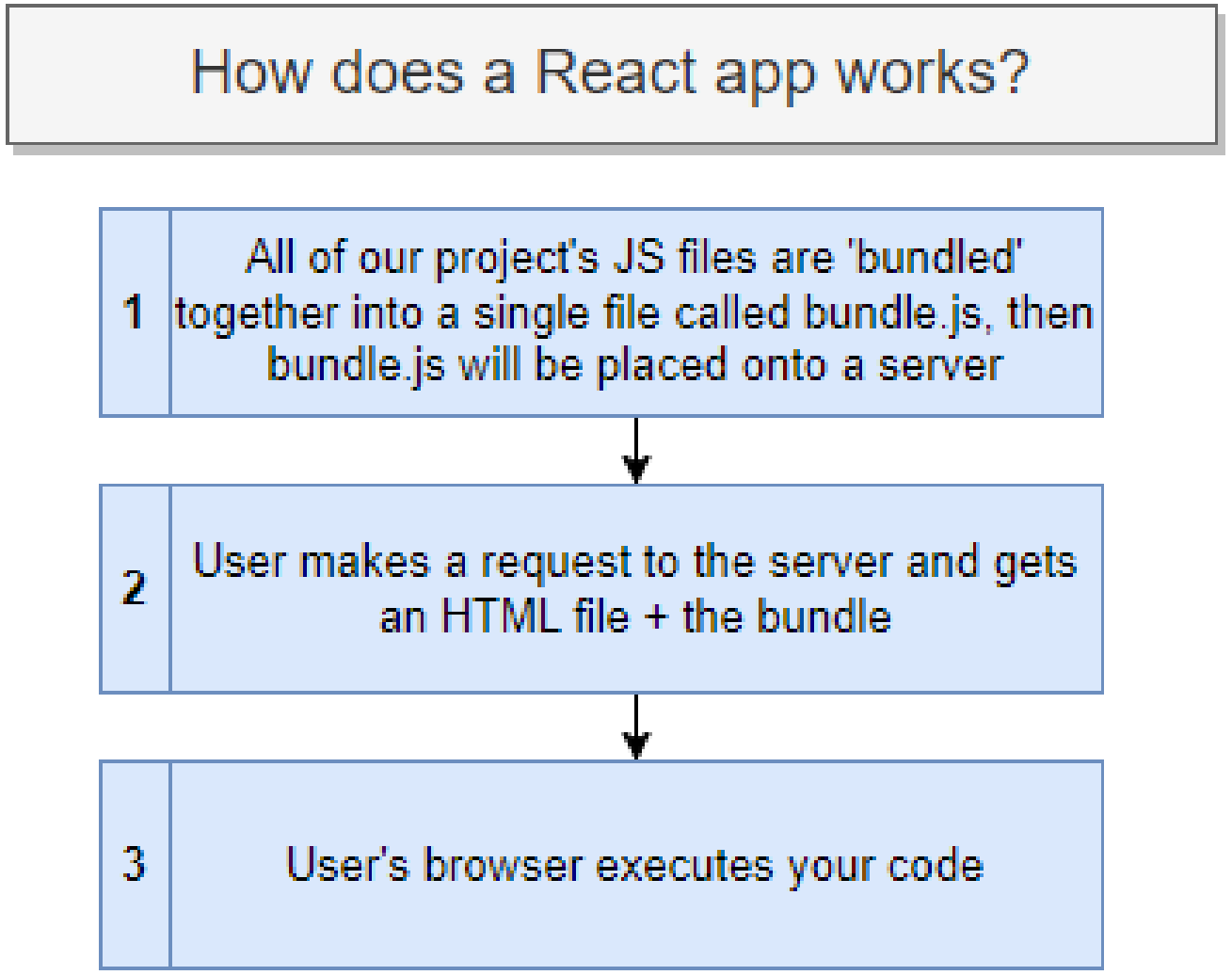






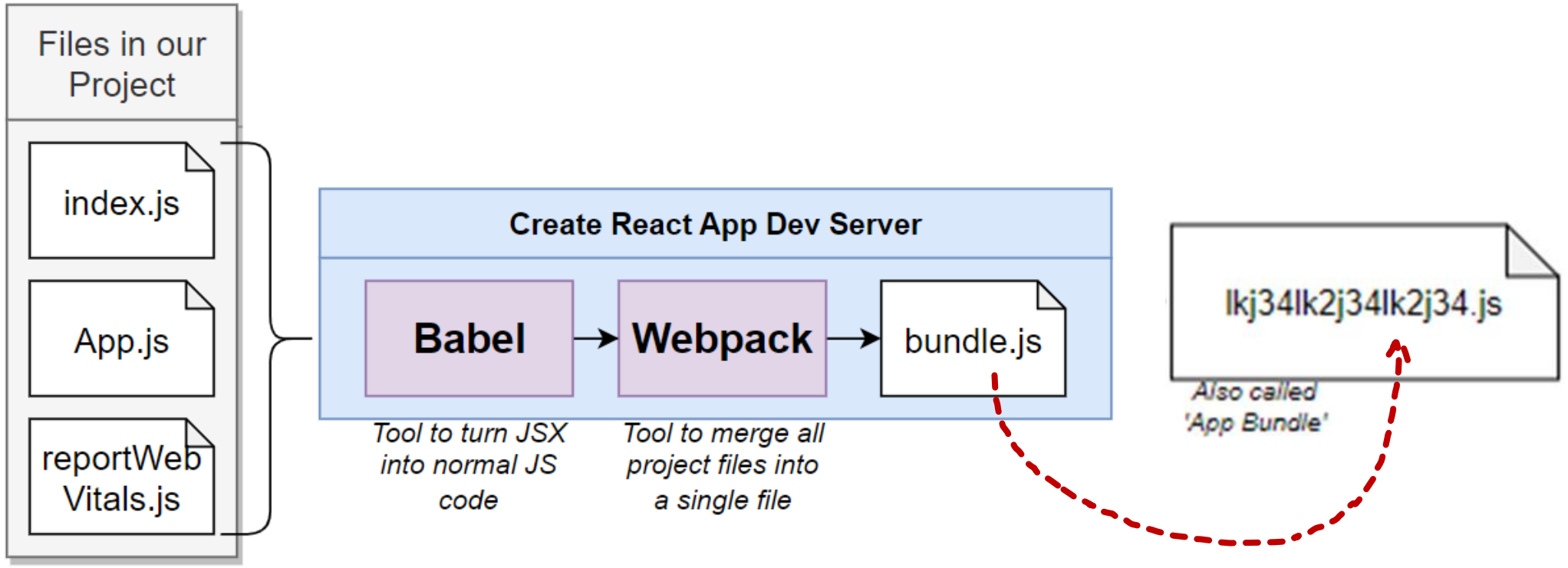
Solution is, We need to transpile the code into browser understandable format

How does React Application Works?



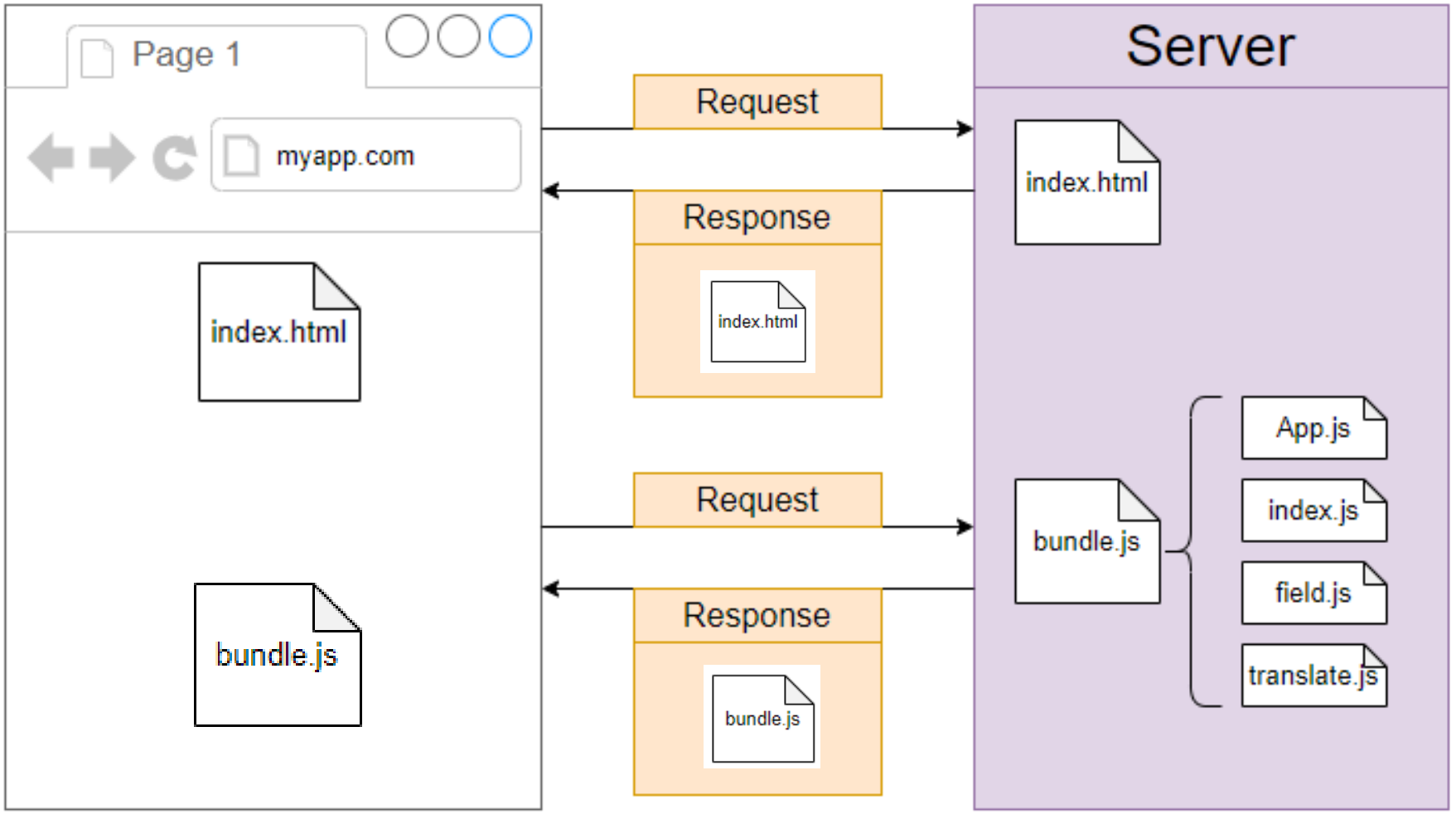
How does React Application Works…?

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

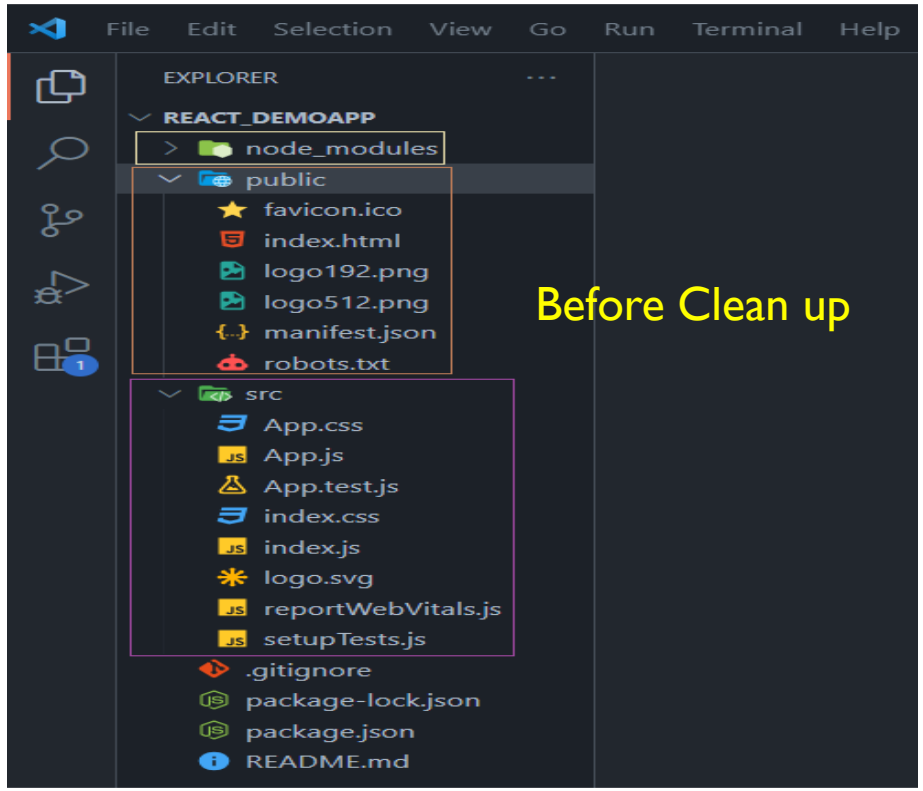
All inside the index.js file

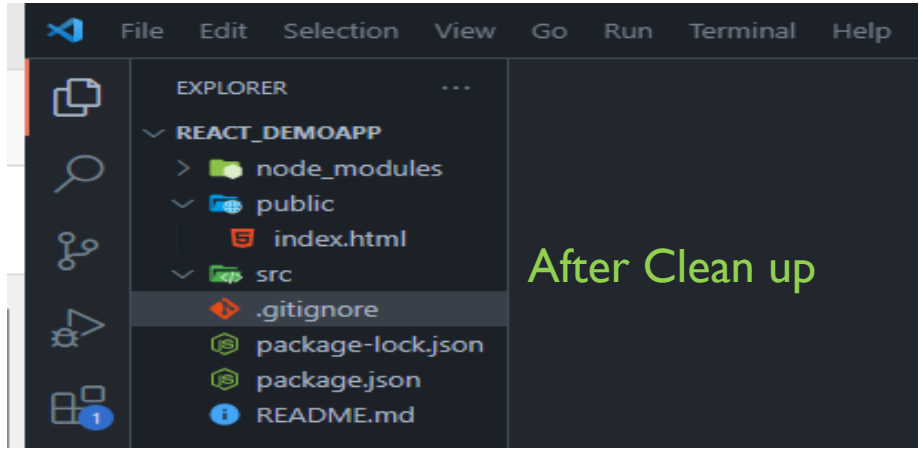
1	Find the div with id of 'root' in the DOM
2	Tell React to take control of that element
3	Tell React to get JSX from the App component, turn it into HTML, and show it in the root

Creating a Plain React Application

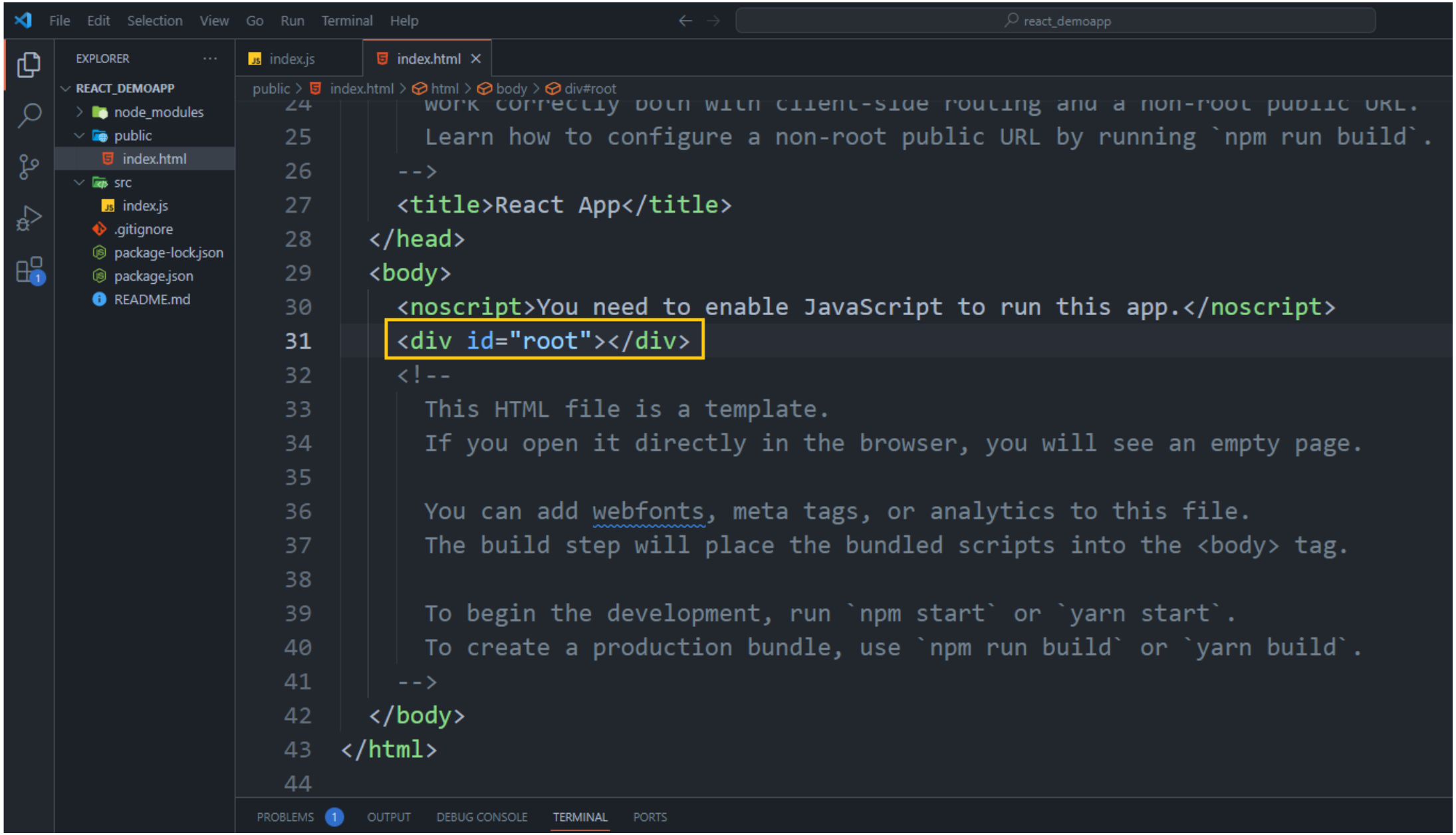
Starting Plain React Application

- 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

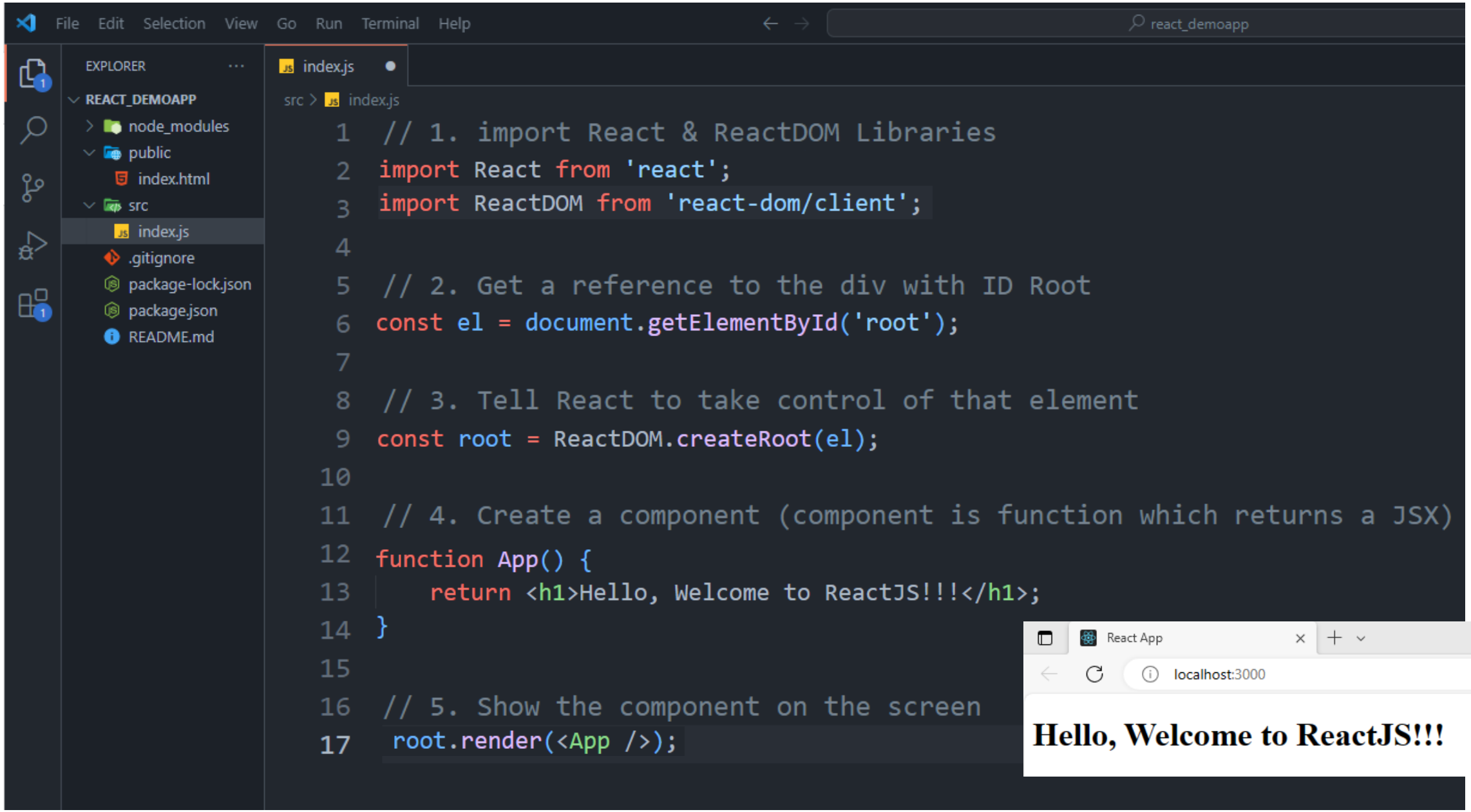




public/index.html



src/index.js


File Edit Selection View Go Run Terminal Help
react_demoapp
EXPLORER
REACT_DEMOAPP
node_modules
public
index.html
src
index.js
.gitignore
package-lock.json
package.json
README.md
index.js
src > index.js
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 return <h1>Hello, Welcome to ReactJS!!!</h1>;
14 }
15
16 // 5. Show the component on the screen
17 root.render(<App />);
React App
localhost:3000
Hello, Welcome to ReactJS!!!

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

JSX we write inside a function

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

Babel

JSX Equalent JS Code runs on browser

```
React.createElement("h1", null, "Hi there!");
```

This is what is returned from calling 'createElement'

```
{
  $$typeof: Symbol(react.element),
  key: null,
  props: { children: 'Hi there!' },
  ref: null
  type: 'h1'
}
```

Why we use th JSX…

babeljs.io/repl

BABEL Docs

```jsx
<div>
  <h1>Hello, Welcome to JSX</h1>
  <ul>
    <li>Ali</li>
    <li>Malli</li>
    <li>Ravi</li>
  </ul>
</div>
```

JSX Code

```jsx
import { jsx as _jsx } from "react/jsx-runtime";
import { jsxs as _jsxs } from "react/jsx-runtime";
/*#__PURE__*/_jsxs("div", {
  children: [/*#__PURE__*/_jsx("h1", {
    children: "Hello, Welcome to JSX"
  }), /*#__PURE__*/_jsxs("ul", {
    children: [/*#__PURE__*/_jsx("li", {
      children: "Ali"
    }), /*#__PURE__*/_jsx("li", {
      children: "Malli"
    }), /*#__PURE__*/_jsx("li", {
      children: "Ravi"
    })]
  })]
});
```

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

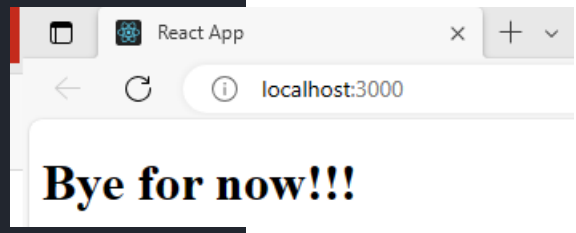
```
function App() {
    <h1>This is extra text from COMPONENT</h1>
    return <h1>Hello, Welcome to ReactJS!!!..</h1>;
}
```

JavaScript Variables in JSX

index.js

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 />);
```



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

```jsx
function App() {
    const date = new Date();
    const time = date.toLocaleDateString();

    return <h1>{time}</h1>;
}
```

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

Good

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

Bad

```jsx
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

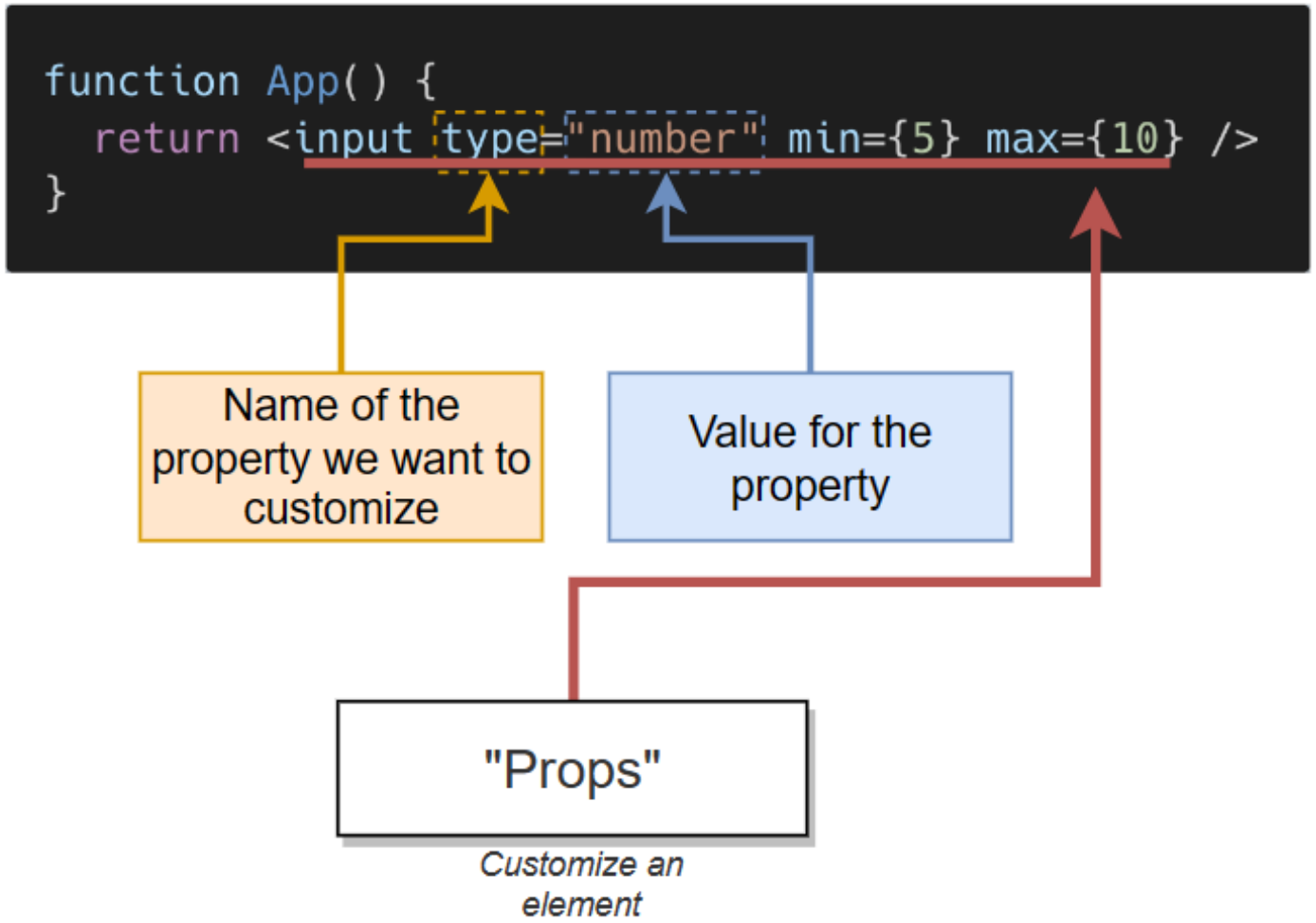
Code to compute values we want to show in our JSX

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}
    />
  );
}
```

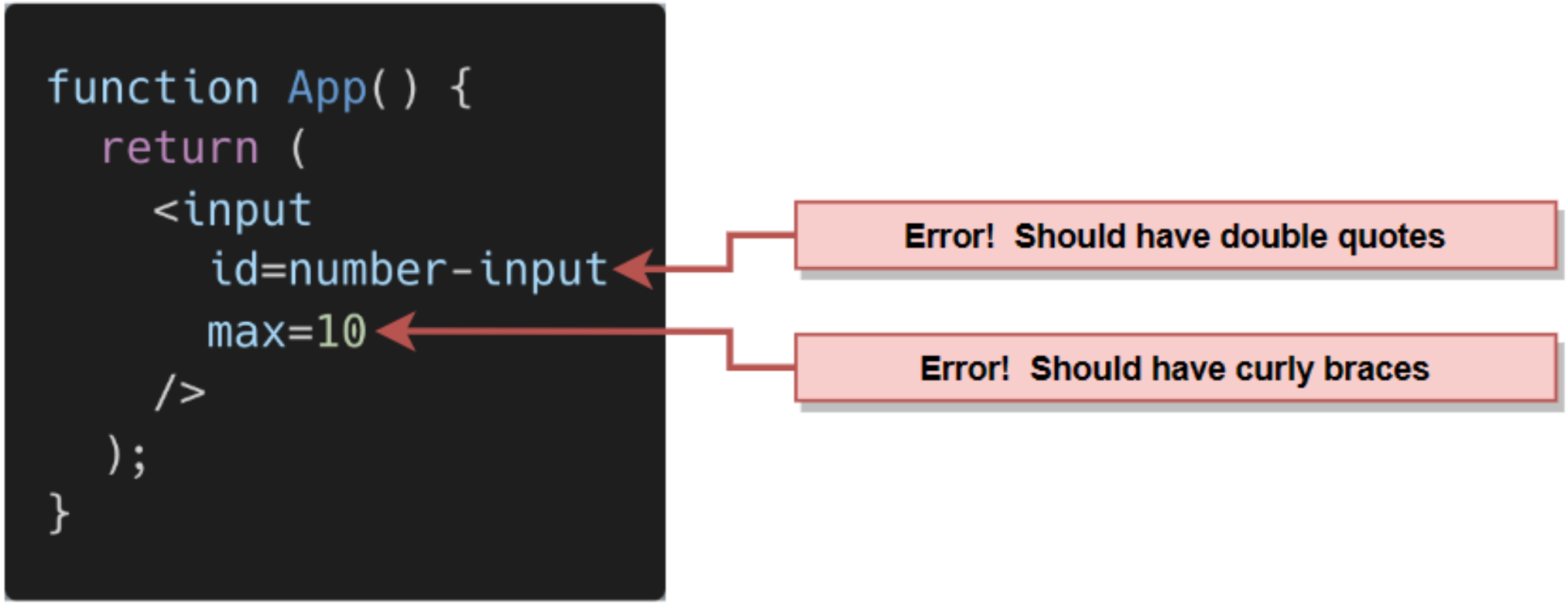
Props don't have to be defined as variables

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



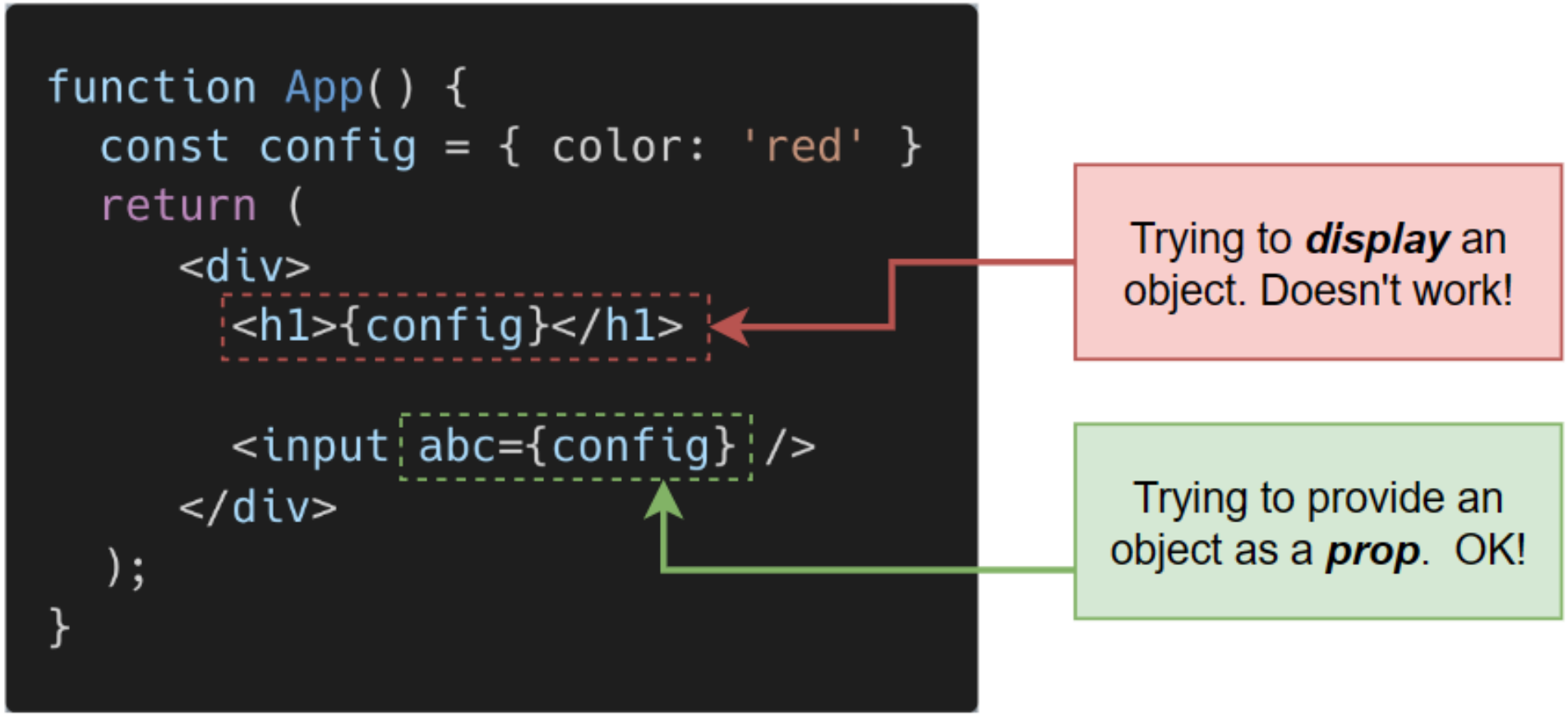
Array, Object as Props

```
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

NOTE

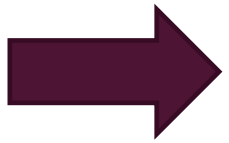

function App() {
const config = { color: 'red' }
return (

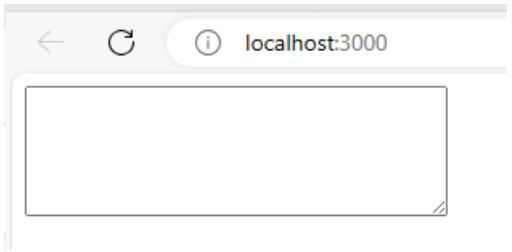
<h1>{config}</h1>
<input abc={config} />

);
}
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

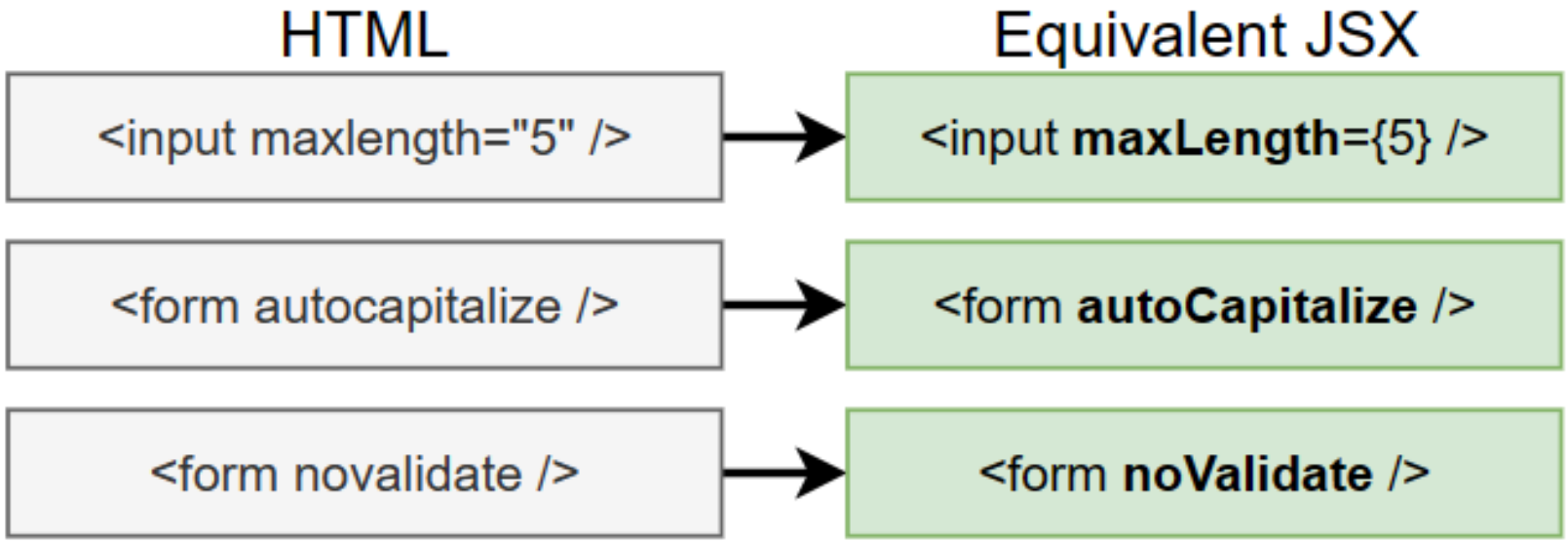
Rules for Converting HTML to JSX

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

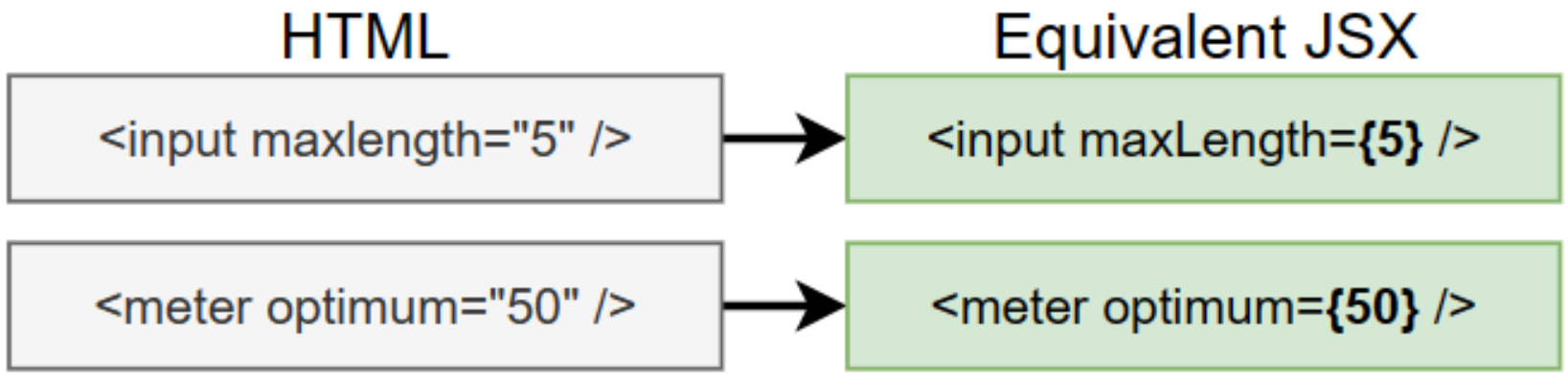
In JSX, all prop names follow camelCase



Rules for Converting HTML to JSX…

Rule #2

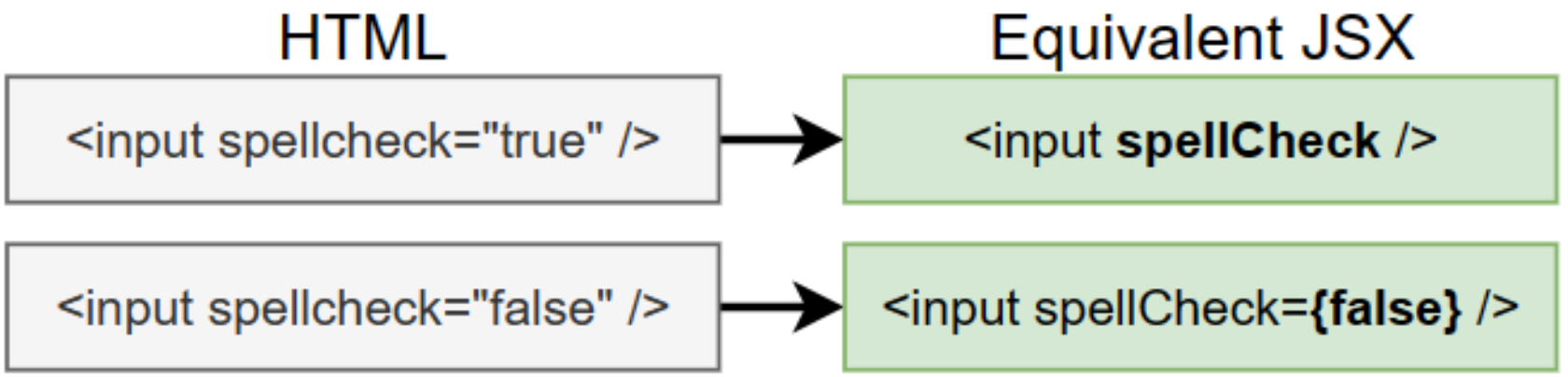
In JSX, attributes meant to be numbers should be provided as numbers with curly braces



Rules for Converting HTML to JSX…

Rule #3

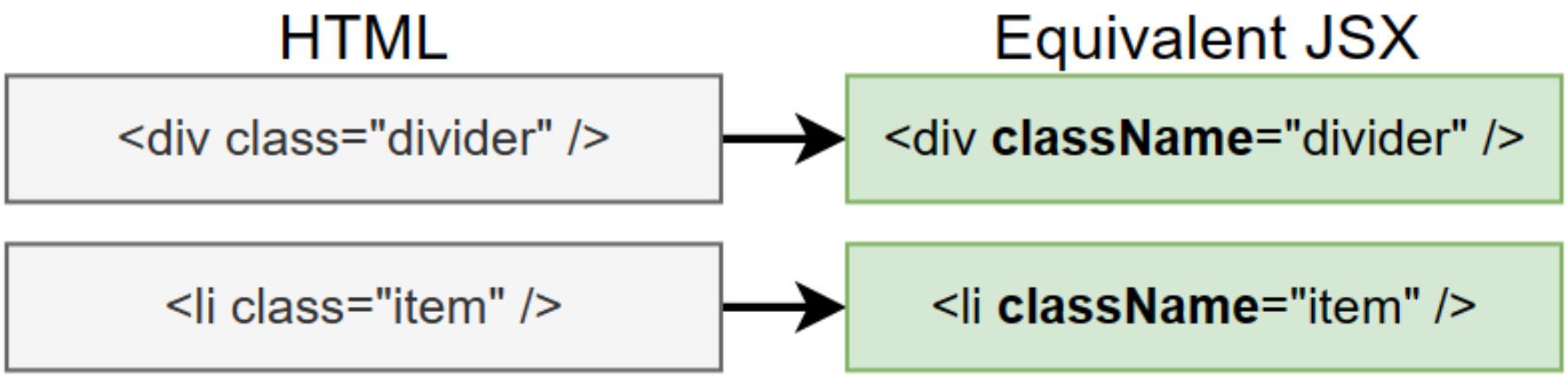
In JSX, boolean 'true' can be written with just the property name. 'False' should be written with curly braces



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

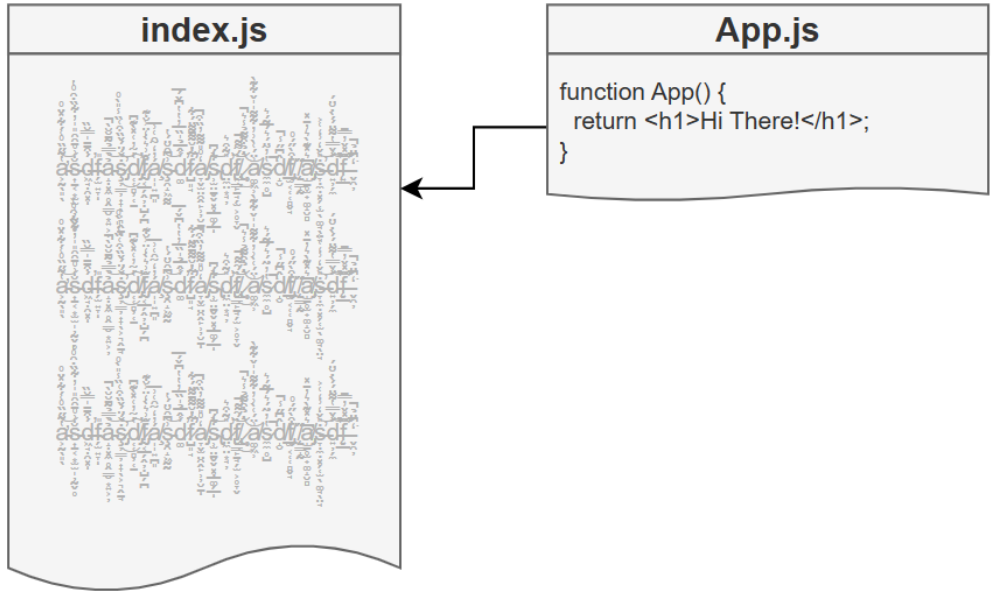
index.js file

1. *Import React and ReactDOM*
2. *Import the App component*
3. *Create a root*
4. *Show the App component*

App.js file

Code to create a component

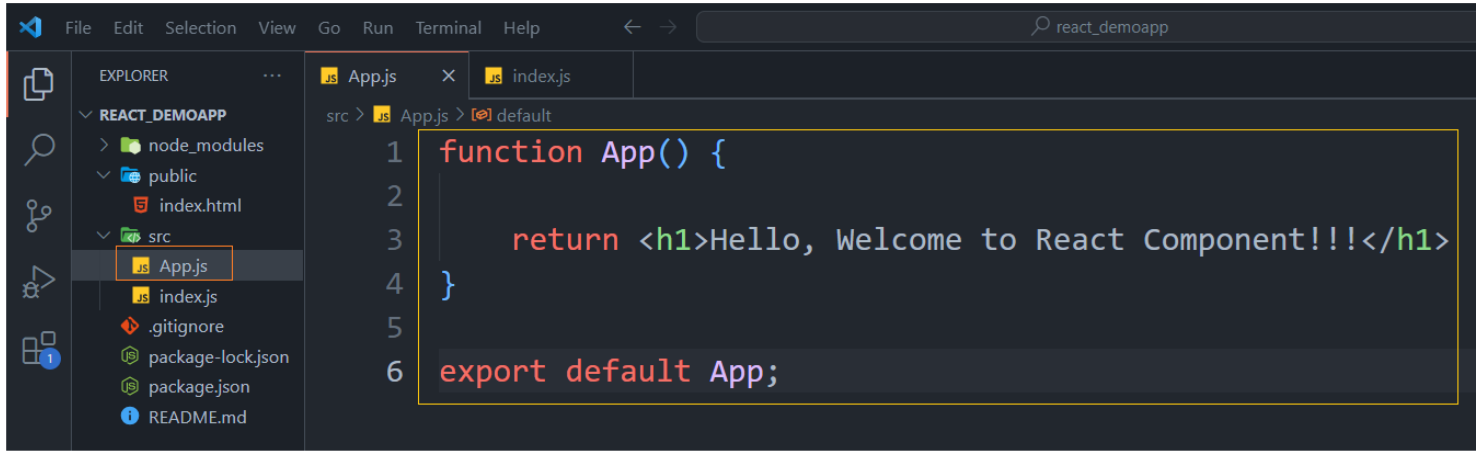
Example

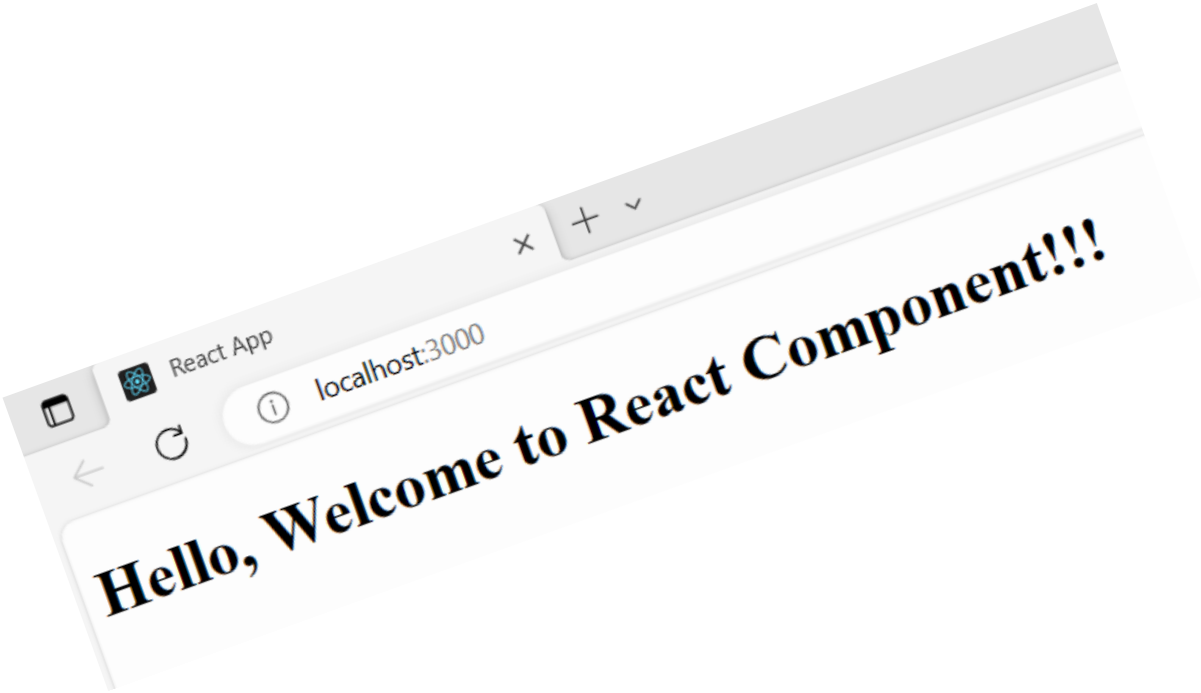


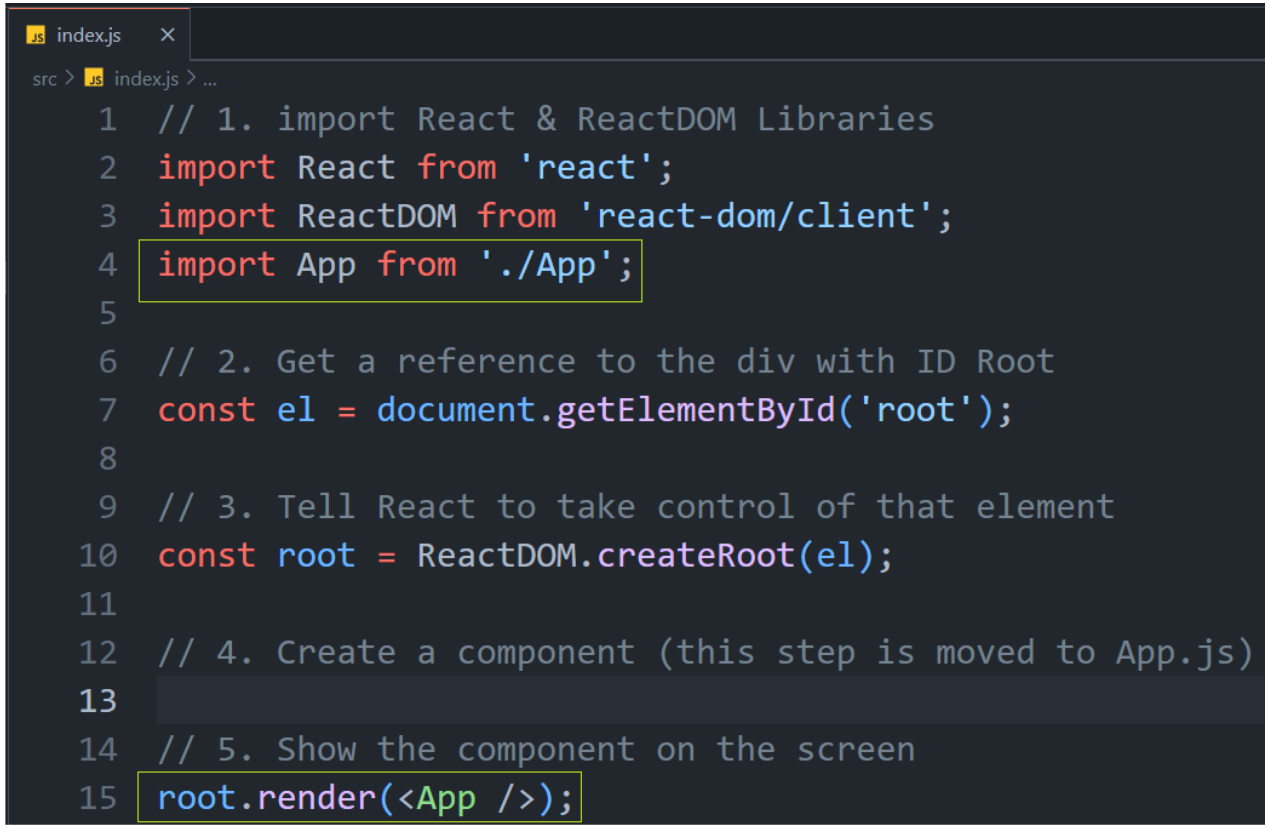
Steps to Create a Components in a separate file

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 <h1>Hi</h1> }`
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



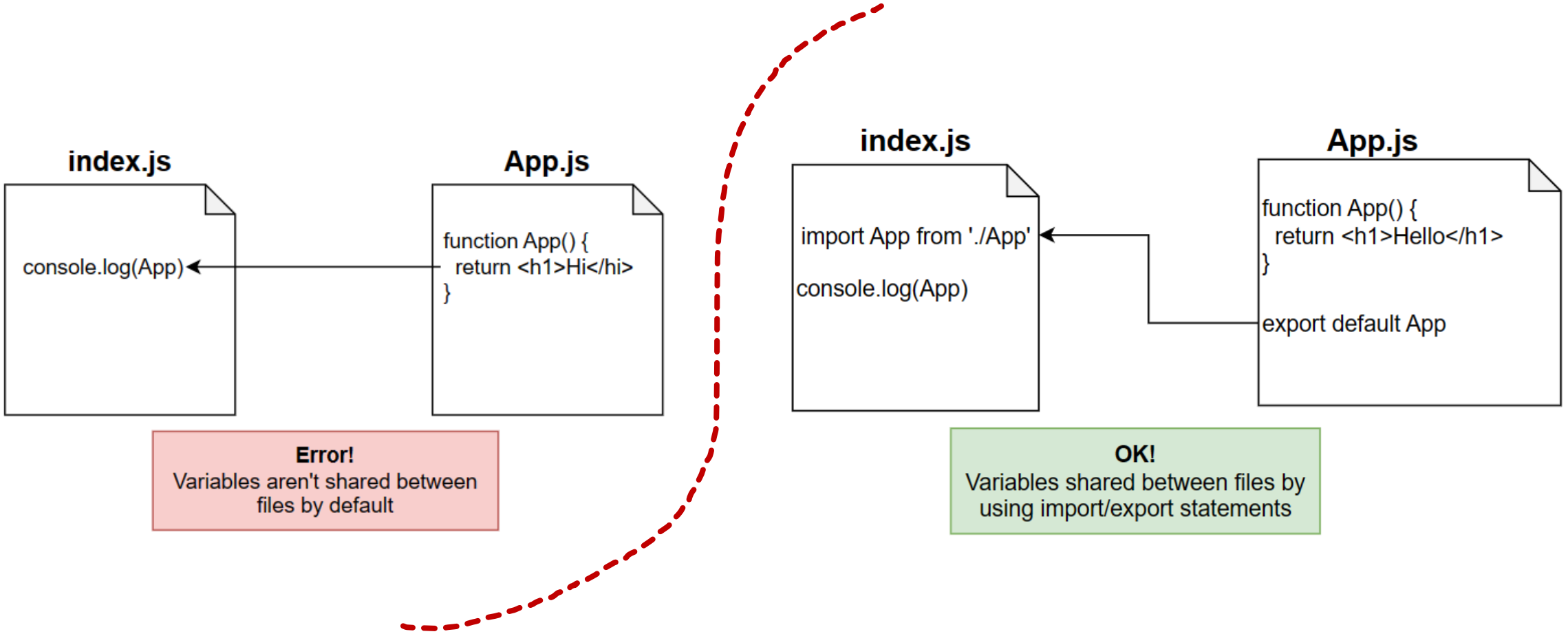




Module System (OR) Import/Exports Statements

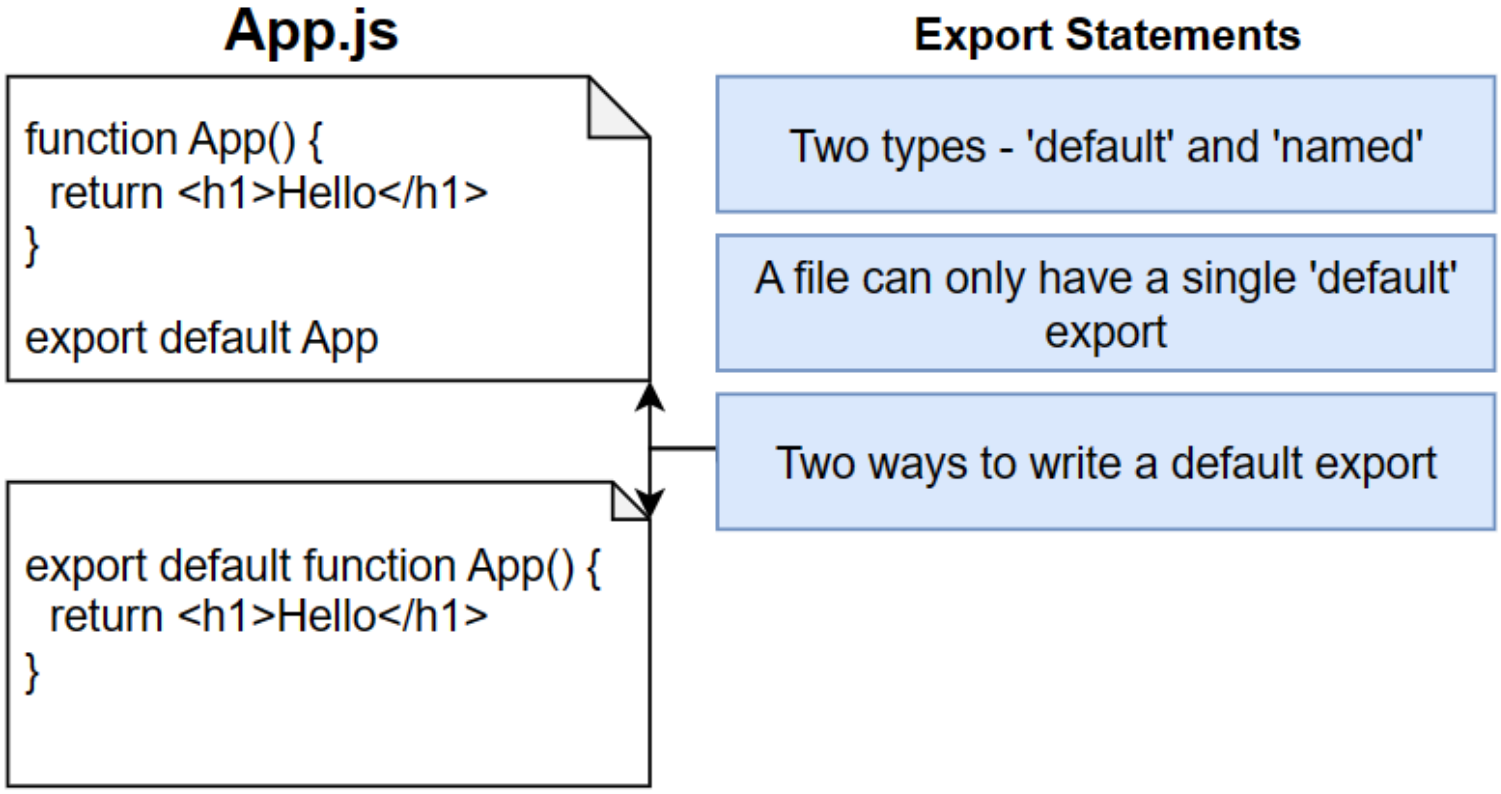
Import/Exports Statements

- 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.

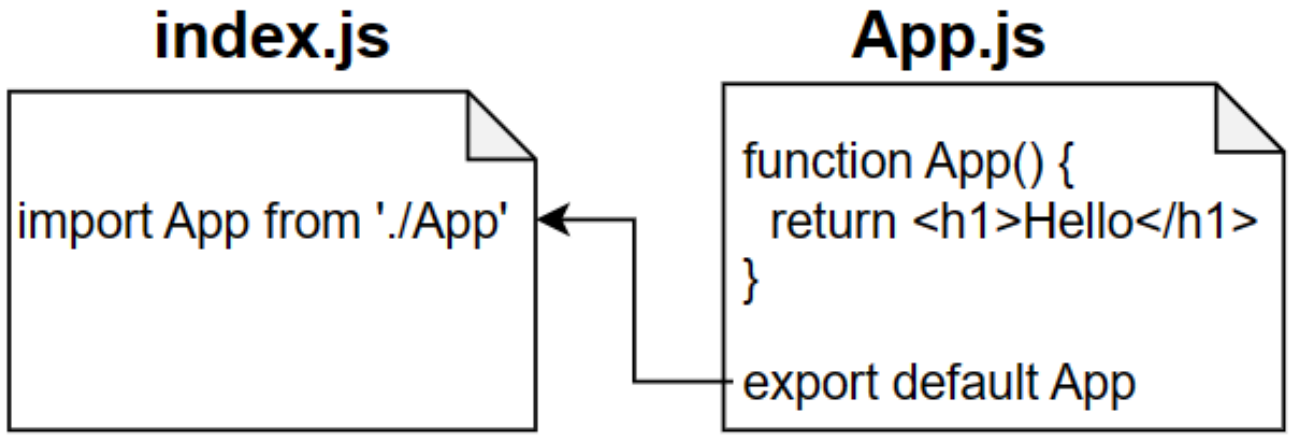


Rules - Exports Statements

- Two kinds of Export statements available
  - ✓ default
  - ✓ named



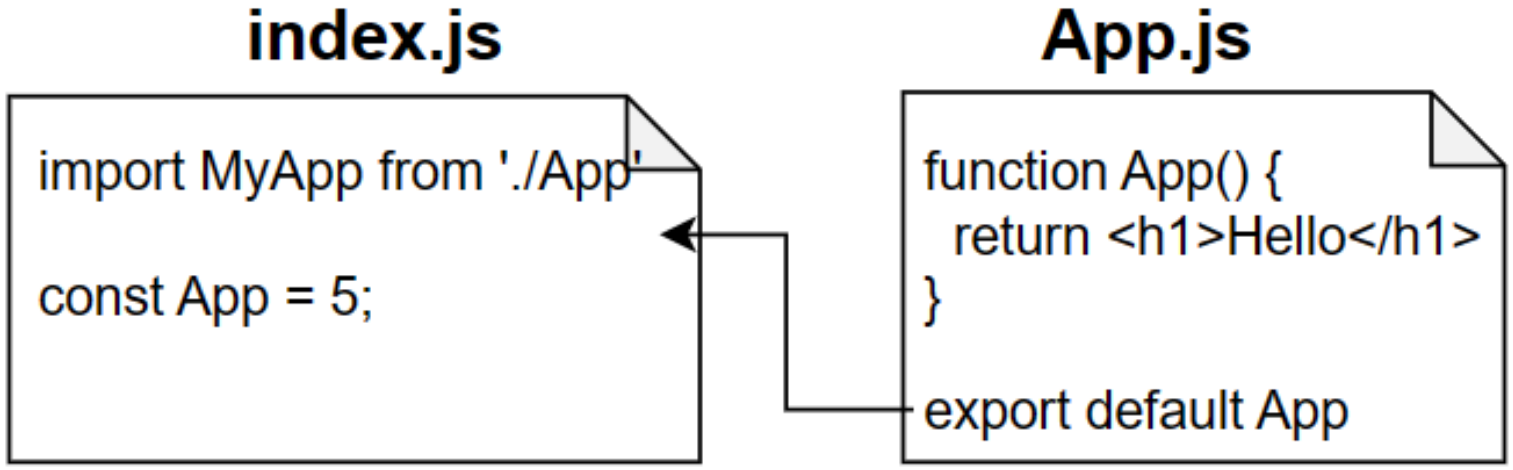
Module System - Import Statements



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

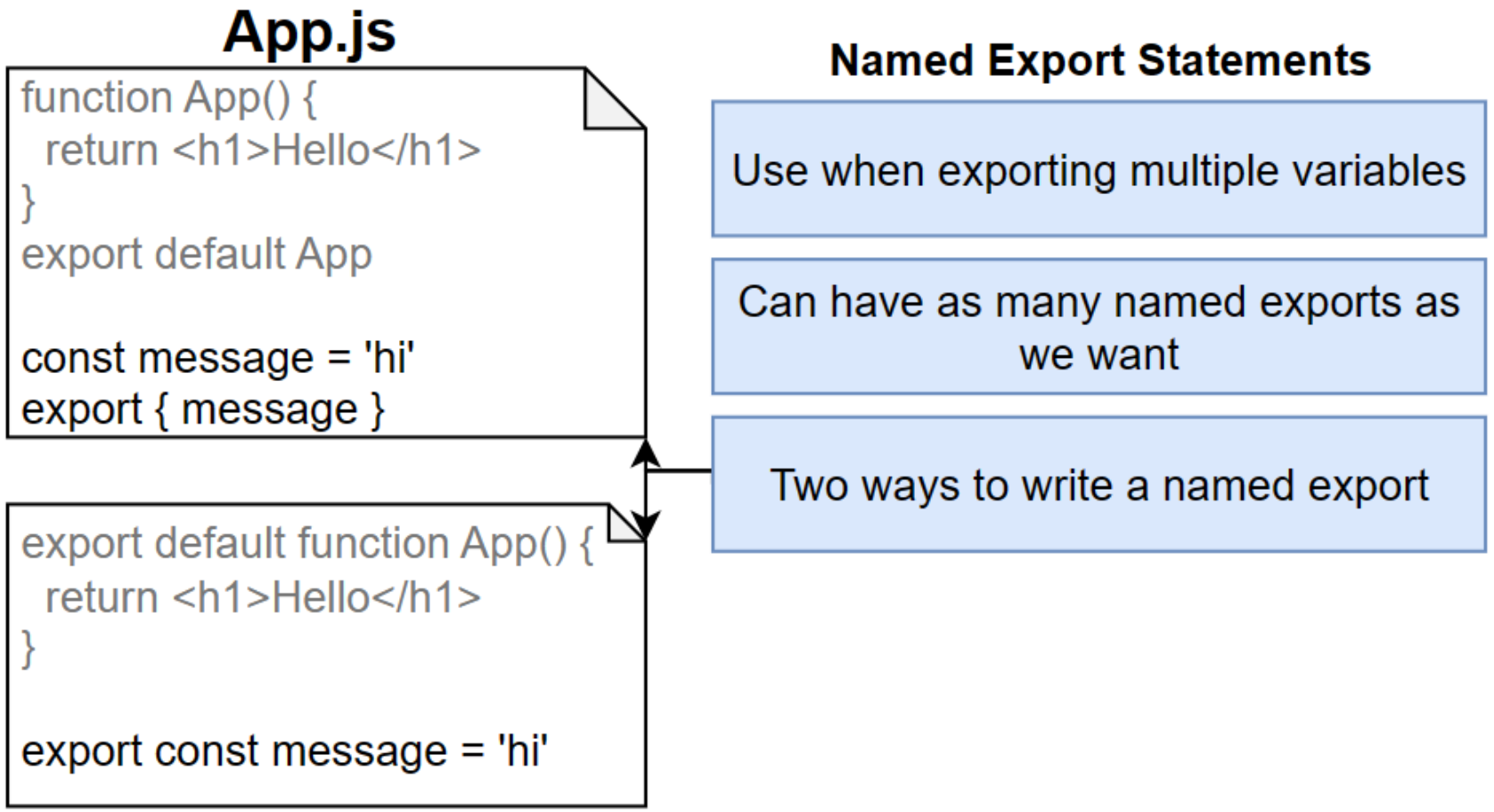


Confusing Thing!
Default exports can be renamed in the *importing* file!!!

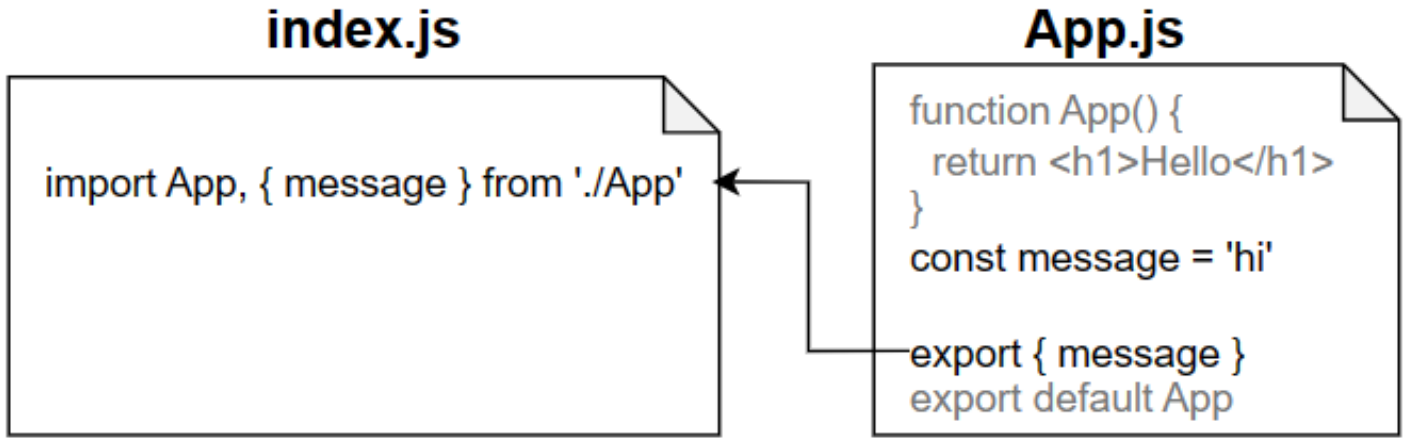
1	Declare a variable called MyApp
2	Find the default export from App.js
3	Assign the default export to MyApp variable

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



Importing - Named exports

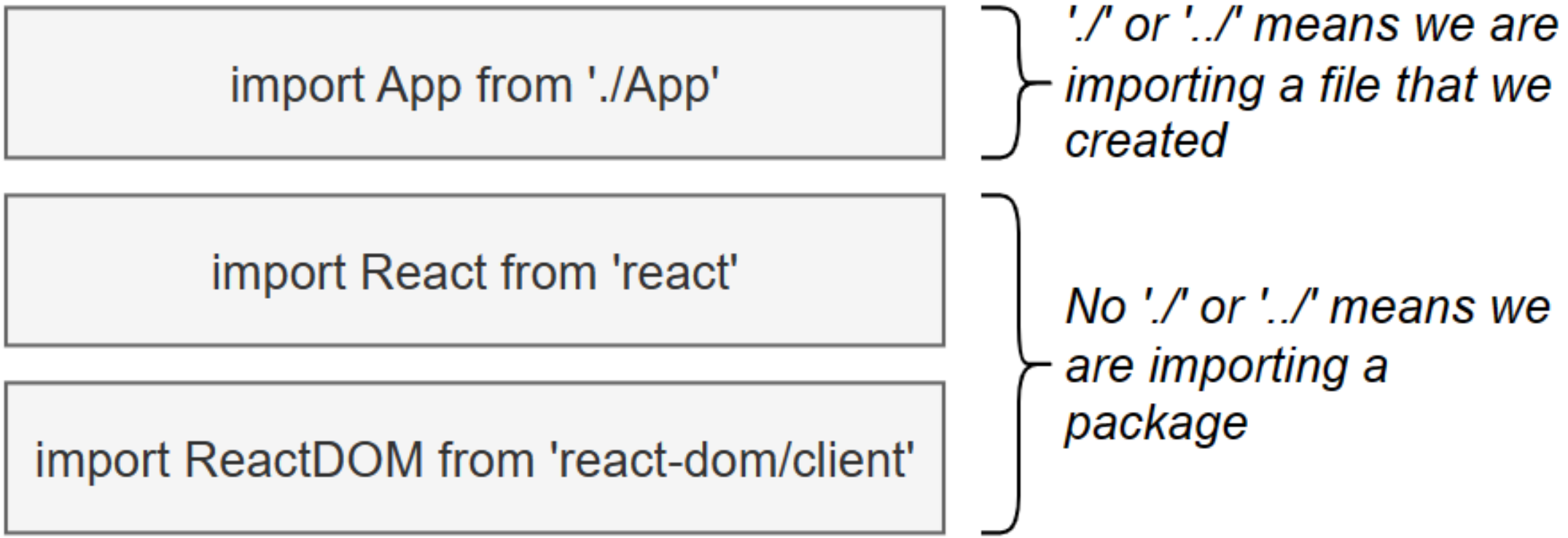


Import Statements

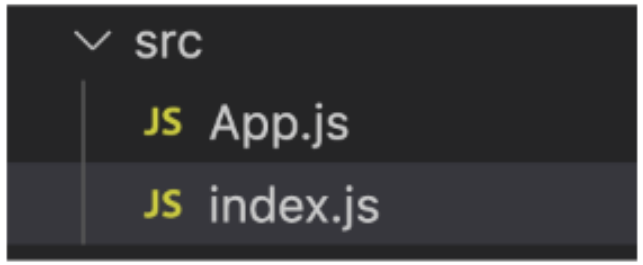
Curly braces indicate we want a 'named' export

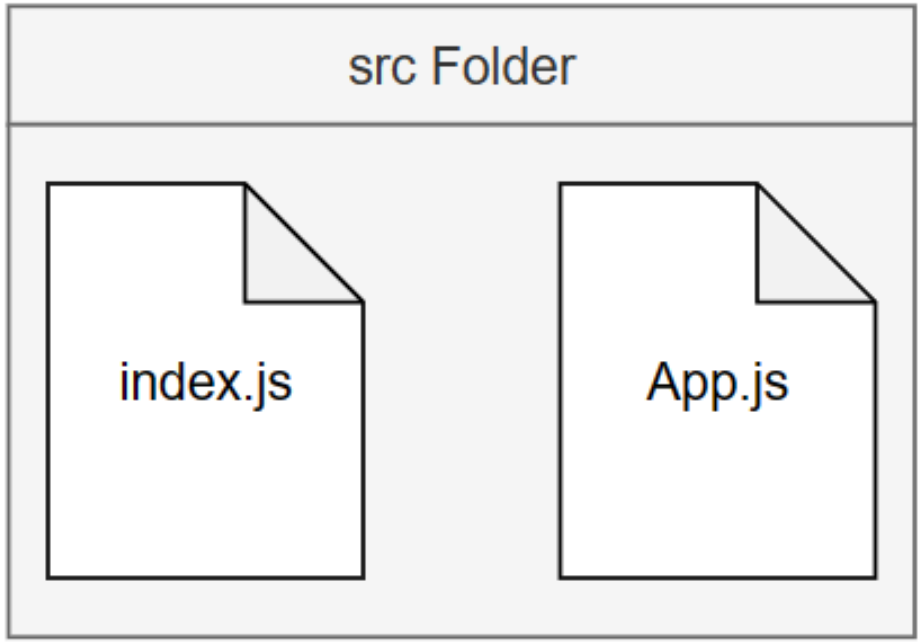
Single import statement can get both default + named exports

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 - 1



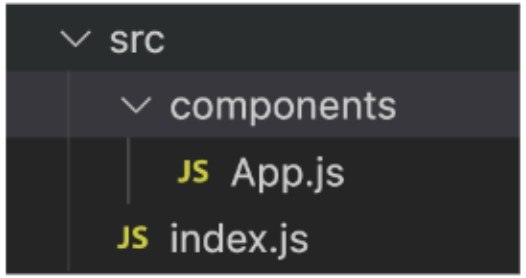


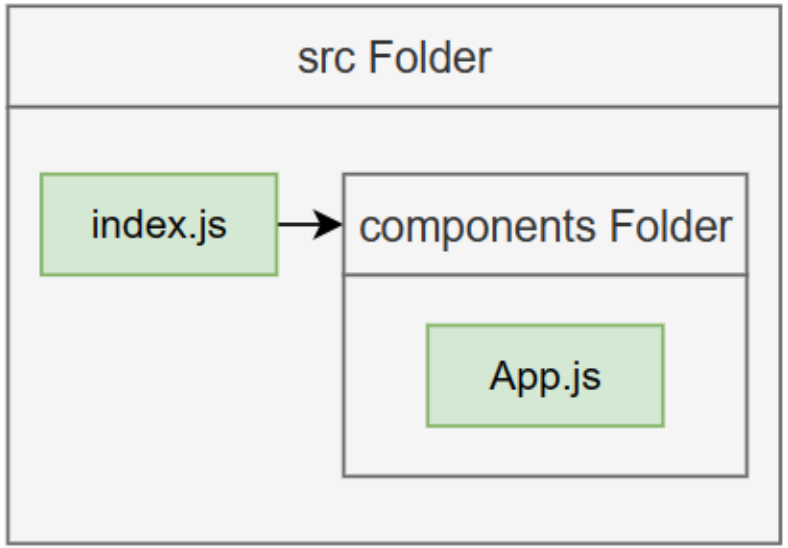
If we are in the index.js file and want to import from App.js....

import App from './App';

NOTE: ./ means same folder/directory

Import Demo - 2

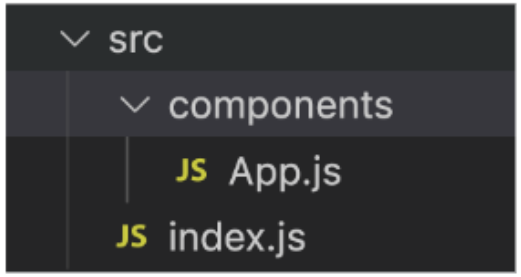


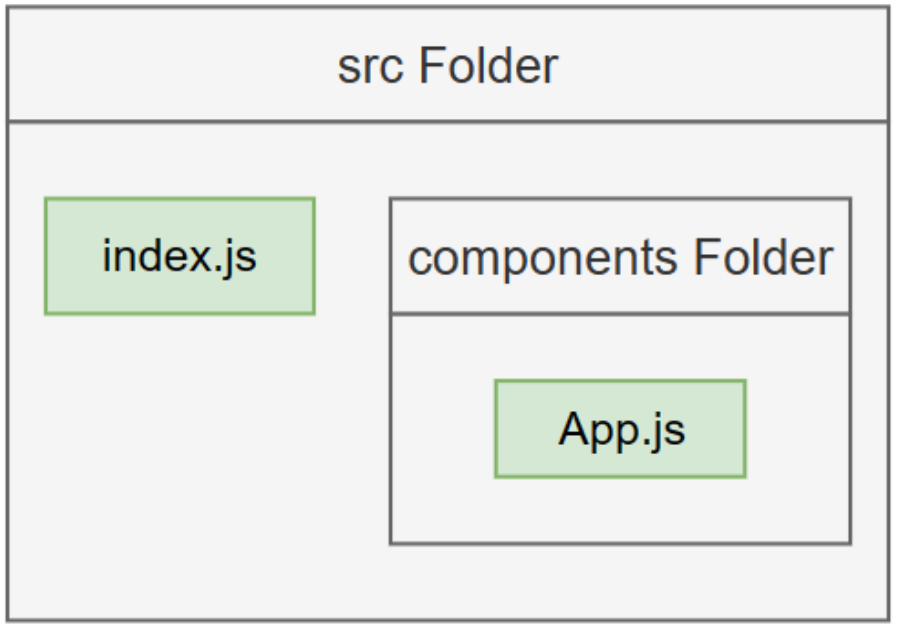


If we are in the index.js file and want to import from App.js....

import App from './components/App';

Import index.js in App.js



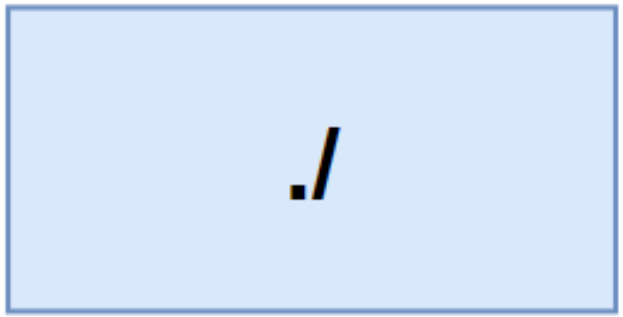


../ means Up one directory

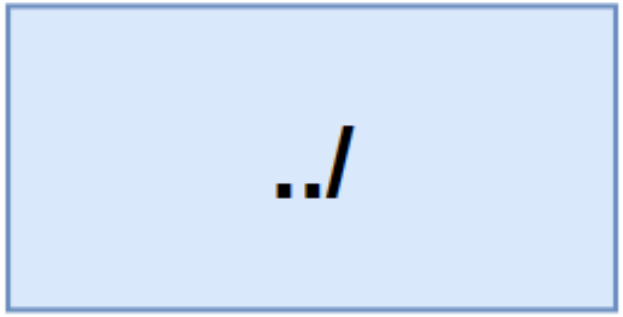
If we are in the App.js file and want to import from index.js....

import index from '../index'

Import Demo - 4

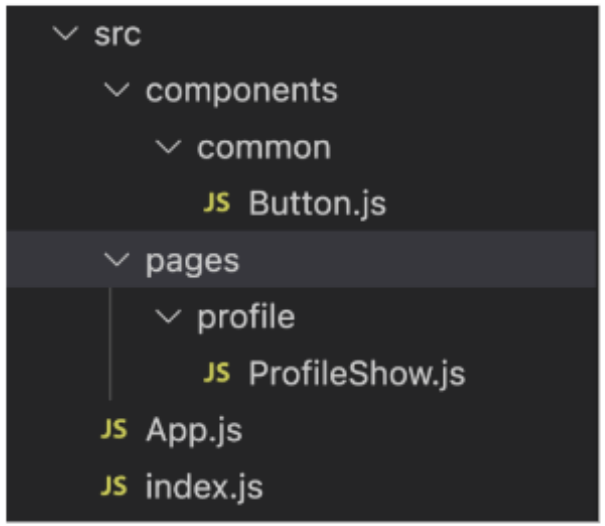


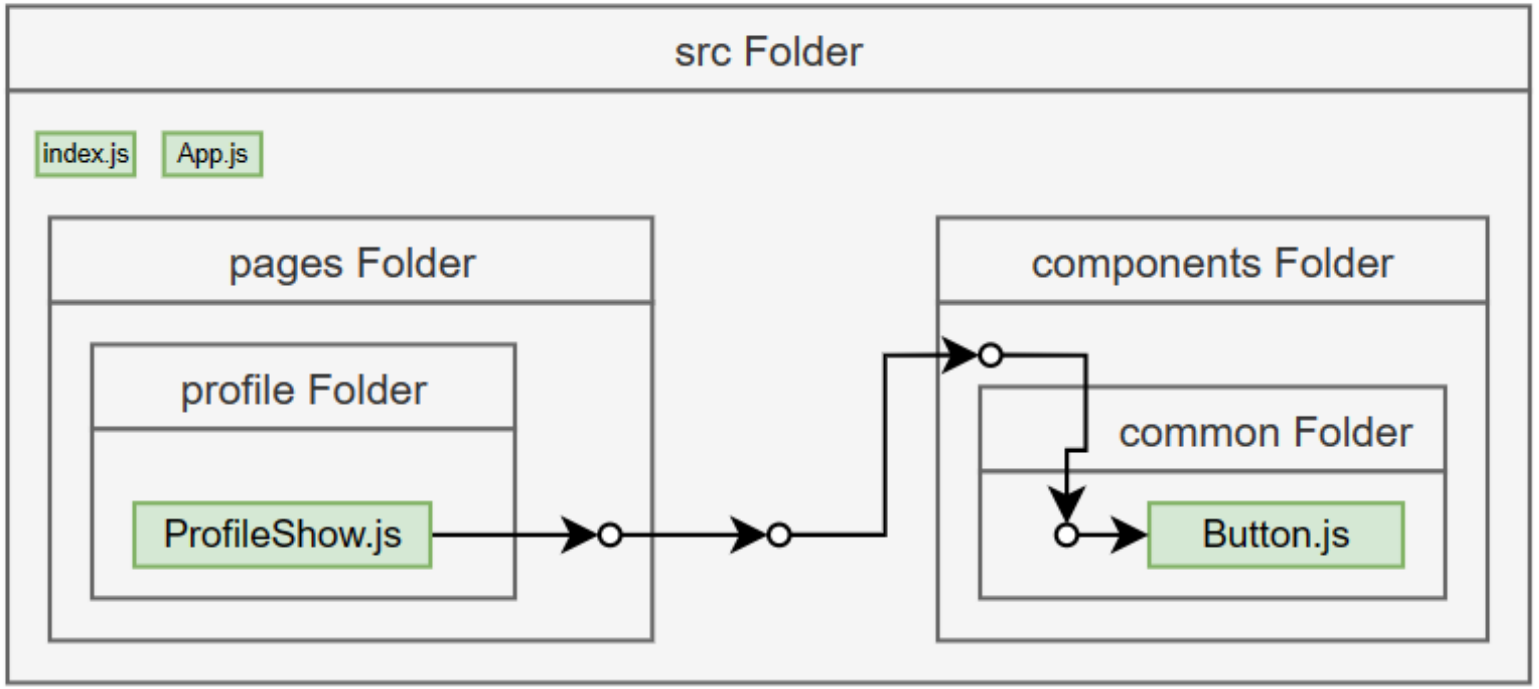
Same directory



Up one directory

Import Demo - 5





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

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

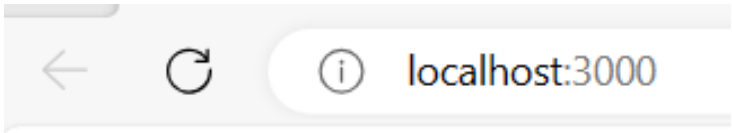
Import Demo - 6

```jsx
function App() {

    return <h1>Hello, Welcome to React Component!!!</h1>
}

const username = 'mdali';
const email = 'mdali@gmail.com';

export { username, email };

export default App;
```

```jsx
// 1. import React & ReactDOM Libraries
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import {username, email} from './App';

```

Import Demo - 7

```jsx
const App = () => {
    const count = 10;
    const name = 'Md Ali';
    const booleanTrue = true;
    const booleanFalse = false;
    const nullValue = null;
    const undefinedValue = undefined;

    return (
      <ul>
        <li>Number: {count}</li>
        <li>String: {name}</li>
        <li>True: {booleanTrue}</li>
        <li>False: {booleanFalse}</li>
        <li>Null: {nullValue}</li>
        <li>Undefined: {undefinedValue}</li>
      </ul>
    );
  };

export default App;
```



Vite

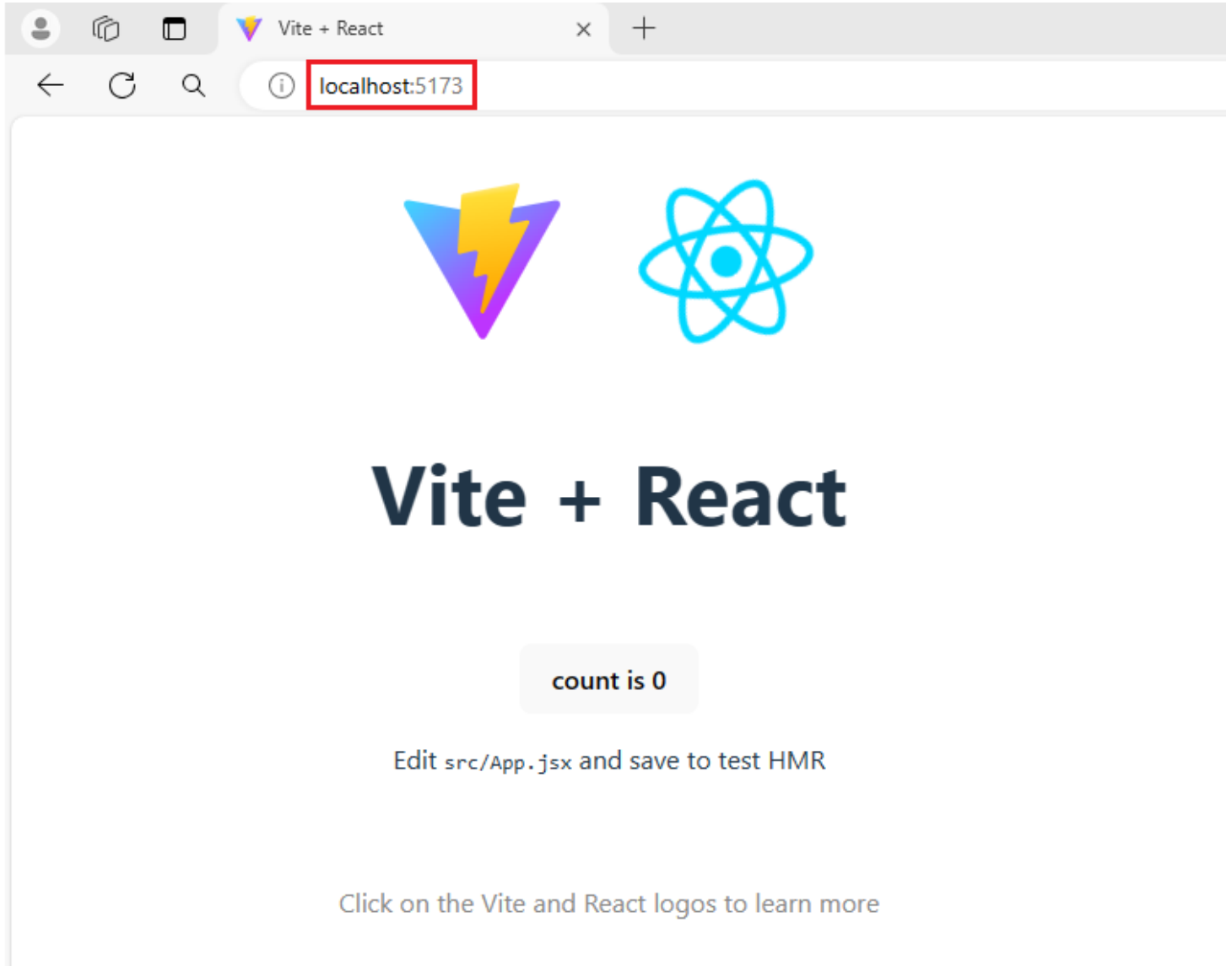
Creating React Application

```
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>
```

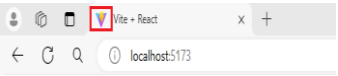
npm run dev

```
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
```


Vite + React
localhost:5173
Vite + React
count is 0
Edit src/App.jsx and save to test HMR
Click on the Vite and React logos to learn more

Understanding Vite-React Application Templated

- **node_module** - is the folder where all our installed packages and library present
- **public** - Contains static assests which will be directly served to the client

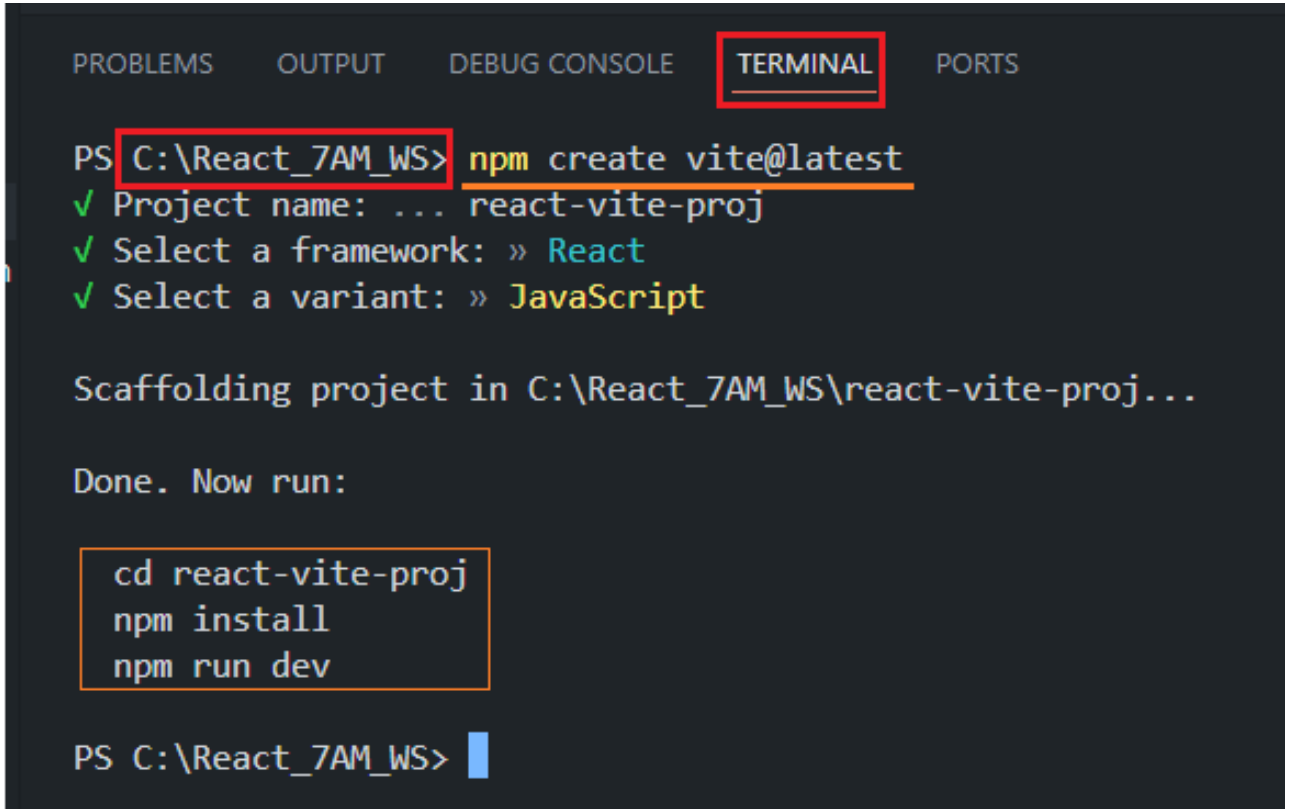


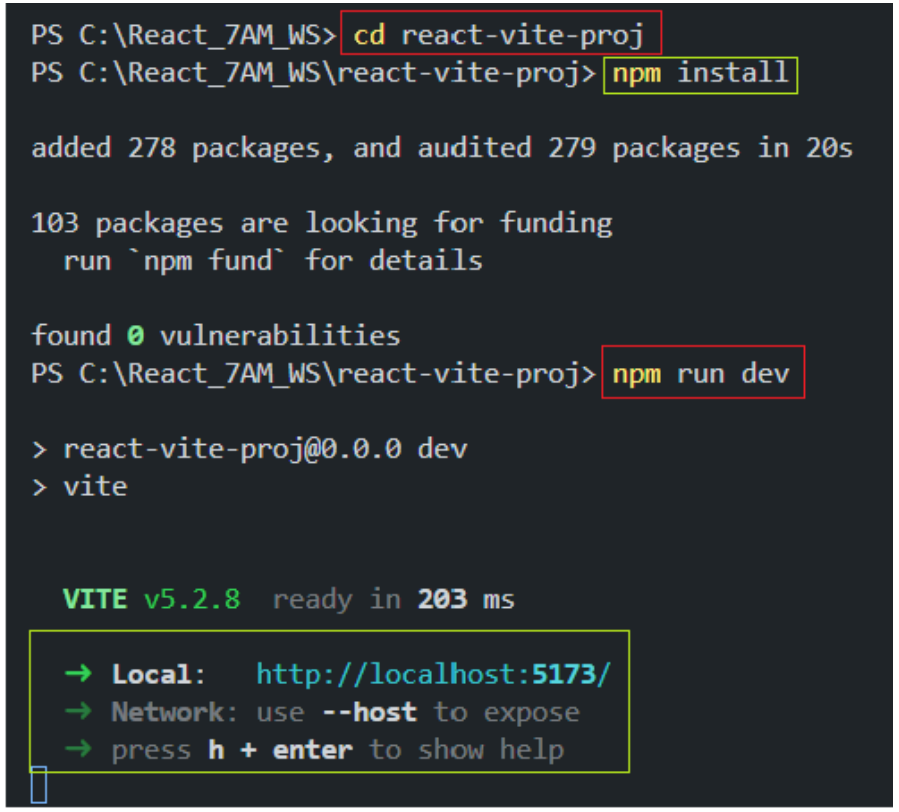
- src
  - **Assets** - we put all our images, icon, fonts used in React Components
  - **App.css** - Used for styling used in App.jsx
  - **App.jsx** - Root component of our application means App.jsx is the starting point of Component Chain. We will call is as "root component". It is similar to **JS**
  - **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

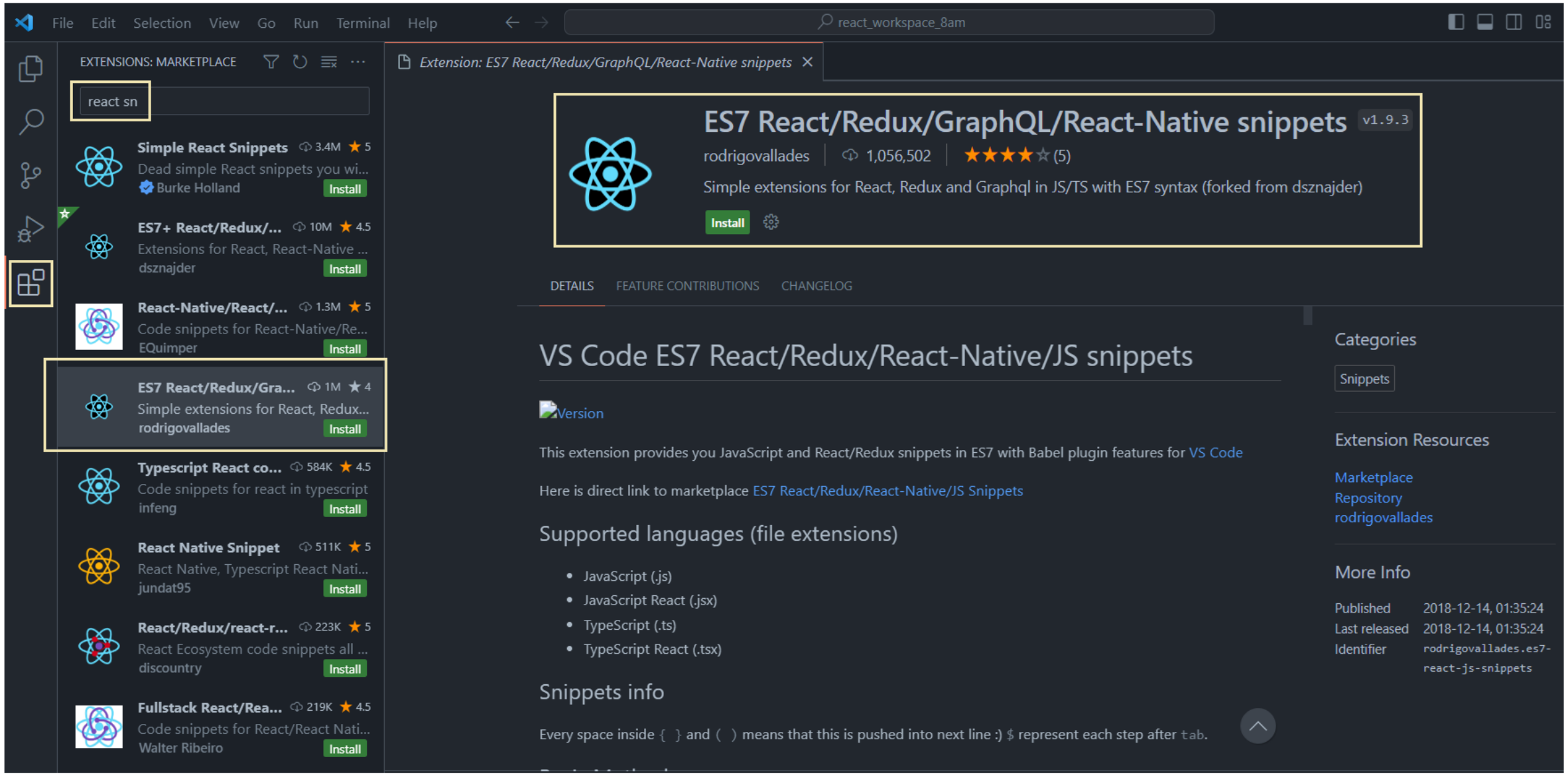




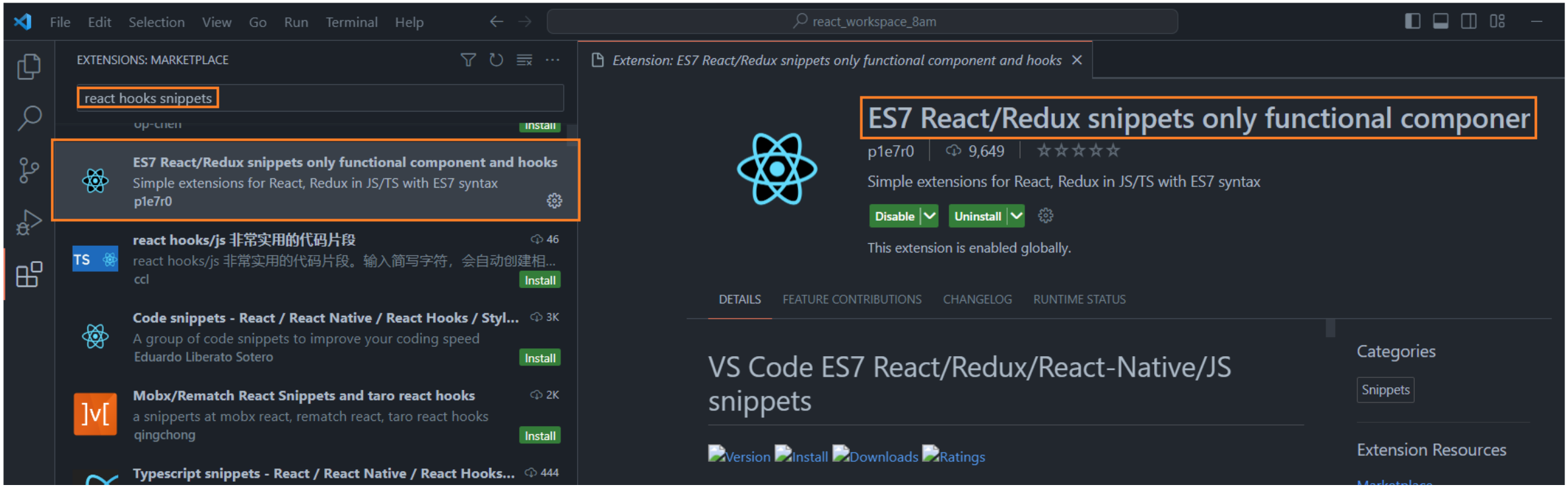
Building First React Component

- 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

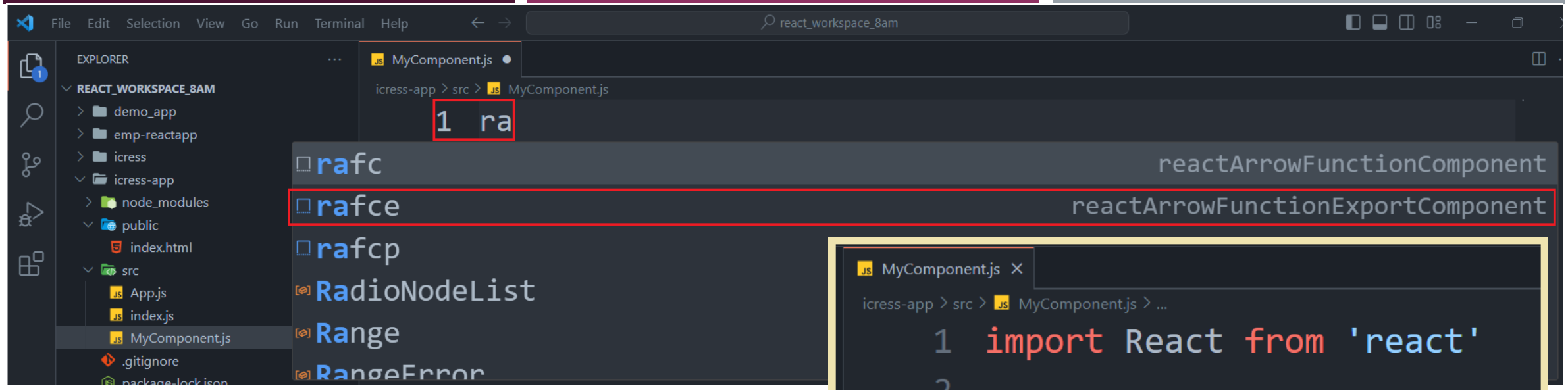
Add React Snippet extension



ES7 React/Redux snippets only functional component and hooks



Create Arrow Function Component & Export the Component



- ✓ **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)
```

MyComponent.js

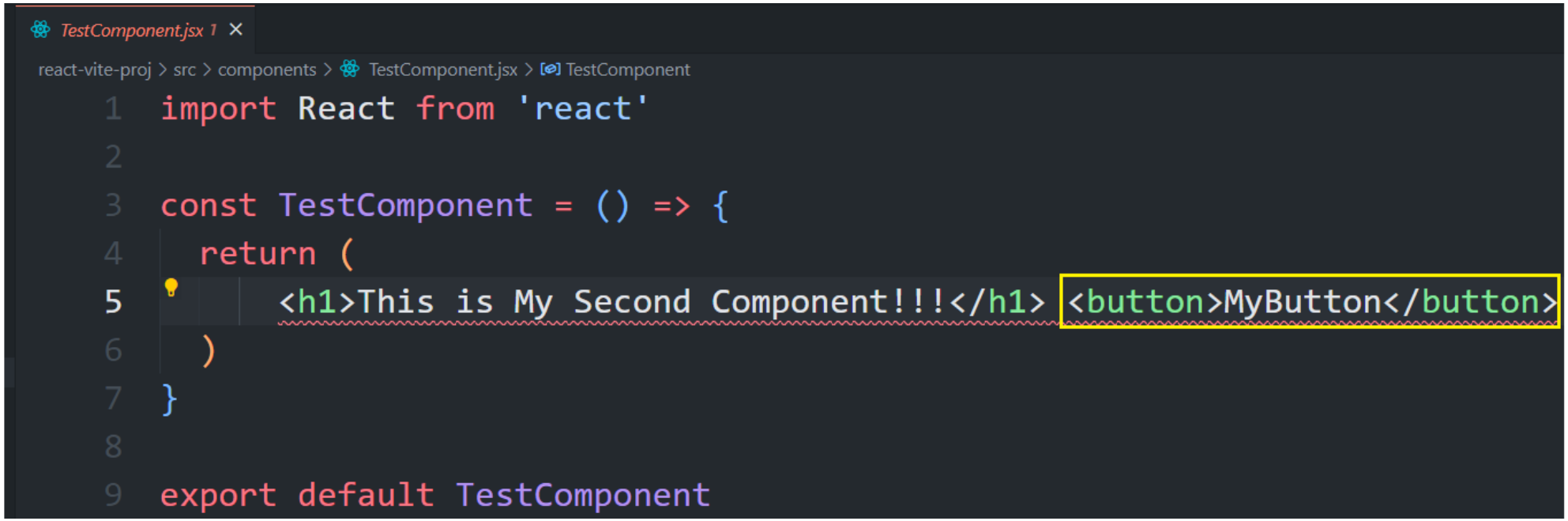
icress-app > src > MyComponent.js > ...

```
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

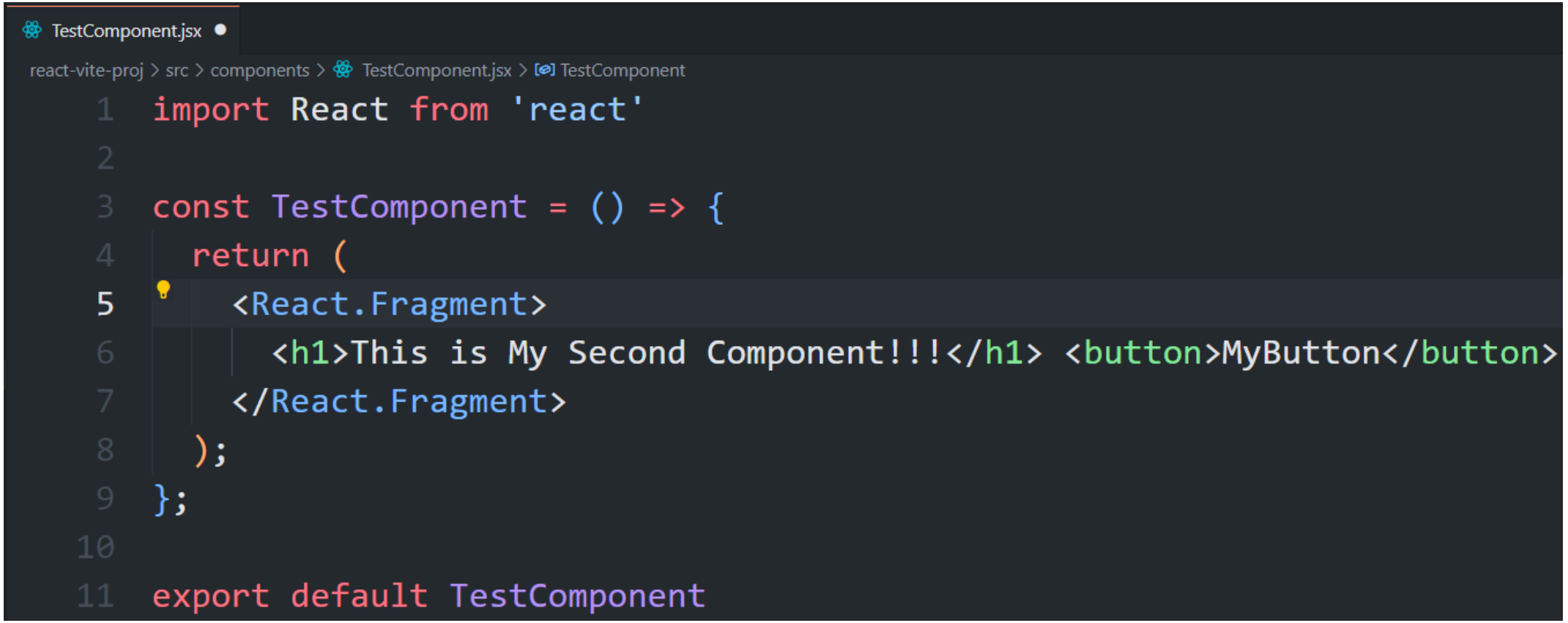


- **NOTE:** In JSX we add elements under one parent element

TestComponent.jsx

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

```jsx
import React from 'react'

const TestComponent = () => {
  return (
    <div>
      <h1>This is My Second Component!!!</h1> <button>MyButton</button>
    </div>
  );
};

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

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

```jsx
import React from 'react'

const TestComponent = () => {
  return (
    <>
      <h1>This is My Second Component!!!</h1> <button>MyButton</button>
    </>
  );
};

export default TestComponent
```

JavaScript expression in JSX

TestComponent.jsx

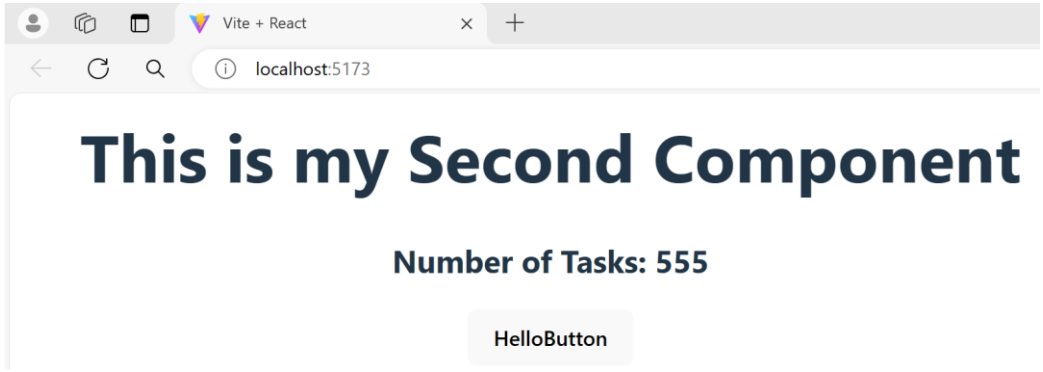
react-vite-proj2 > src > components > TestComponent.jsx > TestComponent

```jsx
import React from 'react'

const TestComponent = () => {
  const tasks = 555;
  return (
    <>
      <h1>This is my Second Component</h1>
      <h2>Number of Tasks: {tasks}</h2>
      <button>HelloButton</button>
    </>
  );
};

export default TestComponent
```

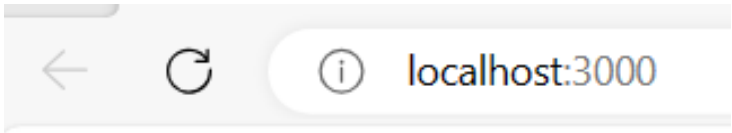
main.jsx

react-vite-proj2 > src > main.jsx

```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import MyComponent from './components/MyComponent.jsx'
import TestComponent from './components/TestComponent.jsx'


ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    {/*<App />
    <MyComponent />*/}
    <TestComponent />
  </React.StrictMode>,
)
```



```jsx
const App = () => {
    const count = 10;
    const name = 'Md Ali';
    const booleanTrue = true;
    const booleanFalse = false;
    const nullValue = null;
    const undefinedValue = undefined;

    return (
      <ul>
        <li>Number: {count}</li>
        <li>String: {name}</li>
        <li>True: {booleanTrue}</li>
        <li>False: {booleanFalse}</li>
        <li>Null: {nullValue}</li>
        <li>Undefined: {undefinedValue}</li>
      </ul>
    );
  };

export default App;
```



Calling function from JSX Expression

TestComponent.jsx

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

```jsx
import React from 'react'

const TestComponent = () => {
  const tasks = 555;

  const countTasks = () => {
    return tasks + 45;
  }

  return (
    <>
      <h1>This is my Second Component</h1>
      <h2>Number of Tasks: {tasks}</h2>
      <h2>countTasks Fun called: {countTasks()}</h2>
      <button>HelloButton</button>
    </>
  );
};

export default TestComponent
```

Calling function from JSX Expression…

TestComponent.jsx

react-vite-proj2 > src > components > TestComponent.jsx > TestComponent > countTasks

```jsx
import React from 'react'

const TestComponent = () => {
  const tasks = 555;

  const countTasks = () => {
    if(tasks === 0) {
      return "Sorry, No tasks available!!!";
    } else {
      return "Number of Tasks = " + tasks;
    }

  }
  return (
    <>
      <h1>This is my Second Component</h1>
      <h2>Number of Tasks: {tasks}</h2>
      <h2>countTasks Fun called: {countTasks()}</h2>
      <button>HelloButton</button>
    </>
  );
};

export default TestComponent
```

Calling function from JSX Expression…

TestComponent.jsx

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

```jsx
import React from 'react'

const TestComponent = () => {
  const tasks = 555;

  const countTasks = () => {
    return tasks === 0? "Sorry, No tasks available!!!": `Number of Tasks = ${tasks}`;
  }

  return (
    <>
      <h1>This is my Second Component</h1>
      <h2>Number of Tasks: {tasks}</h2>
      <h2>countTasks Fun called: {countTasks()}</h2>
      <button>HelloButton</button>
    </>
  );
};

export default TestComponent
```

Events In React

Handling events in React is similar to handling events in HTML

HandlingEvent.jsx

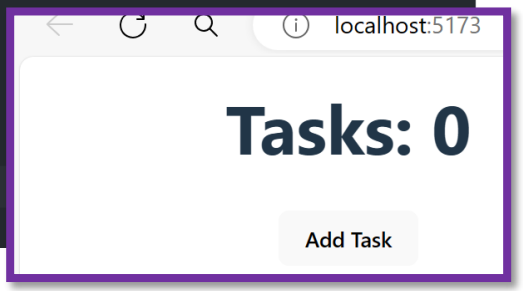
react-vite-proj2 > src > components > HandlingEvent.jsx > ...

```jsx
import React from 'react'

const HandlingEvent = () => {
    const tasks = 0;

    const handleClick = () => {
        console.log("Add Task");
    }

    return (
        <div>
        <h1>Tasks: {tasks}</h1>
        <button onClick={handleClick}>Add Task</button>
        </div>
    );
};

export default HandlingEvent
```

main.jsx

react-vite-proj2 > src > main.jsx

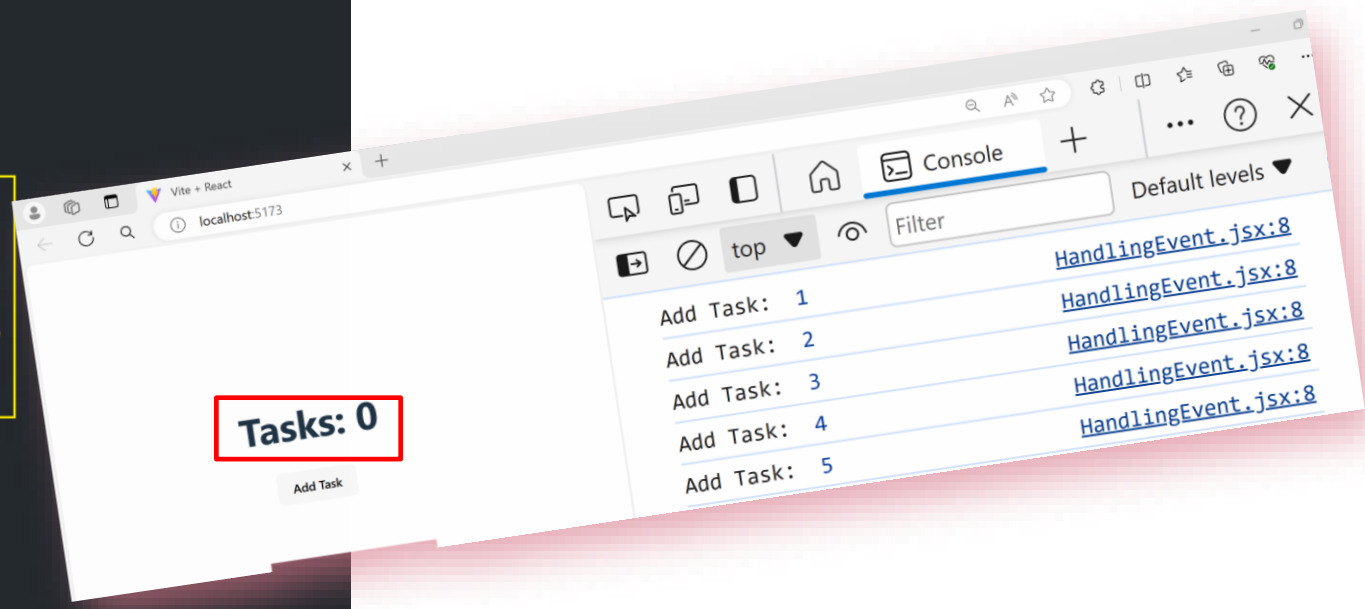
```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import MyComponent from './components/MyComponent.jsx'
import TestComponent from './components/TestComponent.jsx'
import HandlingEvent from './components/HandlingEvent.jsx'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    {/*<App />
    <MyComponent />
    <TestComponent />*/}
    <HandlingEvent />
  </React.StrictMode>,
)
```



HandlingEvent.jsx

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

```jsx
import React from 'react'

const HandlingEvent = () => {
    let tasks = 0;

    const handleClick = () => {
        tasks = tasks + 1;
        console.log("Add Task: ", tasks);
    }

    return (
        <div>
        <h1>Tasks: {tasks}</h1>
        <button onClick={handleClick}>Add Task</button>
        </div>
    );
};

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

```jsx
import React, {useState} from 'react'

const HandlingEvent = () => {
    let tasks = 0;

    const countArray = useState(0); //useState(0) returns array
    console.log("UseState hook = ", countArray);

    const handleClick = () => {
        tasks = tasks + 1;
        console.log("Add Task: ", tasks);
    }

    return (
        <div>
        <h1>Tasks: {tasks}</h1>
        <button onClick={handleClick}>Add Task</button>
        </div>
    );
};

export default HandlingEvent
```

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

HandlingEvent.jsx:7

HandlingEvent.jsx

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

```jsx
import React, {useState} from 'react'

const HandlingEvent = () => {
    let tasks = 0;

    const countArray = useState(0); //useState(0) returns array
    console.log("UseState hook = ", countArray);
    const countValue = countArray[0];
    const setCountValue = countArray[1];//Function used to update countValue

    const handleClick = () => {
        tasks = tasks + 1;
        setCountValue(countValue + 1);
        console.log("Add Task: ", tasks);
    }

    return (
        <div>
        <h1>Tasks: {tasks}</h1>
        <h1>Tasks using Hooks: {countValue}</h1>
        <button onClick={handleClick}>Add Task</button>
        </div>
    );
};

export default HandlingEvent
```

Tasks: 0

Tasks using Hooks: 4

Add Task

```
Add Task:  1                                   HandlingEvent.jsx:14
UseState hook =  ▶ (2) [4, ƒ]                  HandlingEvent.jsx:7
UseState hook =  ▶ (2) [4, ƒ]                  HandlingEvent.jsx:7
```

HandlingEvent.jsx

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

```jsx
import React, {useState} from 'react'

const HandlingEvent = () => {

    //const countArray = useState(0); //useState(0) returns array
    //const countValue = countArray[0];
    //const setCountValue = countArray[1];//Function used to update countValue
    const [countValue, setCountValue] = useState(0); //Array destructuring

    const handleClick = () => {
        setCountValue(countValue + 1);
    }

    return (
        <div>
            <h1>Tasks using Hooks: {countValue}</h1>
            <button onClick={handleClick}>Add Task</button>
        </div>
    );
};

export default HandlingEvent
```

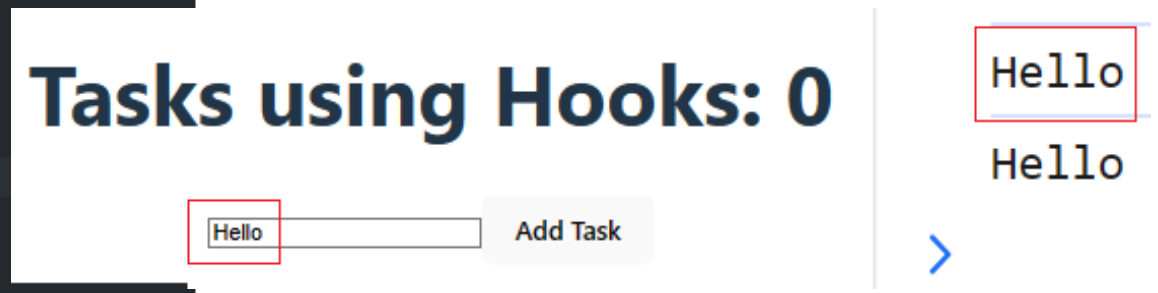
1. State Variable
2. Function to update State Variable

Handling user inputs

HandlingEvent.jsx

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

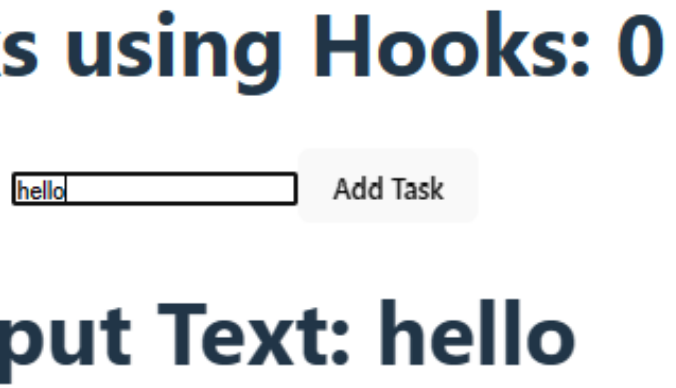
```jsx
import React, {useState} from 'react'

const HandlingEvent = () => {
    const [countValue, setCountValue] = useState(0); //Array destructuring

    const handleClick = () => {
        setCountValue(countValue + 1);
    };

    const handleChange = (event) => {
        console.log(event.target.value);
    };

    return (
        <div>
            <h1>Tasks using Hooks: {countValue}</h1>
            <input type="text" onChange={handleChange} />
            <button onClick={handleClick}>Add Task</button>
        </div>
    );
};

export default HandlingEvent
```



HandlingEvent.jsx

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

```jsx
import React, {useState} from 'react'

const HandlingEvent = () => {
    const [countValue, setCountValue] = useState(0); //Array destructuring
    const [inputText, setInputText] = useState("");

    const handleClick = () => {
        setCountValue(countValue + 1);
    };

    const handleChange = (event) => {
        //console.log(event.target.value);
        setInputText(event.target.value);
    };

    return (
        <div>
            <h1>Tasks using Hooks: {countValue}</h1>
            <input type="text" onChange={handleChange} />
            <button onClick={handleClick}>Add Task</button>

            <h1>Input Text: {inputText}</h1>
        </div>
    );
};

export default HandlingEvent
```



Mapping List / Rendering List in JSX

MappingList.jsx

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

```jsx
import React from 'react';

const MappingList = () => {
    let names = ["Hari", "Giri", "Suri", "Mari", "Siri"];

    return (
        <div>
            <ul>
                for(let name of names) {
                    <li>name</li>
                }
            </ul>
        </div>
    );
};

export default MappingList;
```

- for(let name of names)
- name

Not working

MappingList.jsx

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

```jsx
import React from 'react';

const MappingList = () => {
    let names = ["Hari", "Giri", "Suri", "Mari", "Siri"];

    return (
        <div>
            <ul>
                {
                    names.map(name => <li key={name}>{name}</li>)
                }
            </ul>
        </div>
    );
};

export default MappingList;
```

JSX Expression

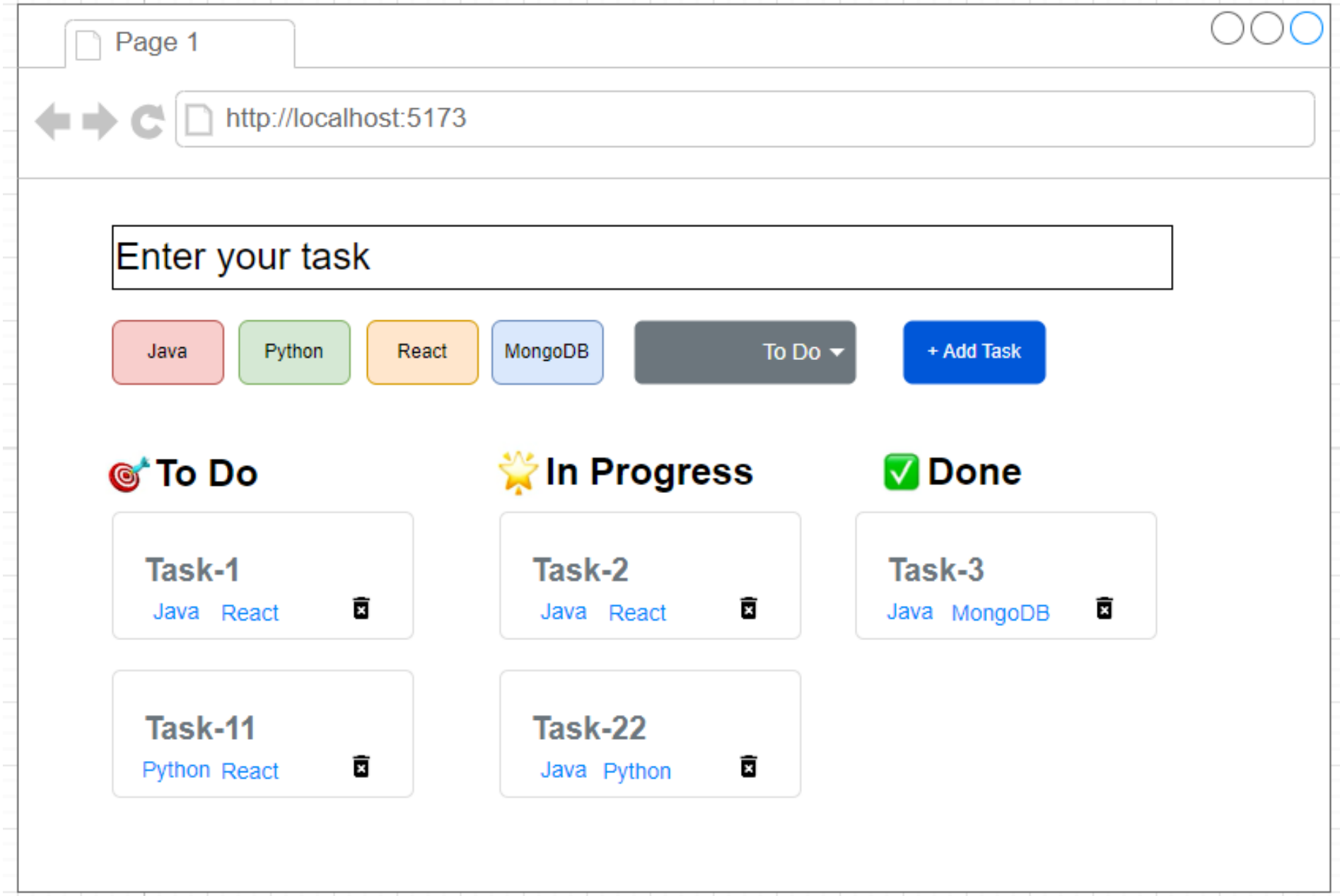
This is Important as Virtual DOM needs this key to uniquely identify each item

- Hari
- Giri
- Suri
- Mari
- Siri

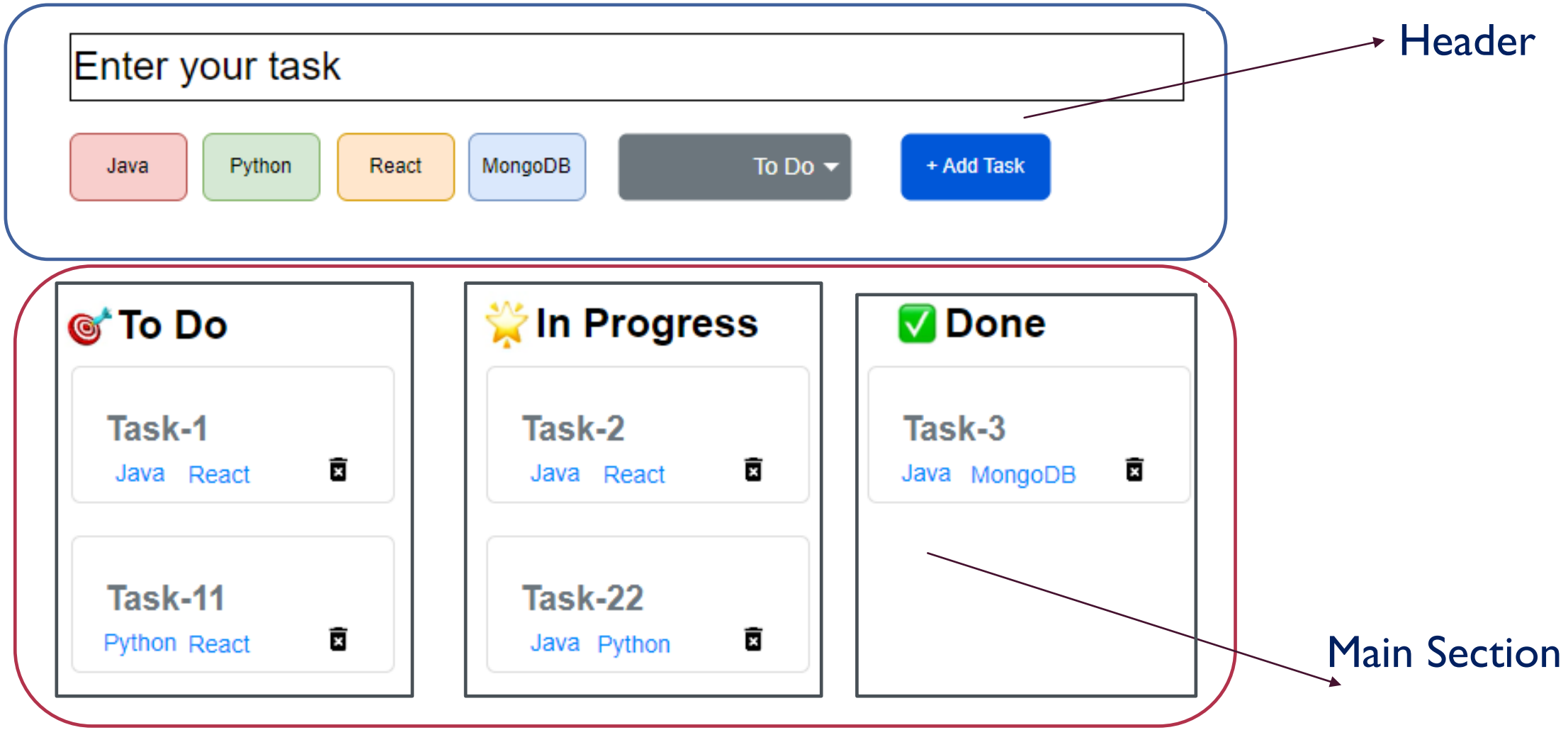
Task Management Project

Using Concepts learned till now

Task Management Application Overview & Planning



Creating Layout Of Application


Enter your task
Java
Python
React
MongoDB
To Do
+ Add Task
Header
To Do
Task-1
Java React
Task-11
Python React
In Progress
Task-2
Java React
Task-22
Java Python
Done
Task-3
Java MongoDB
Main Section

Creating Layout

- Step - 1: 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

index.css

react-vite-proj2 > src > index.css > *

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

App.jsx

react-vite-proj2 > src > App.jsx > App

```jsx
import React from 'react';
import "./App.css";

const App = () => {
  return (
    <div className='app'>
      <header className='app_header'>Header Section</header>

      <main className='app_main'>
        <section className='task_column'>Section - 1</section>
        <section className='task_column'>Section - 2</section>
        <section className='task_column'>Section - 3</section>
      </main>

    </div>
  );
};

export default App;
```

App.css

react-vite-proj2 > src > App.css > .task_column

```css
.app {
  display: grid;
  grid-template-rows: 150px auto;
}

.app_main {
  display: flex;
  justify-content: space-evenly;
  padding: 20px 8%;
}

.task_column {
  width: 33.33%;
  background-color: tomato;
  margin: 20px;
}
```

Create New Component – src/TaskManagementForm.jsx

TaskManagementForm.jsx × src/TaskManagementForm.jsx

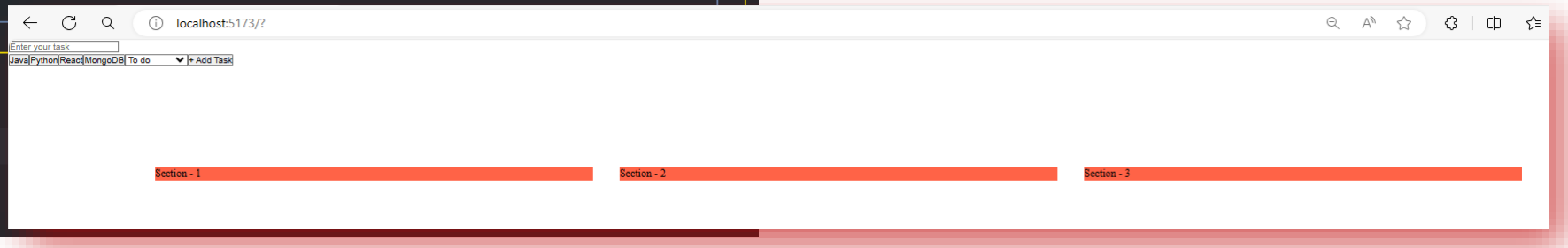
react-vite-proj2 > src > components > TaskManagementForm.jsx > ...

```jsx
import React from 'react';

const TaskManagementForm = () => {
  return (
    <header className='app_header'>
      <form>
        <input type='text' className='task_input' placeholder='Enter your task'/>
        <div className='task_form_bottom_line'>
          <button className='tag'>Java</button>
          <button className='tag'>Python</button>
          <button className='tag'>React</button>
          <button className='tag'>MongoDB</button>

          <select className='task_status'>
            <option value='todo'>To do</option>
            <option value='inprogress'>In progress</option>
            <option value='done'>Done</option>
          </select>

          <button type='submit' className='task_submit'>+ Add Task</button>
        </div>
      </form>
    </header>
  );
};

export default TaskManagementForm;
```

App.jsx ×

react-vite-proj2 > src > App.jsx > default

```jsx
import React from 'react';
import "./App.css";
import TaskManagementForm from './components/TaskManagementForm';

const App = () => {
  return (
    <div className='app'>
      <TaskManagementForm />

      <main className='app_main'>
        <section className='task_column'>Section - 1</section>
        <section className='task_column'>Section - 2</section>
        <section className='task_column'>Section - 3</section>
      </main>

    </div>
  );
};

export default App;
```



Adding Styles for TaskManagementForm Component

TaskManagementForm.css

react-vite-proj2 > src > components > TaskManagementForm.css

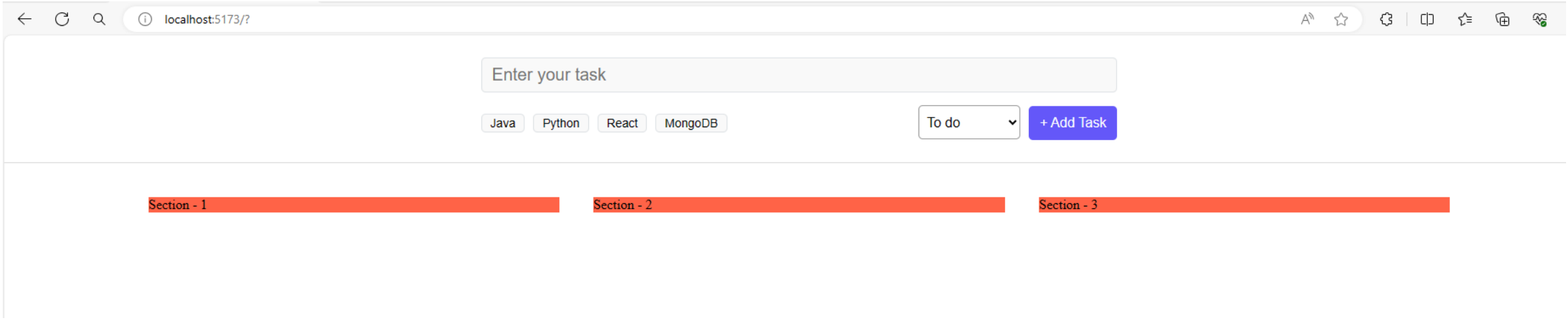
```css
.app_header {
    display: flex;
    align-items: center;
    justify-content: center;
    border-bottom: 1px solid #dcdcdc;
}

.app_header > form {
    width: 40%;
}

.task_input {
    font-size: 20px;
    font-weight: 500;
    background: #f9f9f9;
    color: #000;
    border: 1px solid #dfe3e6;
    border-radius: 5px;
    padding: 8px 12px;
    margin-bottom: 15px;
    width: 100%;
}

.task_form_bottom_line {
    display: flex;
    align-items: center;
    justify-content: space-between;
}
```

```css
.tag {
    font-size: 14px;
    font-weight: 500;
    background-color: #f9f9f9;
    border: 1px solid #dfe3e6;
    border-radius: 5px;
    padding: 2px 10px;
    margin-right: 10px;
    cursor: pointer;
}

.task_status {
    font-size: 16px;
    font-weight: 500;
    border: 1px solid #999;
    border-radius: 5px;
    width: 120px;
    height: 40px;
    padding: 0 5px;
}
```

```css
.task_submit {
    font-size: 16px;
    font-weight: 500;
    background-color: #6457f9;
    color: #fff;
    border-radius: 5px;
    border: 1px solid #dfe3e6;
    height: 40px;
    padding: 3px 13px;
    margin-left: 10px;
    border: none;
    cursor: pointer;
}
```

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

```jsx
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"/>
                <select className='task_status'>
```

NOTE:

We can pass Objects, Arrays, Functions through the props

```jsx
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

```jsx
import React from 'react'
import todo from '../assets/todo.png'
import './TaskComponent.css'

const TaskComponent = () => {
  return (
    <section className='task_column'>
        <h2 className='task_column_heading'>
            <img src={todo} className='task_column_icon' />
            To Do
        </h2>
    </section>
  )
}

export default TaskComponent
```

Designing Task Component

TaskComponent.jsx

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

```jsx
import React from 'react'
import todo from '../assets/todo.png'
import './TaskComponent.css'

const TaskComponent = () => {
  return (
    <section className='task_column'>
        <h2 className='task_column_heading'>
            <img src={todo} className='task_column_icon' />
            To Do
        </h2>
    </section>
  )
}

export default TaskComponent
```

TaskComponent.css

react-vite-proj2 > src > components > TaskC

```css
.task_column_heading {
    display: flex;
    align-items: center;
}

.task_column_icon {
    width: 30px;
    margin-right: 5px;
}
```

Object De-Structuring

TaskComponent.jsx

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

```jsx
import React from 'react'
import './TaskComponent.css'

const TaskComponent = (props) => {
  const {title, task_img} = props;
  return (
    <section className='task_column'>
        <h2 className='task_column_heading'>
            {/*<img src={props.task_img} className='task_column_icon' />
            {props.title}*/}
            <img src={task_img} className='task_column_icon' />
            {title}
        </h2>
    </section>
  )
}

export default TaskComponent
```