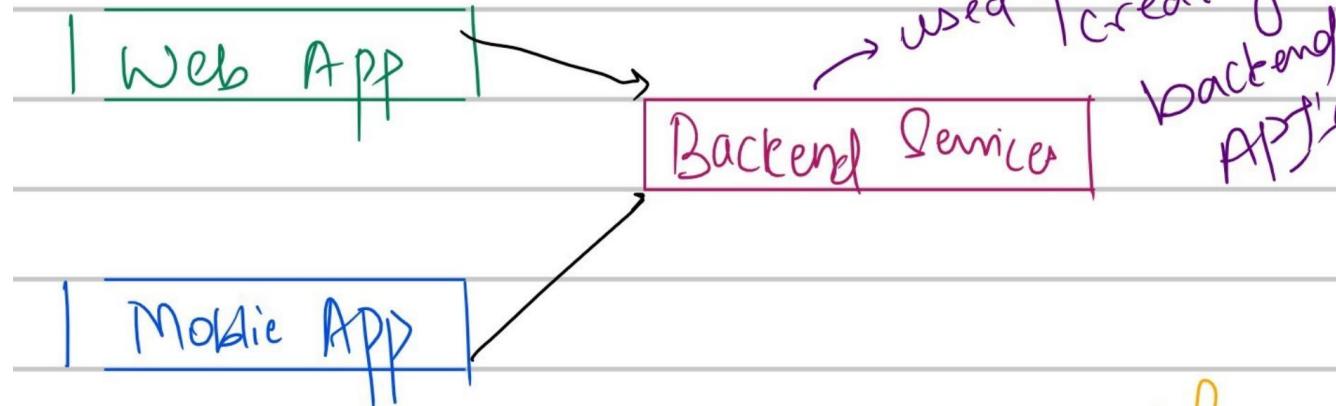




# \* Why Node JS \*

Where is Node JS used a lot?

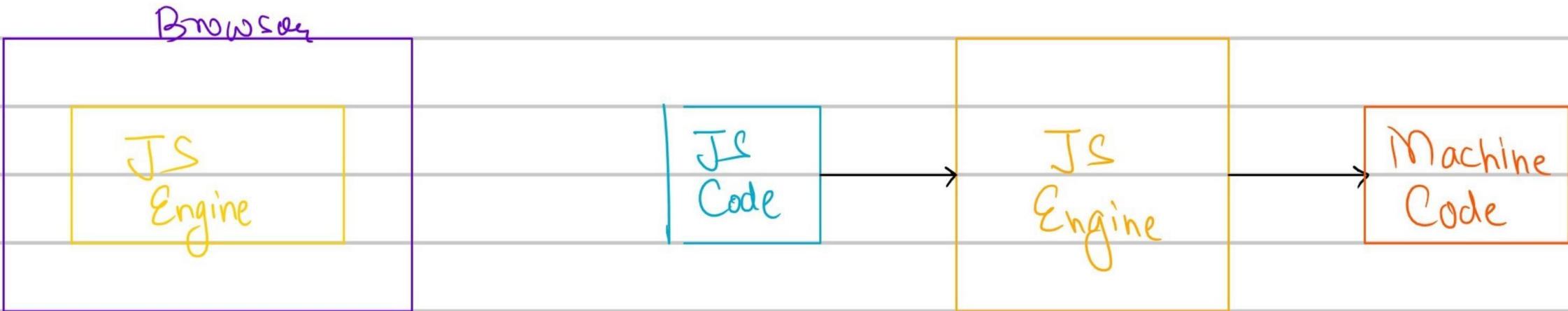


## Features

- \* Super fast
- \* Highly scalable
- \* JavaScript Everywhere
- \* Cleaner & consistent codebase
- \* Largest ecosystem of open source library.

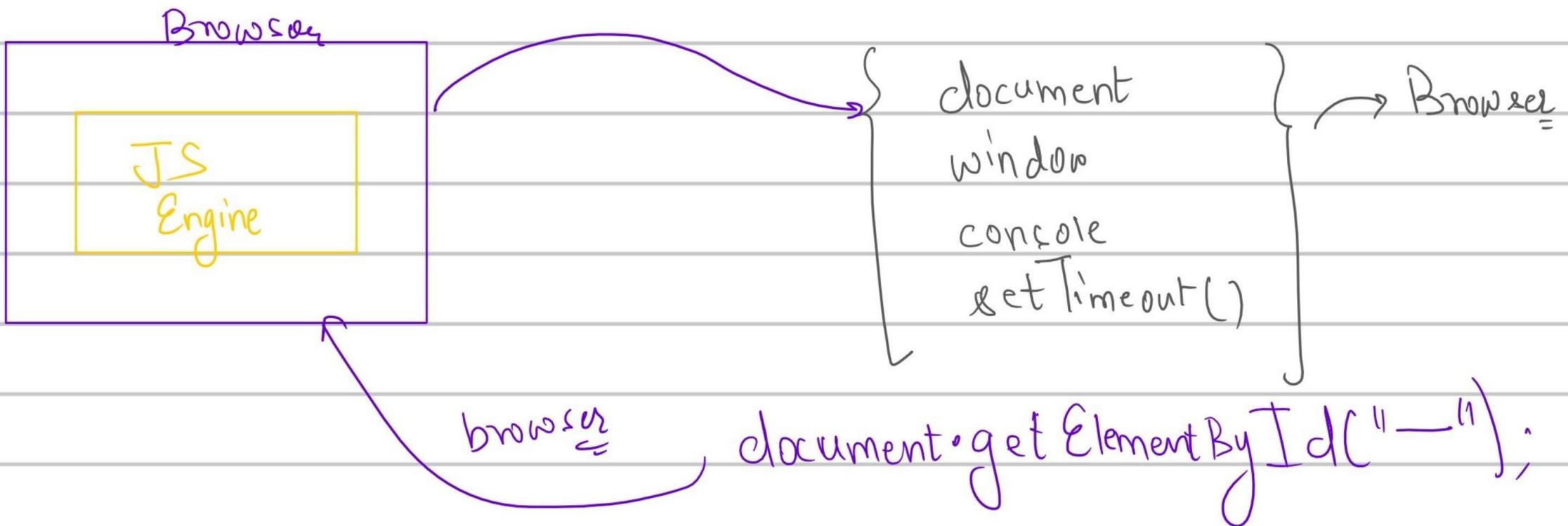
# Node Architecture

Before 2009



Chrome → V8  
Firefox → SpiderMonkey  
Edge → Chakra

# Runtime Environment

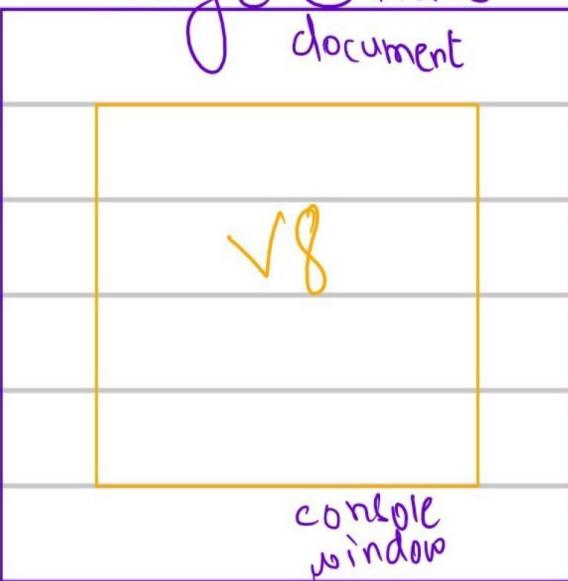


Post 2009

Google open source V8

C++

Google Chrome  
document



document.getElementById("p1"); ✓  
fc ✗  
http ✗

Node.js



document.getElementById("p1") ✗  
fc ✓  
http ✓

- \* Node <sup>is</sup> not a programming language

- \* Node <sup>is</sup> not even a framework like Spring Boot/Django

Node <sup>is</sup> a runtime environment

for executing JavaScript

## Non-Blocking Asynchronous



Waiter

① Take the Order



Table 1

② Hey this is  
the order for  $T_1$

Kitchen

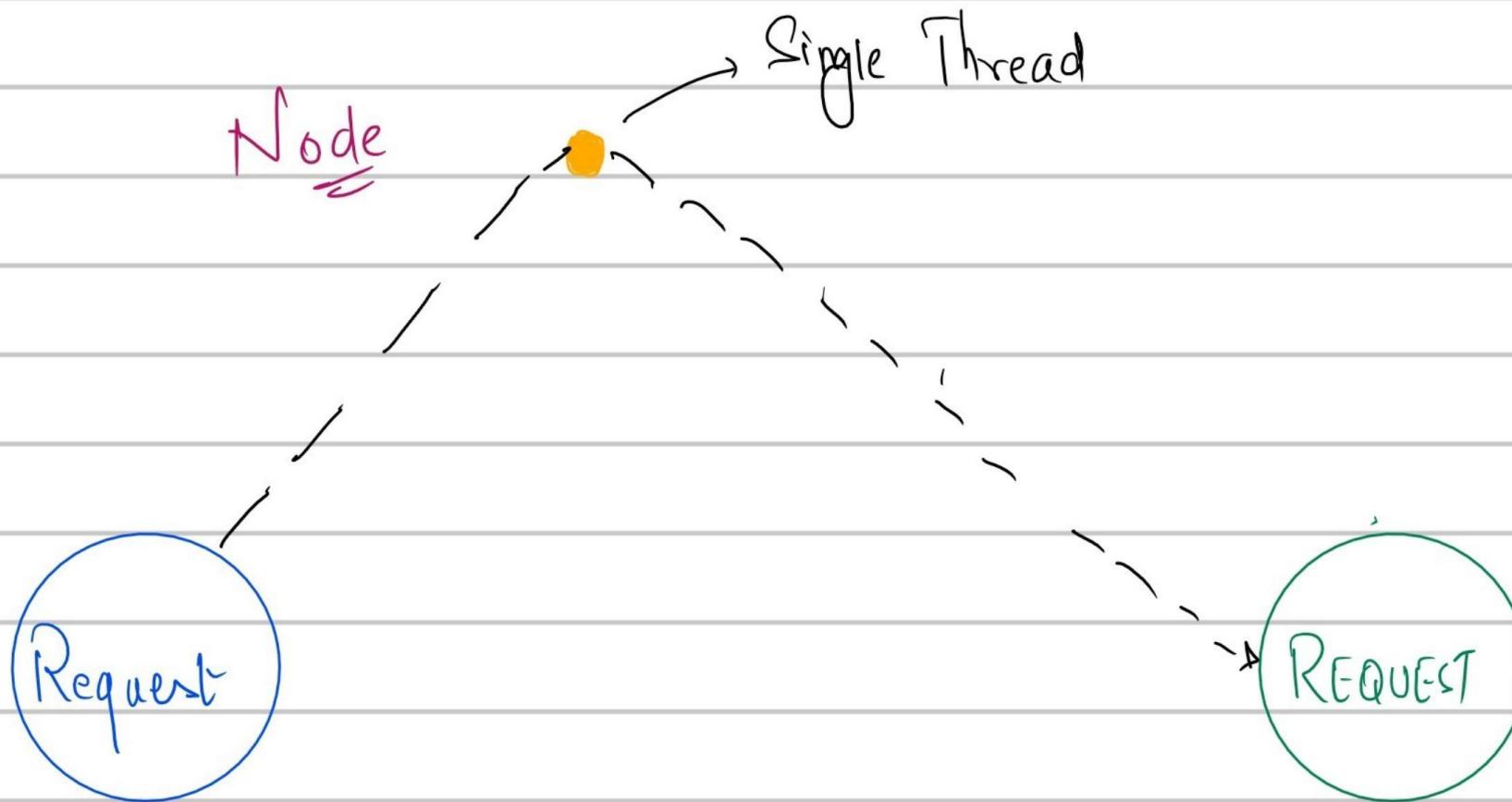
④ This is order for  
 $T_2$

③ Takes the order

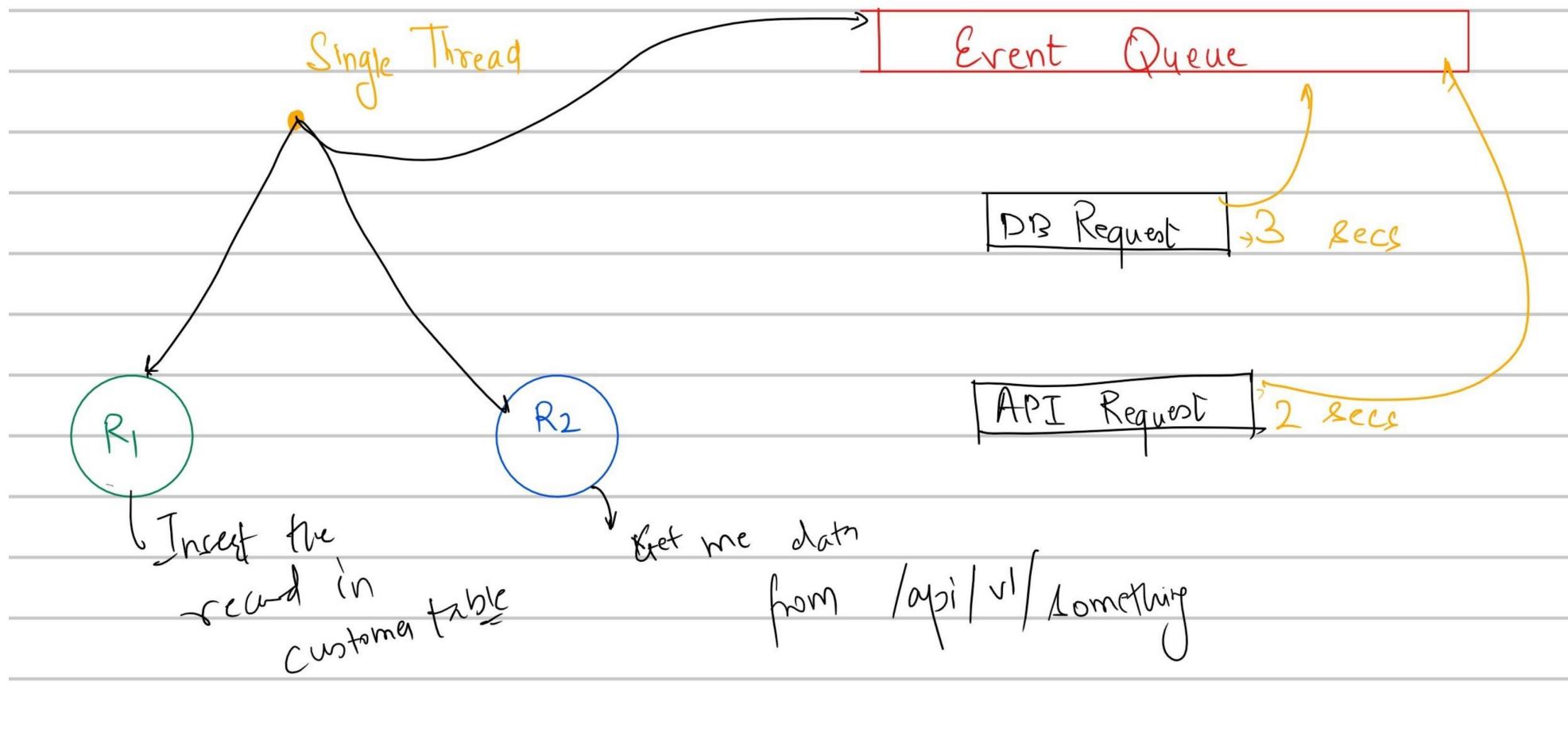


Table 2

\* Node Pic Single Threaded \*

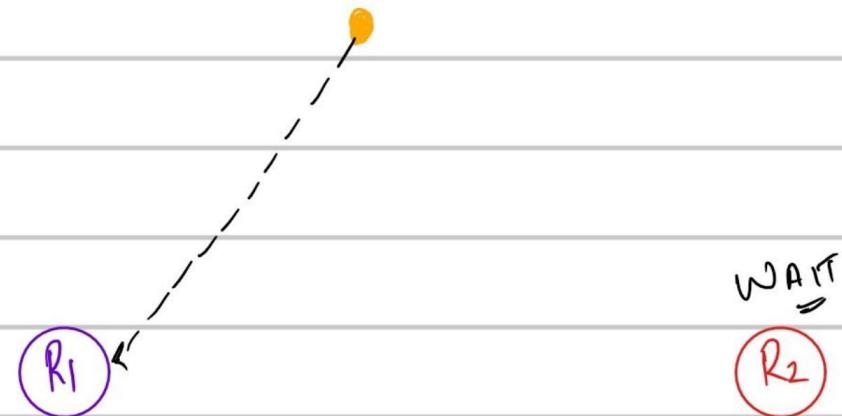


Node applications are **ASYNCHRONOUS** by default.



\* Node is ideal for I/O Intensive App.

\* Node is not ideal for CPU Intensive App



→ Node

Core  
Module → http, fs, url

Local  
Module → user defined

Third Party  
Module → npm



C  $\pm 32767$   
0

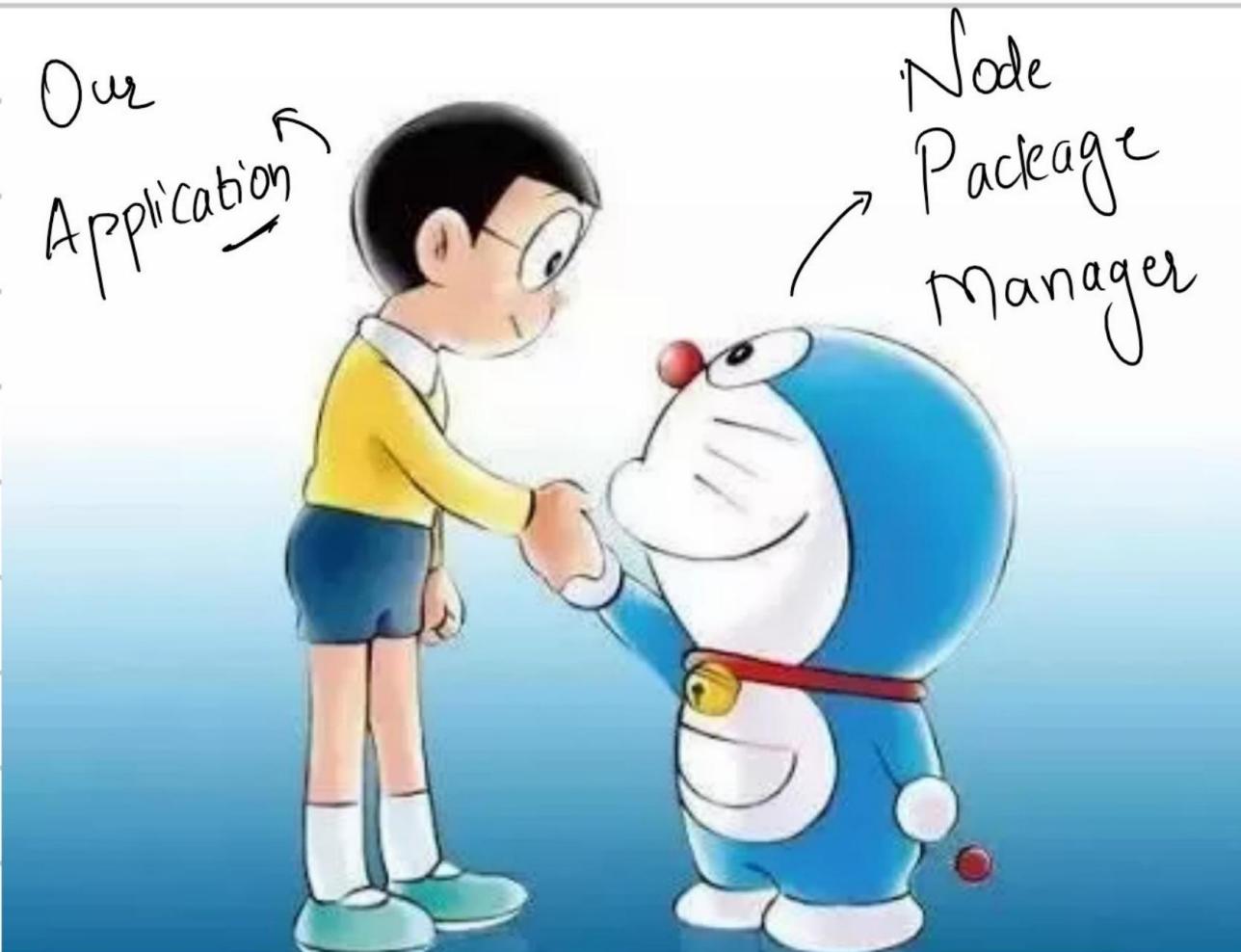
Port is a 2 byte unsigned integer which identifies a software service on a hardware construct.

0 — 65535

0 — 1024 | ~~1025~~ 1025 — 65525



# \* Node Package Manager - npm \*



When I run npm and for initializing  
a node project



npm-demo



package.json

where all the  
meth-data of  
my project  
goes

# Installing an npm package.

Global

npm install package-name -g

anywhere  
↑

Local

npm install package-name

inside project  
↑ folder

Location

current\_folder  
↓

C:\Users\Username\AppData\Roaming\npm\node\_modules

node\_modules

Version

x.x.x

npm install package-name@version

@latest  
=

# Semantic Versioning

Major . Minor . Patch

6 . 3 . 0

↳ found a bug.  
fixed it

6 . 3 . 1

# Semantic Versioning

Major . Minor . Patch  
6 . 3 . 0

↳ found a bug.  
fixed it

6 . 3 . 1

↳ brought some features but  
it doesn't change the  
way of working

6 . 4 . 0



A major overhaul was  
brought it. It changed the  
way application/module  
worked

7 . 0 . 0

After      npm install jquery@3.3.1

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a project named 'NPM-DEMO' which contains a 'node\_modules' folder, a 'jquery' folder, and two lock files: '.package-lock.json' and 'package-lock.json'. The 'package.json' file is open in the main editor. The code in the editor is:

```
scripts": {  
  "test": "echo \\\"Error: no test specified\\\" && exit 1"  
},  
"author": "Zartab Nakhwa",  
"license": "ISC",  
"dependencies": {  
  "jquery": "^3.3.1"  
}
```

A handwritten note with a yellow arrow points from the 'node\_modules' folder in the Explorer to the 'jquery' entry in the 'dependencies' object of the package.json file. Another handwritten note at the bottom right of the screen says: 'an entry for the installed package'.

npm install underscore

The screenshot shows a code editor with two tabs: package.json and index.js. The index.js tab is active, displaying the following code:

```
1 const underscore = require("underscore");
```

To the right of the code, handwritten notes explain the logic for resolving the module name:

→ When a module is used

{

- ① Core Modules → not
- ② File or folder • /underscore → not
- ③ Node\_modules

# Version Upgrade

How node decides for update

```
package.json > {} dependencies > underscore
```

Debug

```
6 "scripts": {  
7   "test": "echo \"Error: no test specified\" && exit 1"  
8 },  
9 "author": "Zartab Nakhwa",  
10 "license": "ISC",  
11 "dependencies": {  
12   "jquery": "^3.3.1",  
13   "underscore": "~1.13.4"  
14 }  
15 }  
16 }
```

↑ → do not update beyond the major

↗ do not update beyond minor

## NPM - Outdated ~ update

```
D:\SCB\2022\Bootcamp July 2022\NODE_NPM_JSON\npm-demo>npm outdated
```

Package	Current	Wanted	Latest	Location	Depended by
jquery	3.3.1	3.6.0	3.6.0	node_modules/jquery	npm-demo

```
D:\SCB\2022\Bootcamp July 2022\NODE_NPM_JSON\npm-demo>npm update
```

```
changed 1 package, and audited 3 packages in 931ms
```

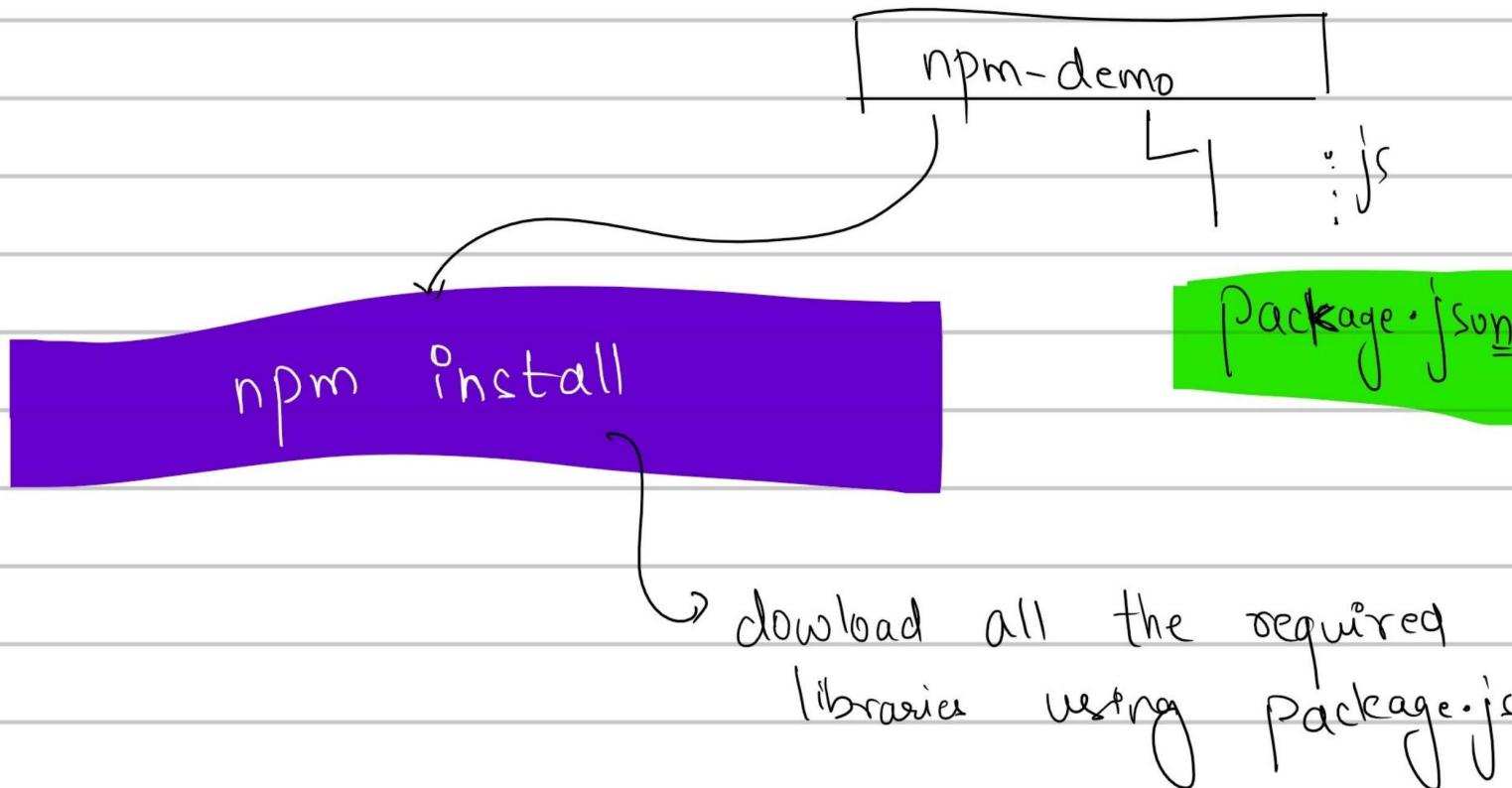
```
found 0 vulnerabilities
```

## Uninstall a module

```
D:\SCB\2022\Bootcamp July 2022\node_npm_json\npm-demo>npm uninstall jquery  
removed 1 package, and audited 2 packages in 881ms  
found 0 vulnerabilities  
D:\SCB\2022\Bootcamp July 2022\node_npm_json\npm-demo>
```

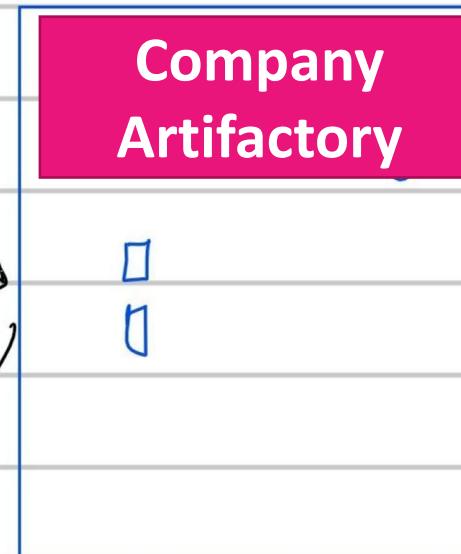
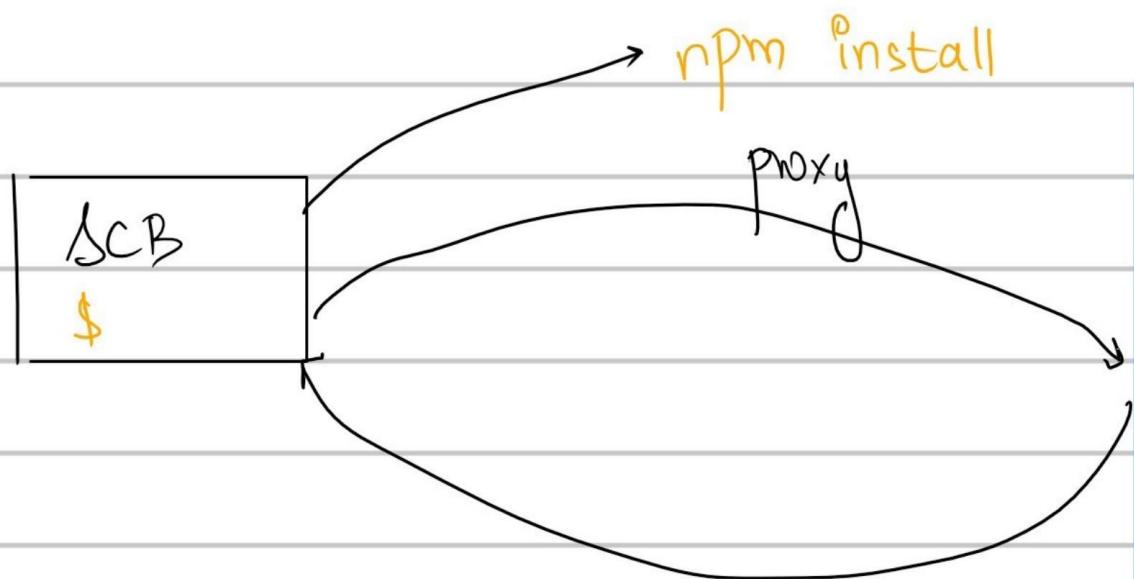
## Transport a node project

node\_modules → .gitignore



Why proxy is used in organization ?

Internet



JSON

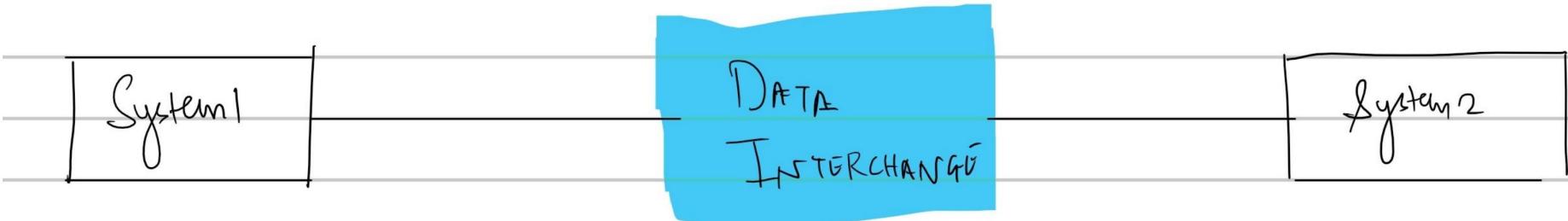
JavaScript Object Notation

data in JSON  
is always in a  
key-value pair

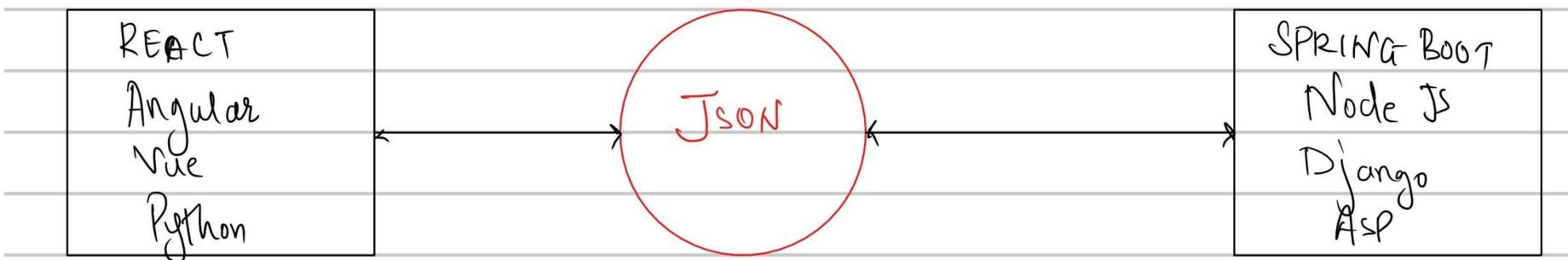
→ is a minimal,  
readable format for structuring  
the data

JSON

is



JSON is English of  
programming language



# JSON



→ a key is always written in "" (string quotation marks)

a value can be a string, number, boolean, array, object

{

"key"

: value,

- a string
- a number
- boolean
- array
- object ()

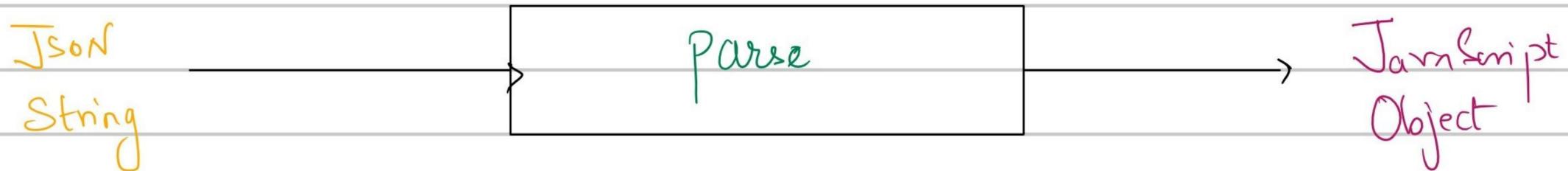
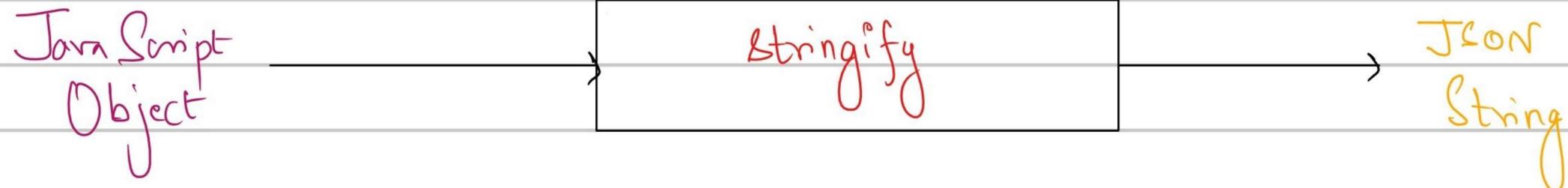
— : — ,

— : — ,

— : —

}

JSON → part of browser



# \* REACT \*

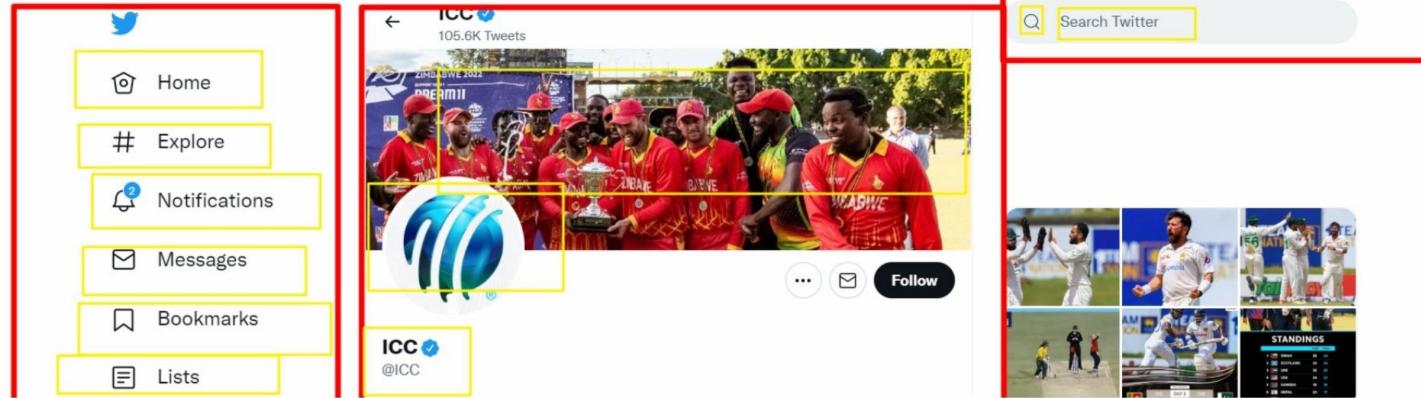
JavaScript based front end / UI

library.

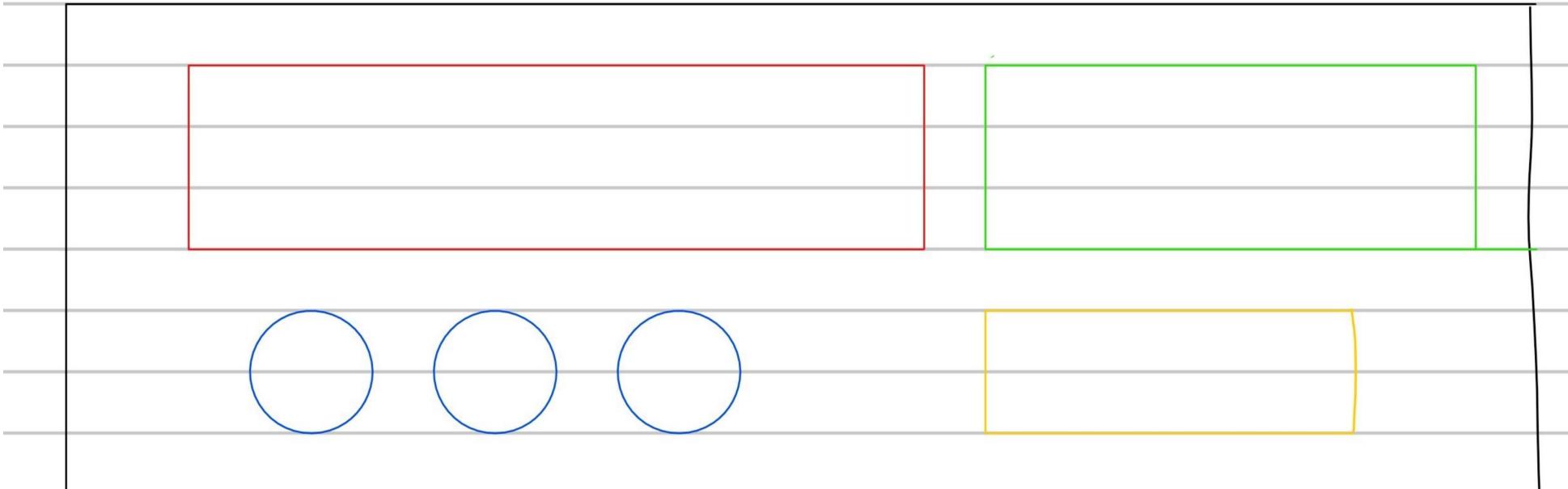
SPA → Single Page Applications

→ React was developed at Facebook

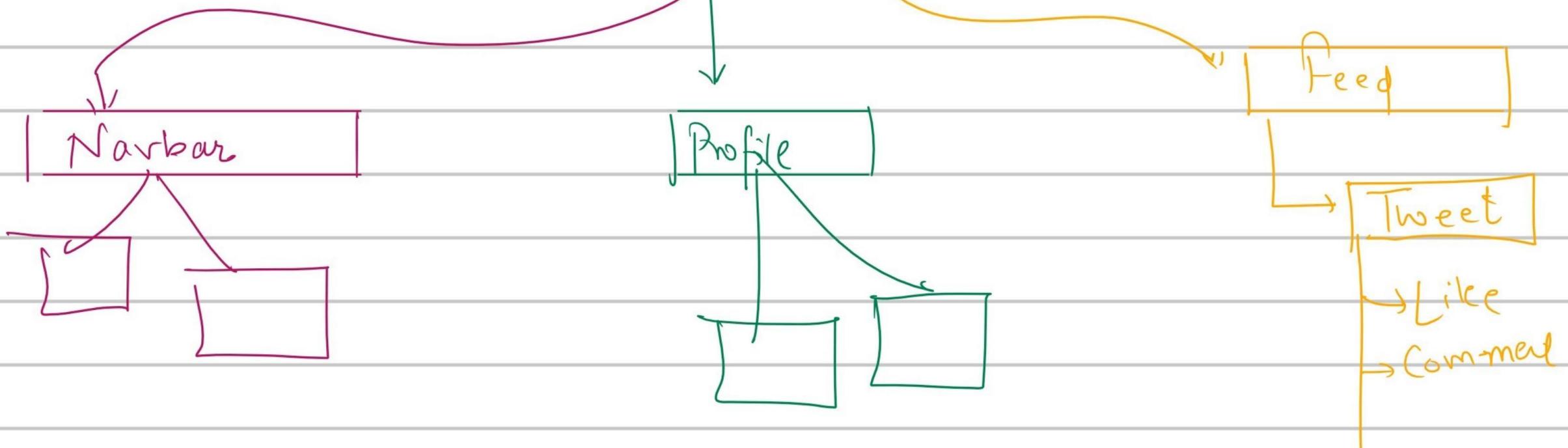
React is made up of Components !!



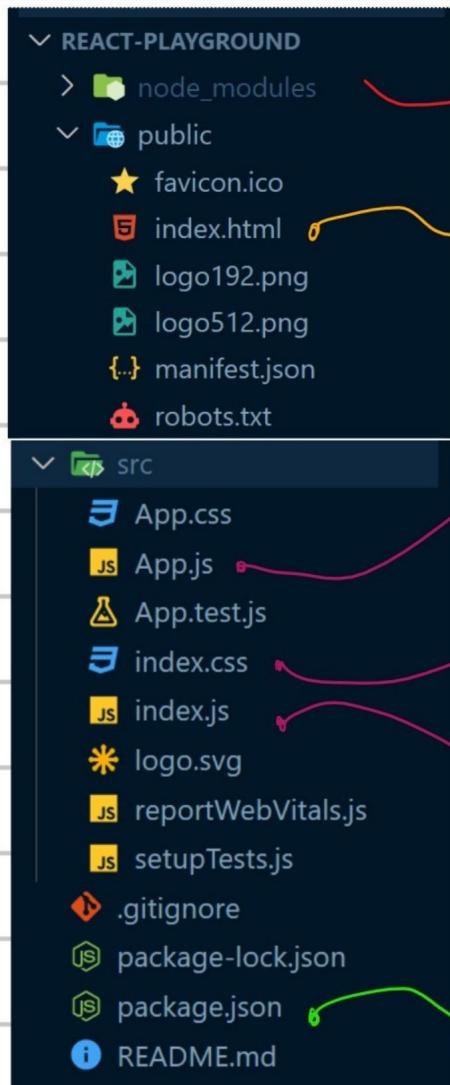
Piece of UI ; who has it's own Component STATE & LIFECYCLE



APP → Root Component



## Structure of React Application



contains the library packages.

renders the data given by `index.js`

Root Component → React Component.

styles `index.html`

main `js` file which loads the Root Component

Configuration file

# \* React Component \*

Class Component

stateful

Functional Component

} stateless  
hooks → stateful

class Tweet extends Component {

state = {  
}

all the variables which  
forms the state of this  
Component; goes here

render() {  
}  
}

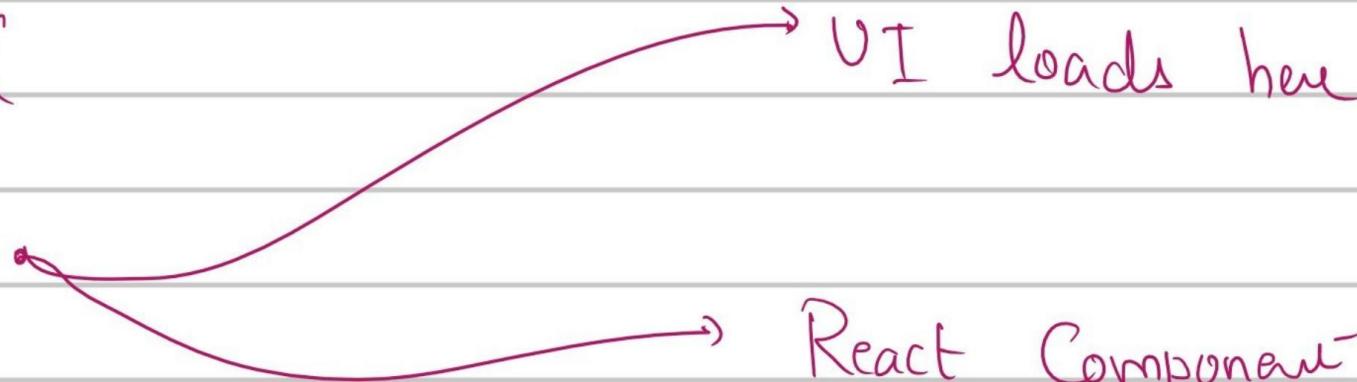
UI loads here

returns a React Component.

Class  
Component

```
function FeedsPage() {
```

```
return (
```



```
);
```

Functional  
Component

Go to react application folder

npm start

The image shows a code editor with three files open:

- index.js**:  
A JavaScript file containing the main application entry point. It imports React, ReactDOM, and App from their respective modules. It then creates a root element using ReactDOM.createRoot and renders the App component into it.
- index.html**:  
An HTML file serving as a template. It includes a manifest link, a title, and a body section. The body contains a `<div id="root"></div>` element where the App component will be rendered.
- App.js**:  
A JavaScript file defining the App component. It returns a JSX structure consisting of a header with a logo and a link to reactjs.org, and a paragraph encouraging users to edit the code.

Handwritten annotations in green and yellow highlight specific parts of the code:

- A large yellow arrow points from the `root.render()` call in index.js down to the `<div id="root"></div>` element in index.html.
- The text "root.render(App)" is written next to the yellow arrow.
- A green arrow points from the `import App from './App';` line in index.js up to the `function App()` definition in App.js.
- The text "imported App" is written above the green arrow.
- A green arrow points from the `<App />` line in index.js up to the `<div className="App">` element in App.js.
- The text "div.id.root" is written next to the green arrow.

JSX  
Java Script XML

JavaScript  
const element = <h1> React </h1>  
HTML

JSX is a Java Script extension syntax used  
to write code in React. It binds  
HTML & Java Script together.

```
const name = "Some Name"
```

```
const e = <h1>Hello {name} </h1>
```

displayName()

}

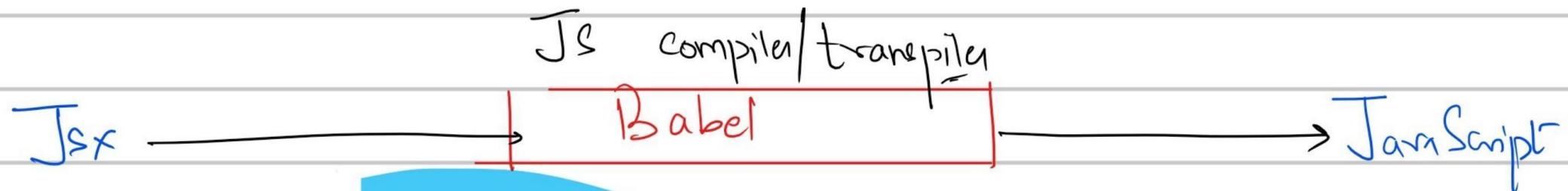
<button onClick={displayName} />



Browser → HTML ✓

Js ✓

Jsx ✗ ↪ not a valid JavaScript code



abc.js

**export** const xyz = 0;

**export** function f1() {

**default export** class Abc {

y

main.js

default → no {}  
named → {} y

import {xyz, f1, Abc}

from './abc'

## Usage of React Components in other Components

App.js

NameComponent.js

<NameComponent />



# What is React?

1. A Javascript Library for building User Interfaces.
2. Developed By Facebook in 2011.
3. A React App/Page is made up of components.
4. Every React App has a Root Component.
5. Root Component contains multiple Child Component.
6. A Component is typically implemented as a Javascript Function.
7. A Component includes some State and a Render Method.
8. React is a Library that takes care of rendering the view & making sure that the view is in sync with the State.

# Snapdeal Component Structure Example

Brand Waali Quality, Bazaar Waali Deal!

Impact@Snapdeal Gift Cards Help Center Sell On Snapdeal Download App

 Search products & brands Q Search Cart Sign In

**Ayushman Bharat Digital Mission**  
Creating India's Digital Health Ecosystem

Health ID- Key to your digital healthcare journey

Create your Health ID

BOB Health Westernwear Kurta Sets Kitchen Need

Your Delivery Pincode

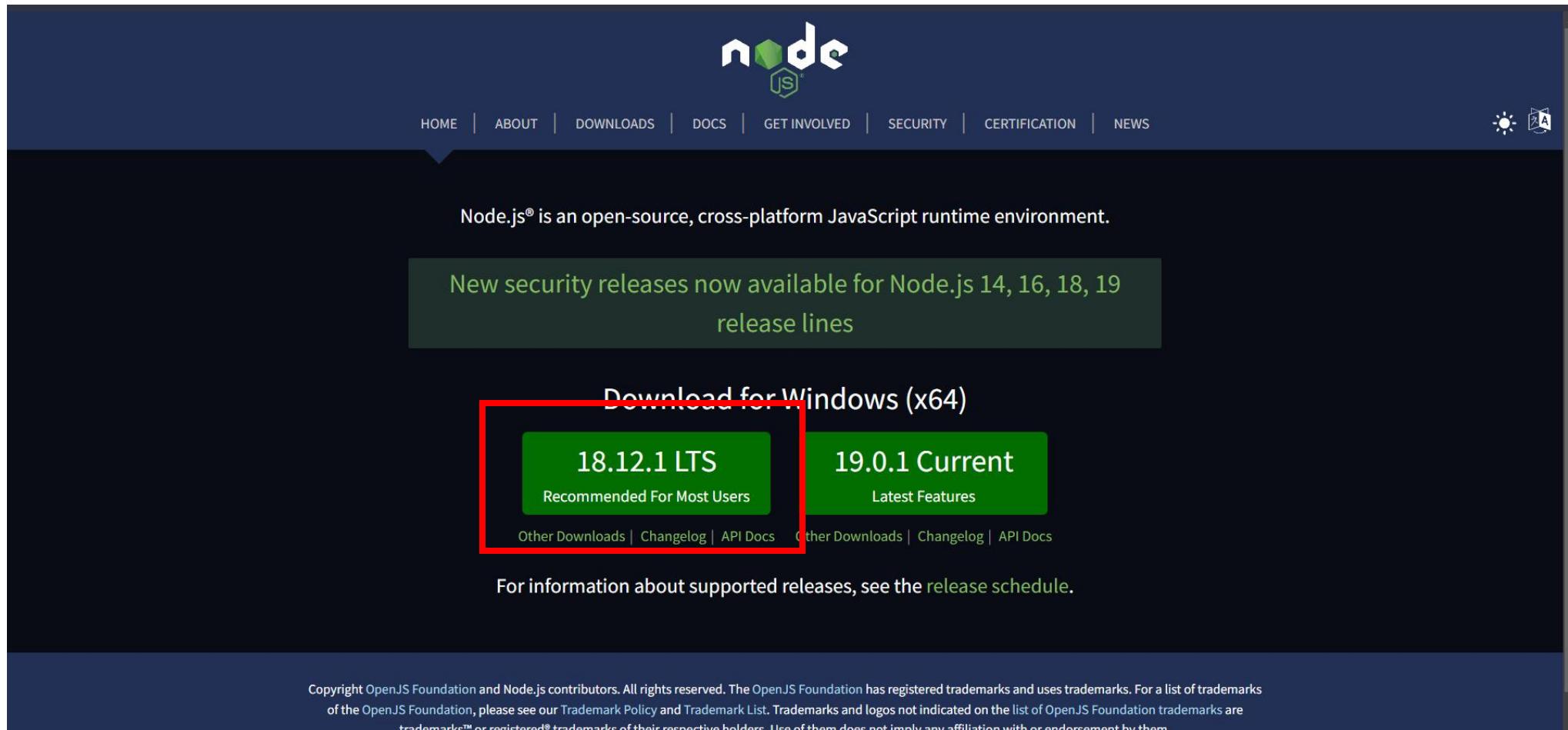
Enter your pincode to check availability and faster delivery options

Enter pincode SUBMIT NEXT

**TRENDING PRODUCTS**

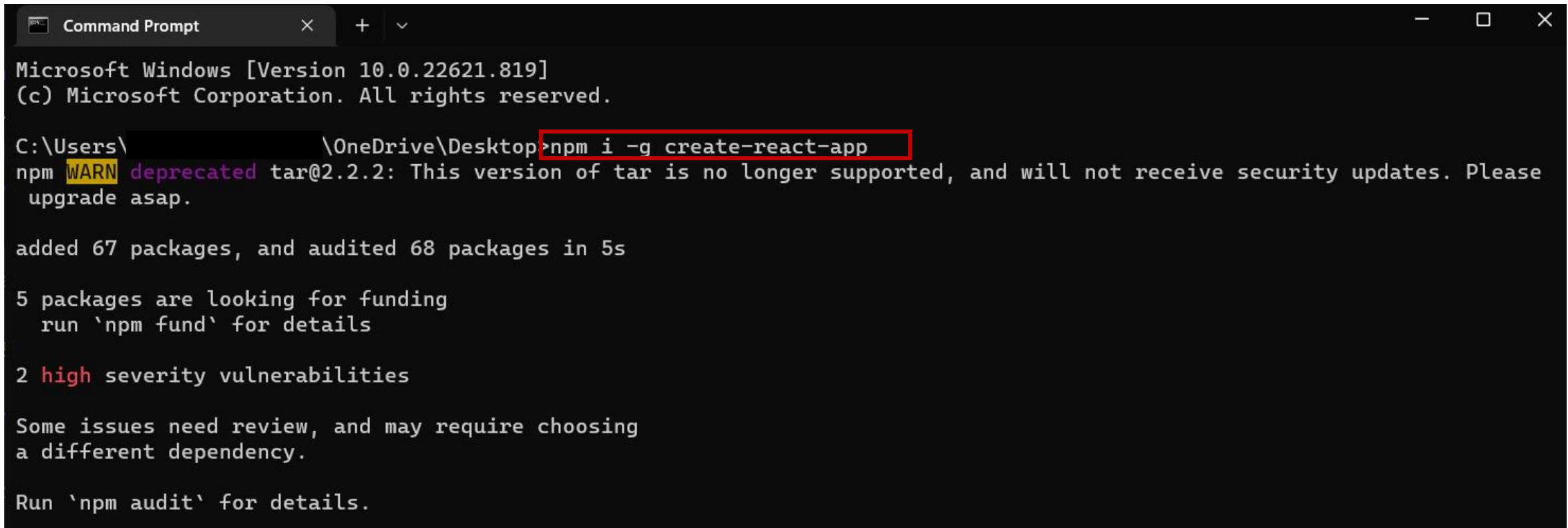
# Setting up the Development Environment.

- Step 1 : Install Node.js from [nodejs.org](https://nodejs.org) .



# Setting up the Development Environment.

- Step 2 : Open command Prompt and Create React App.



```
Command Prompt

Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\          \OneDrive\Desktop>npm i -g create-react-app
npm WARN deprecated tar@2.2.2: This version of tar is no longer supported, and will not receive security updates. Please upgrade asap.

added 67 packages, and audited 68 packages in 5s

5 packages are looking for funding
  run 'npm fund' for details

2 high severity vulnerabilities

Some issues need review, and may require choosing
a different dependency.

Run 'npm audit' for details.
```

# Setting up the Development Environment.

- Step 3 : Download Visual Studio Code from [code.visualstudio.com](https://code.visualstudio.com)

The screenshot shows the official Visual Studio Code download page. At the top, there's a dark navigation bar with the Visual Studio Code logo, a search bar labeled "Search Docs", and a blue "Download" button. A message通知 says "Version 1.73 is now available! Read about the new features and fixes from October." Below the navigation, the main heading is "Download Visual Studio Code" with the subtitle "Free and built on open source. Integrated Git, debugging and extensions." On the left, a large Windows logo is enclosed in a red rectangle, with a "Windows" download button below it. To the right, there are download links for Linux (represented by a Tux icon) and Mac (represented by an Apple icon). Each platform section includes a "deb" and ".rpm" link for Linux, and a ".zip" link for Mac. At the bottom, detailed download links are provided for each platform, including specific architectures like x64, x86, and Arm64.

Visual Studio Code

Docs Updates Blog API Extensions FAQ Learn

Search Docs Download

Version 1.73 is now available! Read about the new features and fixes from October. X

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

 Windows

Windows 8, 10, 11

User Installer x64 x86 Arm64  
System Installer x64 x86 Arm64  
.zip x64 x86 Arm64

 deb .rpm

Debian, Ubuntu Red Hat, Fedora, SUSE

 Mac

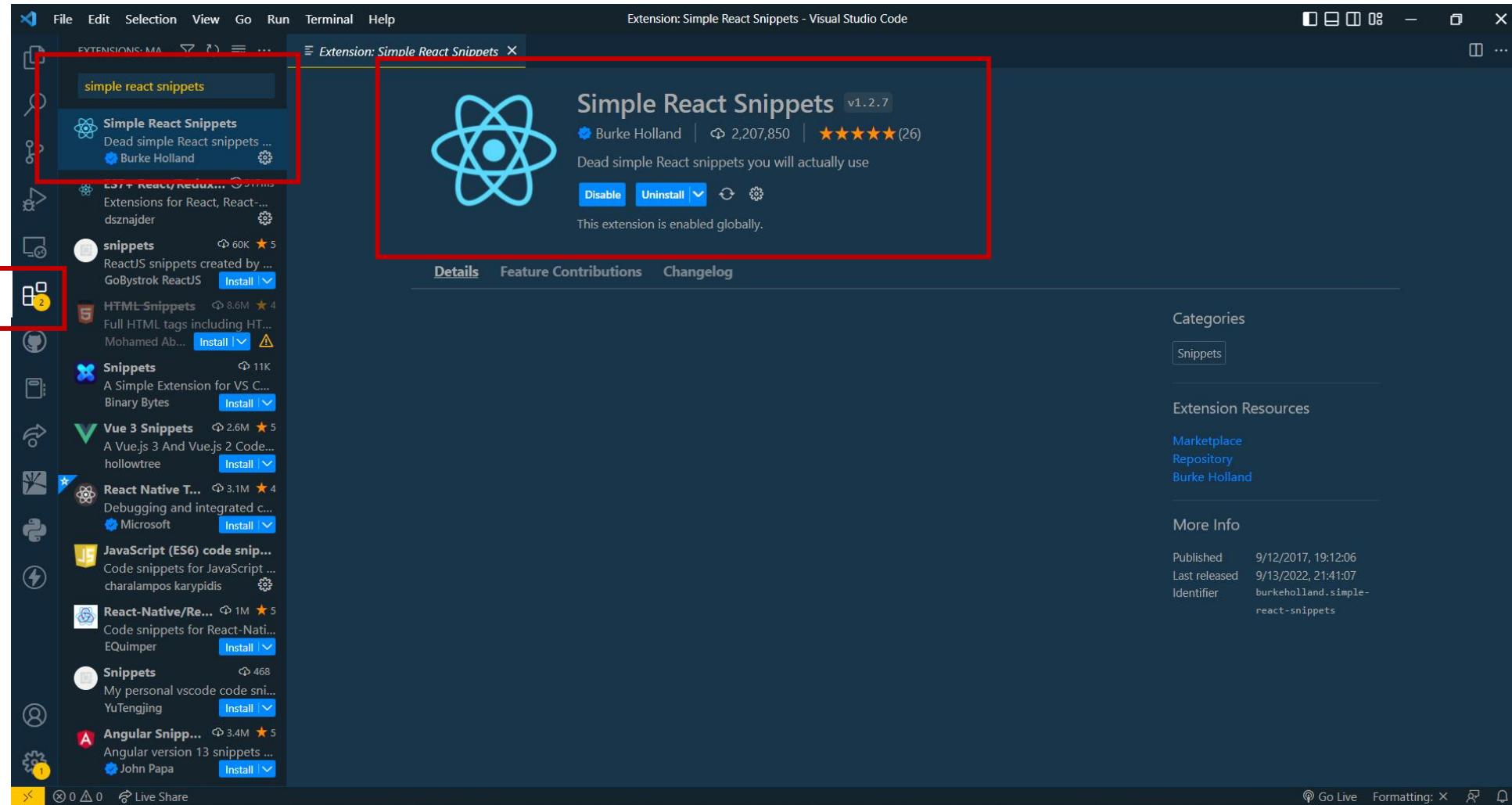
macOS 10.11+

.deb x64 Arm32 Arm64  
.rpm x64 Arm32 Arm64  
.tar.gz x64 Arm32 Arm64  
Snap [Snap Store](#)

.zip Universal Intel chip Apple silicon

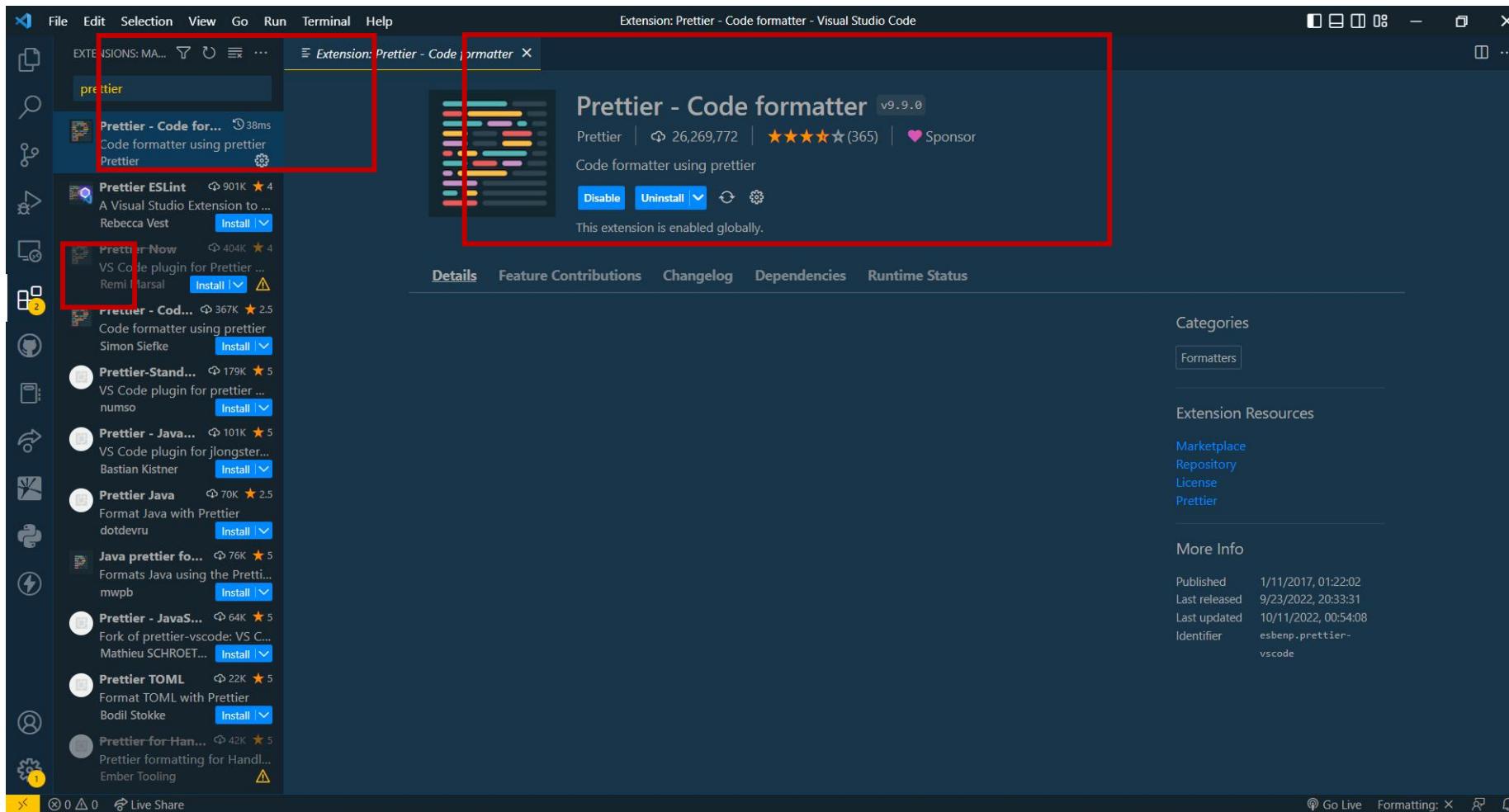
# Setting up the Development Environment.

- Step 4 : Open VS Code and Install ‘Simple React Snippets’ Extension.



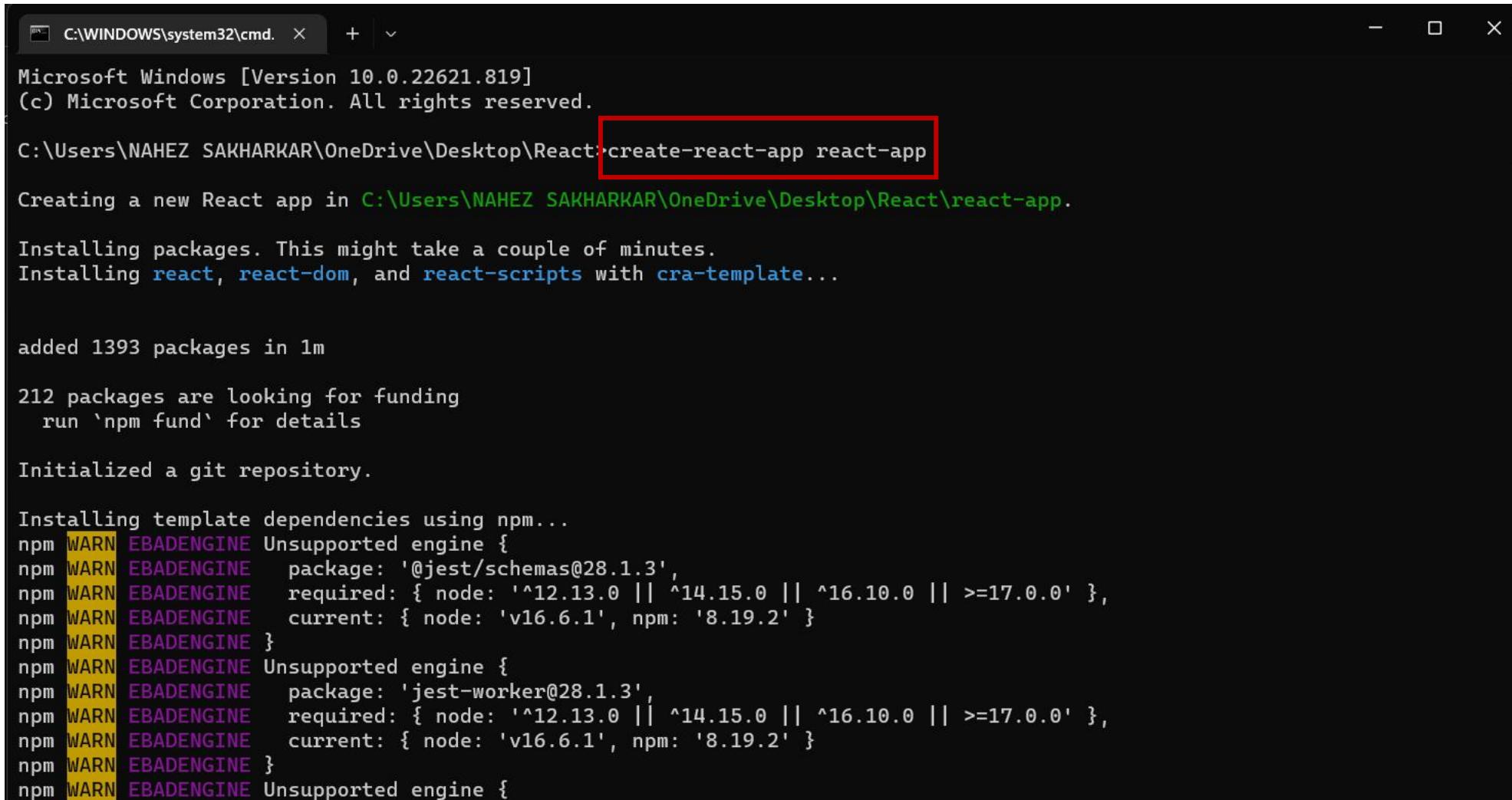
# Setting up the Development Environment.

- Step 5 : Reload VS Code and Install ‘Prettier’ Extension.



# Your First React App

- Step 1 : Open Command Prompt and Type



A screenshot of a Microsoft Windows Command Prompt window titled "C:\WINDOWS\system32\cmd". The window shows the process of creating a new React application named "react-app". The command "create-react-app react-app" is highlighted with a red box. The output text indicates the creation of a new React app in the specified directory, the installation of packages (including react, react-dom, and react-scripts), and the addition of 1393 packages. It also mentions 212 packages looking for funding and initializes a git repository. NPM warning messages related to engines are displayed at the bottom.

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React>create-react-app react-app

Creating a new React app in C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React\react-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1393 packages in 1m

212 packages are looking for funding
  run 'npm fund' for details

Initialized a git repository.

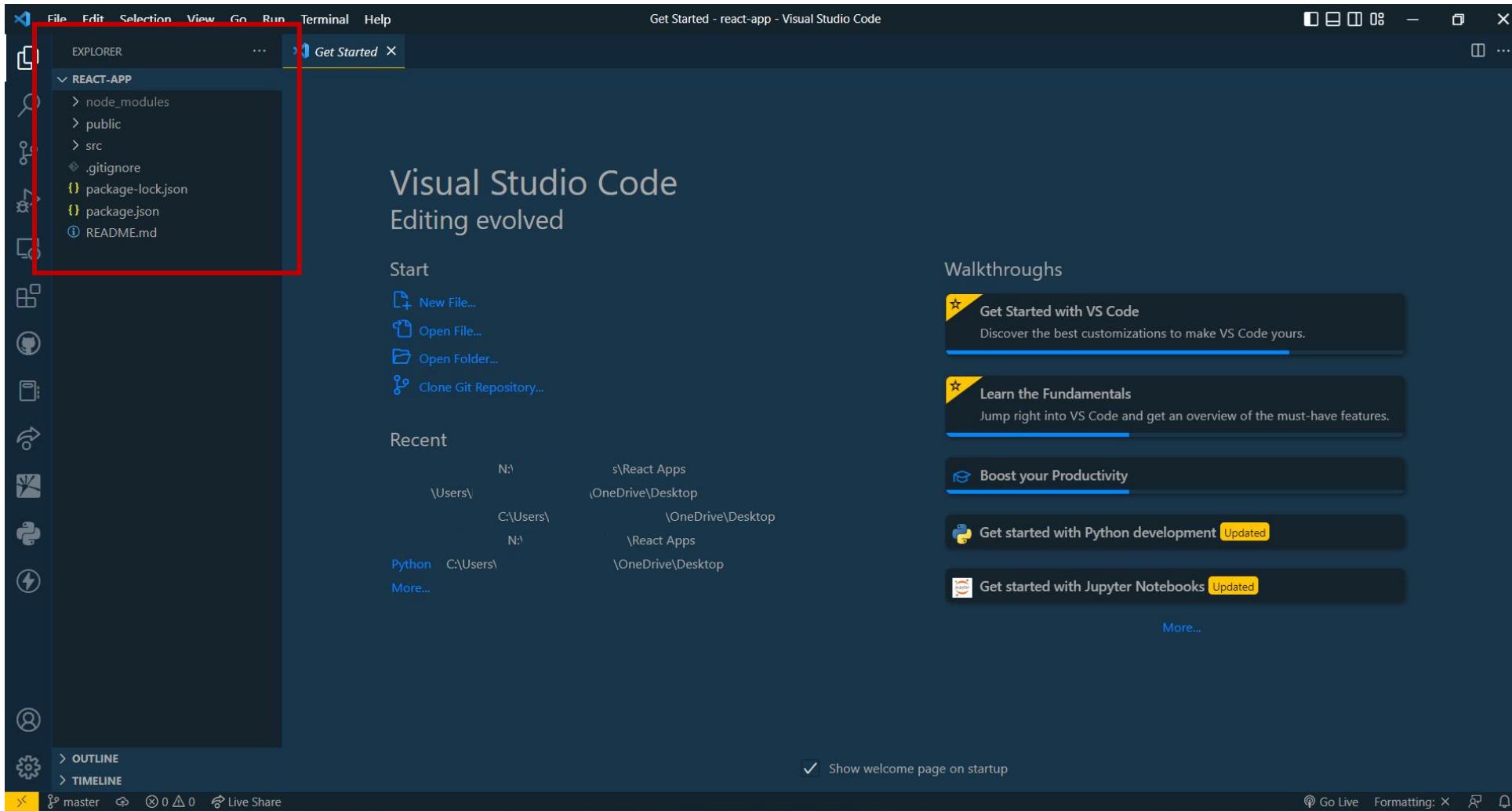
Installing template dependencies using npm...
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@jest/schemas@28.1.3',
npm WARN EBADENGINE   required: { node: '^12.13.0 || ^14.15.0 || ^16.10.0 || >=17.0.0' },
npm WARN EBADENGINE   current: { node: 'v16.6.1', npm: '8.19.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'jest-worker@28.1.3',
npm WARN EBADENGINE   required: { node: '^12.13.0 || ^14.15.0 || ^16.10.0 || >=17.0.0' },
npm WARN EBADENGINE   current: { node: 'v16.6.1', npm: '8.19.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
```

# Your First React App

- This will Install react as well as all the third party libraries.
- Includes Development Server, Webpack & Babel.
- Zero Configuration.

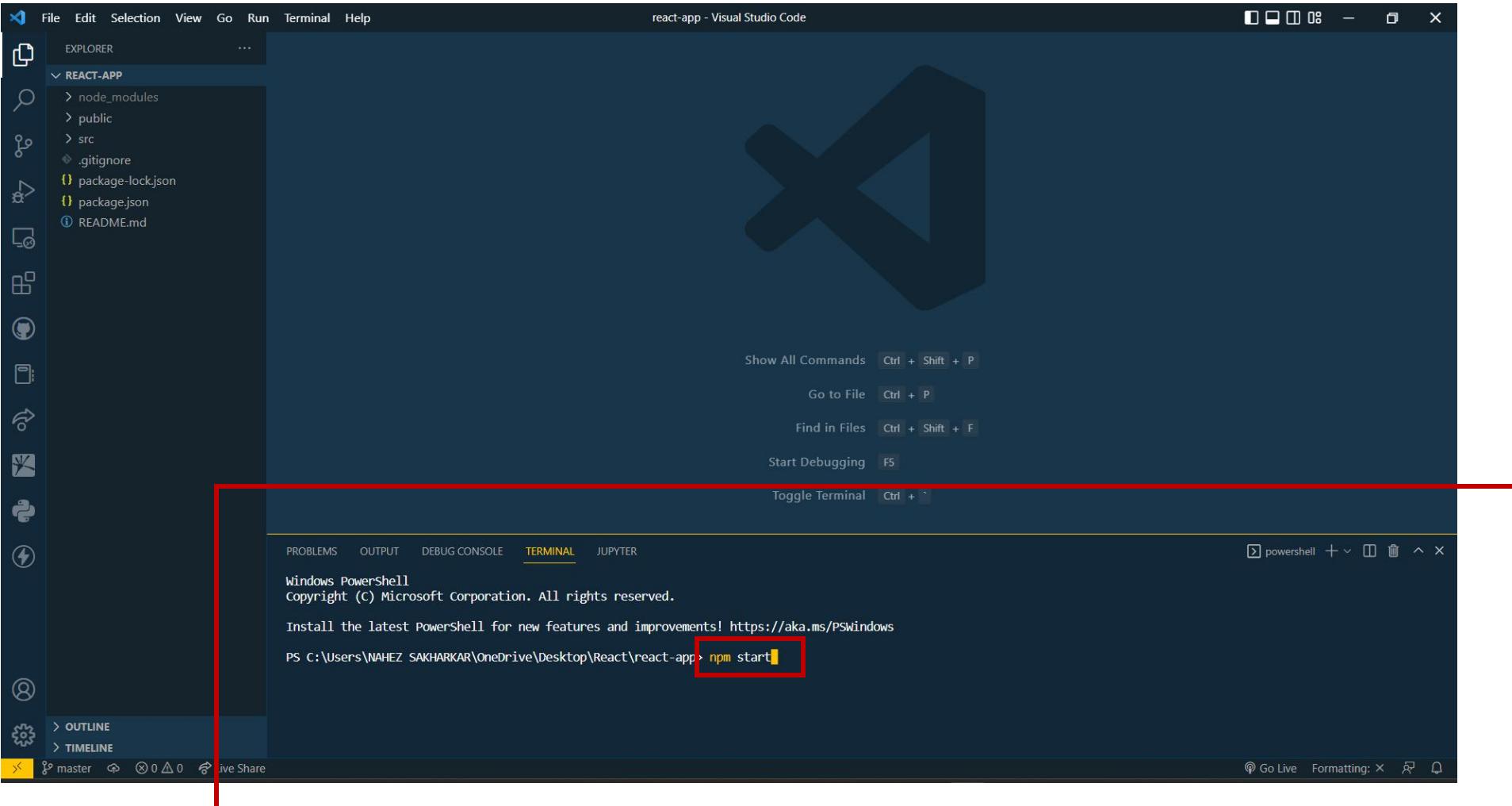
# Your First React App

- Step 2 : Open Folder in VS Code.



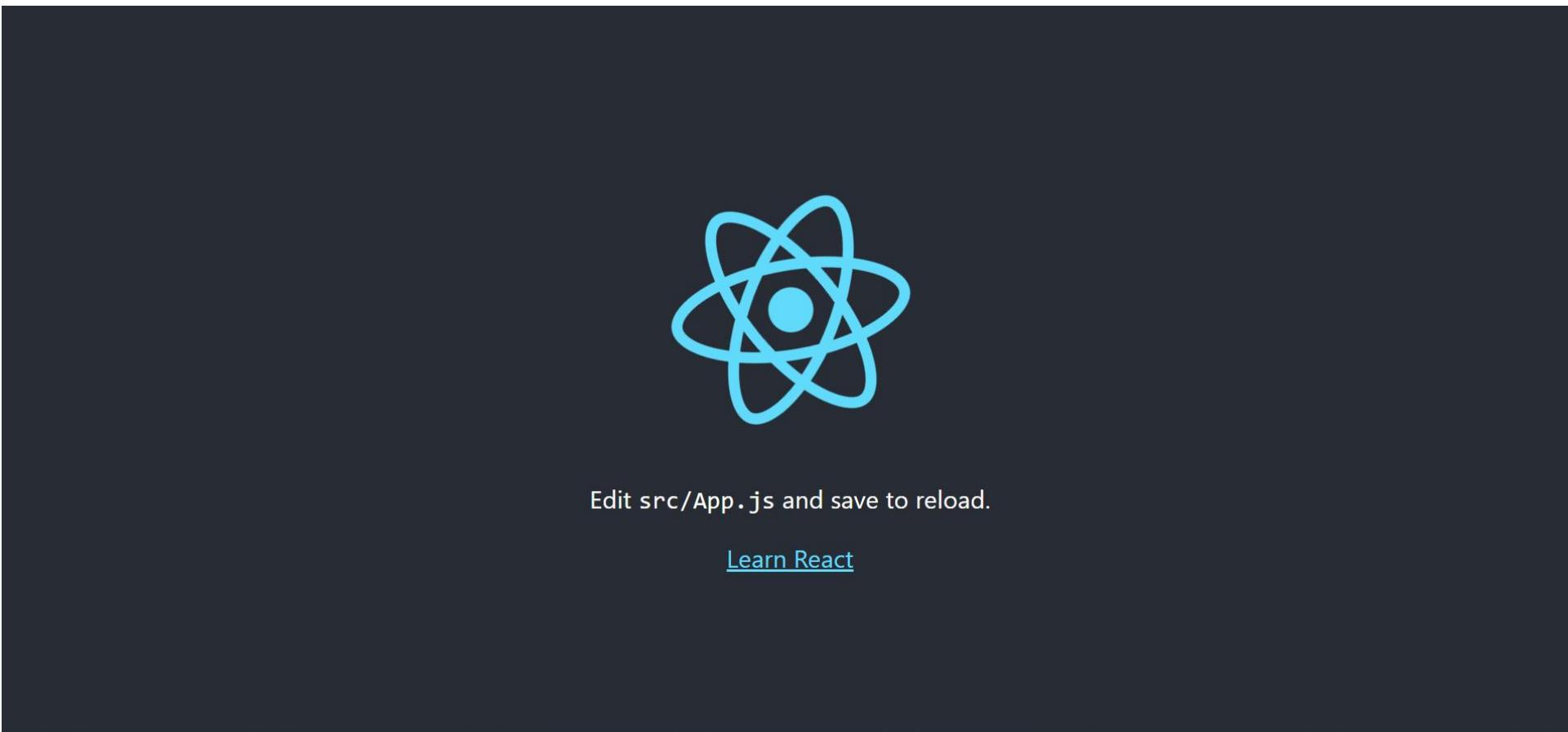
# Your First React App

- Step 3: Open VS Code Terminal and Type ‘npm start’.



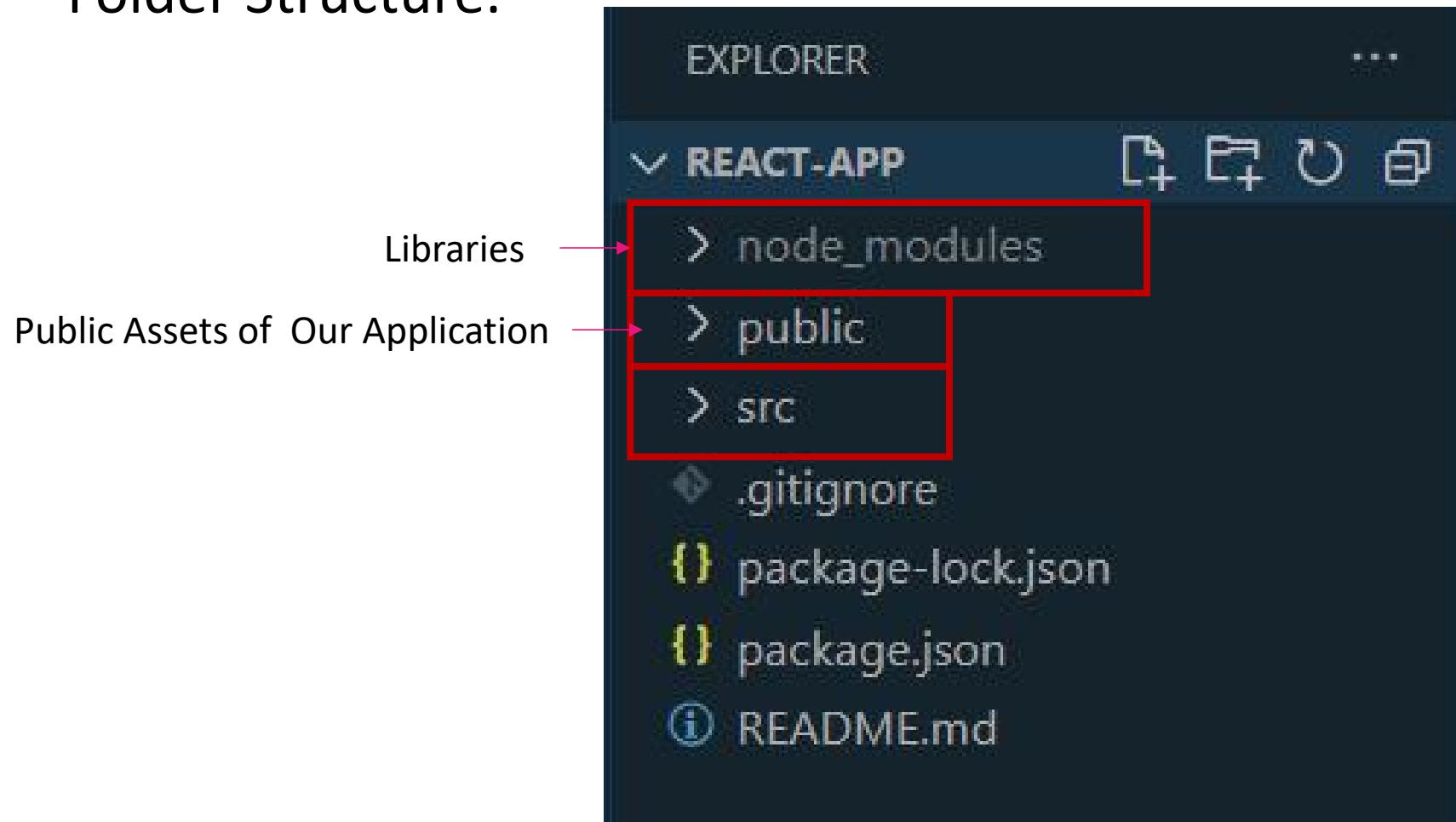
# Your First React App

- Step 4: It will open new browser window.



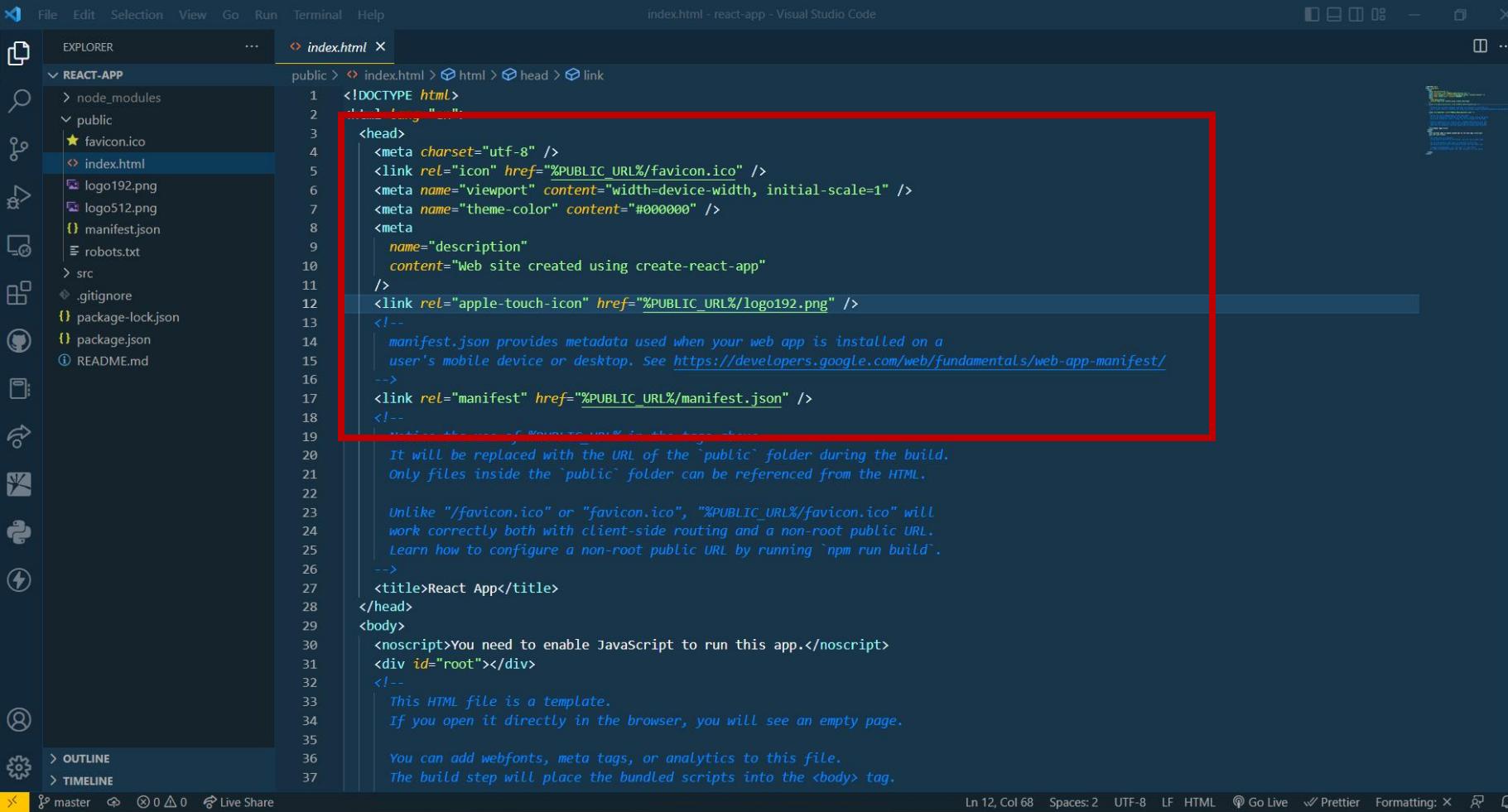
# Your First React App

- Folder Structure.



# Your First React App

- Index.html file in public directory.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "REACT-APP". The "public" folder contains "favicon.ico", "index.html", "logo192.png", "logo512.png", "manifest.json", "robots.txt", "src", ".gitignore", "package-lock.json", "package.json", and "README.md".
- Code Editor (Center):** The "index.html" file is open. The code is displayed in a syntax-highlighted editor. A large red box highlights the entire `<head>` section of the HTML code.
- Terminal (Bottom):** Shows the current branch is "master" and the status bar indicates "Live Share".
- Status Bar (Bottom):** Shows line 12, column 68, spaces: 2, and other file-related information.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using create-react-app" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      This is an empty page of your app in the browser.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.
      Unlike "favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.
    -->
    <!--
      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.
    -->
  </body>
</html>
```

# Your First React App

- This div is the container of our react application.

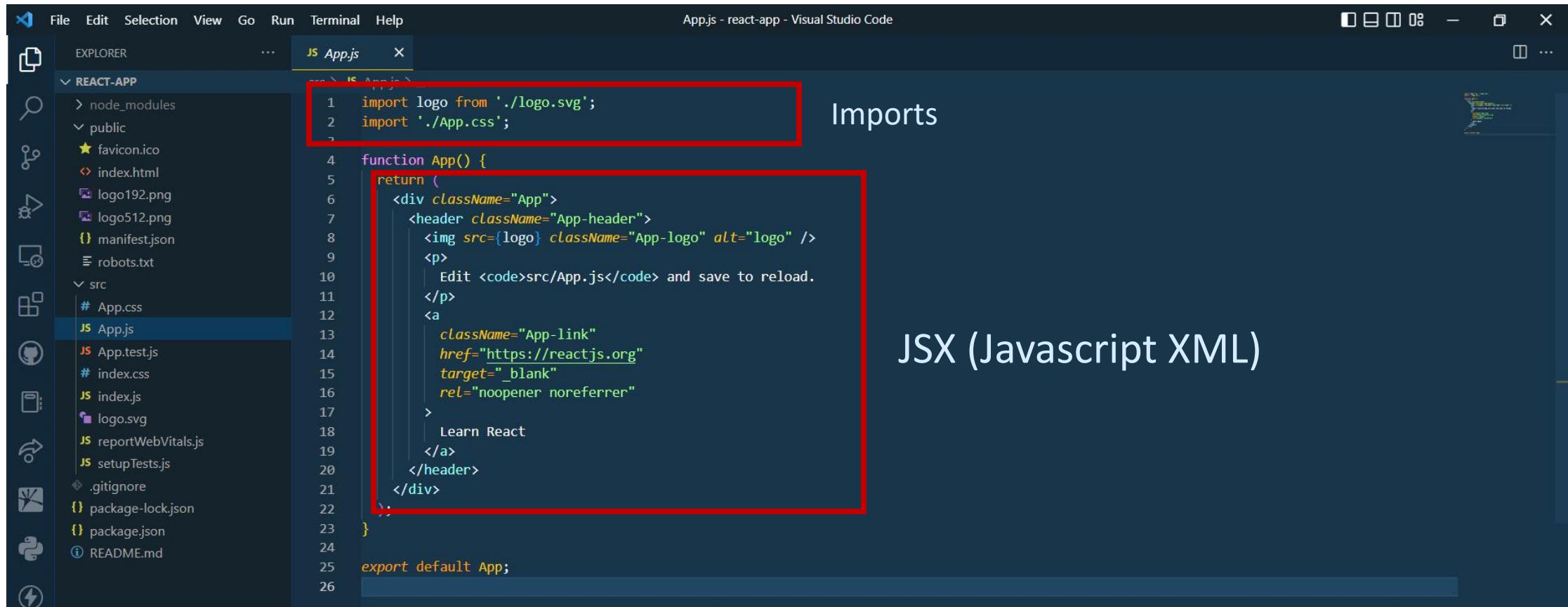
```
<title>React App</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
<!
```

*This HTML file is a template.*

*If you open it directly in the browser, you will see an empty page.*

# Your First React App

- App.js file in src directory.



The screenshot shows the Visual Studio Code interface with the following details:

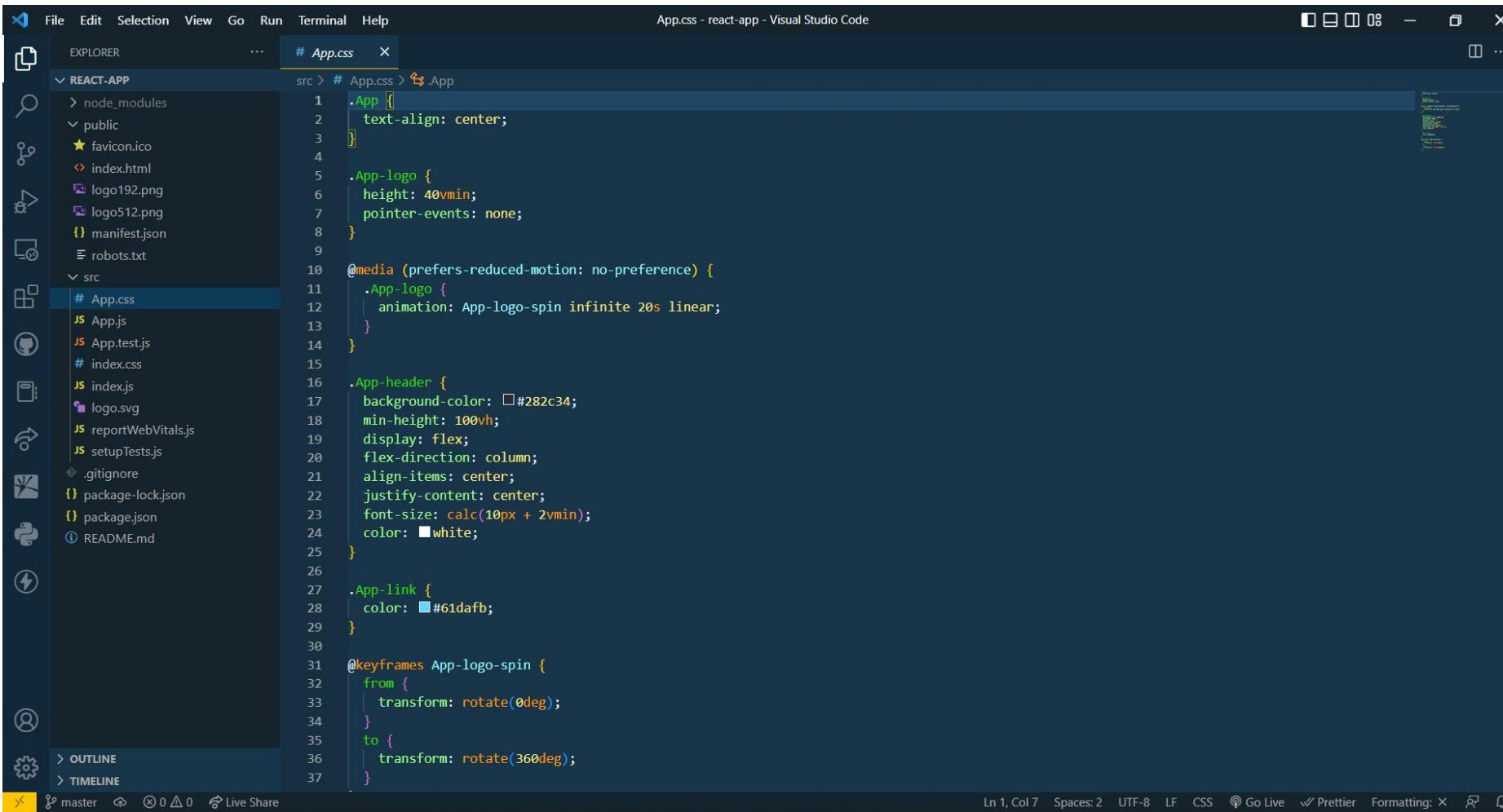
- File Explorer (Left):** Shows the project structure under "REACT-APP". The "src" folder contains "App.css", "App.js", "App.test.js", "index.css", "index.js", "logo.svg", "reportWebVitals.js", "setupTests.js", ".gitignore", "package-lock.json", "package.json", and "README.md".
- Code Editor (Center):** The "App.js" file is open. The code is as follows:

```
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
24
25 export default App;
```

- Annotations:** A red box highlights the first two lines of code (imports). Another red box highlights the entire function body (JSX).
- Right Panel:** Shows the "Imports" and "JSX (Javascript XML)" sections.

# Your First React App

- App.css file in src directory.



The screenshot shows the Visual Studio Code interface with the following details:

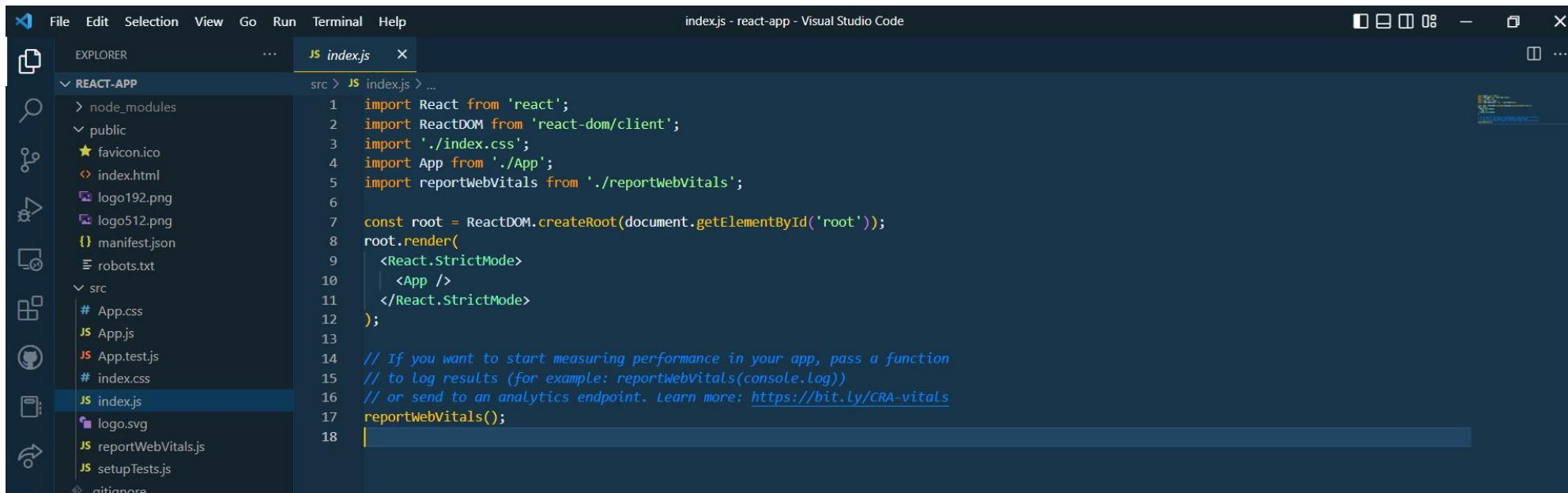
- Title Bar:** App.css - react-app - Visual Studio Code
- File Explorer (Left):** Shows the project structure under "REACT-APP". The "src" folder is expanded, showing "App.css" which is currently selected. Other files include "App.js", "App.test.js", "index.css", "index.js", "logo.svg", "reportWebVitals.js", "setupTests.js", ".gitignore", "package-lock.json", "package.json", and "README.md".
- Code Editor (Center):** Displays the contents of the "App.css" file. The code defines styles for ".App", ".App-logo", ".App-header", ".App-link", and an "@keyframes" rule for ".App-logo-spin".

```
.App {  
  text-align: center;  
}  
  
.App-logo {  
  height: 40vmin;  
  pointer-events: none;  
}  
  
@media (prefers-reduced-motion: no-preference) {  
  .App-logo {  
    animation: App-logo-spin infinite 20s linear;  
  }  
}  

```
- Bottom Status Bar:** Shows "Ln 1, Col 7" and other status indicators like "Spaces: 2", "UTF-8", "LF", "CSS", "Go Live", "Prettier", and "Formatting".

# Your First React App

- Index.js file in src directory, ‘Entry Point of Application’.



The screenshot shows a Visual Studio Code interface with a dark theme. The title bar reads "index.js - react-app - Visual Studio Code". The left sidebar is the Explorer view, showing a project structure for a "REACT-APP" folder. Inside "src", there are files: App.css, App.js, App.test.js, index.css, index.js (which is currently selected and highlighted in blue), logo.svg, reportWebVitals.js, and setupTests.js. The main editor area displays the content of the "index.js" file:

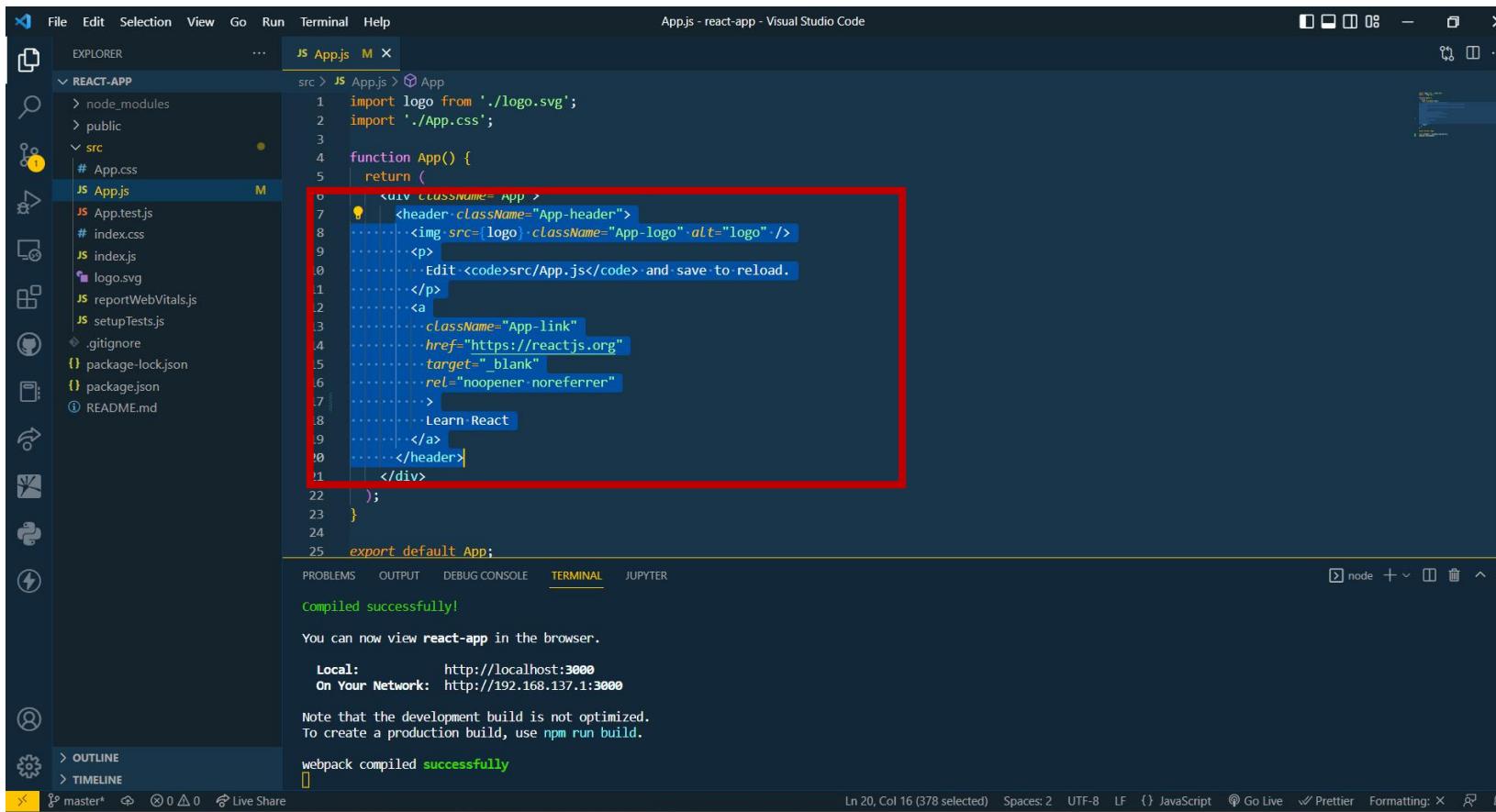
```
src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';

6 const root = ReactDOM.createRoot(document.getElementById('root'));
7 root.render(
8   <React.StrictMode>
9     <App />
10    </React.StrictMode>
11  );
12

14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
18 |
```

# Hello World

- Step 1 : Open App.js File from src directory,
- Step 2 : Delete this part.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "REACT-APP". The "src" folder contains "App.css", "App.js", "index.css", "index.js", "logo.svg", "reportWebVitals.js", "setupTests.js", ".gitignore", "package-lock.json", "package.json", and "README.md".
- Code Editor:** The "App.js" file is open. The code is as follows:

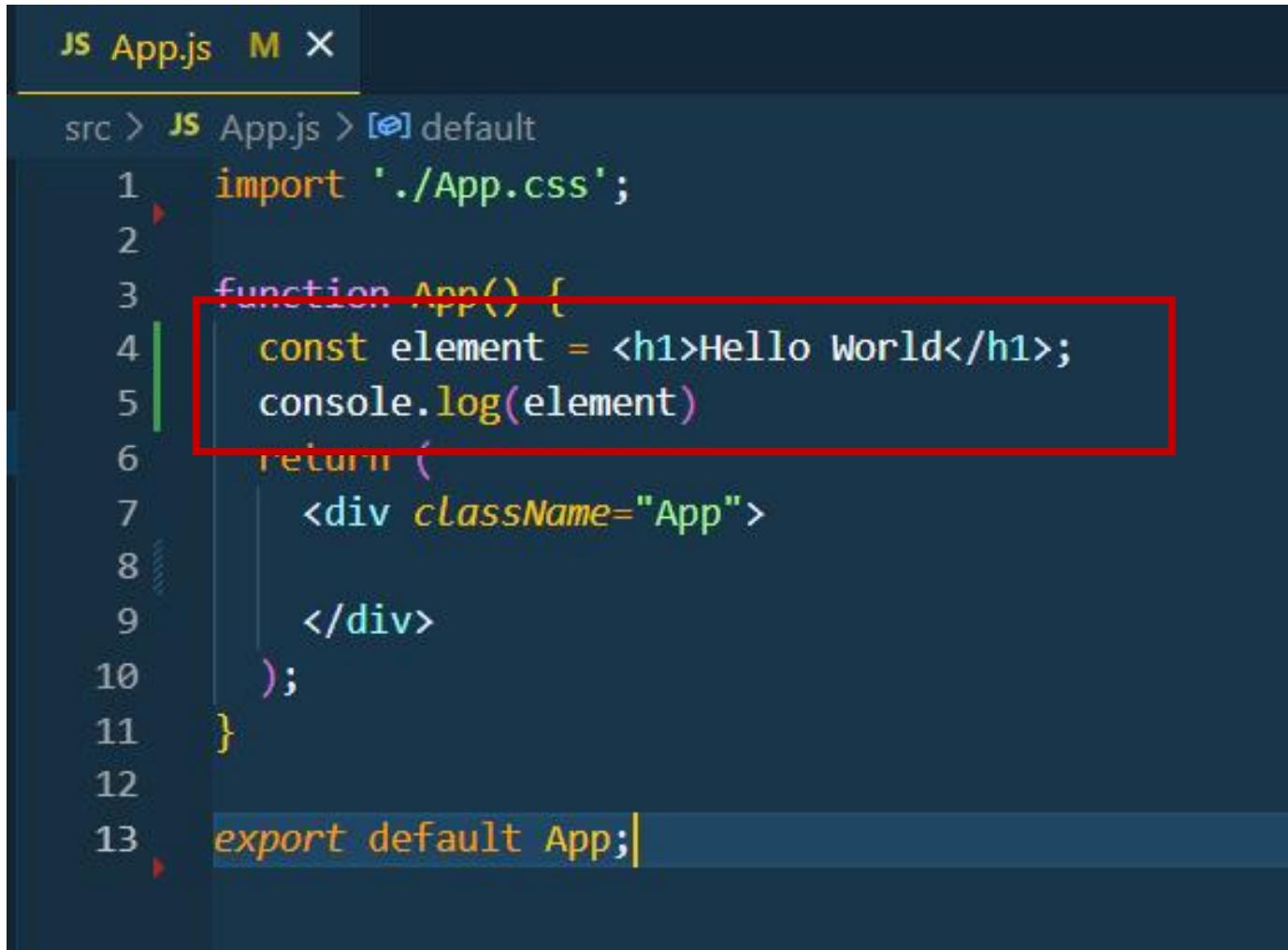
```
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo"/>
9         <p>
10           Edit <a href="https://reactjs.org">src/App.js</a> and save to reload.
11         </p>
12         <a href="https://reactjs.org" target="_blank" rel="noopener noreferrer">
13           Learn React
14         </a>
15       </header>
16     </div>
17   );
18 }
19
20 export default App;
```

- Terminal:** Shows the output of the build process:

```
Compiled successfully!
You can now view react-app in the browser.
Local: http://localhost:3000
On Your Network: http://192.168.137.1:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
```
- Status Bar:** Shows the current file is "master\*", the line and column are "Ln 20, Col 16 (378 selected)", and the encoding is "UTF-8". It also displays icons for JavaScript, Go Live, Prettier, and Formatting.

# Hello World

- Step 3 : Type



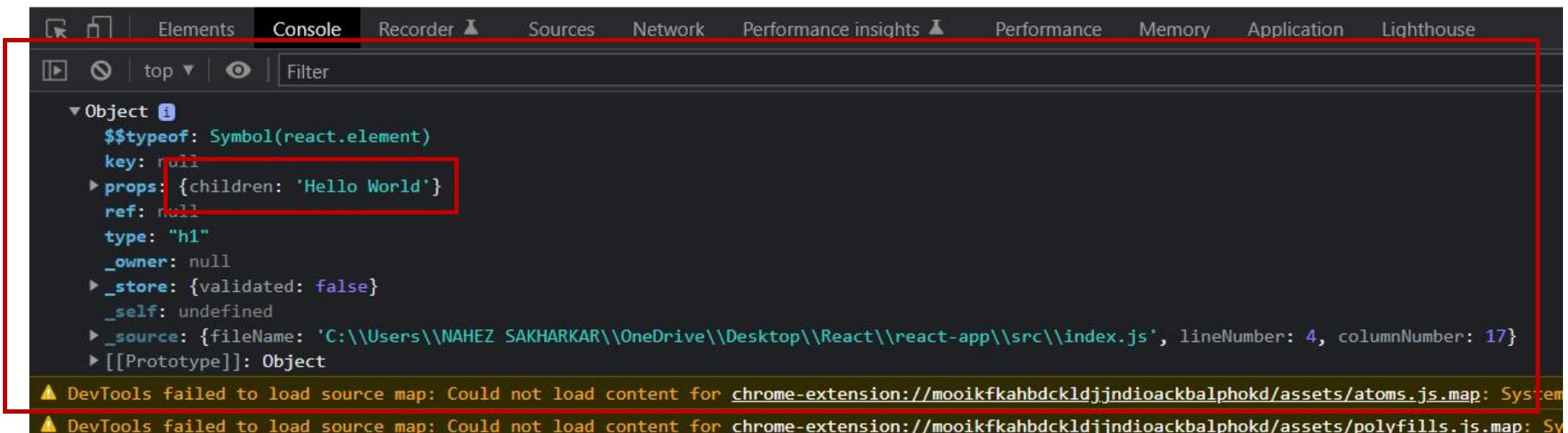
```
JS App.js M X
src > JS App.js > [o] default
1 import './App.css';
2
3 function App() {
4   const element = <h1>Hello World</h1>;
5   console.log(element)
6   return (
7     <div className="App">
8       </div>
9     );
10    }
11  }
12
13 export default App;
```

A screenshot of a code editor showing the file 'App.js'. The code defines a function 'App' that logs an 

# element to the console and returns it wrapped in a div with the class 'App'. The line 'console.log(element)' is highlighted with a red rectangular box. The code editor interface includes tabs for 'JS' and 'M', a status bar with 'X', and a sidebar showing the file structure under 'src'.

# Hello World

- Step 4 : To check your program open console.

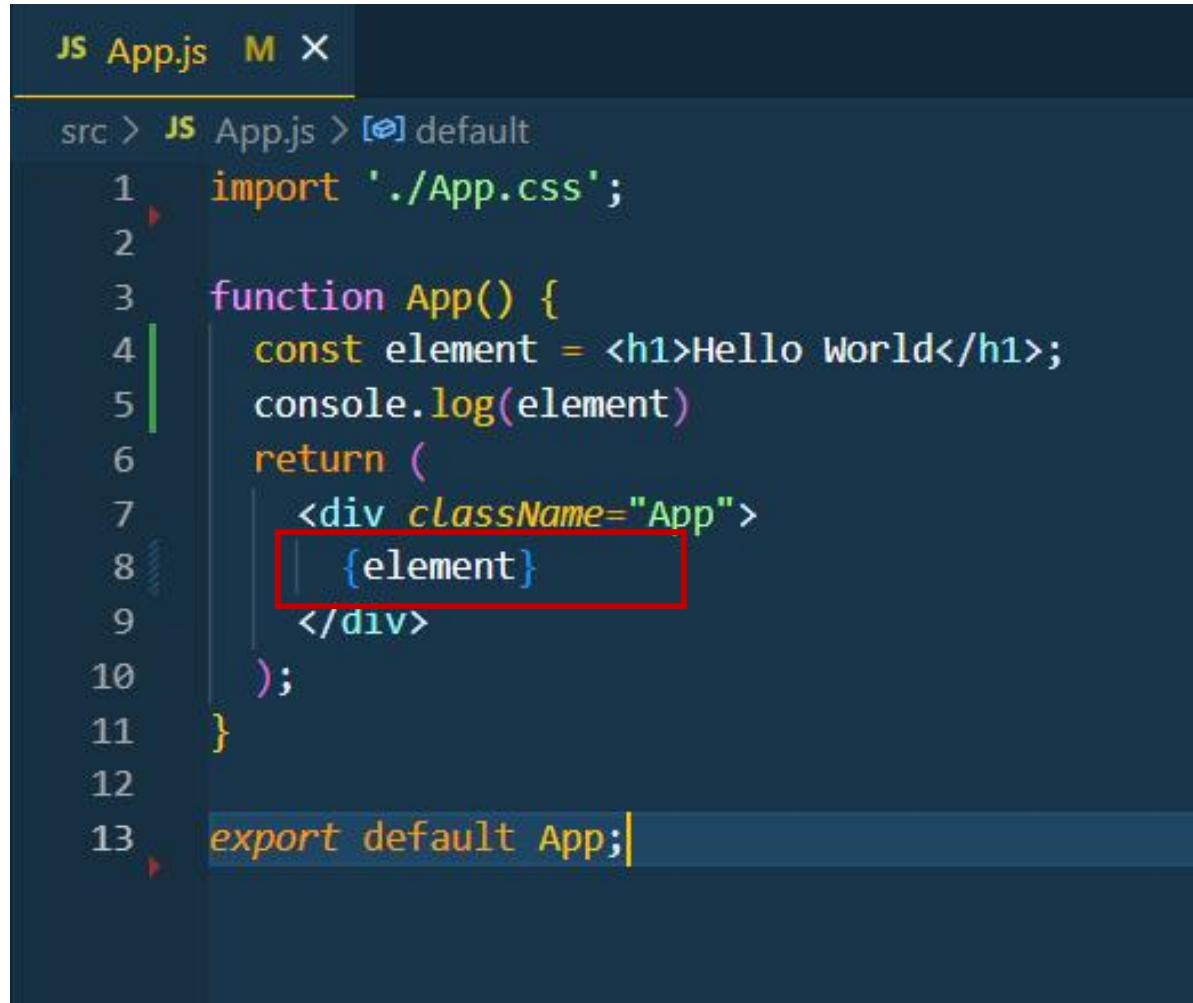


The screenshot shows the Chrome DevTools interface with the "Console" tab selected. A red box highlights the entire content area. Inside, an object representing a React element is expanded, showing its properties. One of the properties, "props", is also highlighted with a red box. The "props" object contains a single child node with the value "Hello World". Below the object tree, two warning messages are displayed in yellow text, both related to failed source map loading for specific extension files.

```
Object {  
  $$typeof: Symbol(react.element)  
  key: null  
  props: {children: 'Hello World'}  
  ref: null  
  type: "h1"  
  _owner: null  
  _store: {validated: false}  
  _self: undefined  
  _source: {fileName: 'C:\\\\Users\\\\NAHEZ SAKHARKAR\\\\OneDrive\\\\Desktop\\\\React\\\\react-app\\\\src\\\\index.js', lineNumber: 4, columnNumber: 17}  
  [[Prototype]]: Object  
}  
  
⚠ DevTools failed to load source map: Could not load content for chrome-extension://mooikfahbdckldjjndioackbalphokd/assets/atoms.js.map: System  
⚠ DevTools failed to load source map: Could not load content for chrome-extension://mooikfahbdckldjjndioackbalphokd/assets/polyfills.js.map: Sy
```

# Hello World

- Step 5 : To Display Hello World in the window type.



The screenshot shows a code editor window titled "JS App.js". The file path is "src > JS App.js > [?] default". The code is as follows:

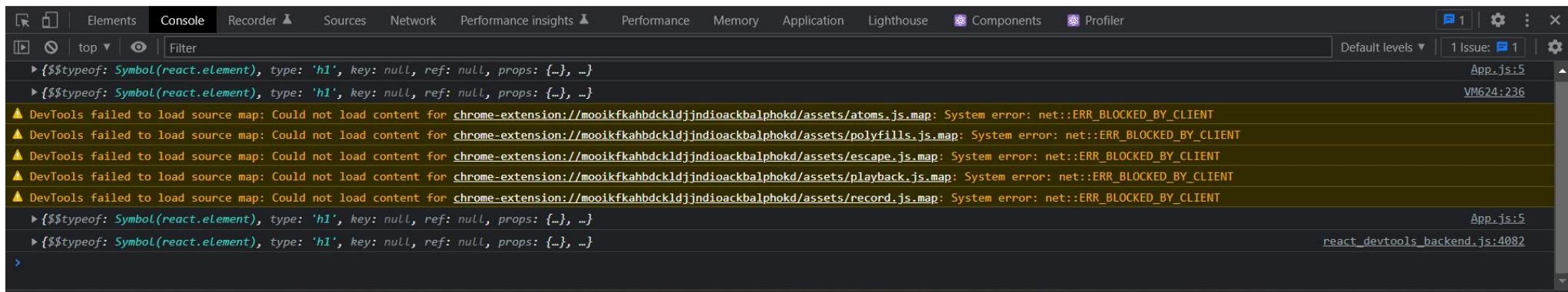
```
1 import './App.css';
2
3 function App() {
4     const element = <h1>Hello World</h1>;
5     console.log(element)
6     return (
7         <div className="App">
8             {element}
9         </div>
10    );
11 }
12
13 export default App;
```

A red box highlights the line of code containing the template literal `{element}` inside the `div` element.

# Hello World

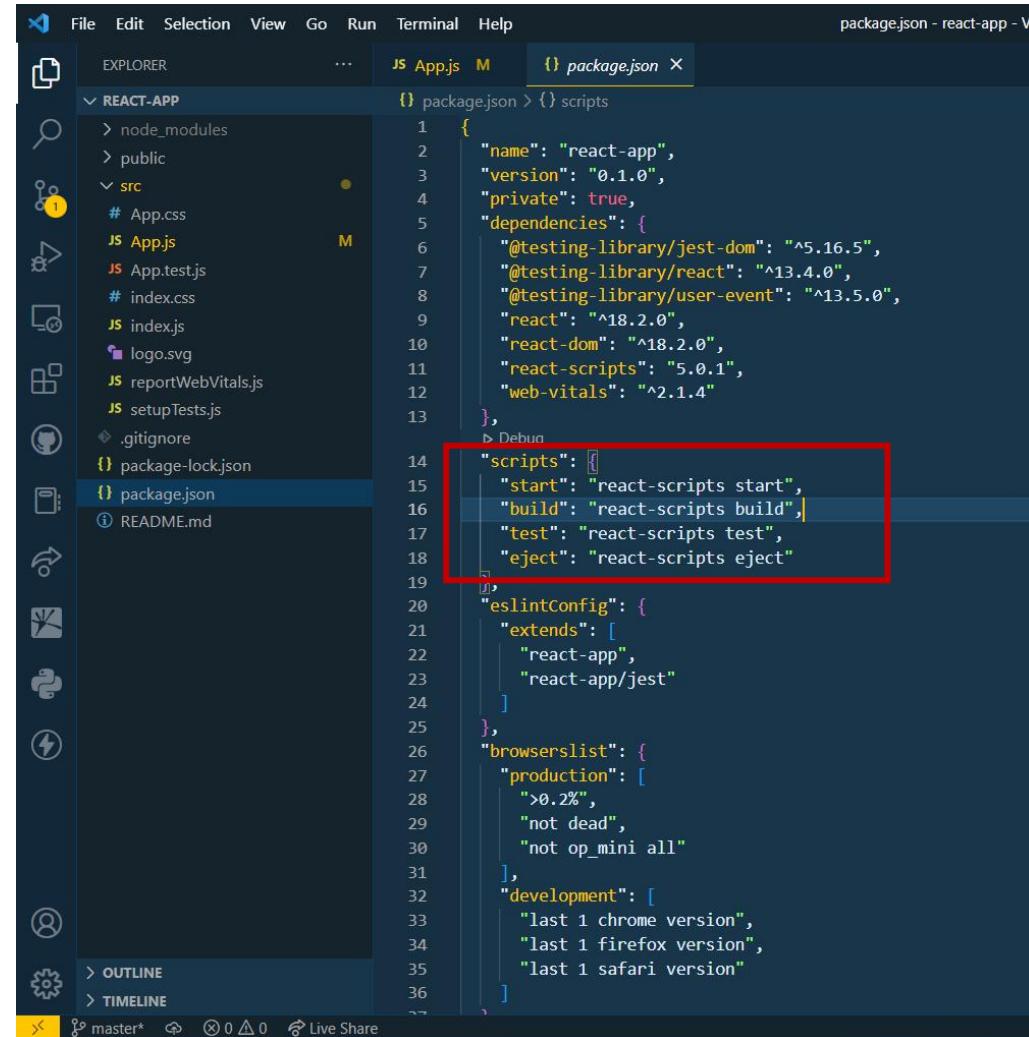
- Step 6 : To Display Hello World in the window type.

Hello World



# Custom Config for Advanced Developers

- Open package.json file



```
File Edit Selection View Go Run Terminal Help package.json - react-app - Vis
EXPLORER JS App.js M package.json X
REACT-APP
  > node_modules
  > public
  > src
    > App.css
    JS App.js M
    JS App.test.js
    # index.css
    JS index.js
    logo.svg
    JS reportWebVitals.js
    JS setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
  > OUTLINE
  > TIMELINE
package.json - react-app - Vis
1  {
2    "name": "react-app",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@testing-library/jest-dom": "^5.16.5",
7      "@testing-library/react": "^13.4.0",
8      "@testing-library/user-event": "^13.5.0",
9      "react": "^18.2.0",
10     "react-dom": "^18.2.0",
11     "react-scripts": "5.0.1",
12     "web-vitals": "^2.1.4"
13   },
14   "scripts": [
15     "start": "react-scripts start",
16     "build": "react-scripts build",
17     "test": "react-scripts test",
18     "eject": "react-scripts eject"
19   ],
20   "eslintConfig": {
21     "extends": [
22       "react-app",
23       "react-app/jest"
24     ]
25   },
26   "browserslist": {
27     "production": [
28       ">0.2%",
29       "not dead",
30       "not op_mini all"
31     ],
32     "development": [
33       "last 1 chrome version",
34       "last 1 firefox version",
35       "last 1 safari version"
36     ]
37   }
}
master* 0 0 △ 0 Live Share
```

# Custom Config

- Type ‘npm run eject’ in Terminal.

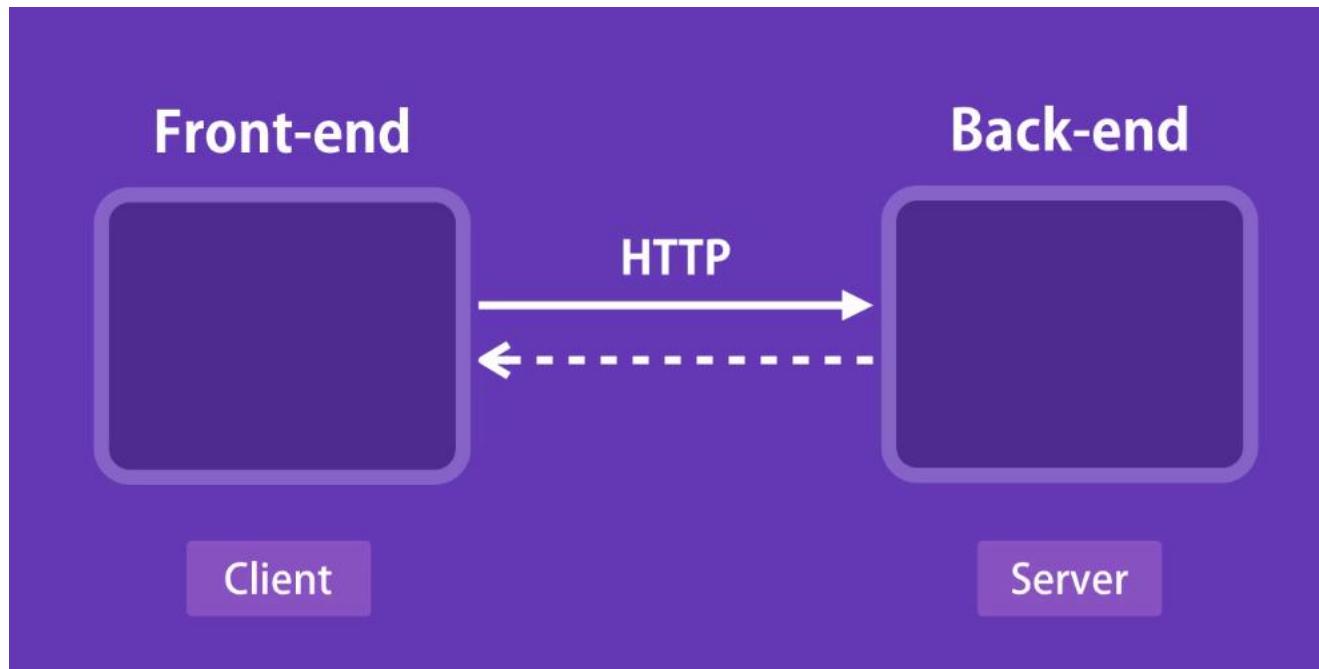
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure for "REACT-APP" containing files like App.css, App.js, App.test.js, index.css, index.js, logo.svg, reportWebVitals.js, setupTests.js, .gitignore, package-lock.json, package.json, and README.md.
- Editor:** The "package.json" file is open, showing its contents. A red box highlights the entire code block of the package.json file.
- Terminal:** The terminal tab is active, showing the command "PS C:\Users\WAHEZ SAKHARKAR\OneDrive\Desktop\React\react-app> npm run eject". Another red box highlights this command.
- Status Bar:** Shows the current file is "master" and the line number is "Ln 137, Col 1".

```
1 {  
2   "name": "react-app",  
3   "version": "0.1.0",  
4   "private": true,  
5   "dependencies": {  
6     "@babel/core": "^7.16.0",  
7     "@pmmmwh/react-refresh-webpack-plugin": "^0.5.3",  
8     "@svgr/webpack": "^5.5.0",  
9     "@testing-library/jest-dom": "^5.16.5",  
10    "@testing-library/react": "^13.4.0",  
11    "@testing-library/user-event": "^13.5.0",  
12    "babel-jest": "^27.4.2",  
13    "babel-loader": "^8.2.3",  
14    "babel-plugin-named-asset-import": "^0.3.8",  
15    "babel-preset-react-app": "^10.0.1",  
16    "bfj": "^7.0.2",  
17    "browserslist": "^4.18.1",  
18    "camelcase": "^6.2.1",  
19    "case-sensitive-paths-webpack-plugin": "^2.4.0",  
20    "css-loader": "^6.5.1",  
21    "css-minimizer-webpack-plugin": "^3.2.0",  
22    "dotenv": "^10.0.0",  
23    "dotenv-expand": "^5.1.0",  
24    "eslint": "^8.3.0",  
25    "eslint-config-react-app": "^7.0.1",  
26  },  
27  "scripts": {  
28    "start": "react-scripts start",  
29    "build": "react-scripts build",  
30    "test": "react-scripts test",  
31    "eject": "react-scripts eject"  
32  }  
33}
```

```
PS C:\Users\WAHEZ SAKHARKAR\OneDrive\Desktop\React\react-app> npm run eject  
> react-app@0.1.0 eject  
> react-scripts eject  
  
NOTE: Create React App 2+ supports TypeScript, Sass, CSS Modules and more without ejecting: https://reactjs.org/blog/2018/10/01/create-react-app-v2.html  
✓ Are you sure you want to eject? This action is permanent. ... yes  
Ejecting...  
  
Copying files into C:\Users\WAHEZ SAKHARKAR\OneDrive\Desktop\React\react-app  
Adding \config\env.js to the project
```

# Full Stack Architecture



```
index.html App.js M HelloReactComponent.jsx 1, U X
```

src > components > HelloReactComponent.jsx > HelloReactComponent

```
1 import React, { Component } from "react";
2
3 class HelloReactComponent extends Component {
4
5     state = {
6
7     }
8
9     render() {
10        return (
11            )
12        }
13    }
14 }
```

When some value is returned from a  
React render(); you need to make  
sure it is a single parent

$\frac{<\text{div}>}{</\text{div}>}$  ✓

$\frac{<\text{h1}>}{<\text{h2}>} \frac{<\text{h2}>}{<\text{h1}>}$  ✗

$\frac{<\text{p}>}{<\text{p}>}$  ✗

```
1  export function HelloFunctionalComponent() {  
2  |  
3  |   return (  
4  |   |     <div>  
5  |   |       <h3>Hello from Functional Component</h3>  
6  |   |     </div>  
7  |   )  
8 }
```

\* Why functional component

① Functional components are  
easy to read & write

② It is easy to separate container & presentational  
components

③ It is 100% JS ④ Can use best practices

⑤ Increases the application performance

## Props

→ Normal HTML have attributes

``

attribute

→ React Component have props

`<NameComponent name="Zantab" />`

} prop → equivalent to  
attribute ;

In HTML attribute

names are pre-fixed

① Attributes in HTML are only strings.

Props can be any JS object

\*

Props in React can have  
any names

String, object, another component,  
function

`<NameComponent  
    name="Zartab" />`

Props are passed to

### ① Class Component:

→ available in class component as an object → props

### ② Functional Component

→ available as an argument.

## State

State is a plain JavaScript object used by React to represent information about the component's current situations.

```
class SomeComponent extends Component{
```

```
state = {
```

```
}
```

```
render() {
```

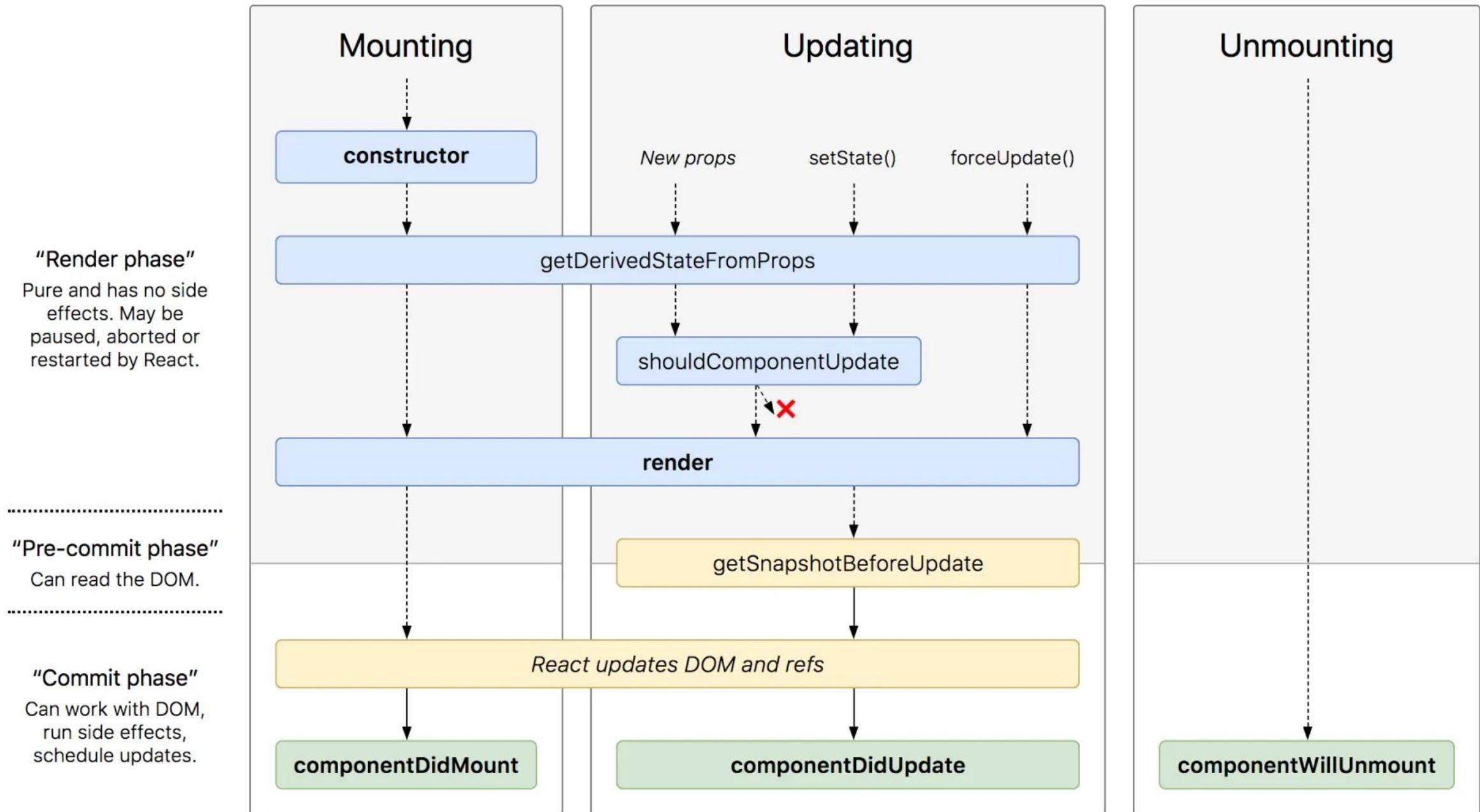
```
}
```

```
}
```

"State is like props; but it is private & it is fully controlled by the Component."

# What are React lifecycle methods?

- You can think of React lifecycle methods as the series of events that happen from the birth of a React component to its death.
- Every component in React goes through a lifecycle of events. I like to think of them as going through a cycle of birth, growth, and death.
- **Mounting** – Birth of your component
- **Update** – Growth of your component
- **Unmount** – Death of your component



# TickerComponent

1. React calls TickerComponent's render()
2. When TickerComponent's output is inserted in DOM React will call the componentDidMount()
3. componentDidMount() --> asks the browser to set up a timer and after every second it calls the tick()
4. After every one second when browser calls the tick() --> tick() internally calls setState() to update the state.
5. setState() as per the lifecycle invokes the render() and hence every second the screen gets updated

## Hooks & State

Hooks were introduced in React 16.8

Special type of functions, which allows us to break a class component into smaller functional components.

- \* In previous versions of React, if a component needed state, only way was using class Component

useState → hook

const [variable, setVariable] = useState(defaultValue)

↳ state      ↳ function to      ↓      ↳  
update the      hook for      default value  
state      creating state      for state  
in functional  
component

const [age, setAge] = useState(0);

const [fruit, setFruit] = useState('Apple');

const [colors, setColors] = useState([]);

const [address, setAddress] = useState({})

{  
  city: 'Mumbai',  
  pin: '400000'}

3

4

5

Decrement

Increment

## \* DOM \*

DOM → Document Object Model

→ represents the UI of the application.

\* Everytime application state gets updated ; Dom is updated  
    ↳ represent the change

→ Dom is represented as a tree data structure.

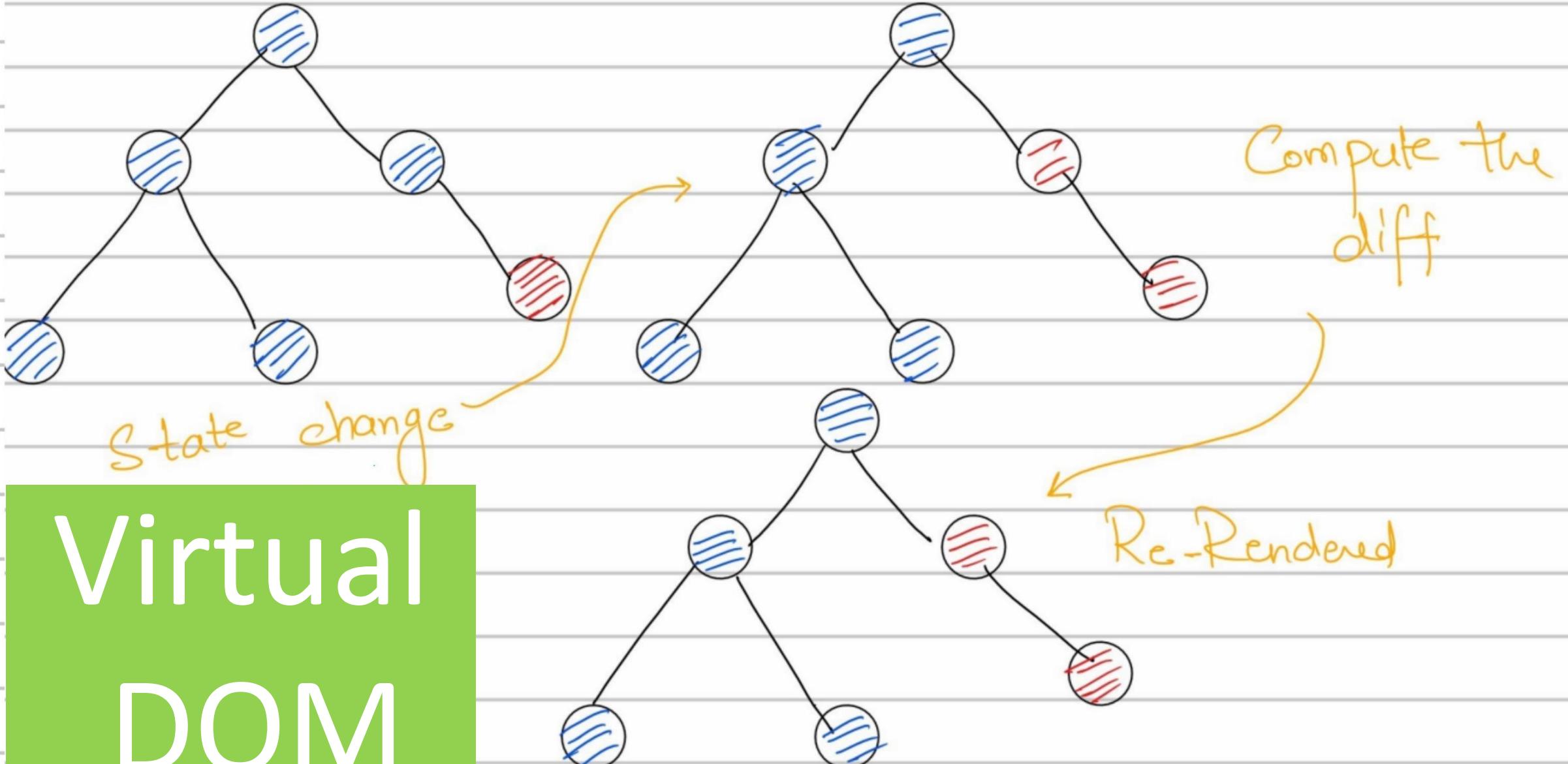
|  
| re-rendering of the page & its  
| child components are time consuming operations

## \*Virtual Dom\*

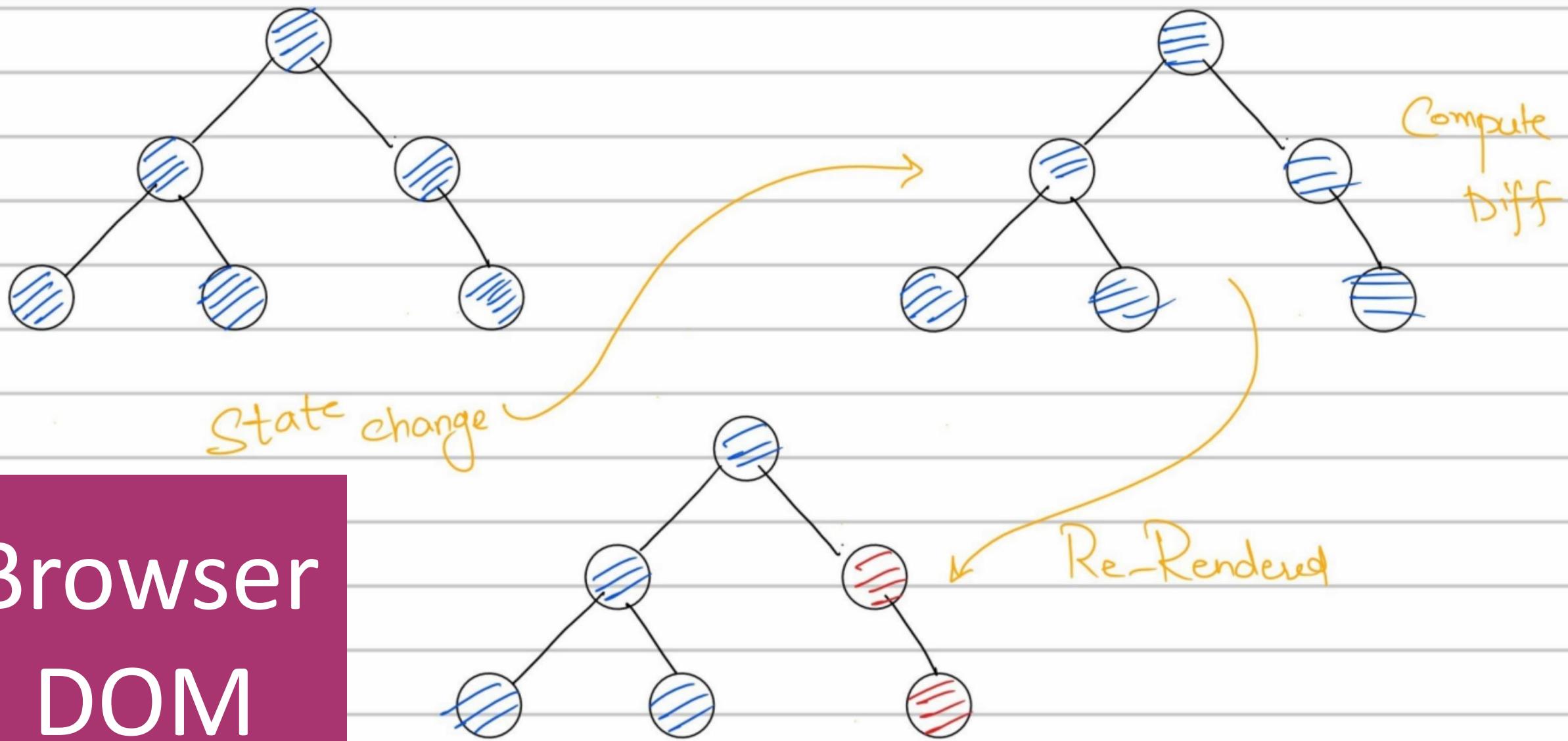
→ When new elements are added to the UI, a Virtual Dom which is represented like a tree is created.  
Each element is a node in a tree.  
If state of these element changes; a new Virtual DOM tree is created.

The tree is compared with previous virtual Dom Tree

→ Differed



# Browser DOM



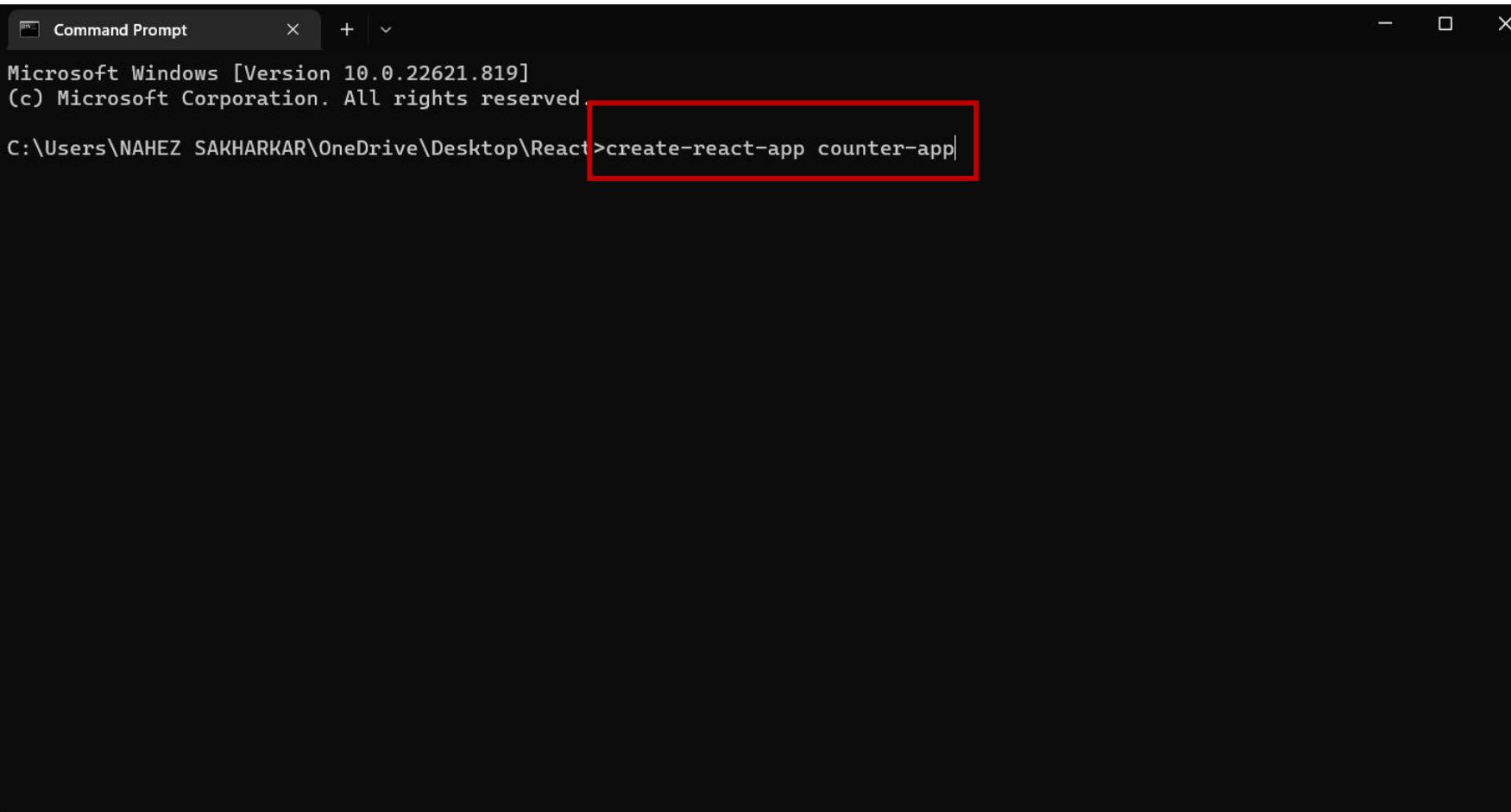


# Components

- *Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.*
- *Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.*
- *A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.*
- *A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial.*

# Setting up the Project

- Step 1 : Open Command Prompt and Type

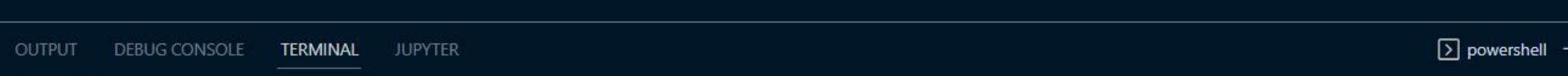


A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following text:  
Microsoft Windows [Version 10.0.22621.819]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React>create-react-app counter-app

The command "create-react-app counter-app" is highlighted with a red rectangular box.

# Setting up the Project

- Step 2 : Open Terminal and Install Bootstrap



Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

```
PS C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React\counter-app> npm i bootstrap
npm WARN EBADENGINE Unsupported engine {
  npm WARN EBADENGINE   package: '@jest/expect-utils@29.3.1',
  npm WARN EBADENGINE   required: { node: '^14.15.0 || ^16.10.0 || >=18.0.0' },
  npm WARN EBADENGINE   current: { node: 'v16.6.1', npm: '8.19.2' }
  npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
```

# Setting up the Project

- Step 3 : Import bootstrap in index.jsx file

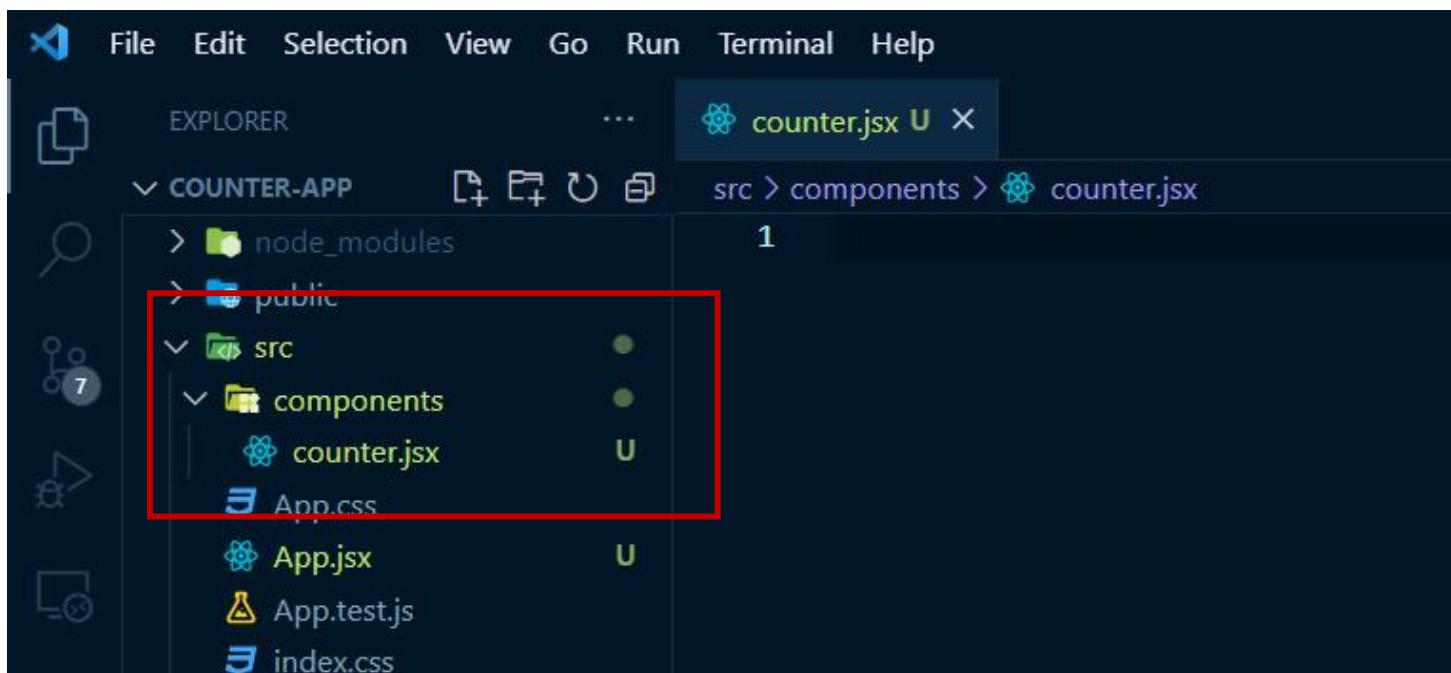
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "COUNTER-APP". It includes "node\_modules", "public", and a "src" folder containing "App.css", "App.jsx", "App.test.js", "index.css", "index.jsx" (which is currently selected), "logo.svg", "reportWebVitals.js", and "setupTests.js". Other files like ".gitignore", "package-lock.json", "package.json", and "README.md" are also listed.
- Terminal (Top Right):** Shows the path "index.jsx - counter-app - Visual Studio Code" and the code editor area.
- Code Editor (Right):** Displays the content of "index.jsx". A red box highlights the line "import 'bootstrap/dist/css/bootstrap.css';".

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6 import 'bootstrap/dist/css/bootstrap.css';
7
8 const root = ReactDOM.createRoot(document.getElementById('root'));
9 root.render(
10   <React.StrictMode>
11     <App />
12   </React.StrictMode>
13 );
14
15 // If you want to start measuring performance in your app, pass a function
16 // to log results (for example: reportWebVitals(console.log))
17 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
18 reportWebVitals();
```

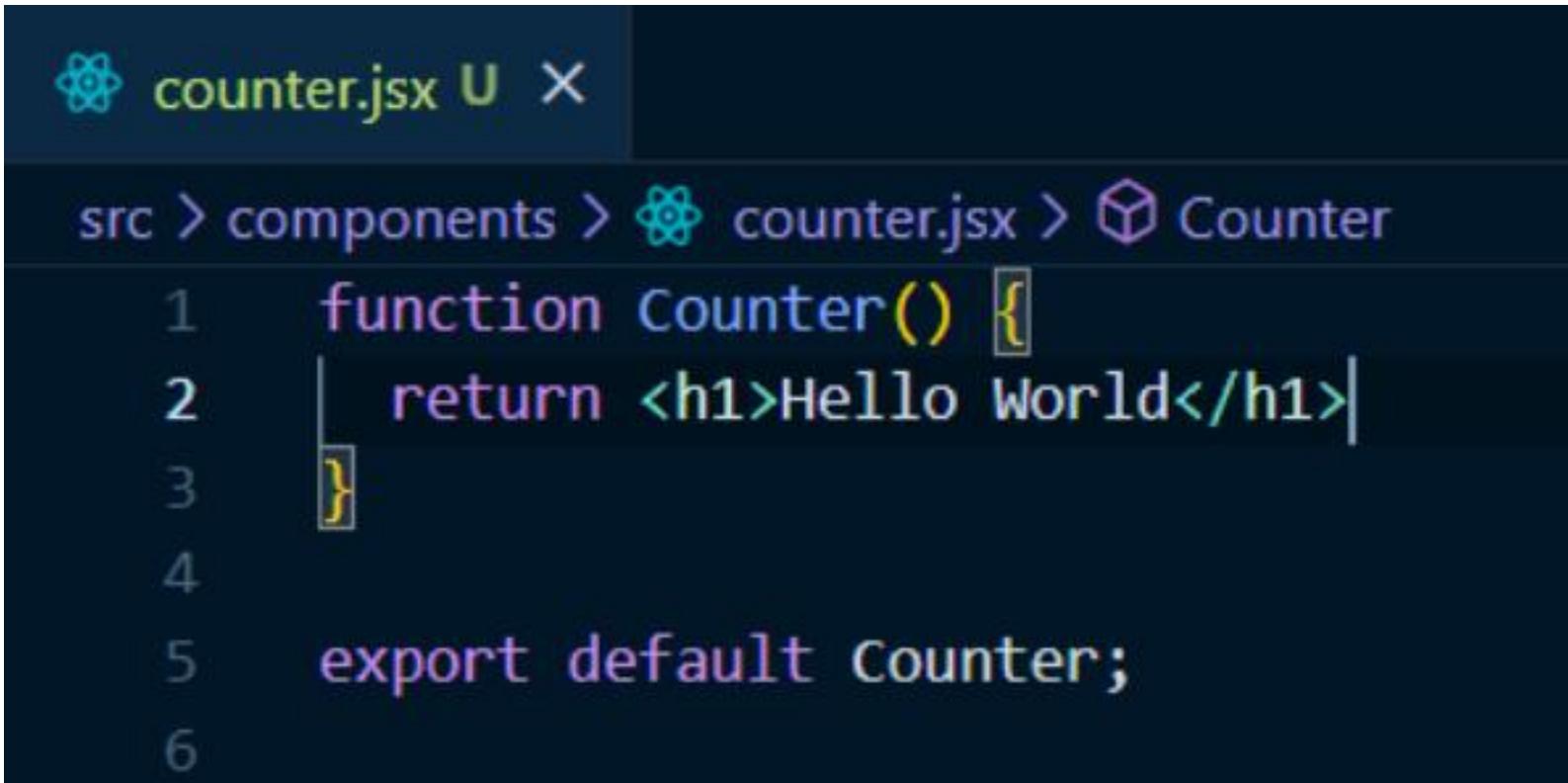
# Your First React Component

- Step 1 : Create New Folder in src Directory named ‘components’.
- Step 2 : Create New file in components folder named ‘counter.jsx’



# Your First React Component

- Step 3 : Type ‘ffc’ for auto generating function component boiler plate. Name the component ‘Counter’.

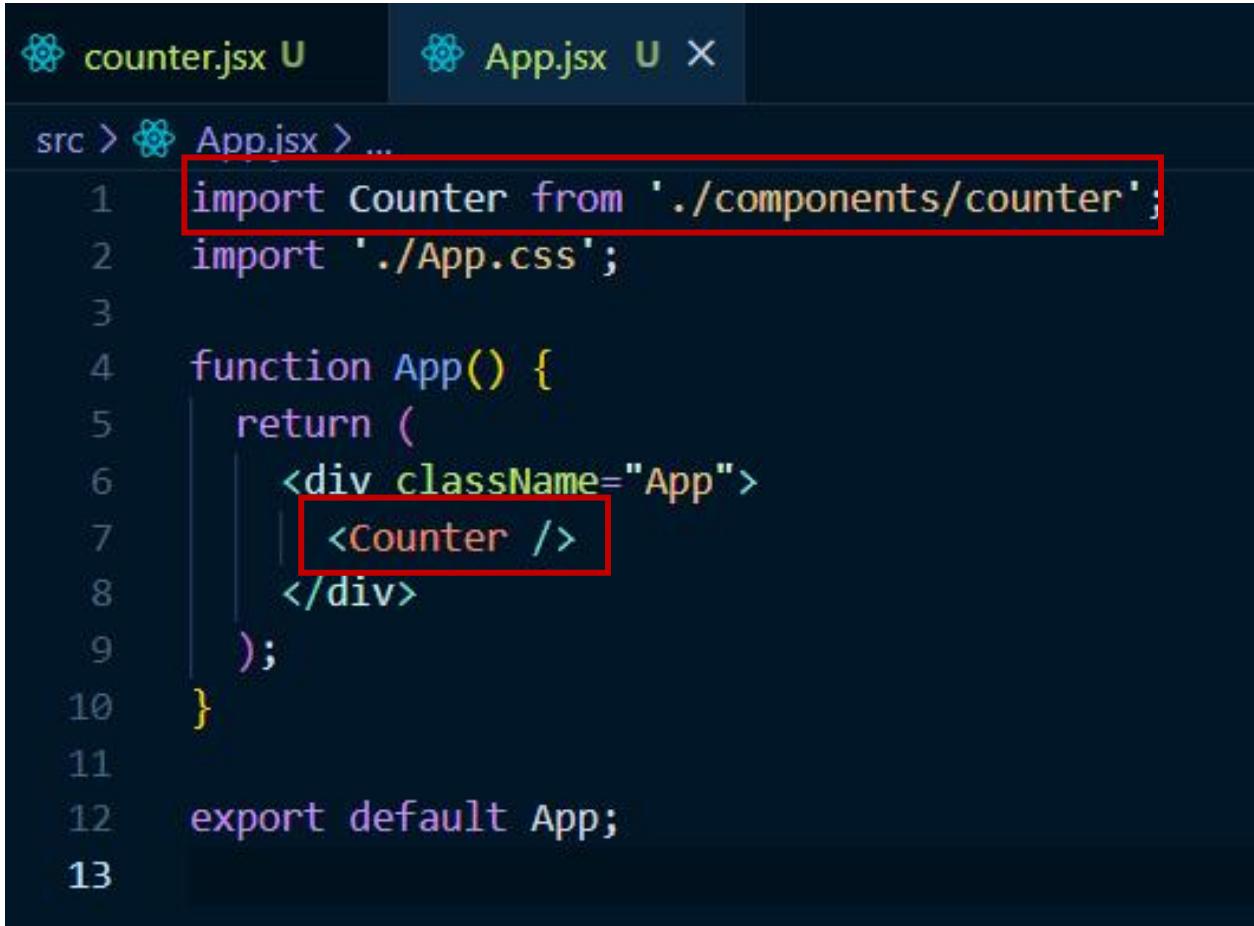


The screenshot shows a code editor window with a dark theme. The file is named "counter.jsx". The file path is "src > components > counter.jsx". The component is named "Counter". The code is as follows:

```
1  function Counter() {  
2      return <h1>Hello World</h1>  
3  }  
4  
5  export default Counter;  
6
```

# Your First React Component

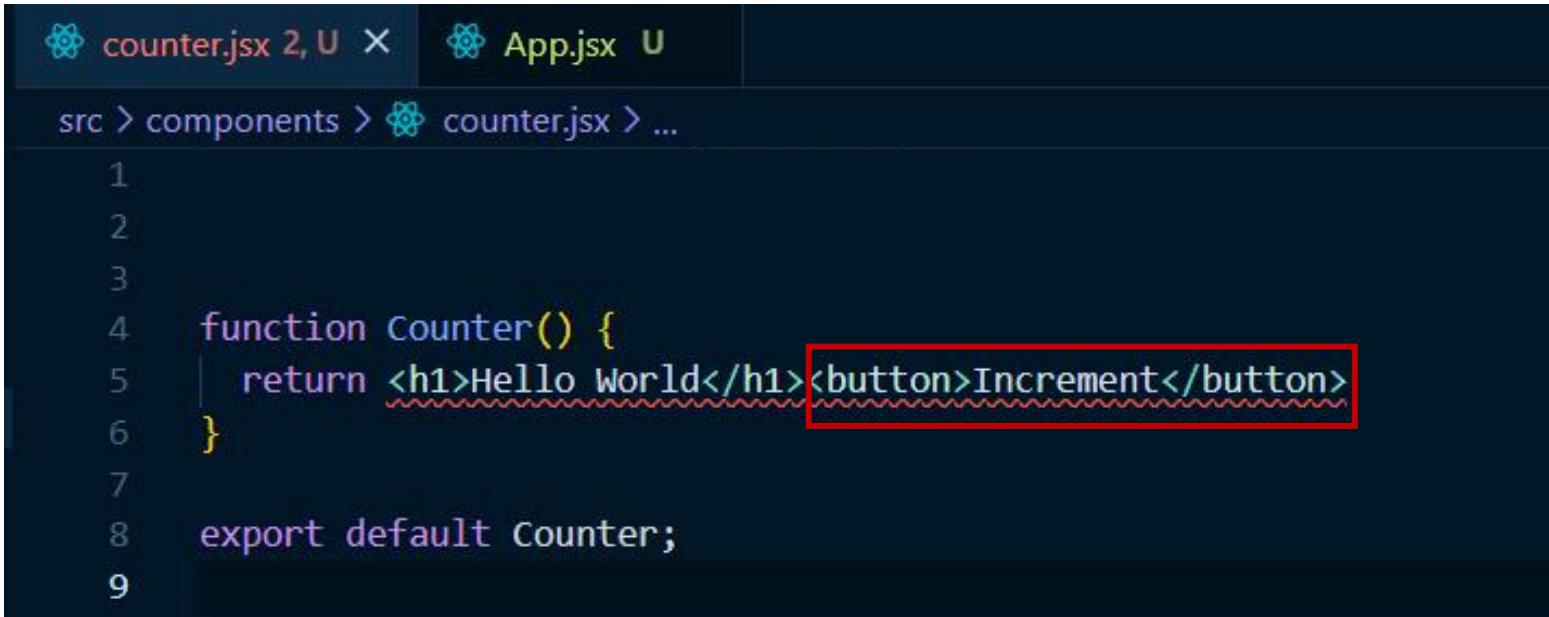
- Step 4 : Open App.jsx file and import the component.



```
counter.jsx U      App.jsx U X
src > App.jsx > ...
1  import Counter from './components/counter';
2  import './App.css';
3
4  function App() {
5    return (
6      <div className="App">
7        <Counter />
8      </div>
9    );
10 }
11
12 export default App;
13
```

# Specifying Children

- Step 5 : Add Button inside counter component .



The screenshot shows a code editor with two tabs: "counter.jsx" and "App.jsx". The "counter.jsx" tab is active, showing the following code:

```
1
2
3
4  function Counter() {
5    return <h1>Hello World</h1><button>Increment</button>
6  }
7
8  export default Counter;
9
```

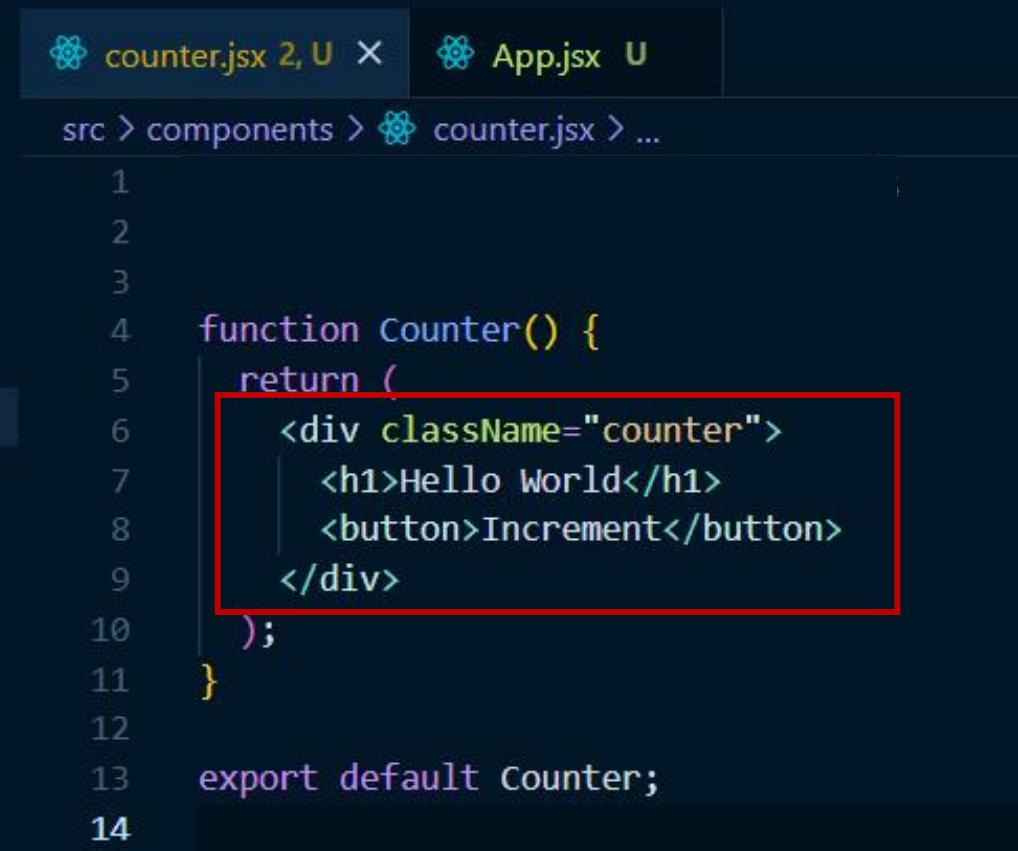
A red box highlights the closing tag of the button element, "</button>", indicating a syntax error.

This will show an error !

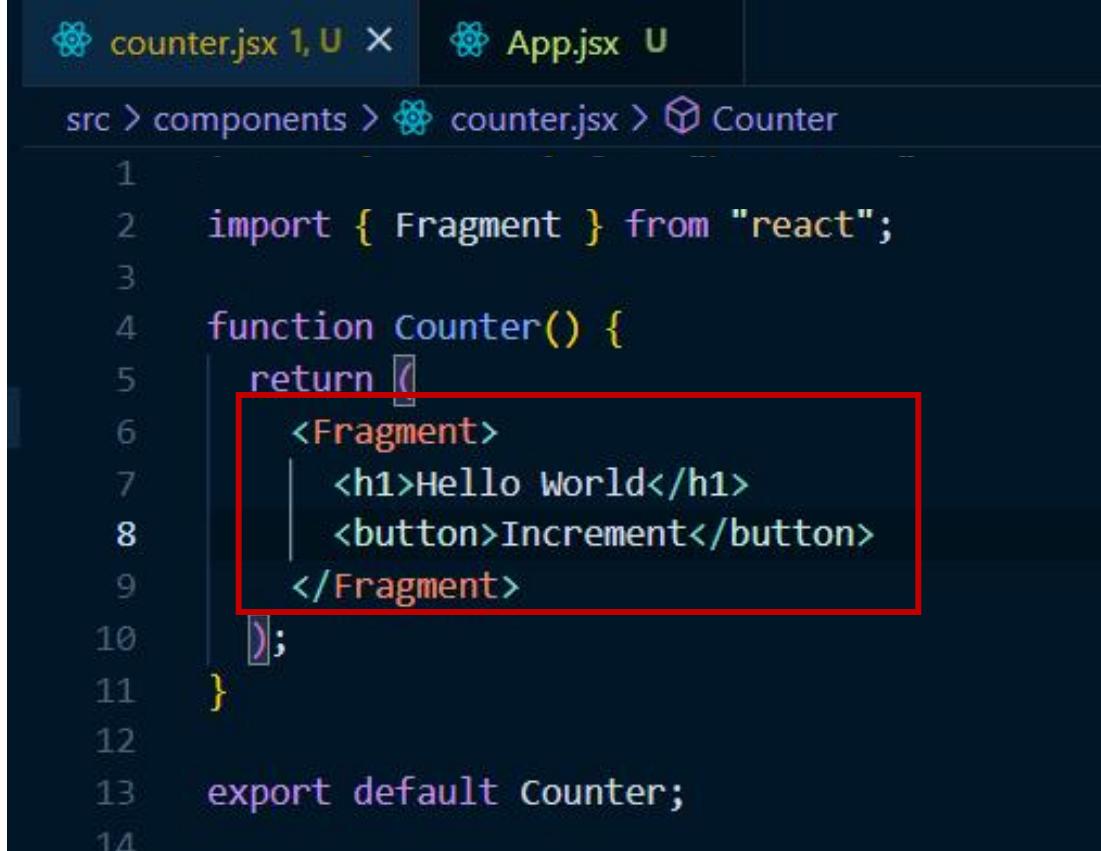
- Fix this by wrapping both elements with a parent div.

# Specifying Children

- Step 6 : Wrapping Elements with parent Div or React Fragment.



```
counter.jsx 2, U X App.jsx U
src > components > counter.jsx > ...
1
2
3
4 function Counter() {
5   return (
6     <div className="counter">
7       <h1>Hello World</h1>
8       <button>Increment</button>
9     </div>
10  );
11}
12
13 export default Counter;
14
```



or

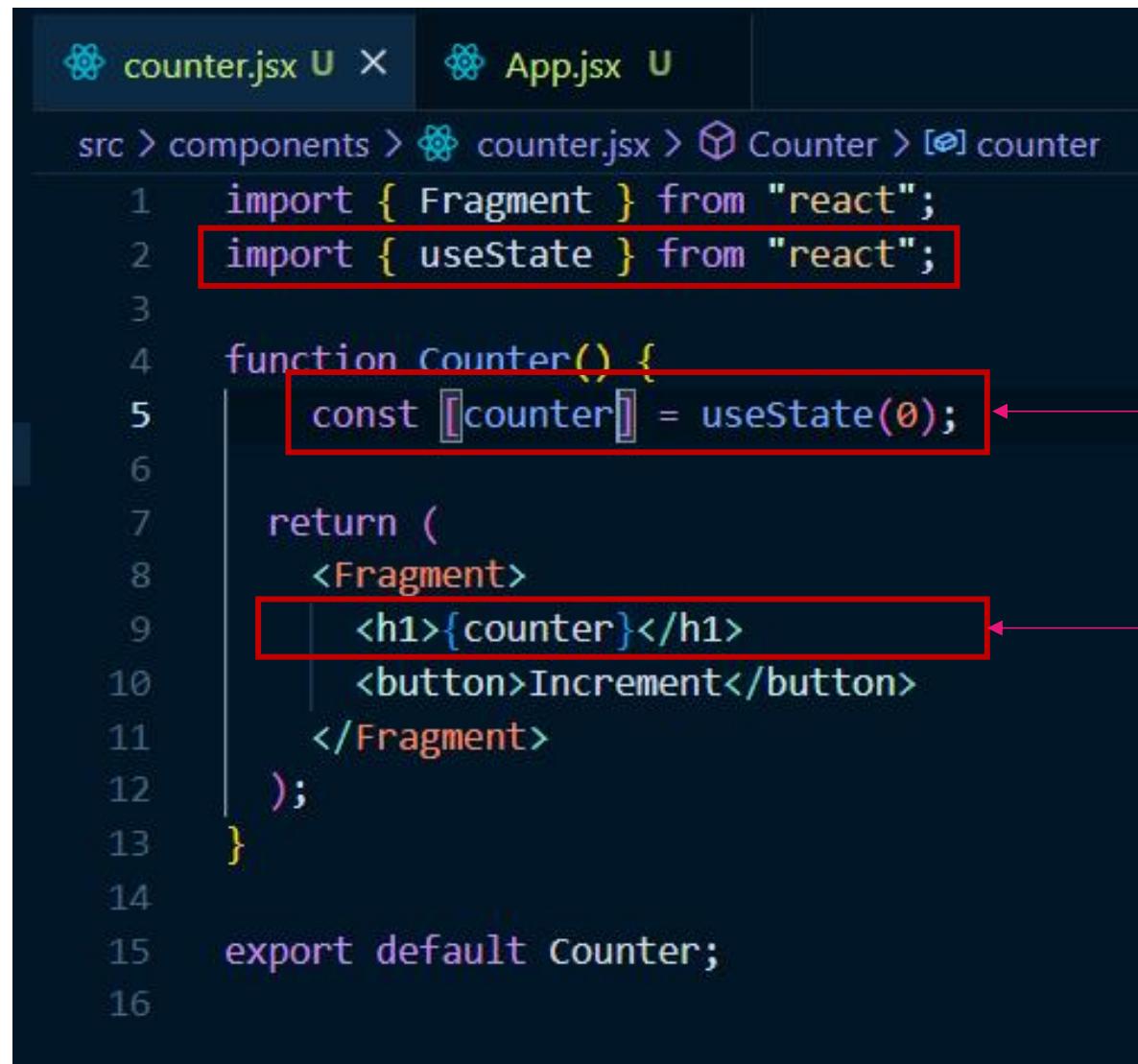
```
counter.jsx 1, U X App.jsx U
src > components > counter.jsx > Counter
1
2 import { Fragment } from "react";
3
4 function Counter() {
5   return (
6     <Fragment>
7       <h1>Hello World</h1>
8       <button>Increment</button>
9     </Fragment>
10  );
11}
12
13 export default Counter;
14
```

# Specifying Children

- Output:



# Embedding Expressions



```
counter.jsx  X  App.jsx  U
src > components > counter.jsx > Counter > [o] counter
1 import { Fragment } from "react";
2 import { useState } from "react";
3
4 function Counter() {
5   const [counter] = useState(0);
6
7   return (
8     <Fragment>
9       <h1>{counter}</h1>
10      <button>Increment</button>
11      </Fragment>
12    );
13  }
14
15 export default Counter;
```

useState Hook

Expression

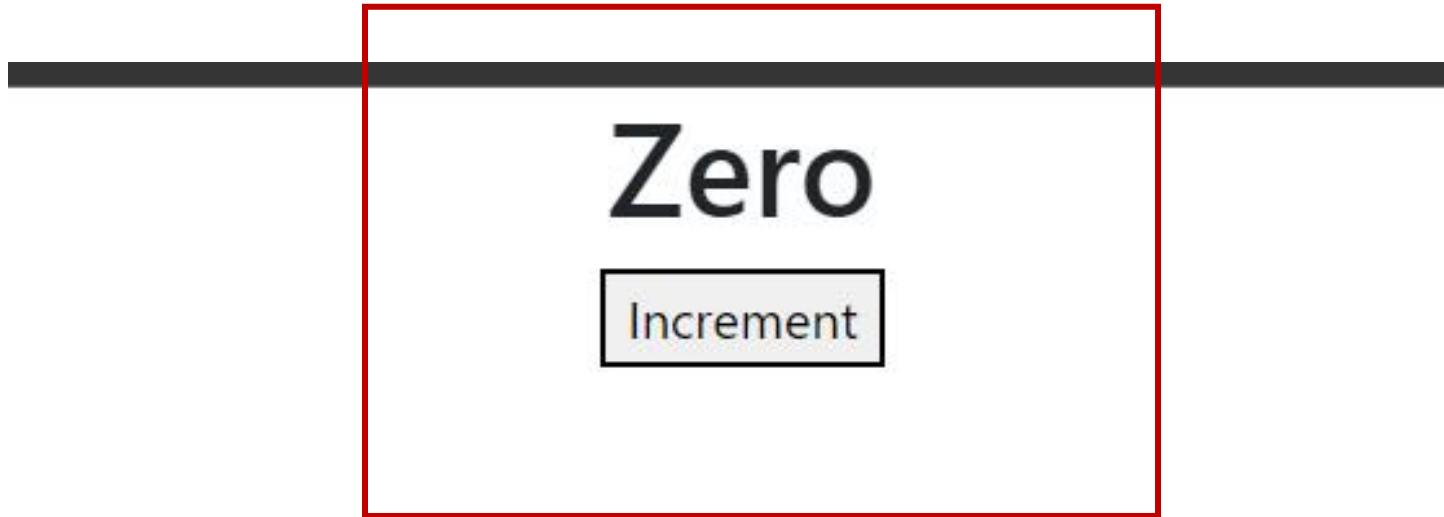
# Embedding Expressions



```
counter.jsx U X App.jsx U
src > components > counter.jsx > ...
1 import { Fragment } from "react";
2 import { useState } from "react";
3
4 function Counter() {
5   const [counter] = useState(0);
6
7   function formatCount() {
8     return counter === 0 ? "Zero" : counter;
9   }
10
11  return (
12    <Fragment>
13      <h1>{formatCount()}</h1>
14      <button>Increment</button>
15    </Fragment>
16  );
17}
18
19 export default Counter;
20 |
```

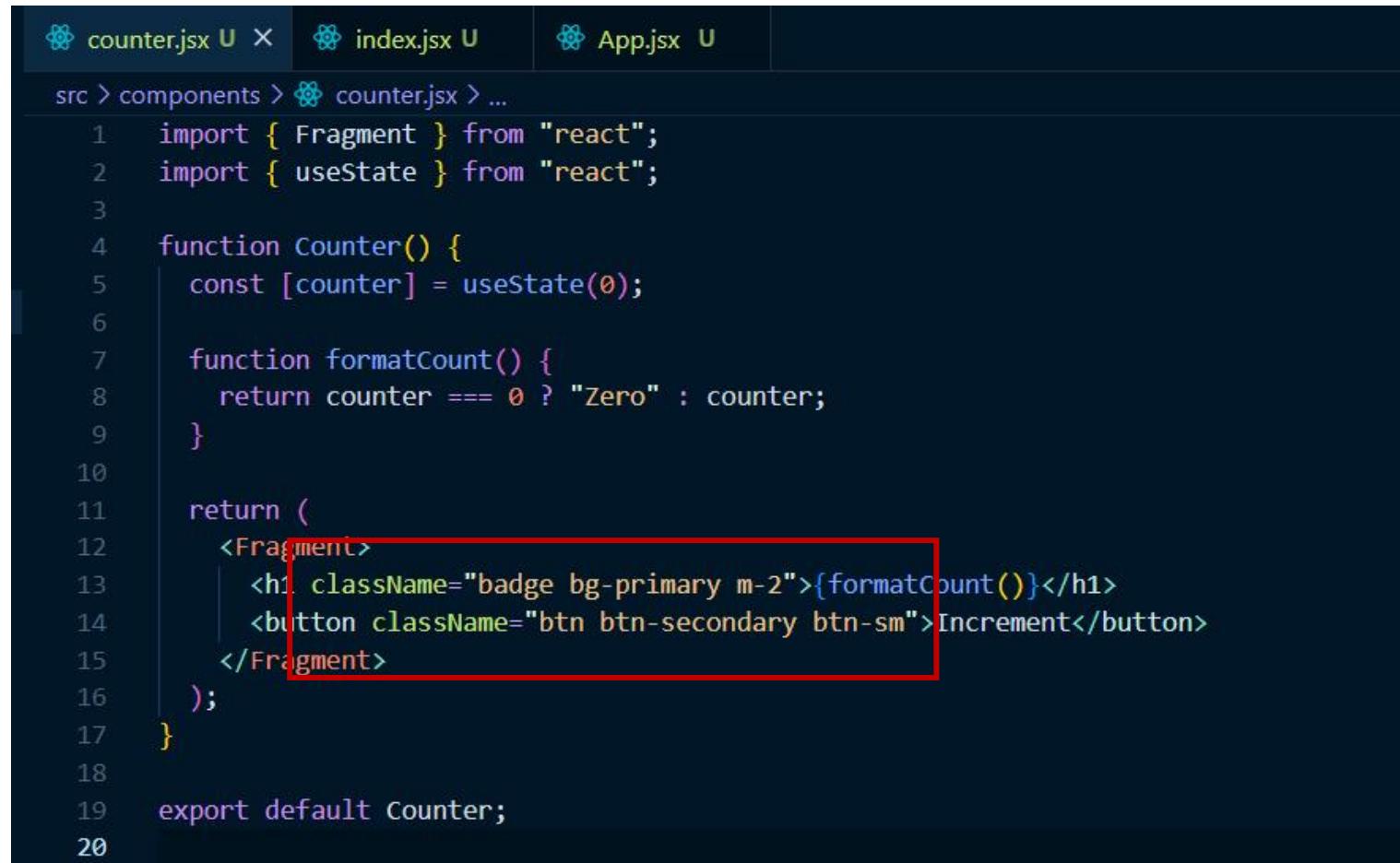
# Embedding Expression

- Output:



# Setting Attributes

- Step 1: Add className instead of class to import bootstrap classes.



```
counter.jsx U X index.jsx U App.jsx U
src > components > counter.jsx > ...
1  import { Fragment } from "react";
2  import { useState } from "react";
3
4  function Counter() {
5    const [counter] = useState(0);
6
7    function formatCount() {
8      return counter === 0 ? "Zero" : counter;
9    }
10
11   return (
12     <Fragment>
13       <h1 className="badge bg-primary m-2">{formatCount()}</h1>
14       <button className="btn btn-secondary btn-sm">Increment</button>
15     </Fragment>
16   );
17 }
18
19 export default Counter;
20
```

# Setting Attributes

- Output:



# Rendering Classes Dynamically

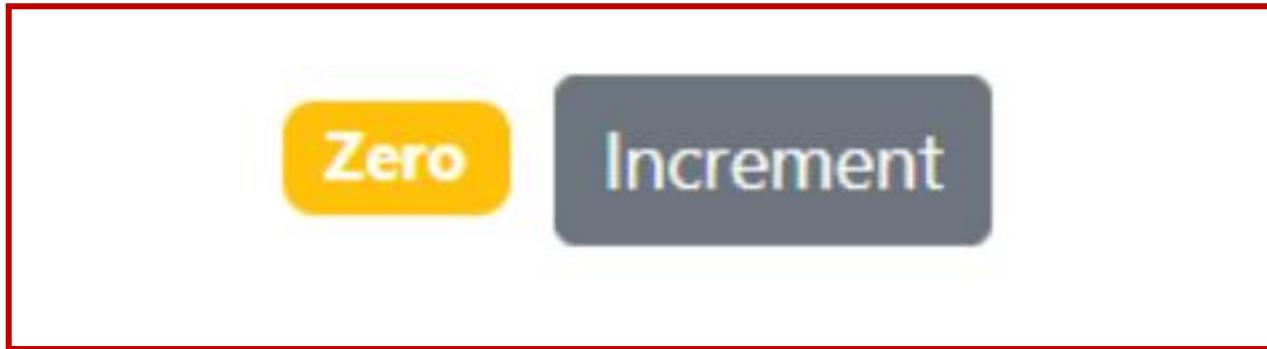
- Create Function and Call it inside className.

Don't forget to add the space

```
6   function formatCount() {
7     return counter === 0 ? "Zero" : counter;
8   }
9
10
11  function getBadgeClasses() {
12    let classes = "badge m-2 ";
13    classes += counter === 0 ? "bg-warning" : "bg-primary";
14    return classes
15  }
16
17  return (
18    <Fragment>
19      <h1 className={getBadgeClasses()}>{formatCount()}</h1>
20      <button className="btn btn-secondary btn-sm">Increment</button>
21    </Fragment>
22  );
23}
24
```

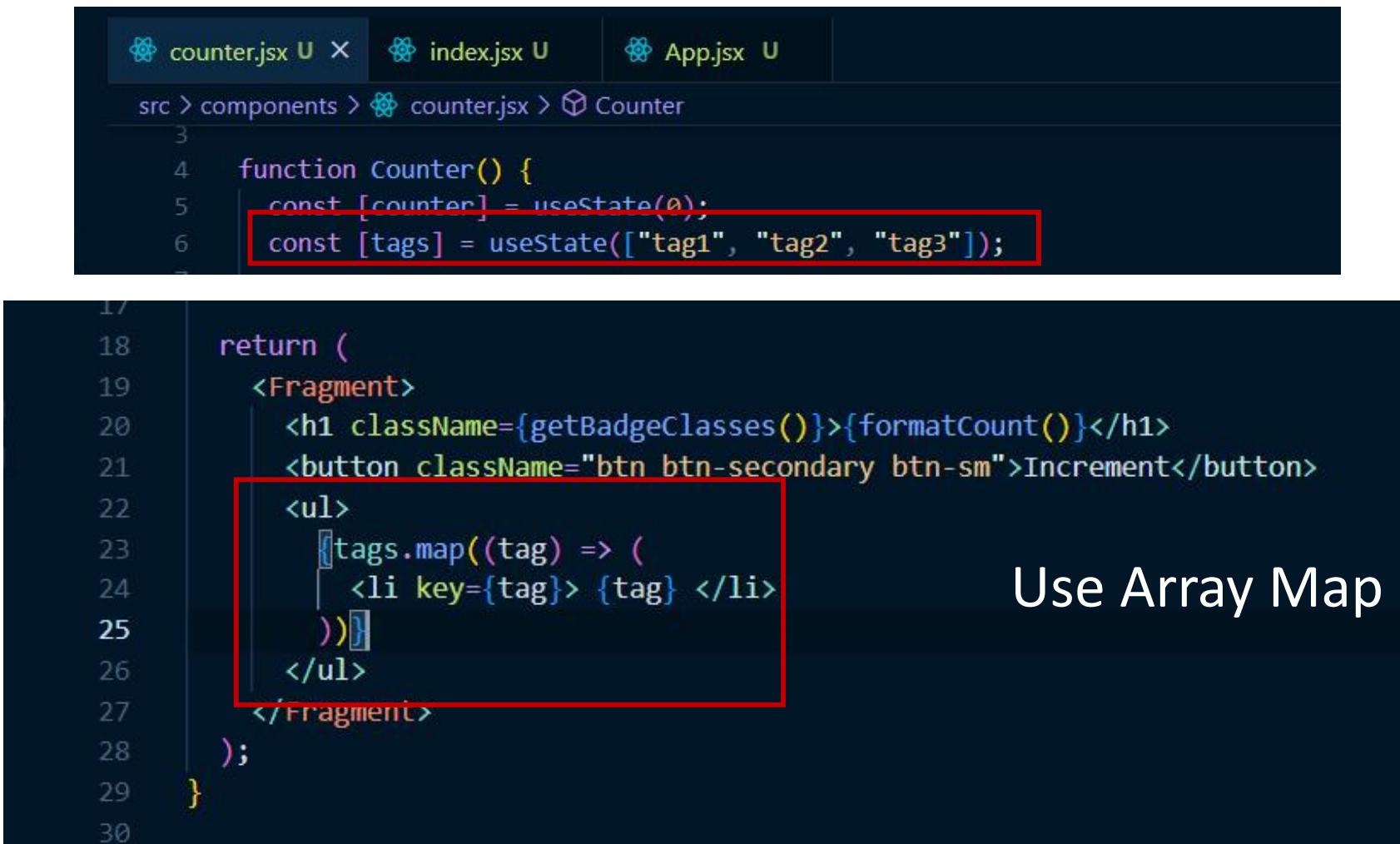
# Setting Attributes

- Output:



# Rendering Lists

Add another useState with Array values.

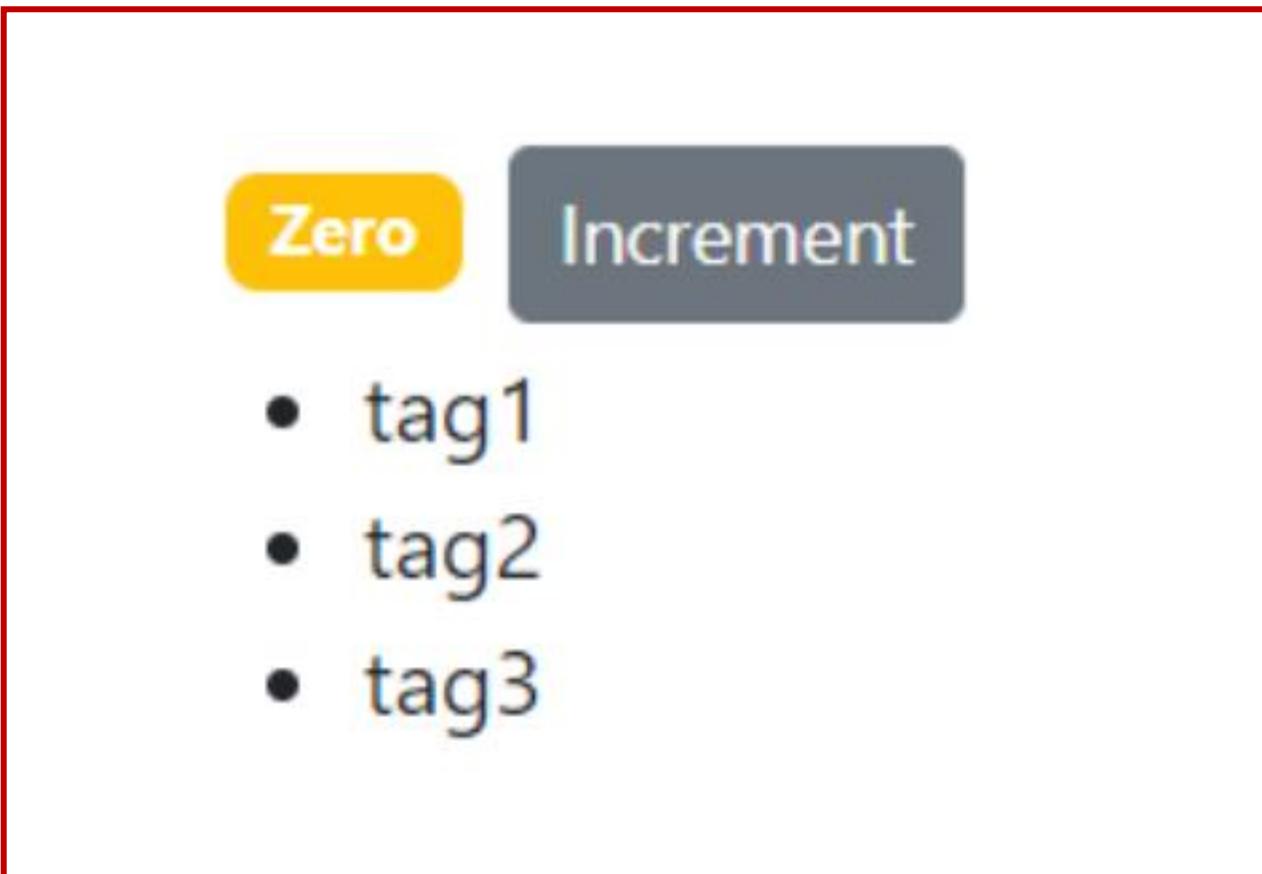


```
counter.jsx U X index.jsx U App.jsx U
src > components > counter.jsx > Counter
3
4   function Counter() {
5     const [counter] = useState(0);
6     const [tags] = useState(["tag1", "tag2", "tag3"]);
7
8
9
10
11
12
13
14
15
16
17
18   return (
19     <Fragment>
20       <h1 className={getBadgeClasses()}>{formatCount()}</h1>
21       <button className="btn btn-secondary btn-sm">Increment</button>
22       <ul>
23         {tags.map((tag) => (
24           <li key={tag}> {tag} </li>
25         ))}
26       </ul>
27     </Fragment>
28   );
29 }
```

Use Array Map

# Rendering Lists

- Output



# Conditional Rendering

The screenshot shows a code editor with three tabs: counter.jsx, index.jsx, and App.jsx. The counter.jsx tab is active, displaying the following code:

```
src > components > counter.jsx > ...
14     classes += counter === 0 ? "bg-warning" : "bg-primary";
15     return classes;
16 }
17
18 function renderTags() {
19     if (tags.length === 0) return <p>"There are No Tags!"</p>;
20     return (
21         <ul>
22             {tags.map((tag) => (
23                 <li key={tag}> {tag} </li>
24             ))
25         </ul>
26     );
27 }
28
29 return (
30     <Fragment>
31         <h1 className={getBadgeClasses()}>{formatCount()}</h1>
32         <button className="btn btn-secondary btn-sm">Increment</button>
33         <div>
34             {tags.length === 0 && "Please create a new Tag!"}
35             {renderTags()}
36         </div>
37     </Fragment>
38 );
39 }
40
41 export default Counter;
```

Two specific sections of the code are highlighted with red boxes:

- A red box surrounds the `renderTags()` function definition and its body, which contains a conditional check and a map operation.
- A red box surrounds the conditional rendering logic within the `<div>` element at lines 34-36, specifically the expression `{tags.length === 0 && "Please create a new Tag!"}`.

# Handling Events

```
29   function handleIncrement() {
30     console.log("Button Clicked!");
31   }
32
33   return (
34     <Fragment>
35       <h1 className={getBadgeClasses()}>{formatCount()}</h1>
36       <button onClick={handleIncrement} className="btn btn-secondary btn-sm">
37         Increment
38       </button>
39     </Fragment>
40   );
41 }
```

# Updating the State

- Add setCounter in useState of counter.

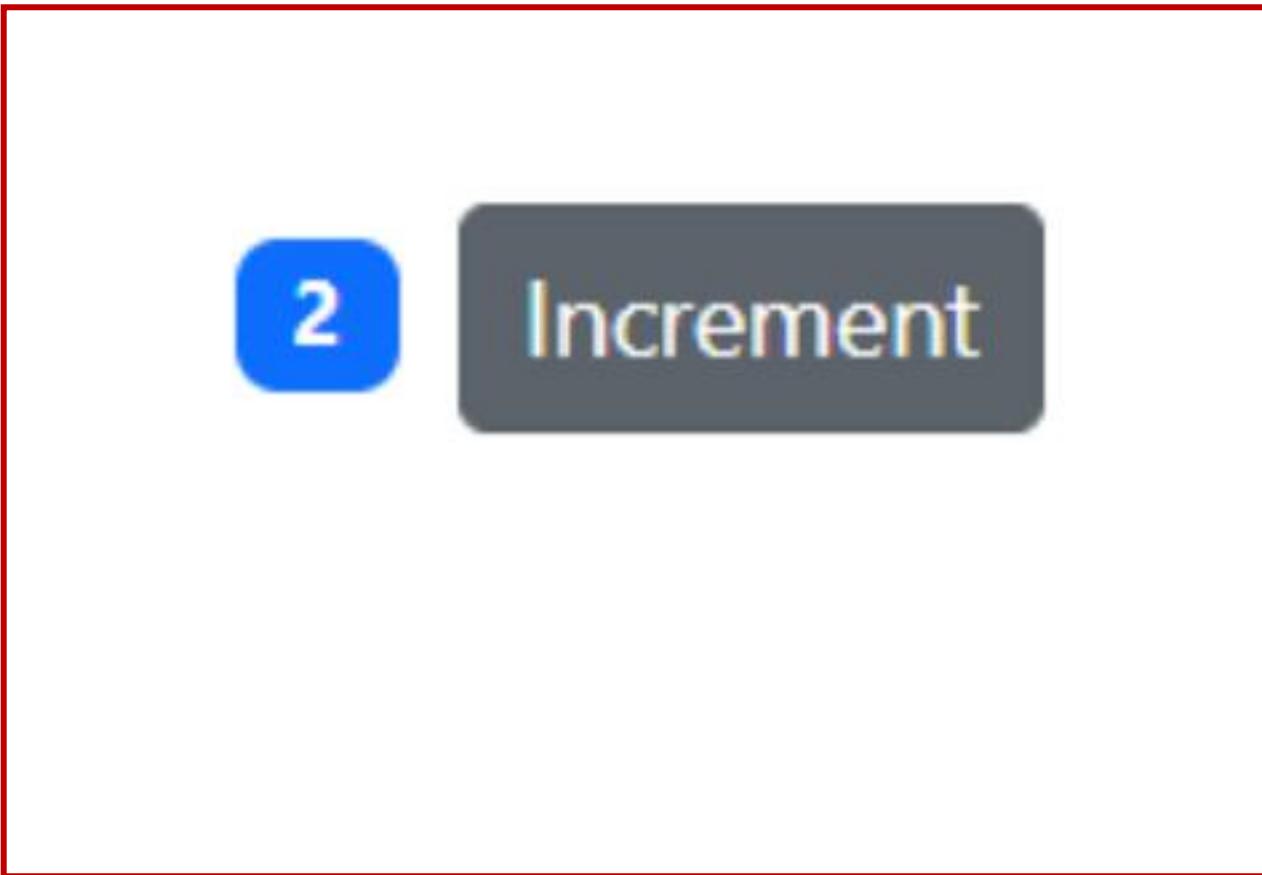
```
5  const [counter, setCounter] = useState(0);
```

- Use call back to increment counter.

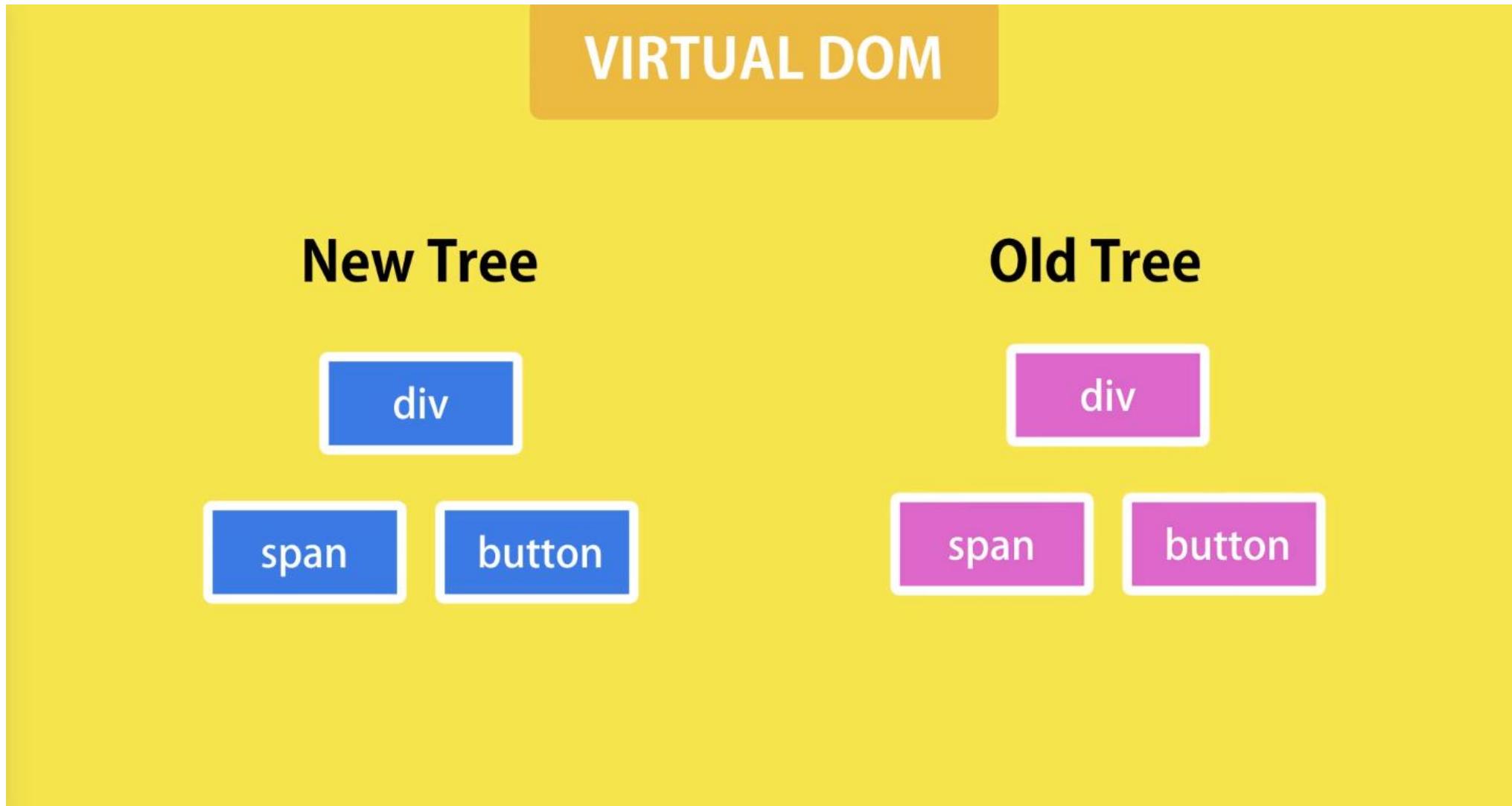
```
28
29  return [
30    <Fragment>
31      <h1 className={getBadgeClasses()}>{formatCount()}</h1>
32      <button onClick={()=> setCounter(counter + 1)} className="btn btn-secondary btn-sm">
33        Increment
34      </button>
35    </Fragment>
36  ];
37 }
```

# Updating the State

- Output

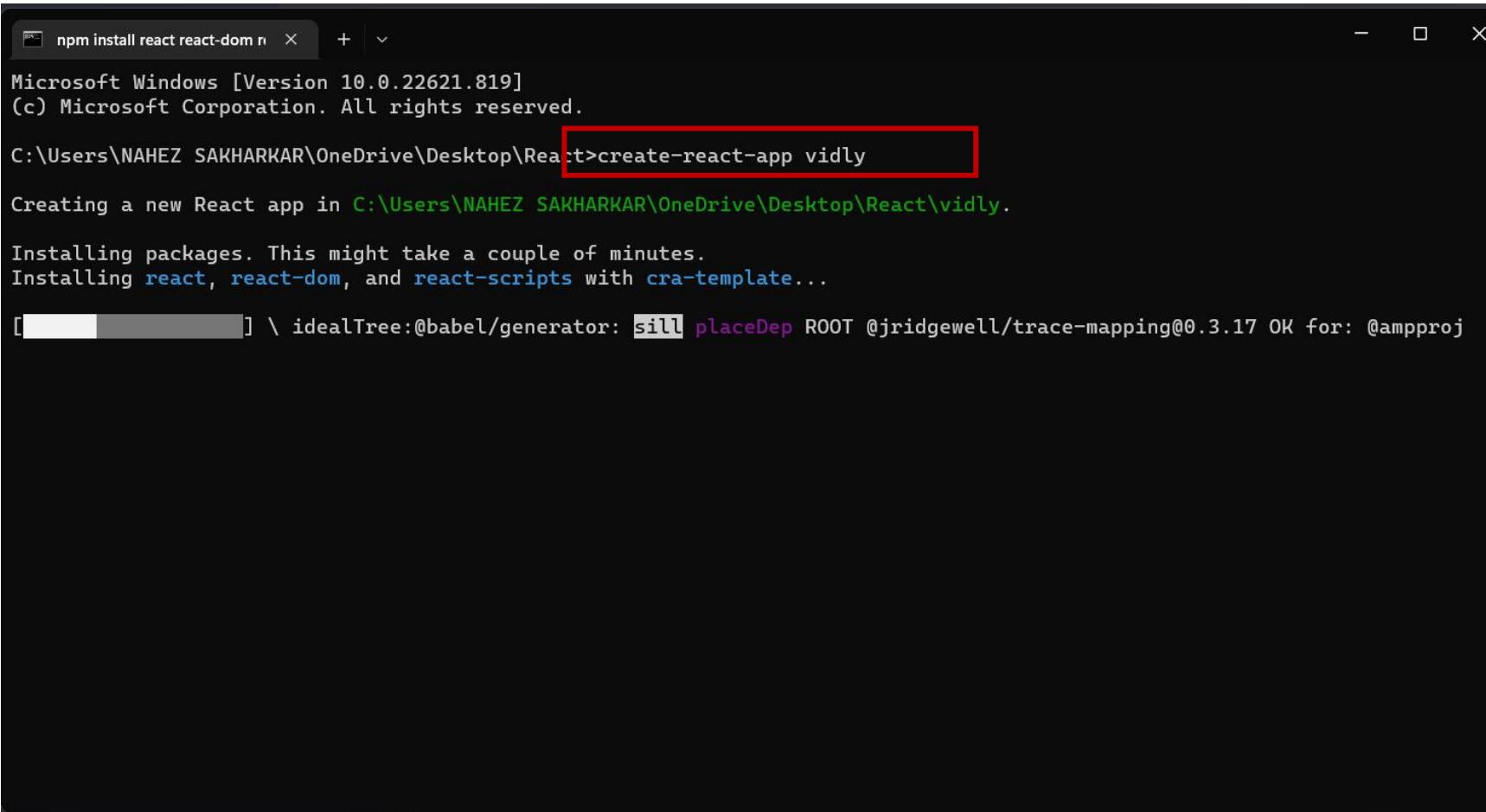


# What Happens when State Changes



# Setting up the Vidly App

Step 1 : Open Command Prompt and Type.



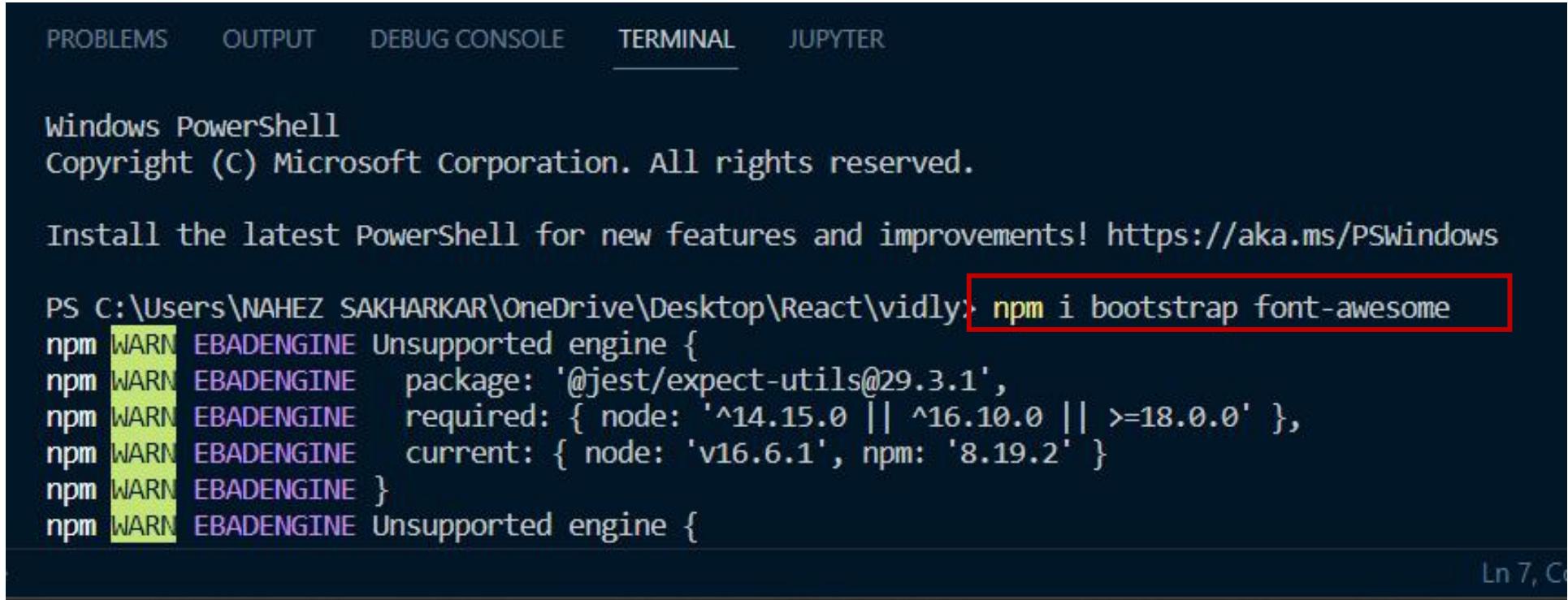
```
npm install react react-dom n  X  +  v  Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React>create-react-app vidly
Creating a new React app in C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React\vidly.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
[██████████] \ idealTree:@babel/generator: sill placeDep ROOT @jridgewell/trace-mapping@0.3.17 OK for: @ampproj
```

# Setting up the Vidly App

Step 2 : Open Project in VS Code and Open Terminal and Install



The screenshot shows a Windows PowerShell terminal window within the Visual Studio Code interface. The terminal tab is active at the top, along with other tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and JUPYTER. The PowerShell prompt indicates it's running on Windows. The user has run the command `npm i bootstrap font-awesome`, which has triggered several warning messages related to an unsupported engine version. A red rectangular box highlights the command line itself.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

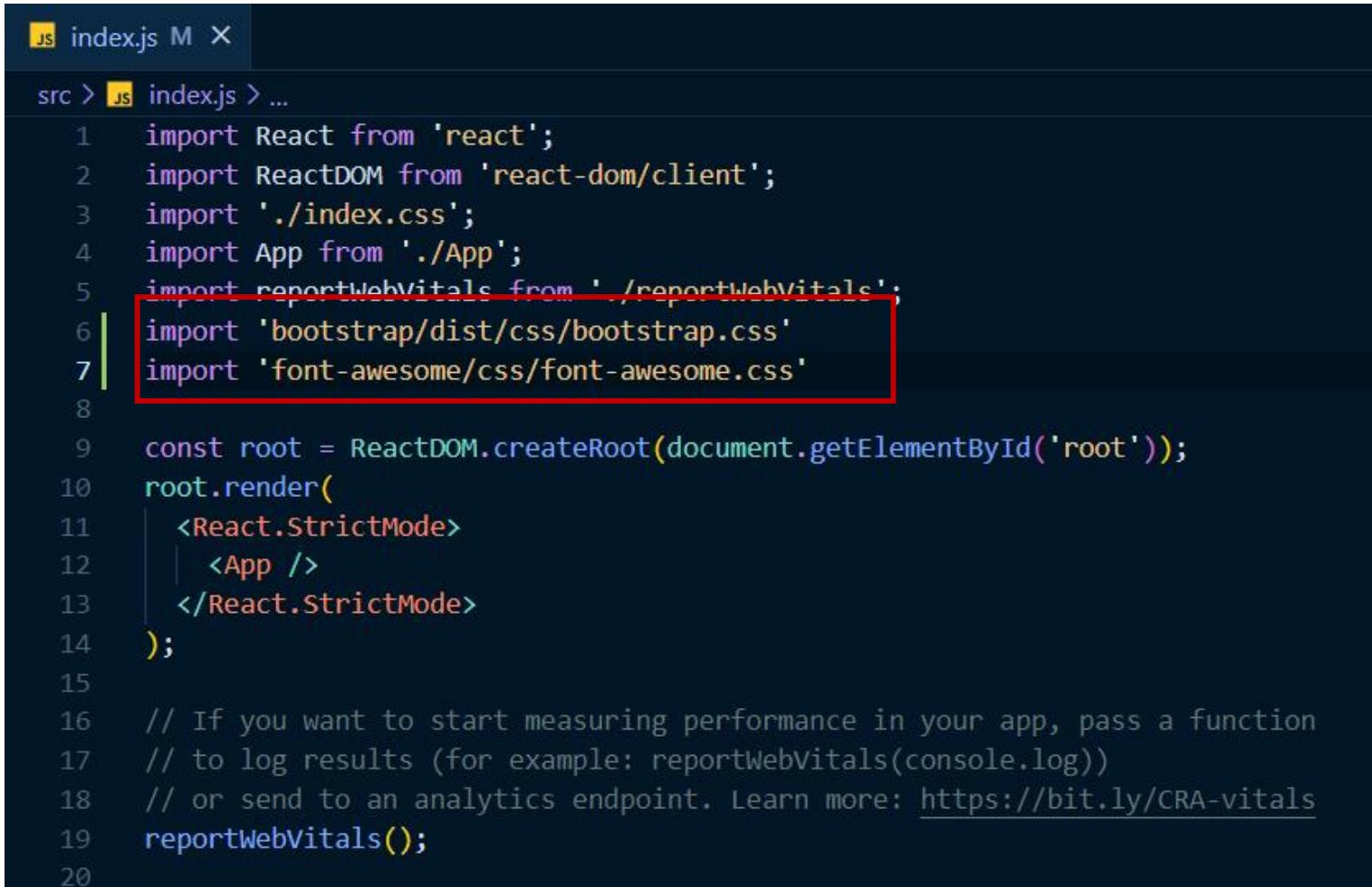
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\NAHEZ SAKHARKAR\OneDrive\Desktop\React\vidly> npm i bootstrap font-awesome
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@jest/expect-utils@29.3.1',
npm WARN EBADENGINE   required: { node: '^14.15.0 || ^16.10.0 || >=18.0.0' },
npm WARN EBADENGINE   current: { node: 'v16.6.1', npm: '8.19.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
```

Ln 7, Col 1

# Setting up the Vidly App

## Step 3 : Open Index.js and Import

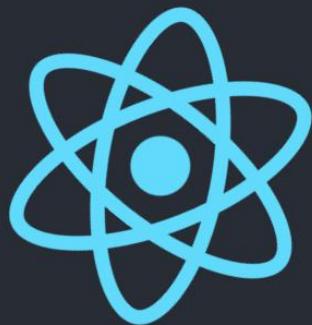


```
JS index.js M X
src > JS index.js > ...
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6 import 'bootstrap/dist/css/bootstrap.css'
7 import 'font-awesome/css/font-awesome.css'

8
9 const root = ReactDOM.createRoot(document.getElementById('root'));
10 root.render(
11   <React.StrictMode>
12   |   <App />
13   </React.StrictMode>
14 );
15
16 // If you want to start measuring performance in your app, pass a function
17 // to log results (for example: reportWebVitals(console.log))
18 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
19 reportWebVitals();
20
```

# Setting up the Vidly App

Step 4 : run ‘npm start’



Edit `src/App.js` and save to reload.

[Learn React](#)

# Setting up the Vidly App

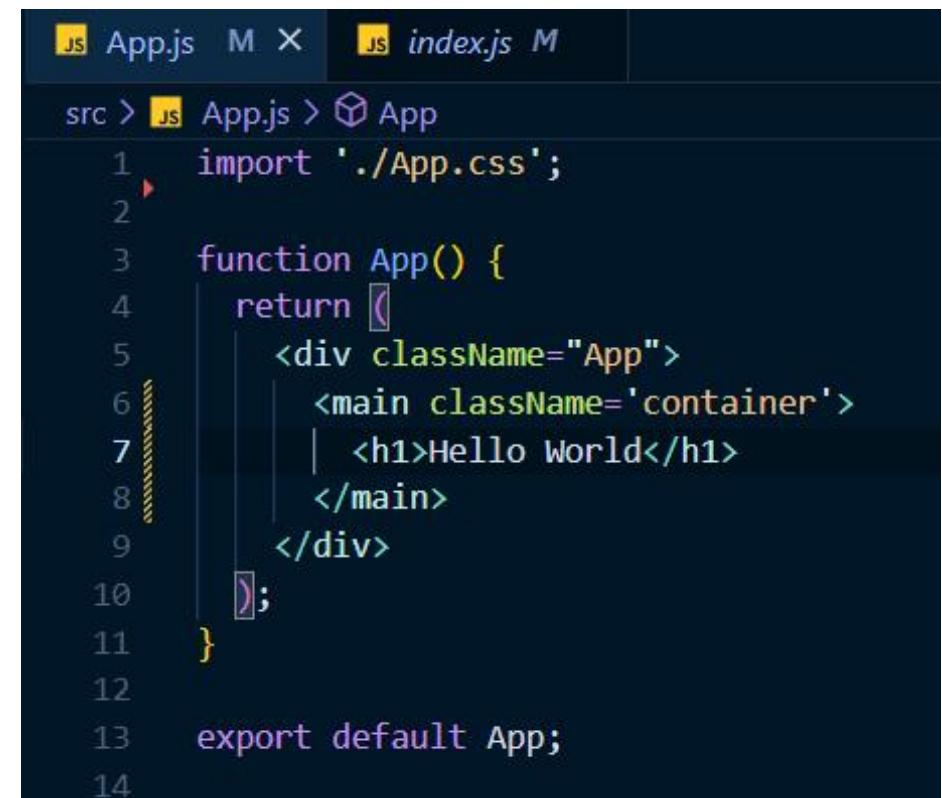
Step 5 : Lets replace default react page with bootstrap template.

Step 6 : Go to <https://getbootstrap.com>

Step 7 : Scroll to Frameworks and inspect starter template.

Step 8 : download [service folder](#) .

Step 9 : Edit App.js code.



The screenshot shows a code editor with a dark theme. The file being edited is 'App.js' located in the 'src' directory. The code is a simple React component named 'App'. It imports 'App.css' and returns a functional component that contains a single 'Hello World' message. The code is numbered from 1 to 14.

```
1 import './App.css';
2
3 function App() {
4   return (
5     <div className="App">
6       <main className='container'>
7         <h1>Hello World</h1>
8       </main>
9     </div>
10    );
11  }
12
13 export default App;
14
```

# Setting up the Vidly App

In **src** create a folder by  
name **services**

For services code :

Go to : <https://github.com/codewithz/cwz-react-requirements>

The screenshot shows a GitHub repository page. At the top, the repository name 'codewithz / cwz-react-requirements' is displayed with a 'Public' badge. To the right are 'Pin' and 'Unwatch' buttons. Below the header is a navigation bar with links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Code' link is underlined. In the main area, there's a summary: 'main' branch, 1 branch, 0 tags. On the right are 'Go to file', 'Add file', and a green 'Code' button with a dropdown arrow. Below this is a commit list. The first commit is by 'codewithz' at 'eff4f03' (1 minute ago), titled 'Add files via upload'. It contains a file named 'services.zip' (highlighted with a red box). Another commit is listed below it, also from 'codewithz' at 'eff4f03' (1 minute ago), titled 'Add files via upload'. At the bottom, there's a note to 'Add a README'.

# Exercise

Create A Project That deletes movies and show number of movies in the db using previously downloaded fake services.

Showing 9 movies in the database.

Title	Genre	Stock	Rate	
Terminator	Action	6	2.5	<button>Delete</button>
Die Hard	Action	5	2.5	<button>Delete</button>
Get Out	Thriller	8	3.5	<button>Delete</button>
Trip to Italy	Comedy	7	3.5	<button>Delete</button>
Airplane	Comedy	7	3.5	<button>Delete</button>

js fakeGenreService.js X

TopicOneHome.jsx

CompanyChangeF

Components > Section 4- Composing Components > finish > vidly > src > services >

```
export const genres = [
  { _id: "5b21ca3eeb7f6fbccd471818", name: "Action" },
  { _id: "5b21ca3eeb7f6fbccd471814", name: "Comedy" },
  { _id: "5b21ca3eeb7f6fbccd471820", name: "Thriller" }
];
```

```
export function getGenres() {
  return genres.filter(g => g);
}
```

fakeGenreService.js

fakeMovieService.js X

TopicOneHome.jsx

CompanyChangeFC

Section 4- Composing Components > Section 4- Composing Components > finish > vidly > src > services > fakeMovieService.js

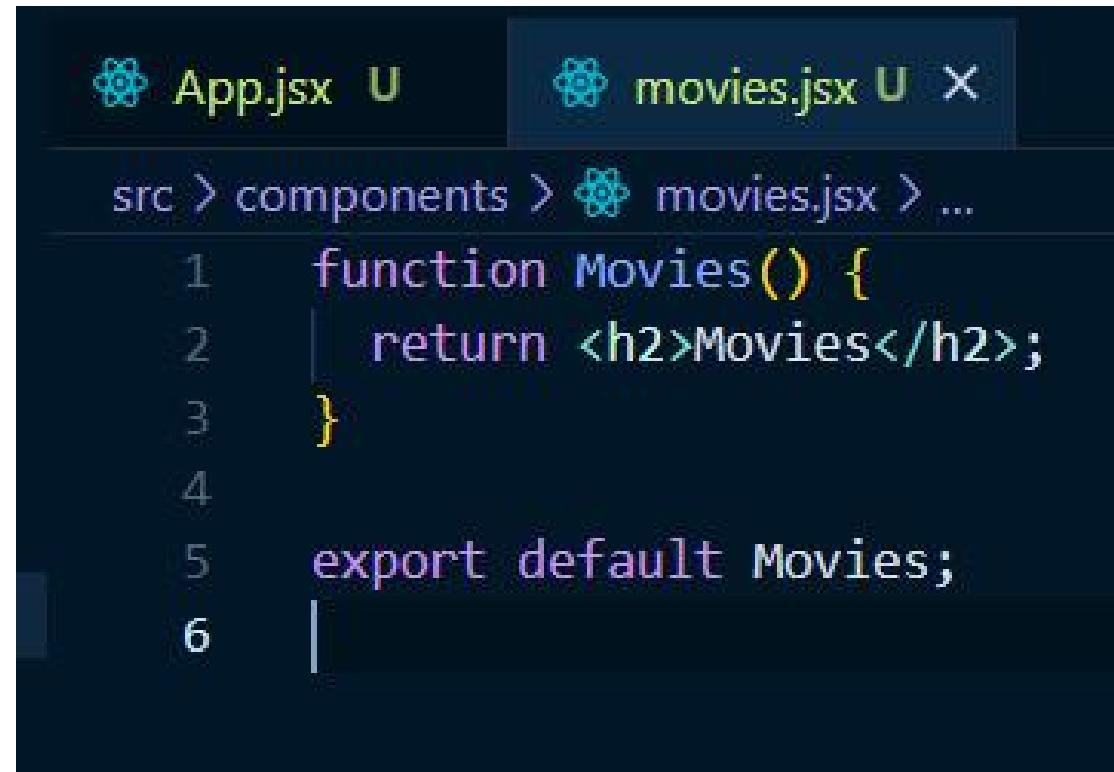
```
import * as genresAPI from "./fakeGenreService";

const movies = [
  {
    _id: "5b21ca3eeb7f6fbccd471815",
    title: "Terminator",
    genre: { _id: "5b21ca3eeb7f6fbccd471818", name: "Action" },
    numberInStock: 6,
    dailyRentalRate: 2.5,
    publishDate: "2018-01-03T19:04:28.809Z",
    Liked: true
  },
  {
    _id: "5b21ca3eeb7f6fbccd471816",
    title: "Die Hard",
    genre: { _id: "5b21ca3eeb7f6fbccd471818", name: "Action" },
    numberInStock: 5,
    dailyRentalRate: 2.5
  }
].
```

# Building the Movies Component

Step 1 : Create A **components** folder in **src** directory.

Step 2 : Create **movies.jsx** component and write code :

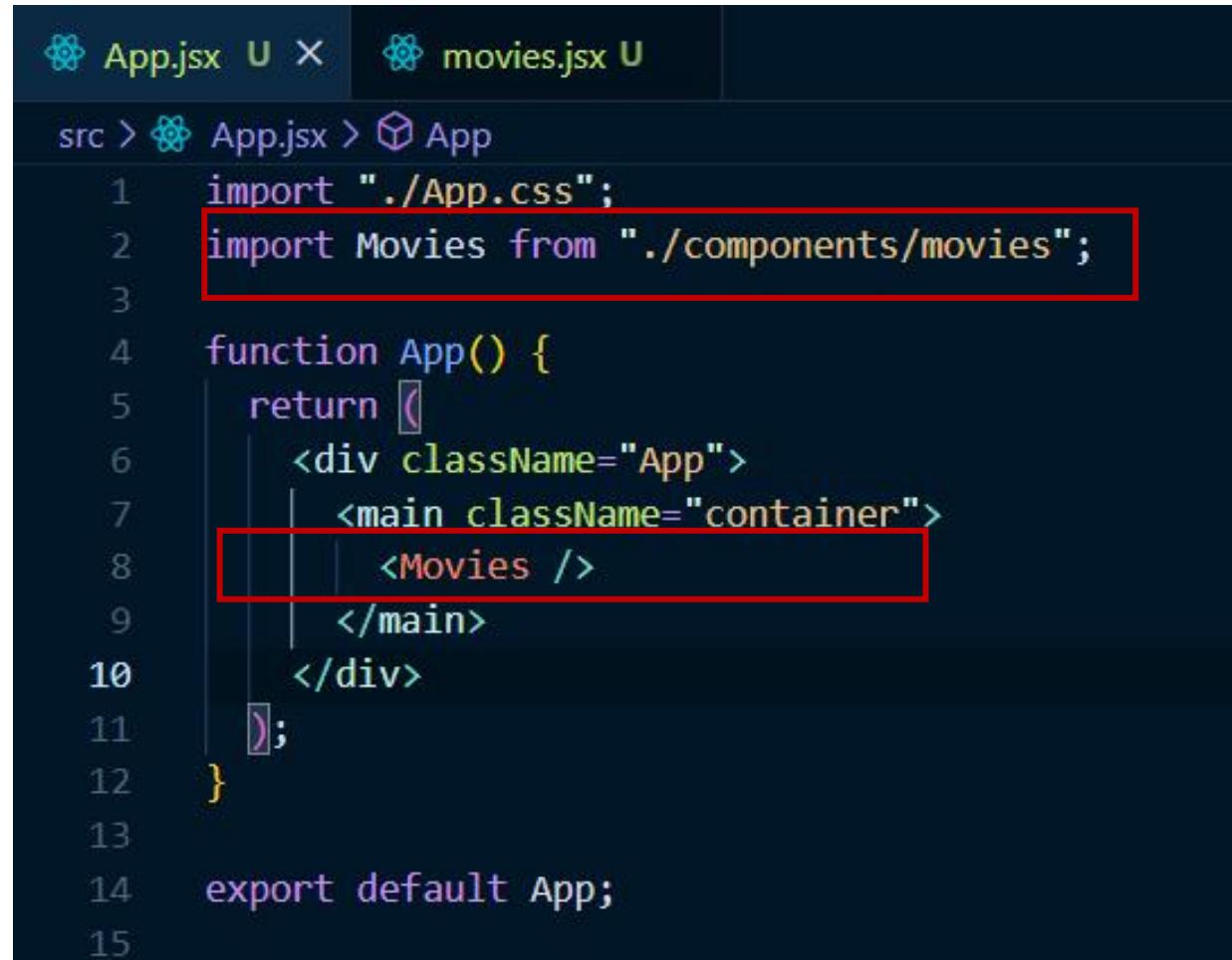


The image shows a screenshot of a code editor with two tabs: "App.jsx" and "movies.jsx". The "movies.jsx" tab is active. Below the tabs, the file path is shown as "src > components > movies.jsx > ...". The code editor displays the following code:

```
1  function Movies() {  
2      return <h2>Movies</h2>;  
3  }  
4  
5  export default Movies;  
6  |
```

# Building the Movies Component

Step 3 : Edit App.jsx file :

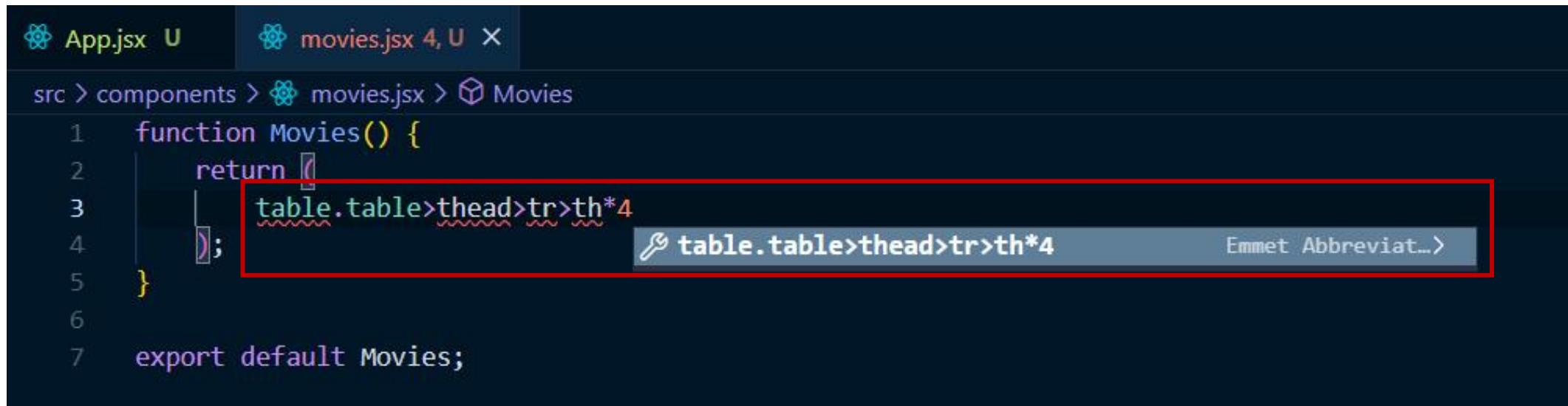


```
App.jsx movies.jsx
src > App.jsx > App
1 import "./App.css";
2 import Movies from "./components/movies";
3
4 function App() {
5   return (
6     <div className="App">
7       <main className="container">
8         <Movies />
9       </main>
10      </div>
11    );
12  }
13
14 export default App;
```

# Building the Movies Component

Step 4 : Research on Bootstrap table from [bootstrap docs.](#)

Step 5 : Open Movies component and edit using zen coding.

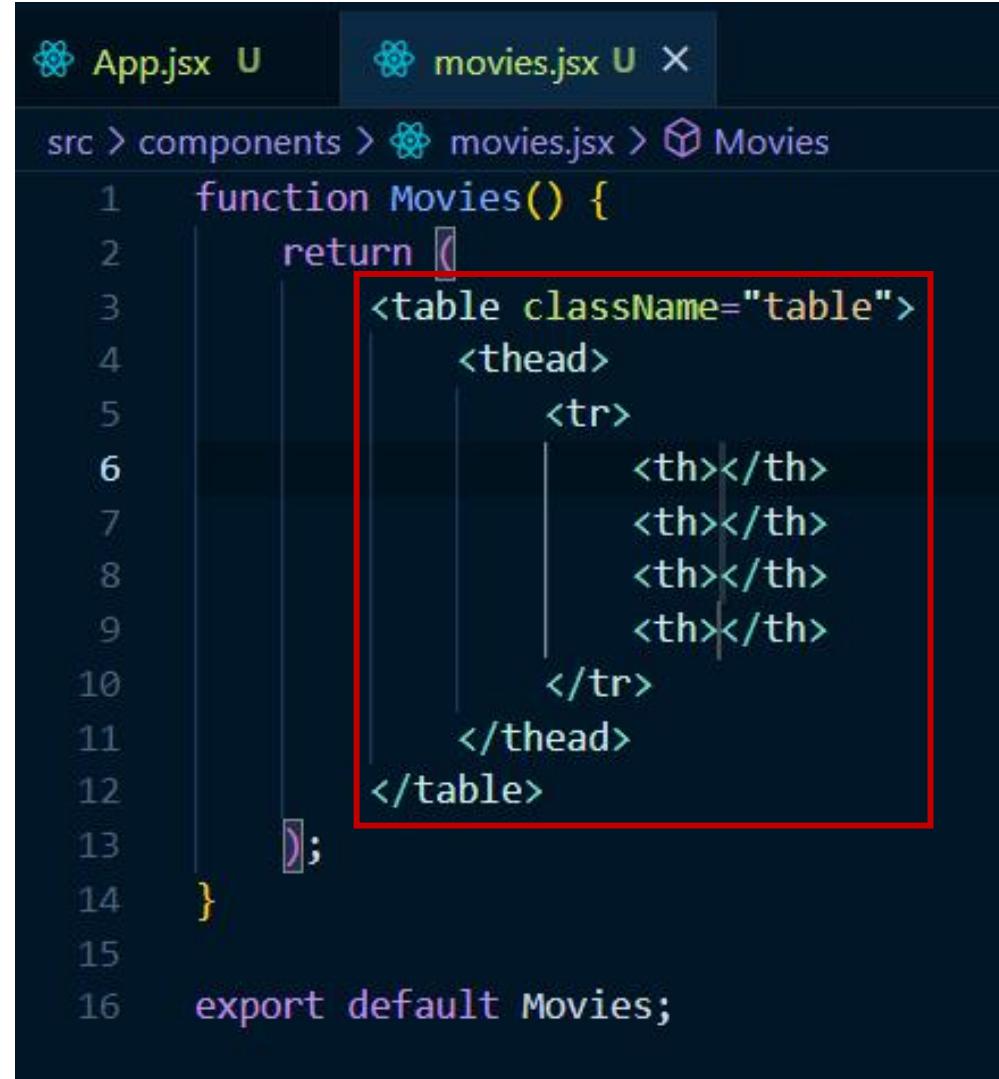


```
App.jsx  U movies.jsx 4, U X
src > components > movies.jsx > Movies
1 function Movies() {
2   return [
3     table.table>thead>tr>th*4
4   ];
5 }
6
7 export default Movies;
```

table.table>thead>tr>th\*4 Emmet Abbreviat...>

# Building the Movies Component

Step 6 : It should convert it into this :



```
App.jsx U movies.jsx U X
src > components > movies.jsx > Movies
1 function Movies() {
2   return [
3     <table className="table">
4       <thead>
5         <tr>
6           <th></th>
7           <th></th>
8           <th></th>
9           <th></th>
10          </tr>
11        </thead>
12      </table>
13    ];
14  }
15
16 export default Movies;
```

# Building the Movies Component

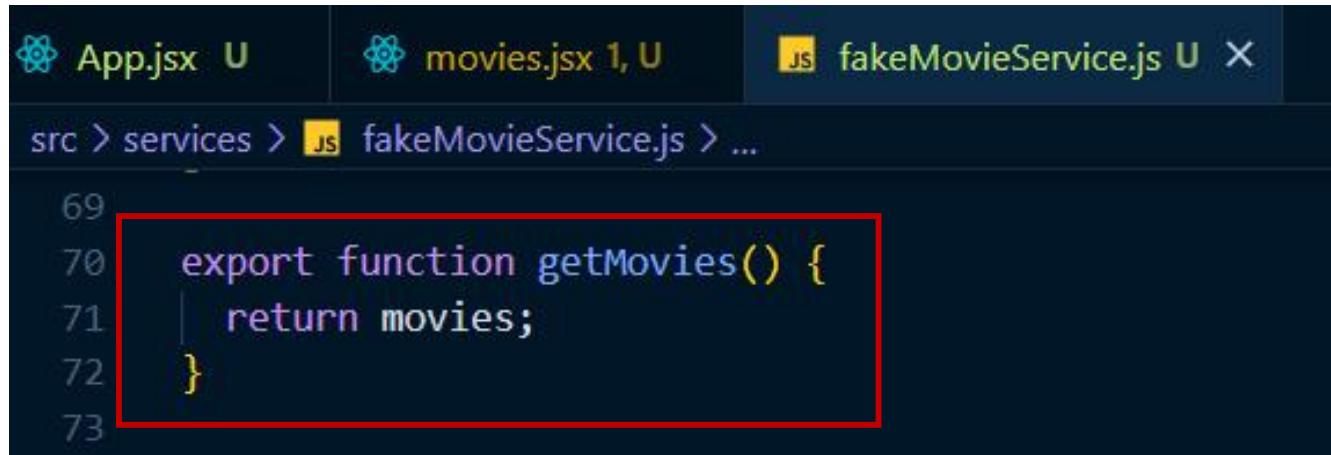
Step 7 : Now Customise Your Table according :

```
<table className="table">
  <thead>
    <tr>
      <th>Title</th>
      <th>Genre</th>
      <th>Stock</th>
      <th>Rate</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
    </tr>
  </tbody>
</table>
```

**It's Genre**

# Building the Movies Component

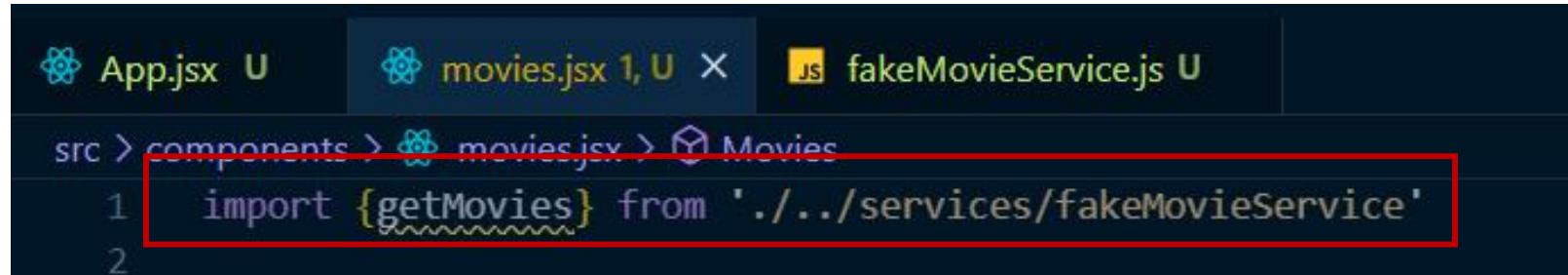
Step 8 : Now import getMovies Function from Movies Services.



The screenshot shows a code editor with three tabs at the top: App.jsx, movies.jsx 1, U, and fakeMovieService.js U X. The fakeMovieService.js tab is active. Below the tabs, the file path is shown as src > services > fakeMovieService.js > ... . The code in the editor is:

```
69
70  export function getMovies() {
71    |   return movies;
72  }
73
```

A red box highlights the entire code block.



The screenshot shows a code editor with three tabs at the top: App.jsx U, movies.jsx 1, U X, and fakeMovieService.js U. The movies.jsx 1, U X tab is active. Below the tabs, the file path is shown as src > components > movies.jsx > Movies. The code in the editor is:

```
1  import {getMovies} from './.../services/fakeMovieService'
2
```

A red box highlights the import statement.

# Building the Movies Component

Step 9 : using useState create a const that stores Movies data

```
function Movies() {  
  const [movies, setMovies] = useState(getMovies);  
  return (  
    <Table>  
      <thead>  
        <tr>  
          <th>Title</th>  
          <th>Genre</th>  
          <th>Stock</th>  
          <th>Daily Rental Rate</th>  
        </tr>  
      </thead>  
      <tbody>  
        {movies.map((movie) => (  
          <tr>  
            <td>{movie.title}</td>  
            <td>{movie.genre.name}</td>  
            <td>{movie.numberInStock}</td>  
            <td>{movie.dailyRentalRate}</td>  
          </tr>  
        ))}  
      </tbody>  
    </Table>  
  );  
}
```

Step 10 : Edit Table Body with Array Mapping

```
<tbody>  
  {movies.map((movie) => (  
    <tr>  
      <th>{movie.title}</th>  
      <th>{movie.genre.name}</th>  
      <th>{movie.numberInStock}</th>  
      <th>{movie.dailyRentalRate}</th>  
    </tr>  
  ))}  
</tbody>
```

# Building the Movies Component

Step 9 : using useState create a const that stores Movies data

```
function Movies() {  
  const [movies, setMovies] = useState(getMovies);  
  return (  
    <Table>  
      <thead>  
        <tr>  
          <th>Title</th>  
          <th>Genre</th>  
          <th>Stock</th>  
          <th>Daily Rental Rate</th>  
        </tr>  
      </thead>  
      <tbody>  
        {movies.map((movie) => (  
          <tr>  
            <td>{movie.title}</td>  
            <td>{movie.genre.name}</td>  
            <td>{movie.numberInStock}</td>  
            <td>{movie.dailyRentalRate}</td>  
          </tr>  
        ))}  
      </tbody>  
    </Table>  
  );  
}
```

Step 10 : Edit Table Body with Array Mapping

```
<tbody>  
  {movies.map((movie) => (  
    <tr>  
      <th>{movie.title}</th>  
      <th>{movie.genre.name}</th>  
      <th>{movie.numberInStock}</th>  
      <th>{movie.dailyRentalRate}</th>  
    </tr>  
  ))}  
</tbody>
```

# Building the Movies Component

Output :

Title	Gnere	Stock	Rate
Terminator	Action	6	2.5
Die Hard	Action	5	2.5
Get Out	Thriller	8	3.5
Trip to Italy	Comedy	7	3.5
Airplane	Comedy	7	3.5
Wedding Crashers	Comedy	7	3.5
Gone Girl	Thriller	7	4.5
The Sixth Sense	Thriller	4	3.5
The Avengers	Action	7	3.5

# Deleting a Movie

- Add Another Head in the table and A button in the Body

```
<thead>
  <tr>
    <th>Title</th>
    <th>Genre</th>
    <th>Stock</th>
    <th>Rate</th>
    <th></th>
  </tr>
</thead>
<tbody>
  {movies.map(movie) => (
    <tr>
      <th>{movie.title}</th>
      <th>{movie.genre.name}</th>
      <th>{movie.numberInStock}</th>
      <th>{movie.dailyRentalRate}</th>
      <th>
        <button className="btn btn-danger btn-sm">Delete</button>
      </th>
    </tr>
  ))}
</tbody>
```

# Deleting a Movie

- Add Key to the element that is repeating.

```
<tbody>
  {movies.map((movie) => [
    <tr key={movie._id}>
      <th>{movie.title}</th>
      <th>{movie.genre.name}</th>
      <th>{movie.numberInStock}</th>
      <th>{movie.dailyRentalRate}</th>
      <th>
```

# Deleting a Movie

- Add Delete Functionality using array filter.

```
function Movies() {
  const [movies, setMovies] = useState(getMovies);

  const handleDelete = (movie) => {
    setMovies(movies.filter((m) => m._id !== movie._id));
  };
}
```

- Adding onclick event to delete button.

```
<th>
  <button
    onClick={() => handleDelete(movie)}
    className="btn btn-danger btn-sm"
  >
    Delete
  </button>
</th>
```

# Conditional Rendering

- Wrap table with React Fragment.



```
return ()  
  <Fragment>  
    <table className="table">  
      <thead>  
        <tr>  
          <th>Title</th>  
          <th>Genre</th>  
          <th>Stock</th>  
          <th>Rate</th>  
          <th></th>  
        </tr>  
      </thead>  
      <tbody>  
        {movies.map((movie) => (  
          <tr key={movie._id}>  
            <th>{movie.title}</th>  
            <th>{movie.genre.name}</th>  
            <th>{movie.numberInStock}</th>  
            <th>{movie.dailyRentalRate}</th>  
            <th>  
              <button  
                onClick={() => handleDelete(movie)}  
                className="btn btn-danger btn-sm">  
                Delete  
              </button>  
            </th>  
          </tr>  
        ))}  
      </tbody>  
    </table>  
  </Fragment>  
};
```

- If there are No Movies :



```
const { length: count } = movies;  
  
if (count === 0) return <p>There are No Movies in the Database</p>;
```

- To show Movies Count :



```
return (  
  <Fragment>  
    <p>Showing {count} Movies in the Database</p>  
    <table className="table">  
      <thead>
```

- Add padding in index.css

```
body {  
  margin: 0;  
  padding: 20px 0 0 0;  
  font-family: -apple-system
```

# Movies App Output

Showing 9 Movies in the Database

Title	Gnere	Stock	Rate	
Terminator	Action	6	2.5	<button>Delete</button>
Die Hard	Action	5	2.5	<button>Delete</button>
Get Out	Thriller	8	3.5	<button>Delete</button>
Trip to Italy	Comedy	7	3.5	<button>Delete</button>
Airplane	Comedy	7	3.5	<button>Delete</button>
Wedding Crashers	Comedy	7	3.5	<button>Delete</button>
Gone Girl	Thriller	7	4.5	<button>Delete</button>
The Sixth Sense	Thriller	4	3.5	<button>Delete</button>
The Avengers	Action	7	3.5	<button>Delete</button>

There are No Movies in the Database

# Composing Components

Pass Data

Raise and Handle Events

Multiple Components in Sync

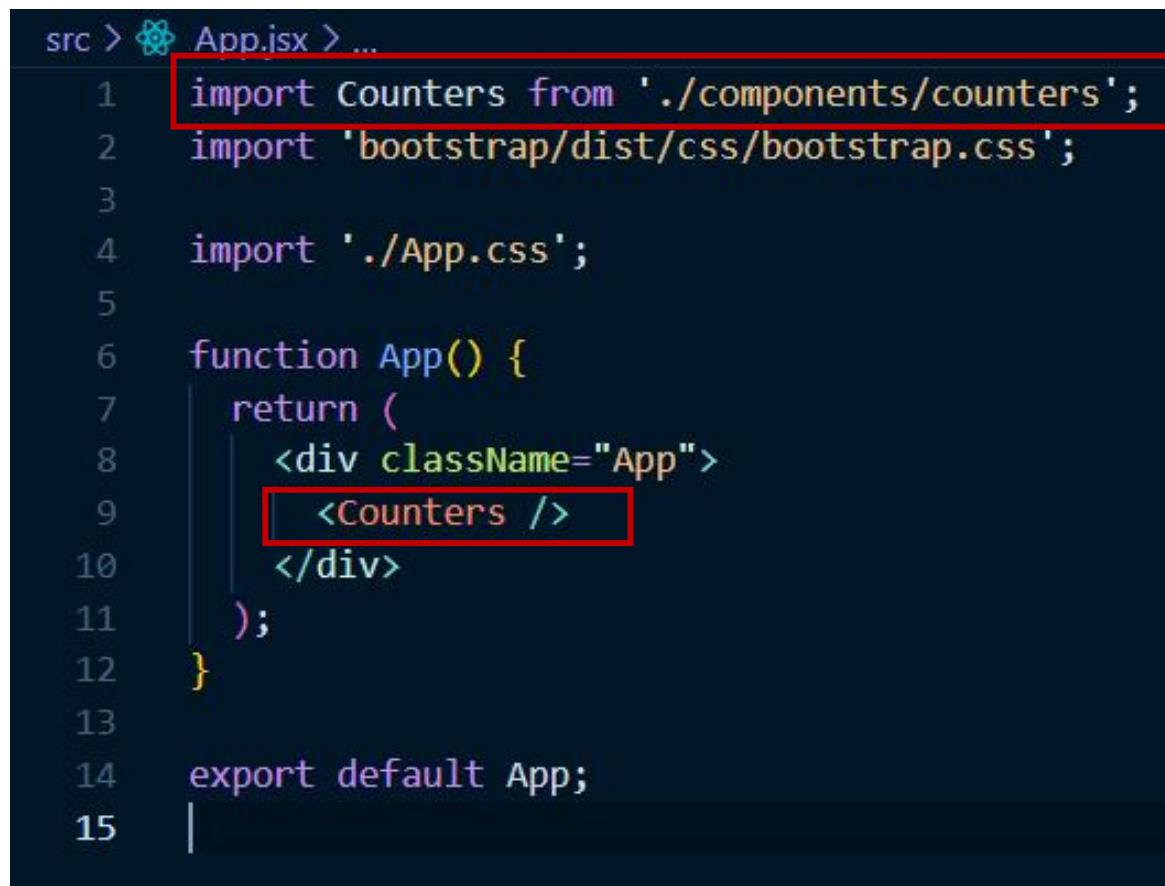
Functional Components

Lifecycle Hooks

Shift back to counters  
app

# Composing Components

- Step 1 : Create `counters.jsx` file in component folder.
- Step 2 : Open `App.js` file and replace ‘counter’ imports with ‘counters’.



```
src > App.js > ...
1 import Counters from './components/counters';
2 import 'bootstrap/dist/css/bootstrap.css';
3
4 import './App.css';
5
6 function App() {
7   return (
8     <div className="App">
9       <Counters />
10      </div>
11    );
12  }
13
14 export default App;
15 |
```

# Composing Components

- Step 3 : Open `counters.jsx` and write.

```
src > components > ⚙ counters.jsx > ...
1 import Counter from "./counter";
2
3 function Counters() {
4   return (
5     <div>
6       <Counter />
7       <Counter />
8       <Counter />
9       <Counter />
10    </div>
11  );
12}
13 export default Counters;
14 |
```

# Composing Components

- Step 4 : Edit Counters.jsx to make and create Array Object using useState.

```
src > components > 🐛 counters.jsx > ...
1  import Counter from "./counter";
2  import { useState } from "react";
3
4  function Counters() {
5      const [counters] = useState([
6          { id: 1, value: 0 },
7          { id: 2, value: 0 },
8          { id: 3, value: 0 },
9          { id: 4, value: 0 },
10     ]);
11
12     return (
13         <div>
14             {counters.map(counter) => (
15                 <Counter key={counter.id}></Counter>
16             ))}
17         </div>
18     );
19 }
20 export default Counters;
21 |
```

# Composing Components

- Output will be same for both codes.
- 



- After Increment
- 



# Passing Data to Components

- Adding Props to Components :

```
src > components >  counters.jsx > ...
12   return (
13     <div>
14       {counters.map((counter) => (
15         <Counter key={counter.id} value={counter.value} selected={true} />
16       )));
17     </div>
18   );
19 }
20 export default Counters;
21
```

A screenshot of a code editor showing a portion of a React component named 'Counters'. The component returns a `<div>` element containing a `.map` operation over the `counters` array. Each item in the array is rendered as a `<Counter>` component. The `<Counter>` component has three props: `key`, `value`, and `selected`. The `value` and `selected` props are highlighted with a red box and an arrow points from the text 'Props' to this box.

```
<div>
  {counters.map((counter) => (
    <Counter key={counter.id} value={counter.value} selected />
  )));
</div>
```

A screenshot of a code editor showing a portion of a React component. It starts with a `<div>` tag, followed by a `.map` operation over the `counters` array. Each item in the array is rendered as a `<Counter>` component. The `<Counter>` component has three props: `key`, `value`, and `selected`. The `selected` prop is highlighted with a red box and an arrow points from the text 'No Value = True' to this box.

# Passing Data to Components

- Open `counter.jsx` and edit :

```
4  function Counter(props) {  
5    const [counter, setCounter] = useState(props.value);  
6    const [tags] = useState(["tag1", "tag2", "tag3"]);  
7  }
```

# After changing Values from counters.jsx

```
const [counters] = useState([  
    { id: 1, value: 10 },  
    { id: 2, value: 7 },  
    { id: 3, value: 2 },  
    { id: 4, value: 80 },  
]);
```

10

Increment

7

Increment

2

Increment

80

Increment

# Passing Children

- Edit **Counters.jsx** :

```
<div>
  {counters.map((counter) => (
    <Counter key={counter.id} value={counter.value} selected>
      <h4>Counter #{counter.id}</h4>
    </Counter>
  ))}
</div>
```

Child

- Edit **Counter.jsx**

```
<Fragment>
  {props.children}
  <h1 className={getBadgeClasses()}>{formatCount()}</h1>
  <button onClick={()=> setCounter(counter + 1)} className="btn btn-secondary btn-sm">
    Increment
  </button>
</Fragment>
```

# Passing Children Output

Counter #1

10 Increment

Counter #2

7 Increment

Counter #3

2 Increment

Counter #4

80 Increment

# Passing H4 in props instead of children

- Edit Counters.jsx :

```
<div>
  {counters.map(counter) => (
    <Counter key={counter.id} value={counter.value} id={counter.id}>
      </Counter>
  )}
</div>
```

- Edit Counter.jsx

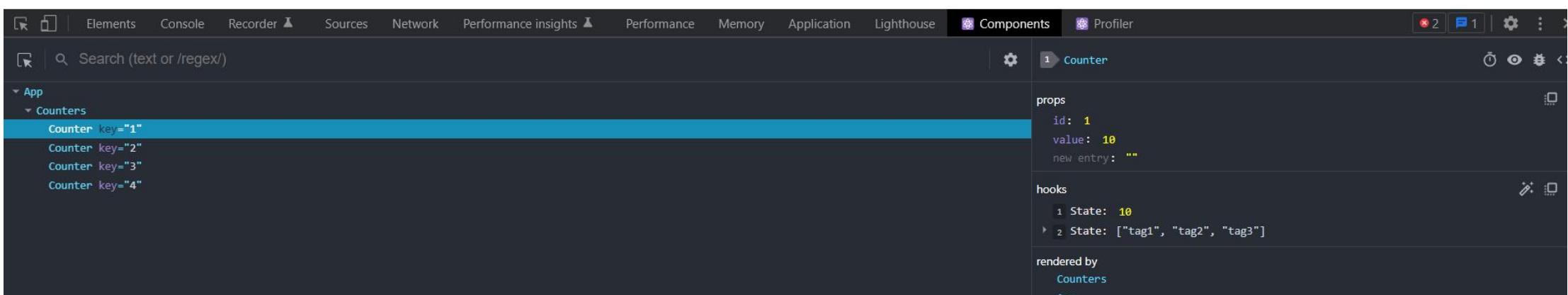
```
<Fragment>
  <h4>Counter #{props.id}</h4>
  <h1 className={getBadgeClasses()}>{formatCount()}</h1>
  <button onClick={()=> setCounter(counter + 1)} className="btn btn-secondary btn-sm">
    Increment
  </button>
</Fragment>
```

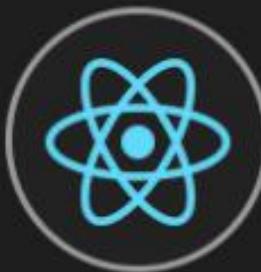
# Debugging the React Apps

- Download Chrome *Extension* ‘React Developer Tools’.



- Inspect react app & open components tab





## Props vs State

✓ props are read-only

✓ props can not be modified

✓ state changes can be asynchronous

✓ state can be modified using `this.setState`

State of the component should  
only be updated by the  
component it belongs to

## Counters (Parent)



As the state belongs to Counters ,  
**Counters** component is responsible for updating the **state**.  
But the state update of **Counters** is dependent upon  
the **event** which is getting generated from **Counter**  
component which is child

**Counter**  
*(Child)*



## Counters (Parent)



Counters component will now write a function to update the state and pass it to the Counter component using props

Counter  
(Child)



# Raising and Handling Event

- Creating handleDelete in `counters.jsx` and passing it as a prop.

```
const handleDelete = () => {
  console.log("Counter has been Deleted!");
};
```

```
return (
  <div>
    {counters.map((counter) => (
      <Counter
        key={counter.id}
        onDelete={handleDelete}
        value={counter.value}
        id={counter.id}
      ></Counter>
    ))}
  </div>
```

## Counters (Parent)



Now the Counter component will invoke the function which it received from the Counters component via props and it will update the state.

Counter  
(Child)

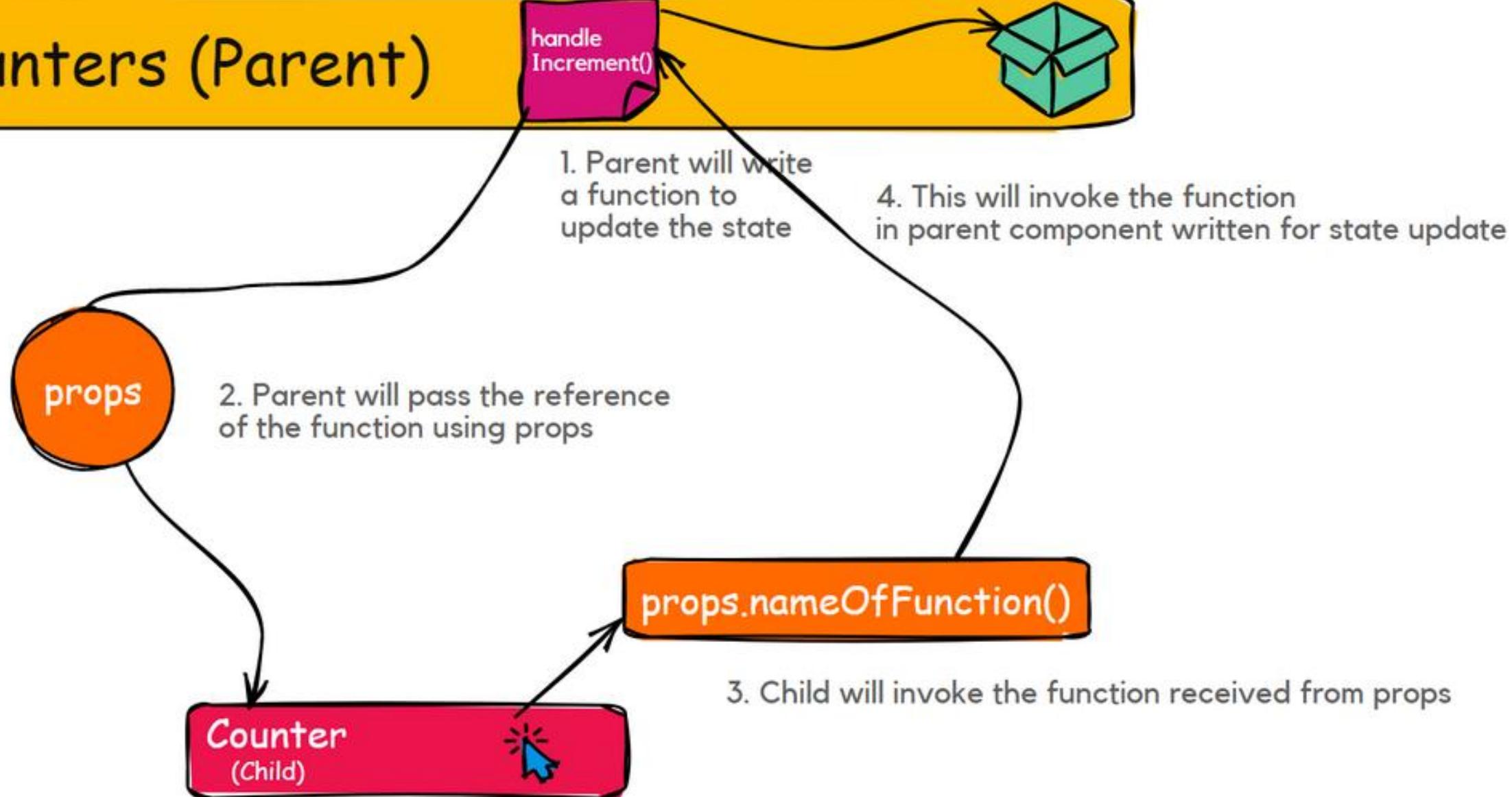


# Raising and Handling Event

- Create a Delete Button with onClick Event in **counter.jsx**

```
<button
  onClick={() => props.onDelete()}
  className="btn btn-danger btn-sm m-2">
  Delete
</button>
```

# Counters (Parent)



# Updating the State

- Update Events

counter.jsx

```
<button  
  onClick={() => props.onDelete(props.id)}  
  className="btn btn-danger btn-sm m-2"  
>  
  Delete  
</button>
```

counters.jsx

```
const handleDelete = (counterId) => {  
  console.log("Counter " + counterId + " has been Deleted!")  
}
```

Counter 1 has been Deleted!

Counter 3 has been Deleted!

# Updating the State

- Update using setState

counters.jsx

```
const handleDelete = (counterId) => {
  setCounters(counters.filter((c) => c.id !== counterId));
};
```

- Update props to simplify

```
<Counter
  key={counter.id}
  onDelete={handleDelete}
  counter={counter}>
</Counter>
```

# Updating the State

- Update props to counter.prev-prop-name.

counter.jsx

```
function Counter(props) {
  const [counter, setCounter] = useState(props.counter.value);
  const [tags] = useState(["tag1", "tag2", "tag3"]);
```

```
  return (
    <Fragment>
      <h4>Counter #${props.counter.id}</h4>
      <h1 className={getBadgeClasses()}>{formatCount()}</h1>
      <button
        onClick={() => setCounter(counter + 1)}
        className="btn btn-secondary btn-sm"
      >
        Increment
      </button>
      <button
        onClick={() => propsonDelete(props.counter.id)}
        className="btn btn-danger btn-sm m-2"
      >
        Delete
      </button>
    </Fragment>
  );
} /* <div>
```

# Single Source of Truth

- Create a Reset Button with onClick Event

counters.jsx

```
const handleReset = () => {
  setCounters(
    counters.map((c) => {
      c.value = 0;
      return c;
    })
  );
};

return (
  <div>
    <button onClick={handleReset} className="btn btn-primary btn-sm m-2">
      Reset
    </button>
    {counters.map((counter) => (
      <Counter
        key={counter.id}
        onDelete={handleDelete}
        counter={counter}
      ></Counter>
    ))}
  </div>
);
}

export default Counters;
```

# Single Source of Truth

- Changes wont affect DOM but you can see it in react debugging tool

The screenshot shows the React developer tools interface. At the top, there's a preview of the application with four counter components labeled Counter #1, Counter #2, Counter #3, and Counter #4. Each counter has a value (10, 7, 2, 80) and two buttons: 'Increment' and 'Delete'. A red box highlights the first counter. Below the preview, the developer tools navigation bar includes Elements, Console, Recorder, Sources, Network, Performance insights, Performance, Memory, Application, Lighthouse, Components (which is selected), and Profiler. The Components tab shows the 'Counters' component with its props: 'new entry: ""'. The state is expanded, showing an array of four objects: {id: 1, value: 0}, {id: 2, value: 0}, {id: 3, value: 0}, and {id: 4, value: 0}. A red box highlights this state section. At the bottom, a footer states 'This is because we don't have a single source of Truth'.

Reset

Counter #1  
10 Increment Delete

Counter #2  
7 Increment Delete

Counter #3  
2 Increment Delete

Counter #4  
80 Increment Delete

Elements Console Recorder Sources Network Performance insights Performance Memory Application Lighthouse Components Profiler

App

Counters

props

new entry: ""

Collapsible prop value

State: [{...}, {...}, {...}, {...}]

0: {id: 1, value: 0}

1: {id: 2, value: 0}

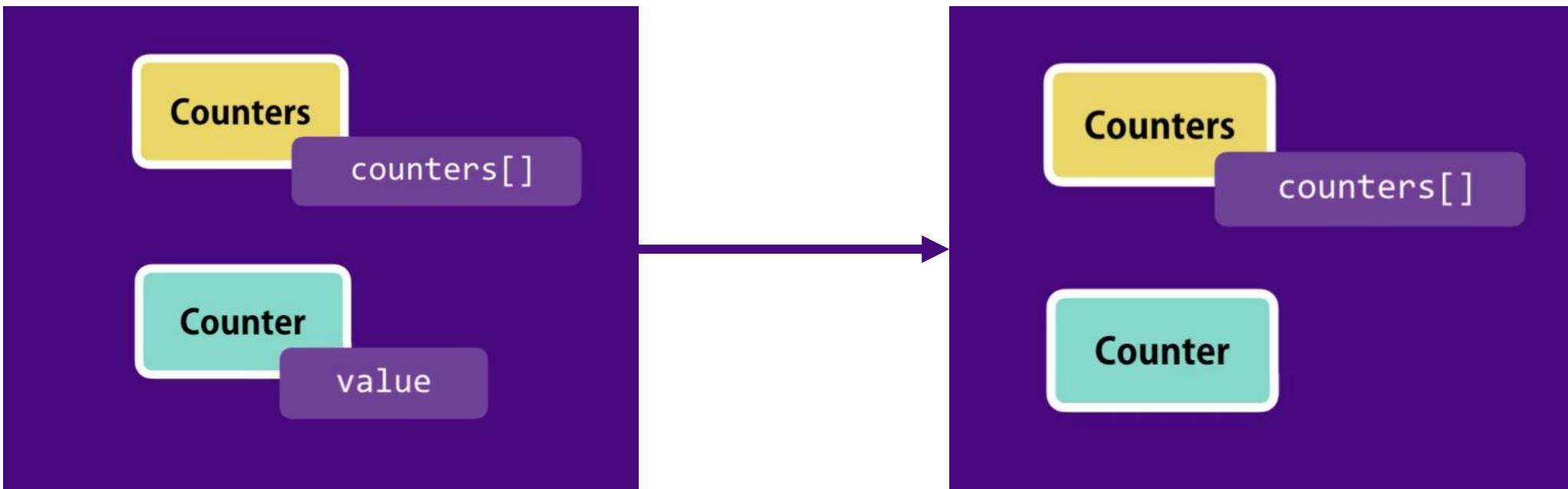
2: {id: 3, value: 0}

3: {id: 4, value: 0}

This is because we don't have a single source of Truth

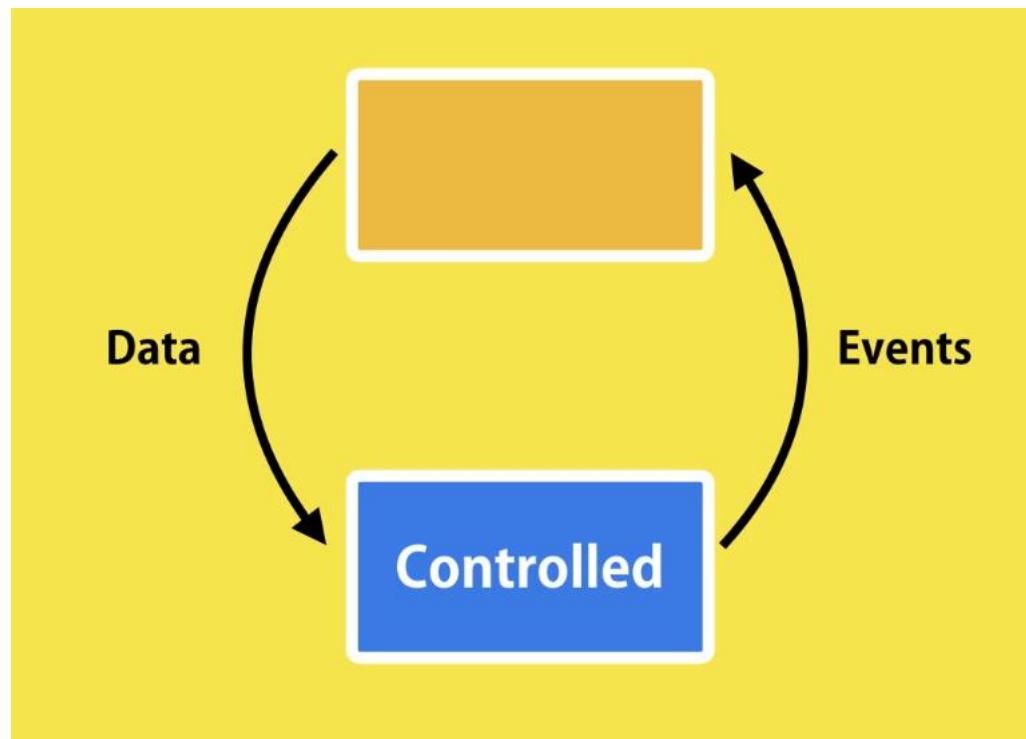
# Single Source of Truth

- To Fix this, we need to remove local state in our counter component



# Removing the Local State

- Remove Local state and only rely on the props.
- **Controlled Component** : Controlled Components are those in which form's data is handled by the component's state. It takes its current value through props and makes changes through callbacks like onClick, onChange, etc. A parent component manages its own state and passes the new values as props to the controlled component.



# Removing the Local State

- In counter.jsx update

counter.jsx

```
src > components > counter.jsx > Counter > getBadgeClasses
1 import { Fragment } from "react";
2 import { useState } from "react";
3
4 function Counter(props) {
5   const [counter, setCounter] = useState(props.counter.value);
6
7   function formatCount() {
8     return props.counter.value === 0 ? "Zero" : props.counter.value;
9   }
10
11  function getBadgeClasses() {
12    let classes = "badge m-2 ";
13    classes += props.counter.value === 0 ? "bg-warning" : "bg-primary";
14    return classes;
15  }
16
```

```
<button
  onClick={() => props.onIncrement(props.counter)}
  className="btn btn-secondary btn-sm"
>
  Increment
</button>
```

# Removing the Local State

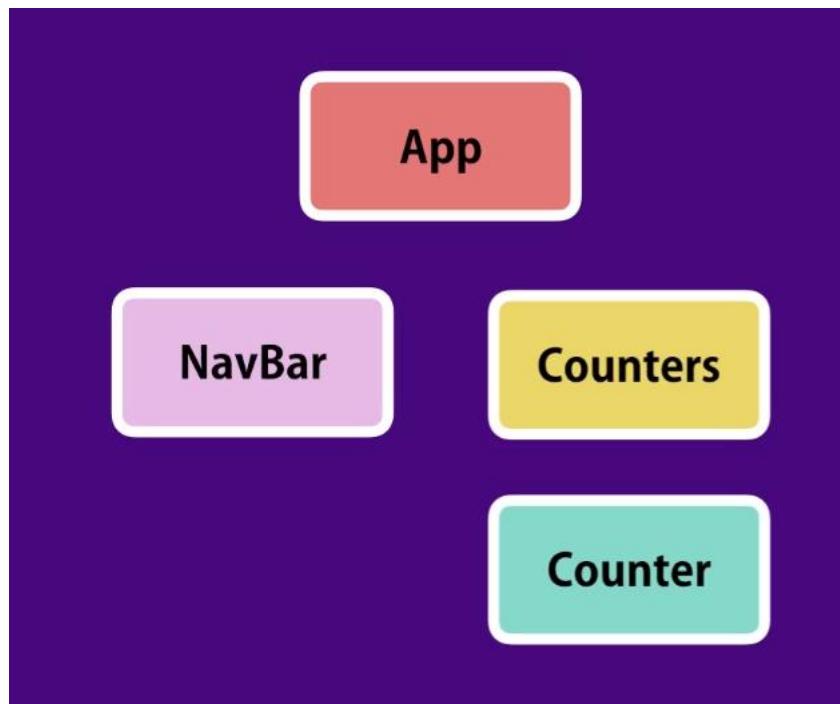
- In `counters.jsx` add

```
const handleIncrement = (counter) => [
  counters.filter((c) => c.id === counter.id)[0].value++;
  setCounters(
    counters.map((c) => {
      return c;
    })
  );
];
```

```
<Counter
  key={counter.id}
  onDelete={handleDelete}
  onIncrement={handleIncrement}
  counter={counter}
></Counter>
```

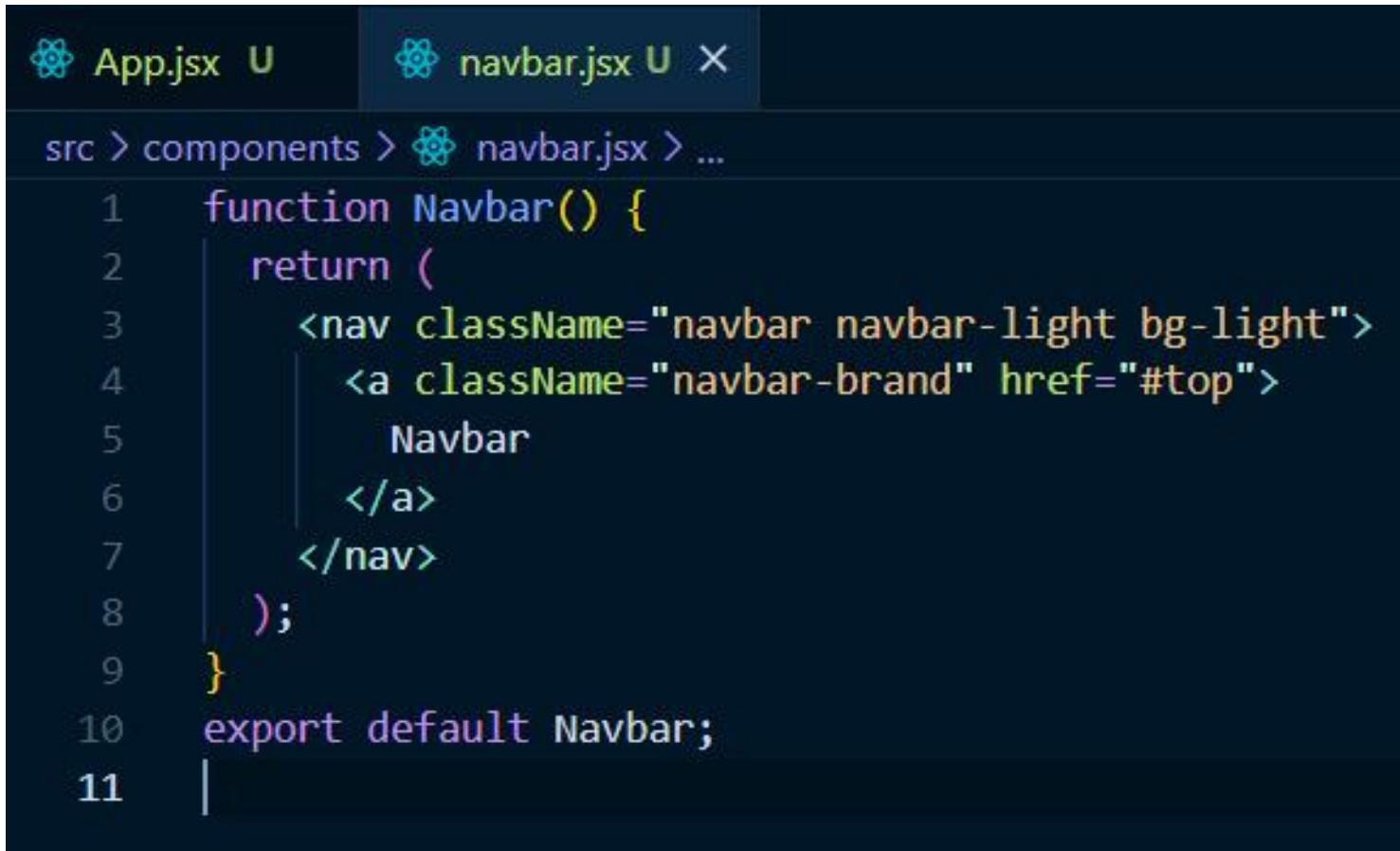
# Multiple Components in Sync

- Ideal Component Tree :



# Multiple Components in Sync

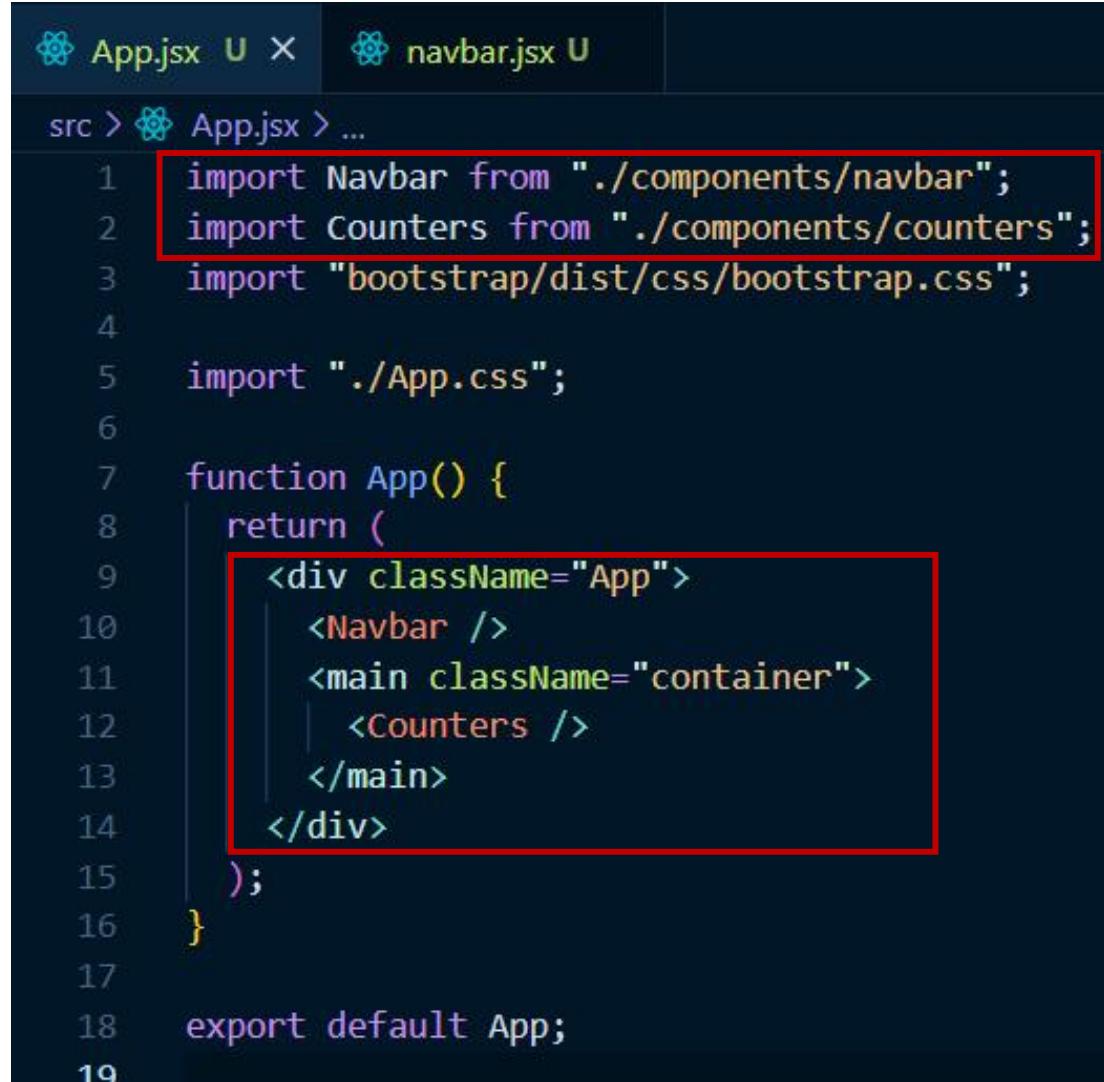
- Create `navbar.jsx` component and write



```
App.jsx U navbar.jsx U X
src > components > navbar.jsx > ...
1  function Navbar() {
2    return (
3      <nav className="navbar navbar-light bg-light">
4        <a className="navbar-brand" href="#top">
5          Navbar
6        </a>
7      </nav>
8    );
9  }
10 export default Navbar;
11 |
```

# Multiple Components in Sync

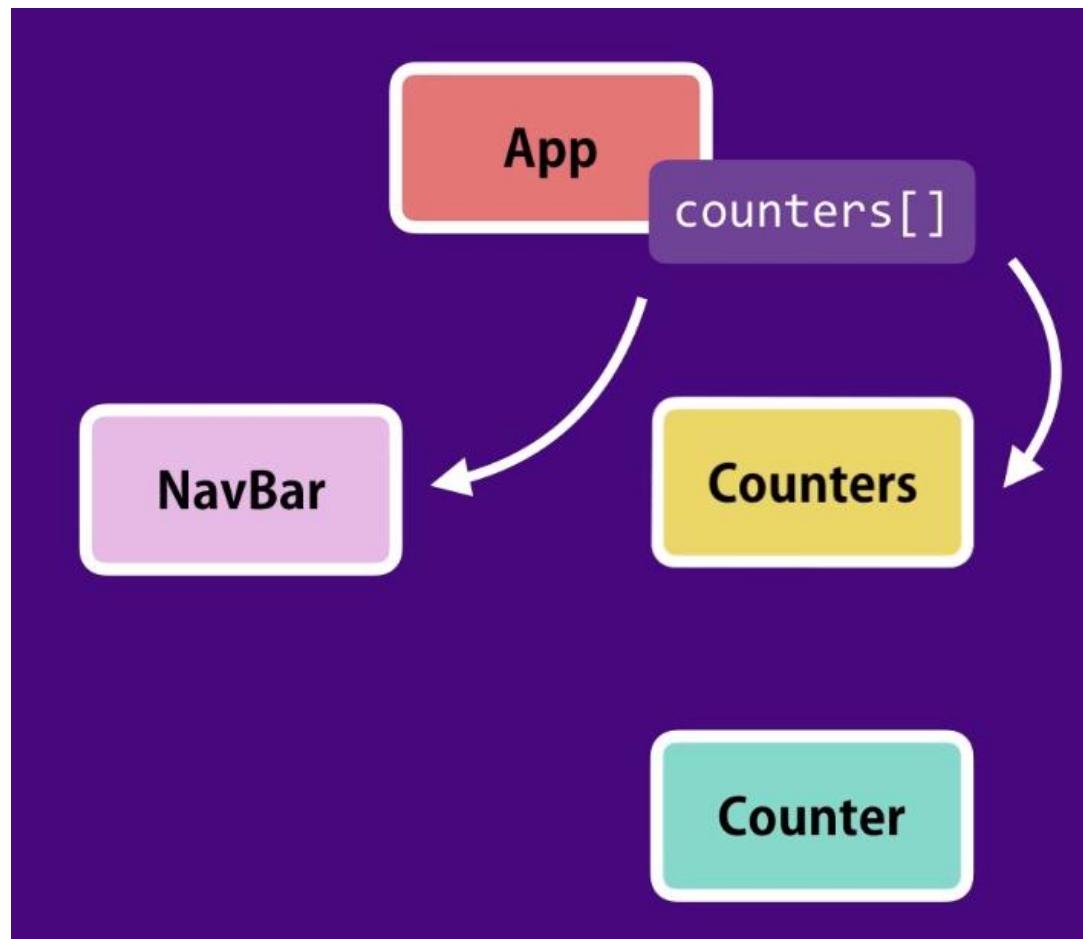
- update App.js



```
App.jsx  U X  navbar.jsx  U
src > App.jsx > ...
1 import Navbar from "./components/navbar";
2 import Counters from "./components/counters";
3 import "bootstrap/dist/css/bootstrap.css";
4
5 import "./App.css";
6
7 function App() {
8   return (
9     <div className="App">
10      <Navbar />
11      <main className="container">
12        <Counters />
13      </main>
14    </div>
15  );
16}
17
18 export default App;
19
```

# Lifting the State Up

- Ideal



# Lifting the State Up

- Move States & All Methods to **App.jsx** and Pass props in Counters Component.

```
return (
  <div className="App">
    <Navbar />
    <main className="container">
      <Counters
        counters={counters}
        onReset={handleReset}
        onIncrement={handleIncrement}
        onDelete={handleDelete}>
      </Counters>
    </main>
  </div>
);
```

```
import Navbar from "./components/navbar";
import Counters from "./components/counters";
import "bootstrap/dist/css/bootstrap.css";
import { useState } from "react";

import "./App.css";
function App() {
  const [counters, setCounters] = useState([
    { id: 1, value: 10 },
    { id: 2, value: 0 },
    { id: 3, value: 2 },
    { id: 4, value: 80 },
  ]);

  const handleDelete = (counterId) => {
    setCounters(counters.filter((c) => c.id !== counterId));
  };

  const handleIncrement = (counter) => {
    counters.filter((c) => c.id === counter.id)[0].value++;
    setCounters(
      counters.map((c) => {
        return c;
      })
    );
  };

  const handleReset = () => {
    setCounters(
      counters.map((c) => {
        c.value = 0;
        return c;
      })
    );
  };
}

export default App;
```

# Lifting the State Up

- Update Counters Component

counters.jsx

```
App.jsx U navbar.jsx U counters.jsx X
src > components > counters.jsx > ...
1 import Counter from "./counter";
2
3 function Counter(props) {
4   return (
5     <div>
6       <button
7         onClick={props.onReset}
8         className="btn btn-primary btn-sm m-2"
9       >
10      Reset
11    </button>
12    {props.counters.map((counter) => (
13      <Counter
14        key={counter.id}
15        onDelete={props.onDelete}
16        onIncrement={props.onIncrement}
17        counter={counter}
18      ></Counter>
19    )));
20  </div>
21);
22
23 export default Counter;
```

# Lifting the State Up

- Pass Prop to navbar for count of counters.

App.jsx

```
<div className="App">
  <Navbar totalCounters={counters.filter((c) => c.value > 0).length} />
  <main className="container">
    <Counters
      counters={counters}
      onReset={handleReset}
      onIncrement={handleIncrement}
      onDelete={handleDelete}>
    </Counters>
  </main>
</div>
```

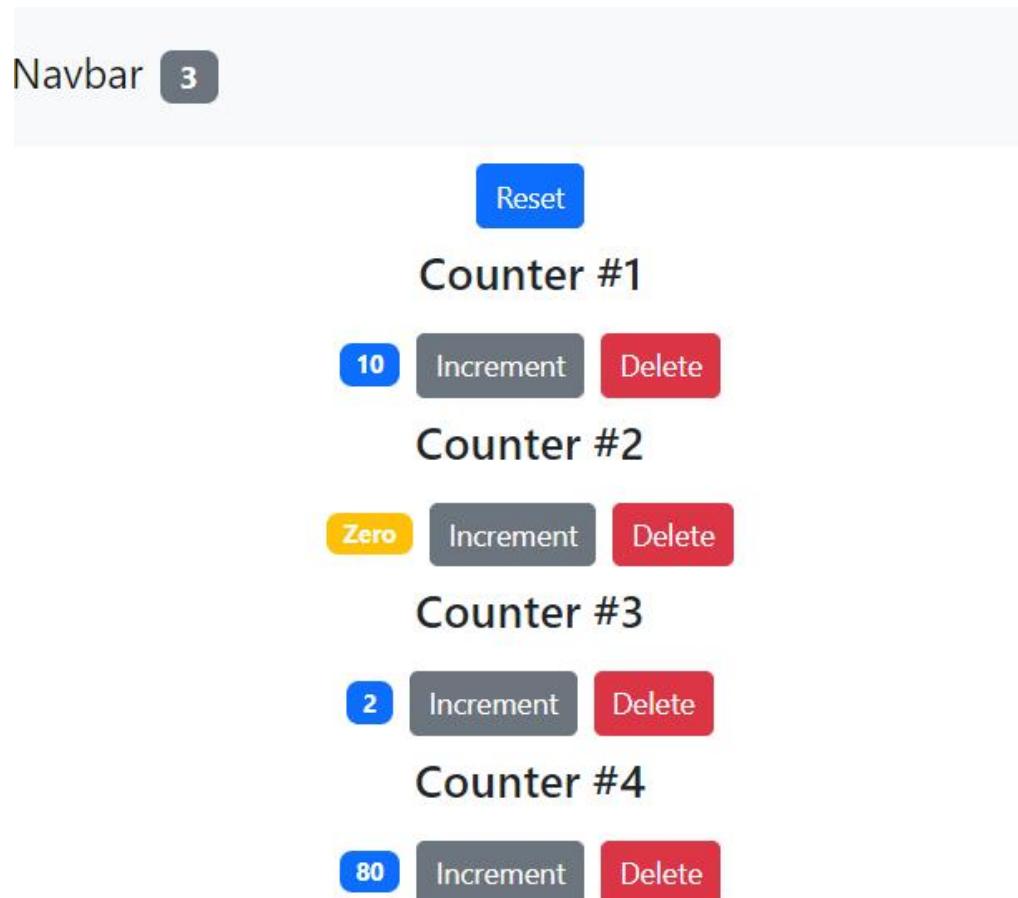
- Update Navbar

navbar.jsx

```
<nav className="navbar navbar-light bg-light">
  <a className="navbar-brand" href="#top">
    Navbar
    <span className="badge badge-pill bg-secondary m-2">
      {props.totalCounters}
    </span>
  </a>
</nav>
```

# Lifting the State Up

- Output :





# Destructuring Arguments

- In navbar.jsx instead of taking props as argument we can :

```
src > components > navbar.jsx > ...
1  function Navbar({ totalCounters }) {
2    return (
3      <nav className="navbar navbar-light bg-light">
4        <a className="navbar-brand" href="#top">
5          Navbar
6          <span className="badge badge-pill bg-secondary m-2">
7            {totalCounters}
8          </span>
9        </a>
10       </nav>
11     );
12   }
13   export default Navbar;
```

# Lifecycle Hooks

- In React, each component has stages during its lifecycle, generally referred to as React component's lifecycle.
- There are different phases in the lifecycle, viz. Mounting, Updating, and Unmounting.
- Each phase has its methods – `componentDidMount`, `componentDidUpdate`, `componentWillUnmount`, etc. that help you perform life cycle operations on the components.

## MOUNT

`constructor`

`render`

`componentDidMount`

## UPDATE

`render`

`componentDidUpdate`

## UNMOUNT

`componentWillUnmount`

# Mounting Phase

- 1) Constructor() :-
- Since we added a constructor we need to call the constructor of parent class using *super()*.
- Constructor is called once when an object is created this is why we used constructor to set properties of an object.
- We will set the state of component based on props that we receive from outside. i.e we will map state to props here.

```
import React,{Component} from 'react';

export default class App extends Component {

  state = {}

  //we dont have access to this.props inside constructor
  //to get access of this.props inside constructor we need to pass it as parameter
  //to constructor
  //also call super method with props as parameter else it will give us undefined
  constructor(props){
    super(props);
    console.log('App - Constructor');
    this.state = this.props.something;
  }

  render() {
    return (
      <div>App</div>
    )
  }
}
```

# Mounting Phase

- 2) `ComponentDidMount()` :-
- This lifecycle method is called after our component is rendered into the DOM.
- This method is perfect place to make ajax calls and get data from server.

```
import React,{Component} from 'react';

export default class App extends Component {

  state = {}

  //we dont have access to this.props inside constructor
  //to get access of this.props inside constructor we need to pass it as parameter to
  constructor
    //also call super method with props as parameter else it will give us undefined
  constructor(props){
    super(props);
    console.log('App - Constructor');
    this.state = this.props.something;
  }

  componentDidMount() {
    //Make Ajax call
    //Call setState with new data
    this.setState({}); 
    console.log('App - Mounted');
  }

  render() {
    return (
      <div>App</div>
    )
  }
}
```

# Mounting Phase

- 3) Render() :-

```
import React,{Component} from 'react';

export default class App extends Component {
    state = {}

    //we dont have access to this.props inside constructor
    //to get access of this.props inside constructor we need to pass it as parameter to
constructor
    //also call super method with props as parameter else it will give us undefined
    constructor(props){
        super(props);
        console.log('App - Constructor');
        this.state = this.props.something;
    }

    componentDidMount() {
        //Make Ajax call
        //Call setState with new data
        this.setState({});
        console.log('App - Mounted');
    }

    render() {
        console.log('App - Rendered');
        return (
            <div>App</div>
        )
    }
}
```

# Mounting Phase

## Execution Order :-

Render returns the react element that represents our virtual DOM  
Then React gets that virtual DOM and renders it in our actual DOM  
Then Component is Mounted

App - Constructor	App.js:14
App - Constructor	VM51:236
App - Rendered	App.js:27
App - Rendered	VM51:236
App - Mounted	App.js:22
App - Mounted	App.js:22

# Updating Phase

- Whenever state is changed our entire component tree is rerendered.
- React does rerender actual dom instead it updates the virtual dom.
- React has a copy of old virtual dom that is why we should not update the state directly so that we have two different objects reference in the memory
- The react will compare old virtual dom and new virtual dom and then it will figure out what is changed and then update our actual dom accordingly.

# Updating Phase

- `componentDidMount()` :-
- `componentDidMount` allows us to decide if we want to make an api call and get new data based on changes of props and state object.

```
handleChange = () => {
  this.setState({key:"value"});
}

//we dont have access to this. props ins
```

```
render() {
  console.log('App - Rendered');
  return (
    <Page onUpdate={this.handleChange} />
  )
}
```

```
import React, { Component } from 'react';

class Page extends Component{

  componentDidUpdate(prevProps,prevState){
    console.log("prevProps",prevProps)
    console.log("prevState",prevState)
    //Here We can compare previousProps with currentProps and make some api calls
  }

  render(){
    console.log('Page - Rendered')
    return(
      <>
        <Navbar />
        <button onClick={()=>this.props.onUpdate()} >change</button>
      </>
    )
  }
}
```

```
const Navbar = () => {
  console.log('Navbar - Rendered');
  return (
    <h1>Navbar</h1>
  )
}
```

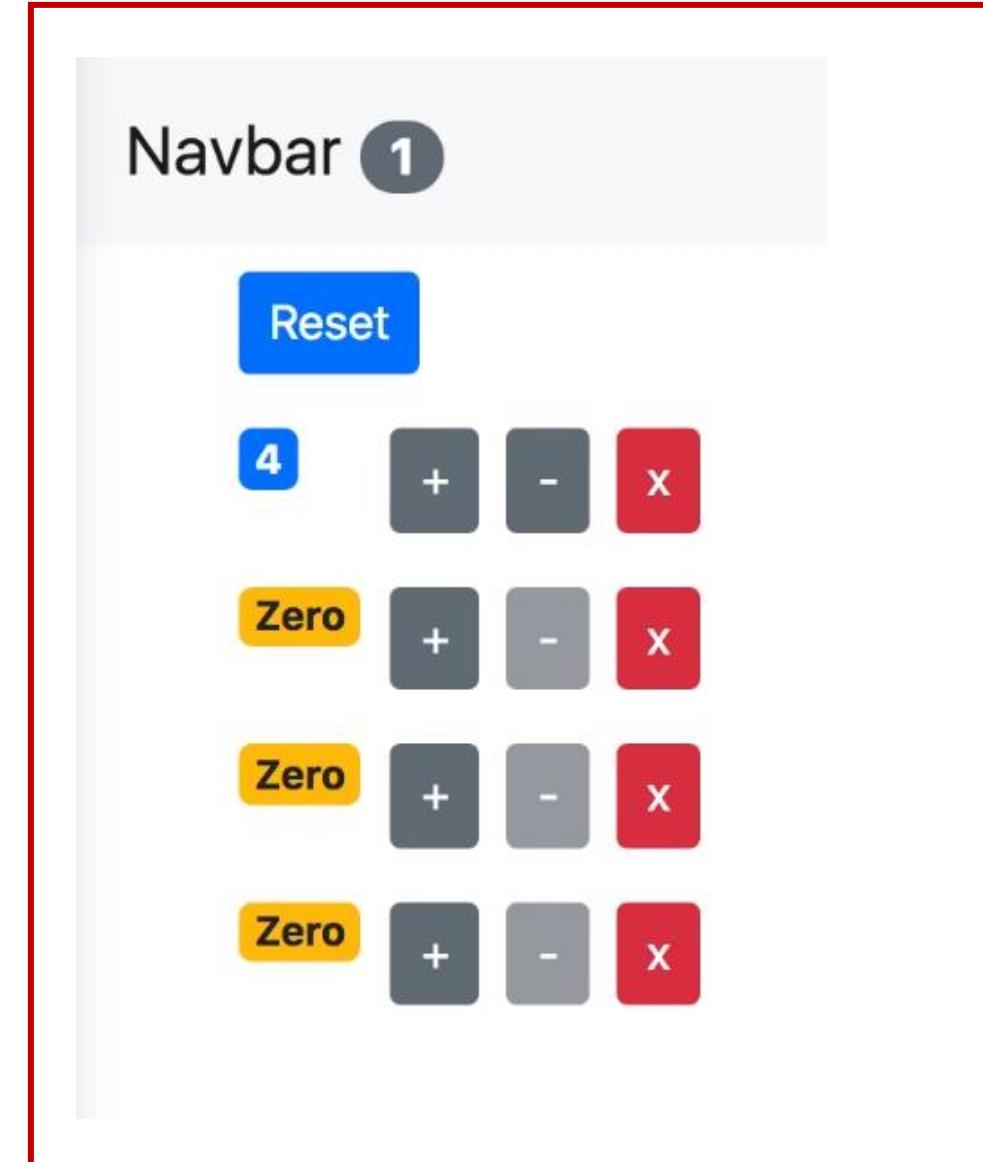
# UnMounting Phase

- In this phase we have one lifecycle hook
- `componentWillUnmount()` :-
- This Method is called just before a component is removed from the DOM.
- We use this to do cleanup process to avoid memory leaks

```
componentWillUnmount() {  
  console.log('Counter - UnMounted');  
}
```

# Exercise- Decrement Button

- Output to Achieve



# Exercise- Decrement Button

- In `counter.jsx` Add and Update



```
<Fragment>
  <h4>Counter #{props.counter.id}</h4>
  <h1 className={getBadgeClasses()}>{formatCount()}</h1>
  <button
    onClick={() => props.onIncrement(props.counter)}
    className="btn btn-secondary btn-sm">
    >
    +</button>
  <button
    onClick={() => props.onDecrement(props.counter)}
    className="btn btn-secondary btn-sm">
    >
    -</button>
  <button
    onClick={() => propsonDelete(props.counter.id)}
    className="btn btn-danger btn-sm m-2">
    >
    X</button>
</Fragment>
```

# Exercise- Decrement Button

- In `counters.jsx` Update

```
<Counter  
  key={counter.id}  
  onDelete={props.onDelete}  
  onIncrement={props.onIncrement}  
  onDecrement={props.onDecrement}  
  counter={counter}  
></Counter>
```

- In `App.jsx` Update

```
const handleDecrement = (counter) => {  
  counters.filter((c) => c.id === counter.id)[0].value--;  
  setCounters(  
    counters.map((c) => {  
      return c;  
    })  
  );  
};
```

```
<Counters  
  counters={counters}  
  onReset={handleReset}  
  onIncrement={handleIncrement}  
  onDecrement={handleDecrement}  
  onDelete={handleDelete}>  
</Counters>
```

# Exercise- Like Component

- Output to Achieve

Showing 9 movies in the database.

Title	Genre	Stock	Rate	Heart	Delete
Terminator	Action	6	2.5	🖤	<button>Delete</button>
Die Hard	Action	5	2.5	🤍	<button>Delete</button>
Get Out	Thriller	8	3.5	🤍	<button>Delete</button>
Trip to Italy	Comedy	7	3.5	🤍	<button>Delete</button>
Airplane	Comedy	7	3.5	🤍	<button>Delete</button>

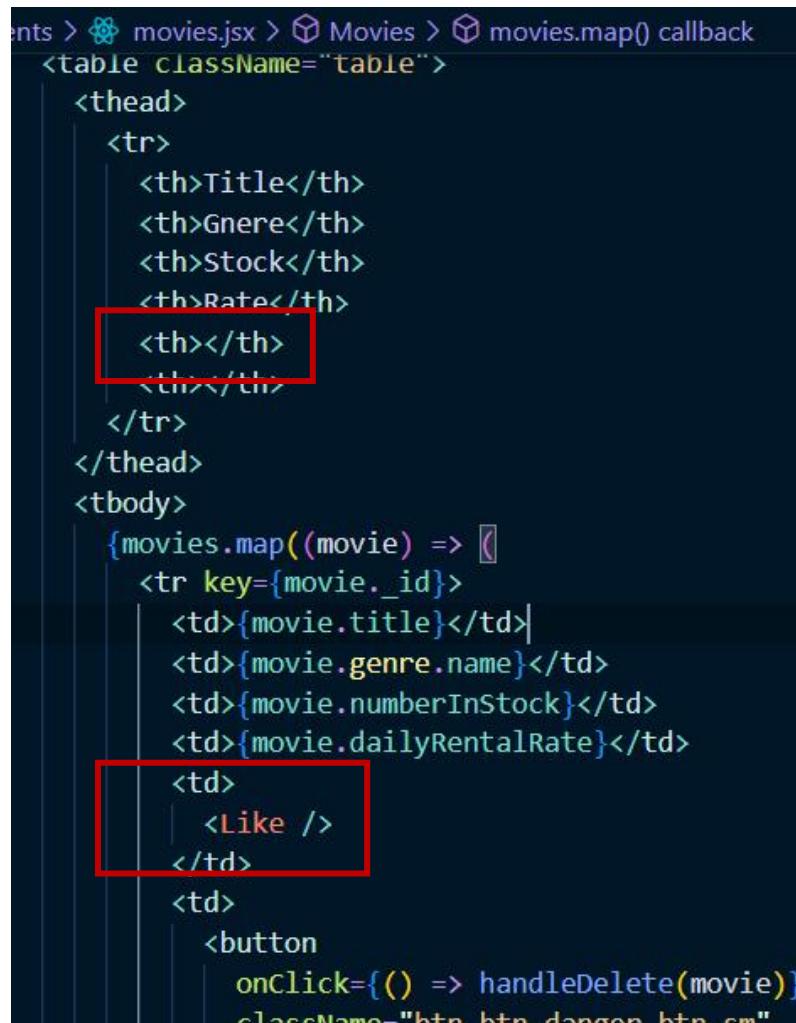
# Exercise- Like Component

- Create like.jsx
- Find Heart Icon from font-awesome

```
src > components > like.jsx > ...
1  function Like() {
2    return <i style={{color: "black",}} className="fa fa-regular fa-heart"></i>;
3  }
4
5  export default Like;
6
```

# Exercise- Like Component

- Import like component in movies component



The image shows a screenshot of a code editor displaying a portion of a React component. The component is a table with a header and a body. The header has five columns: Title, Genre, Stock, Rate, and a blank column. The body maps over an array of movies, displaying their title, genre, stock count, and daily rental rate. In the last column of the body, there is a `<td>` element containing a `<Like />` component, which is highlighted with a red rectangular box. Below this, there is another `<td>` element containing a `<button>` element with an `onClick` handler and a specific class name.

```
  <thead>
    <tr>
      <th>Title</th>
      <th>Genre</th>
      <th>Stock</th>
      <th>Rate</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    {movies.map((movie) => (
      <tr key={movie._id}>
        <td>{movie.title}</td>
        <td>{movie.genre.name}</td>
        <td>{movie.numberInStock}</td>
        <td>{movie.dailyRentalRate}</td>
        <td>
          <Like />
        </td>
        <td>
          <button
            onClick={() => handleDelete(movie)}
            className="btn btn-danger btn-sm"
          >
        </td>
      </tr>
    ))}
  </tbody>

```

# Exercise- Like Component

- Update

```
<td>
  <Like liked={movie.liked} onClick={() => handleLike(movie)} />
</td>
```

movies.jsx

```
const handleLike = (movie) => {
  const index = movies.indexOf(movie);
  movies[index].liked = !movies[index].liked;
  setMovies(
    movies.map((m) => {
      return m;
    })
  );
};
```

likes.jsx

```
<i
  onClick={props.onClick}
  style={{ color: props.liked ? "red" : "black", cursor: "pointer" }}
  className="fa fa-regular fa-heart"
></i>
```



# Exercise- Like Component

- Output

Showing 9 Movies in the Database

Title	Gnere	Stock	Rate		
Terminator	Action	6	2.5	❤	Delete
Die Hard	Action	5	2.5	🖤	Delete
Get Out	Thriller	8	3.5	🖤	Delete
Trip to Italy	Comedy	7	3.5	🖤	Delete
Airplane	Comedy	7	3.5	❤	Delete
Wedding Crashers	Comedy	7	3.5	🖤	Delete
Gone Girl	Thriller	7	4.5	❤	Delete
The Sixth Sense	Thriller	4	3.5	❤	Delete
The Avengers	Action	7	3.5	🖤	Delete

# Pagination, Filtering & Sorting

All Genres
Action
Comedy
Thriller

Showing 9 movies in the database.

Title 	Genre	Stock	Rate	
Airplane	Comedy	7	3.5	 <button>Delete</button>
Die Hard	Action	5	2.5	 <button>Delete</button>
Get Out	Thriller	8	3.5	 <button>Delete</button>
Gone Girl	Thriller	7	4.5	 <button>Delete</button>

# Pagination Component

- Create a folder by name `common` in `components`
- Create `pagination.jsx` component

```
src > components > common > pagination.jsx > [e] c
  1  function Pagination(props) {
  2    return null;
  3  }
  4
  5  export default Pagination;
```

- Update `movies.jsx`

```
</table>
<Pagination
  itemsCounts={count}
  pageSize={pageSize}
  onPageChange={handlePageChange}
/>
</Fragment>
```

# Pagination Component

- Update movies.jsx

```
function Movies() {
  const [movies, setMovies] = useState(getMovies);
  const [pageSize] = useState(4);

  const handleDelete = (movie) => {
    setMovies(movies.filter((m) => m._id !== movie._id));
  };

  const handleLike = (movie) => {
    const index = movies.indexOf(movie);
    movies[index].liked = !movies[index].liked;
    setMovies(
      movies.map((m) => {
        return m;
      })
    );
  };

  const handlePageChange = (page) => {
    console.log(page);
  };
}
```

# Displaying Pages

- Install lodash : `npm i lodash` and update `pagination.jsx`

Showing 9 Movies in the Database

Title	Gnere	Stock	Rate	Heart	Delete
Terminator	Action	6	2.5	♥	Delete
Die Hard	Action	5	2.5	♥	Delete
Get Out	Thriller	8	3.5	♥	Delete
Trip to Italy	Comedy	7	3.5	♥	Delete
Airplane	Comedy	7	3.5	♥	Delete
Wedding Crashers	Comedy	7	3.5	♥	Delete
Gone Girl	Thriller	7	4.5	♥	Delete
The Sixth Sense	Thriller	4	3.5	♥	Delete
The Avengers	Action	7	3.5	♥	Delete

1 2 3

```
movies.jsx U pagination.jsx X
src > components > common > pagination.jsx > Pagination
1 import _ from "lodash";
2
3
4 function Pagination(props) {
5   const { itemsCounts, pageSize } = props;
6   const pagesCount = Math.ceil(itemsCounts / pageSize);
7
8   if (pagesCount === 1) return null;
9   const pages = _.range(1, pagesCount + 1);
10
11  return (
12    <nav>
13      <ul className="pagination">
14        {pages.map((page) => (
15          <li key={page} className="page-item">
16            <a className="page-link">{page}</a>
17          </li>
18        ))}
19      </ul>
20    </nav>
21  );
22}
23
24 export default Pagination;
```

# Handling Page Changes

- In `movies.jsx` add

```
function Movies() {
  const [movies, setMovies] = useState(getMovies);
  const [pageSize] = useState(4);
  const [currentPage, setCurrentPage] = useState(1);
```

```
<Pagination
  itemsCounts={count}
  pageSize={pageSize}
  currentPage={currentPage}
  onPageChange={handlePageChange}
/>
```

```
const handlePageChange = (page) => {
  setCurrentPage(page)
};
```

# Handling Page Changes

- In `pagination.jsx` Update

```
function Pagination(props) {
  const { itemsCounts, pageSize, onPageChange, currentPage } = props;
  const pagesCount = Math.ceil(itemsCounts / pageSize);
  console.log(currentPage)

  if (pagesCount === 1) return null;
  const pages = _.range(1, pagesCount + 1);

  return (
    <nav>
      <ul className="pagination">
        {pages.map((page) => (
          <li key={page} className={page === currentPage ? "page-item active" : "page-item"}>
            <a className="page-link" onClick={()=> onPageChange(page)}>{page}</a>
          </li>
        ))
      </ul>
    </nav>
  );
}
```

# Pagination Data

- Create **utils** folder and inside that create **paginate.jsx**

```
src > components > utils > paginate.jsx > ...
1  import _ from "lodash";
2
3  export function Paginate(items, pageNumber, pageSize) {
4    const startIndex = (pageNumber - 1) * pageSize;
5    return _(items).slice(startIndex).take(pagesize).value();
6  }
7  |
```

# Pagination Data

- Update and add in `movies.jsx`

```
src > components > 🏠 movies.jsx > 📄 Movies
1  import { Fragment, useState } from "react";
2  import { getMovies } from "../../services/fakeMovieService";
3  import Like from "./common/like";
4  import Pagination from "./common/pagination";
5  import { Paginate } from "./utils/paginate";
6
7  function Movies() {
8    const [movies, setMovies] = useState(getMovies);
9    const [pageSize] = useState(4);
10   const [currentPage, setCurrentPage] = useState(1);
11
12   const allMovies = Paginate(movies, currentPage, pageSize);
13
14   const handleDelete = (movie) => {
15     setMovies(movies.filter((m) => m._id !== movie._id));
16   };
17
```

```
</thead>
<tbody>
  {allMovies.map((movie) => (
    <tr key={movie._id}>
      <td>{movie.title}</td>
      <td>{movie.genre.name}</td>
      <td>{movie.numberInStock}</td>
      <td>{movie.dailyRentalRate}</td>
      <td>
        <Like liked={movie.liked} onClick={() =>
          handleDelete(movie)
        }>
      </td>
      <td>
        <button
          onClick={() =>
            handleDelete(movie)
          }>
          Delete
        </button>
      </td>
    </tr>
  ))}
</tbody>
```

# Type Checking with PropTypes

- Install `npm i prop-types`
- Open `pagination.jsx`

```
import PropTypes from "prop-types";
```

```
Pagination.propTypes = {
  itemsCounts: PropTypes.number.isRequired,
  pageSize: PropTypes.number.isRequired,
  currentPage: PropTypes.number.isRequired,
  onPageChange: PropTypes.func.isRequired,
};
```

✖ Warning: Failed prop type: [index.js:2178](#)  
Invalid prop `itemsCount` of type `string`  
supplied to `Pagination`, expected `number`.  
in Pagination (at movies.jsx:78)  
in Movies (at App.js:9)  
in main (at App.js:8)  
in App (at index.js:9)

# ListGroup Components

- Create `listGroup.jsx` in `common` folder.

```
src > components > common > ✨ listGroup.  
1  function ListGroup() {  
2      |     return null;  
3  }  
4  
5  export default ListGroup;
```

```
src > components > movies.jsx > Movies > handleGenreSelect
1 import { useState, useEffect } from "react";
2 import { getMovies } from "../../services/fakeMovieService";
3 import { getGenres } from "../services/fakeGenreService";
4 import Like from "./common/like";
5 import Pagination from "./common/pagination";
6 import { Paginate } from "./utils/paginate";
7 import ListGroup from "./common/listGroup";
8
9 function Movies() {
10   const [movies, setMovies] = useState([]);
11   const [genres, setGenres] = useState([]);
12   const [pagesize] = useState(4);
13   const [currentPage, setCurrentPage] = useState(1);
14
15   useEffect(() => {
16     setMovies(getMovies);
17     setGenres(getGenres);
18   }, []);
19 }
```

- Update  
Movies component

# ListGroup Component Interface

```
src > components > 🎬 movies.jsx > 📄 Movies
  31   |   |   return m;
  32   |   | }
  33   |   );
  34   };
  35
● 36   const handleGenreSelect = (genre) => {
  37   |   console.log(genre)
  38   }
  39
  40   const handlePagechange = (page) => {
  41   |   setCurrentPage(page);
  42   };
  43
  44   const { length: count } = movies;
  45
  46   if (count === 0) return <p>There are No Movies in the Database</p>;
  47
  48   return [
  49     <div className="row">
  50       <div className="col-2">
  ● 51         <ListGroup items={genres} onItemSelected={handleGenreSelect} />
  52       </div>
  53       <div className="col">
  54         <p>Showing {count} Movies in the Database</p>
    <table className="table">
```

# Filtering – Displaying Items

- Update listGroup :

listGroup.jsx

```
const ListGroup = (props) => {
  const { items, textProperty, valueProperty } = props;
  return (
    <ul className="list-group">
      {items.map((item) => (
        <li key={item[valueProperty]} className="list-group-item">
          {item[textProperty]}
        </li>
      ))}
    </ul>
  );
};

export default ListGroup;
```

# Filtering – Displaying Items

- Update Movies :

```
<div className="row">
  <div className="col-3">
    <ListGroup
      items={genres}
      textProperty="name"
      valueProperty="_id"
      onItemSelect={handleGenreSelect}>
    />
  </div>
```

The screenshot shows a user interface for managing movies. On the left, there is a sidebar with a list of genres: Action, Comedy, and Thriller. A red box highlights this sidebar. To the right, a main area displays a table of movies with columns: Title, Genre, Stock, Rate, and Delete button. The table shows five movies: Terminator (Action, Stock 6, Rate 2.5), Die Hard (Action, Stock 5, Rate 2.5), Get Out (Thriller, Stock 8, Rate 3.5), and Trip to Italy (Comedy, Stock 7, Rate 3.5). Each movie row has a 'Delete' button. At the bottom, there is a navigation bar with pages 1, 2, and 3, where page 1 is highlighted.

Title	Genre	Stock	Rate	
Terminator	Action	6	2.5	Delete
Die Hard	Action	5	2.5	Delete
Get Out	Thriller	8	3.5	Delete
Trip to Italy	Comedy	7	3.5	Delete

Showing 9 Movies in the Database

1 2 3

# Default Props

- Update movies.jsx :

```
<ListGroup
  items={genres}
  onItemSelect={handleGenreSelect}
/>
```

- Update listGroup.jsx :

```
ListGroup.defaultProps = {
  textProperty: "name",
  valueProperty: "_id",
};
```

```
export default ListGroup;
```

# Filtering – Handling Selection

- Update ListGroup

```
const ListGroup = (props) => {
  const { items, textProperty, valueProperty, selectedItem, onItemSelected } =
    props;
  return (
    <ul className="list-group">
      {items.map((item) => (
        <li
          onClick={() => onItemSelected(item)}
          key={item[valueProperty]}
          className={
            item === selectedItem ? "list-group-item active" : "list-group-item"
          }
        >
          {item[textProperty]}
        </li>
      ))}
    </ul>
  );
};
```

# Filtering – Handling Selection

- Update movies.jsx

```
src > components > movies.jsx > Movies
11 | const [genres, setGenres] = useState([]);
12 | const [selectedGenre, setSelectedGenre] = useState([]); [selectedGenre, setSelectedGenre]
13 | const [pageSize] = useState(4);
14 | const [currentPage, setCurrentPage] = useState(1);
15 |
16 | useEffect(() => {
```

```
};

const handleGenreSelect = (genre) => {
  setSelectedGenre(genre);
};

const handlePageChange = (page) => {
  setCurrentPage(page);
};

const { length: count } = movies;

if (count === 0) return <p>There are No Movies in the Database</p>

return (
  <div className="row">
    <div className="col-3">
      <ListGroup
        items={genres}
        selectedItem={selectedGenre}
        onItemSelect={handleGenreSelect}
      />
    </div>
  </div>
);
```

# Filtering – Implementing Filtering

- In movies.jsx

```
> components > movies.jsx > Movies

function Movies() {
  const [movies, setMovies] = useState([]);
  const [genres, setGenres] = useState([]);
  const [selectedGenre, setSelectedGenre] = useState(); // Line 1
  const [pageSize] = useState(4);
  const [currentPage, setCurrentPage] = useState(1);

  useEffect(() => {
    setMovies(getMovies);
    setGenres(getGenres);
  }, []);

  const filtered = selectedGenre
    ? movies.filter((m) => m.genre._id === selectedGenre._id)
    : movies; // Line 2

  const allMovies = Paginate(filtered, currentPage, pageSize);

  const handleDelete = (movie) => {
    setMovies(movies.filter((m) => m._id !== movie._id)); // Line 3
  };
}
```

# Filtering – Adding All Genre

- Update Movies.jsx

```
const handleGenreSelect = (genre) => {
  setSelectedGenre(genre);
  setCurrentPage(1);
};
```

```
function Movies() {
  const [movies, setMovies] = useState([]);
  const [genres, setGenres] = useState([]);
  const allGenres = { _id: 0, name: "All Genres" };
  const [selectedGenre, setSelectedGenre] = useState(allGenres);
  const [pageSize] = useState(4);
  const [currentPage, setCurrentPage] = useState(1);

  useEffect(() => {
    setMovies(getMovies);
    setGenres([allGenres, ...getGenres()]);
  }, []);

  const filtered =
    selectedGenre && selectedGenre._id
      ? movies.filter((m) => m.genre._id === selectedGenre._id)
      : movies;
```

# Sorting – Extracting MoviesTable

- Create `moviesTable.jsx` in Components folder
- Copy table from movies component and paste

```
components > 📂 moviesTable.jsx > 📁 MoviesTable > 📁 allMovies.map() c
import Like from "./common/like";

function MoviesTable(props) {
  const { allMovies, onDelete, onLike } = props;

  return (
    <table className="table">
      <thead>
```

```
<tbody>
  {allMovies.map((movie) => [
    <tr key={movie._id}>
      <td>{movie.title}</td>
      <td>{movie.genre.name}</td>
      <td>{movie.numberInStock}</td>
      <td>{movie.dailyRentalRate}</td>
      <td>
        <Like liked={movie.liked} onClick={() => onLike(movie)} />
      </td>
      <td>
        <button
          onClick={() => onDelete(movie)}
          className="btn btn-danger btn-sm"
        >
          Delete
        </button>
      </td>
    </tr>
  ])}
</tbody>
```

# Sorting - Raising the Sort Event

In Movies.jsx

```
const handleSort = (path) => {
  console.log(path);
};
```

```
<MoviesTable
  onDelete={handleDelete}
  onLike={handleLike}
  allMovies={allMovies}
  onSort={handleSort}>
```

/>

```
<Pagination
```

# Sorting - Raising the Sort Event

- In moviesTable

```
function MoviesTable(props) {  
  const { allMovies, onDelete, onLike, onSort } = props;  
  return [  
    <table className="table">  
      <thead>  
        <tr>  
          <th onClick={() => onSort("title")}>Title</th>  
          <th onClick={() => onSort("genre.name")}>Genre</th>  
          <th onClick={() => onSort("numberInStock")}>Stock</th>  
          <th onClick={() => onSort("dailyRentalRate")}>Rate</th>  
          <th></th>  
          <th></th>  
        </tr>  
    </thead>  
    <tbody>  
      {allMovies.map(movie => (  
        <tr key={movie.id}>  
          <td>{movie.title}</td>  
          <td>{movie.genre.name}</td>  
          <td>{movie.numberInStock}</td>  
          <td>{movie.dailyRentalRate}</td>  
          <td>  
            <button onClick={() => onDelete(movie.id)}>Delete</button>  
            <button onClick={() => onLike(movie.id)}>Like</button>  
          </td>  
        </tr>  
      ))}  
    </tbody>  
  </table>  
}
```

# Sorting - Implementing Sorting

In movies.jsx

```
import ListGroup from "./common/listGroup";
import MoviesTable from "./moviesTable";
import _ from "lodash";

function Movies() {
  const [movies, setMovies] = useState([]);
  const [genres, setGenres] = useState([]);
  const allGenres = { _id: 0, name: "All Genres" };
  const [selectedGenre, setSelectedGenre] = useState(allGenres);
  const [pageSize] = useState(4);
  const [currentPage, setCurrentPage] = useState(1);
  const [sortColumn, setSortColumn] = useState({ path: "title", order: "asc" });
```

# Sorting - Implementing Sorting

In movies.jsx

```
const sorted = _.orderBy(filtered, [sortColumn.path], [sortColumn.order]);
const allMovies = Paginate(sorted, currentPage, pageSize);

const handleSort = (path) => {
  if (sortColumn.path === path) {
    setSortColumn({
      path,
      order: sortColumn.order === "asc" ? "desc" : "asc",
    });
  } else {
    setSortColumn([
      {
        path,
        order: "asc",
      },
    ]);
  }
};
```

# Routing and it's component

- In non-single page applications, when you click on a link in the browser, a request is sent to the server before the HTML page gets rendered.
- In React, the page contents are created from our components. So what React Router does is intercept the request being sent to the server and then injects the contents dynamically from the components we have created.
- This is the general idea behind SPAs which allows content to be rendered faster without the page being refreshed.

## `<BrowserRouter>`

- ▶ Type declaration

A `<BrowserRouter>` stores the current location in the browser's address bar using clean URLs and navigates using the browser's built-in history stack.

`<BrowserRouter window>` defaults to using the current document's defaultView, but it may also be used to track changes to another window's URL, in an `<iframe>`, for example.

```
1 import * as React from "react";
2 import * as ReactDOM from "react-dom";
3 import { BrowserRouter } from "react-router-dom";
4
5 ReactDOM.render(
6   <BrowserRouter>
7     {/* The rest of your app goes here */}
8   </BrowserRouter>,
9   root
10 );
```

## `<Routes>`

Rendered anywhere in the app, `<Routes>` will match a set of child routes from the current location.

Whenever the location changes, `<Routes>` looks through all its child routes to find the best match and renders that branch of the UI.

`<Route>` elements may be nested to indicate nested UI, which also correspond to nested URL paths. Parent routes render their child routes by rendering an <Outlet>.

```
1  <Routes>
2    <Route path="/" element={<Dashboard />}>
3      <Route
4        path="messages"
5        element={<DashboardMessages />}
6      />
7      <Route path="tasks" element={<DashboardTasks />} />
8    </Route>
9    <Route path="about" element={<AboutPage />} />
10   </Routes>
```

## ``Route``

Routes are perhaps the most important part of a React Router app. They couple URL segments to components, data loading and data mutations. Through route nesting, complex application layouts and data dependencies become simple and declarative.

## ``path``

The path pattern to match against the URL to determine if this route matches a URL, link href, or form action.

## ``element``

The element to render when the route matches the URL.

```
1 <Route path="/for-sale" element={<Properties />} />
```

# Setup

- Create a new application named **router**
- **npx create-react-app router**
- Create **navbar** component
- Update **app.js**

```
src > JS App.js > ...
1 import './App.css';
2 import Navbar from './components/navbar';
3
4 function App() {
5   return (
6     <div className="App">
7       <Navbar />
8     </div>
9   );
10 }
11
12 export default App;
13 |
```

```
components > navbar.jsx > ...
function Navbar() {
  return (
    <ul>
      <li>
        <a href="/">Home</a>
      </li>
      <li>
        <a href="/products">Products</a>
      </li>
      <li>
        <a href="/posts/2018/06">Posts</a>
      </li>
      <li>
        <a href="/admin">Admin</a>
      </li>
    </ul>
  );
}

export default Navbar;
```

# Routing

## How to Install React Router

To install React Router, all you have to do is run

**npm install react-router-dom@6** in your project terminal and  
then wait for the installation to complete.

If you are using yarn then use this command:

**yarn add react-router-dom@6.**

# Adding Routing

- npm install react-router-dom
- Update index.js

```
import { reportWebVitals } from './reportWebVitals';
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

# Create multiple components

- Create home.jsx

```
c > components > ⚛ home.jsx > ...
1  import React from "react";
2
3  const Home = () => {
4      return <h1>Home</h1>;
5  };
6
7  export default Home;
8 |
```

# Create multiple components

- Create products.jsx

```
import { useState } from "react";

function Products() {
  const [products, setProducts] = useState([
    { id: 1, name: "Product 1" },
    { id: 2, name: "Product 2" },
    { id: 3, name: "Product 3" },
  ]);

  return (
    <div>
      <h1>Products</h1>
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            <a href={`/products/${product.id}`}>{product.name}</a>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default Products;
```

# Create multiple components

- Create posts.jsx

```
rc > components > posts.jsx > [?] default
  1  const Posts = () => {
  2    return (
  3      <div>
  4        <h1>Posts</h1>
  5        Year: , Month:
  6      </div>
  7    );
  8  };
  9
10  export default Posts;
11
```

# Create multiple components

- Create a folder by name `admin` in `components`
- Create `dashboard.jsx` in admin folder in components

```
const Dashboard = ({ match }) => {
  return (
    <div>
      <h1>Admin Dashboard</h1>
    </div>
  );
}

export default Dashboard;
```

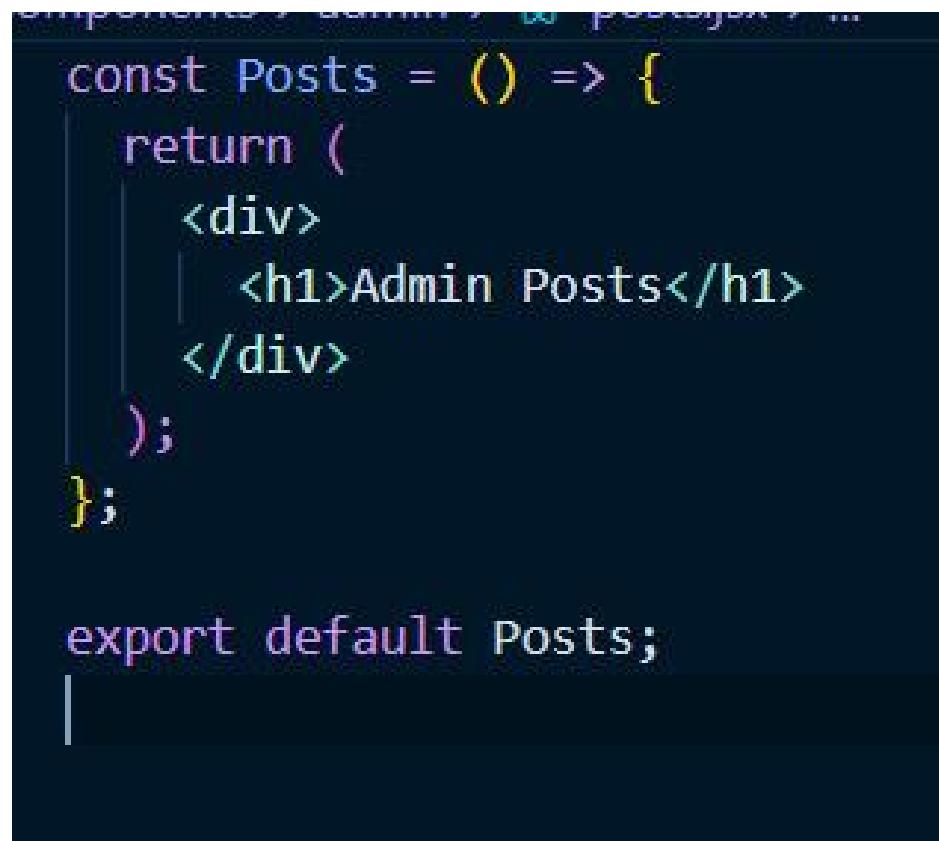
# Create multiple components

- Create `users.jsx` in `admin` folder in components

```
src > components > admin > ✨ users.jsx > ...
1  const Users = () => {
2    return <h1>Admin Users</h1>;
3  };
4
5  export default Users;
6
```

# Create multiple components

- Create `posts.jsx` in `admin` folder in components



A screenshot of a code editor showing a file named `Posts.jsx`. The code defines a functional component named `Posts` that returns a `<div>` containing an `<h1>Admin Posts</h1>`. The code is written in JSX syntax.

```
const Posts = () => {
  return (
    <div>
      <h1>Admin Posts</h1>
    </div>
  );
};

export default Posts;
```

# Define Routes

- In App.js

```
import './App.css';
import { Routes, Route } from "react-router-dom"

import Navbar from './components/navbar';
import Dashboard from './components/admin/dashboard'
import Home from './components/home';
import Products from './components/products'
import Posts from './components/posts'

function App() {
  return (
    <div className="App">
      <Navbar />
      <div className='content'>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="products" element={<Products />} />
          <Route path="posts" element={<Posts />} />
          <Route path="admin" element={<Dashboard />} />
        </Routes>
      </div>
    </div>
  );
}
```

# Link

- Update Navbar

```
> components > navbar.jsx > ...
1  import { Link } from "react-router-dom";
2
3  function Navbar() {
4      return (
5          <ul>
6              <li>
7                  <Link to="/">Home</Link>
8              </li>
9              <li>
10                 <Link to="/products">Products</Link>
11             </li>
12             <li>
13                 <Link to="/posts">Posts</Link>
14             </li>
15             <li>
16                 <Link to="/admin">Admin</Link>
17             </li>
18         </ul>
19     );
20 }
21
22 export default Navbar;
```

# Error Page

- Create NotFound.jsx component

```
const NotFound = () => {
  return <h1>Not Found</h1>;
};

export default NotFound;
```

# Error Page

- Edit App.jsx

```
import Home from './components/home';
import Products from './components/products'
import Posts from './components/posts'
import NotFound from './components/notFound';

function App() {
  return (
    <div className="App">
      <Navbar />
      <div className='content'>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="products" element={<Products />} />
          <Route path="posts" element={<Posts />} />
          <Route path="admin" element={<Dashboard />} />
          <Route path="*" element={<NotFound />} />
        </Routes>
      </div>
    </div>
  );
}
```

# Route Parameters

- In App.js

```
import Home from './components/home';
import Products from './components/products'
import ProductDetails from './components/productDetails';
import Posts from './components/posts';
import NotFound from './components/notFound';

function App() {
  return (
    <div className="App">
      <Navbar />
      <div className='content'>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="products" element={<Products />} />
          <Route path="products/:id" element={<ProductDetails />} />
          <Route path="posts/:year/:month" element={<Posts />} />
          <Route path="admin" element={<Dashboard />} />
          <Route path="*" element={<NotFound />} />
        </Routes>
      </div>
    </div>
  );
}
```

# Route Parameters

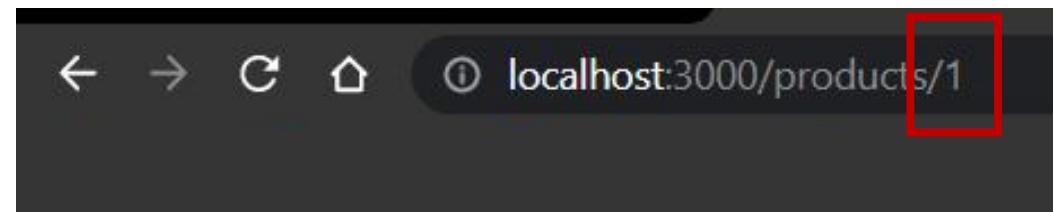
- Create `productDetails.jsx`

```
components > productDetails.jsx > ...
import { useParams } from "react-router-dom";

function ProductDetails() {
  let { id } = useParams();

  return (
    <div>
      <h1>Product Details - {id}</h1>
      <button>Save</button>
    </div>
  );
}

export default ProductDetails;
```



- [Home](#)
- [Products](#)
- [Posts](#)
- [Admin](#)

**Product Details - 1**

Save

# Optional Parameters

- In App.js

```
<div className='content'>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="products" element={<Products />} />
    <Route path="products/:id" element={<ProductDetails />} />
    <Route path="/posts">
      <Route index element={<Posts />} />
      <Route path=":year" element={<Posts />} />
      <Route path=":year/:month" element={<Posts />} />
    </Route>
    <Route path="admin" element={<Dashboard />} />
    <Route path="*" element={<NotFound />} />
  </Routes>
</div>
</div>
```

# Optional Parameters

- In `posts.js`

```
1 import { useParams } from "react-router-dom";
2
3 const Posts = () => {
4     const { year, month } = useParams();
5     return (
6         <div>
7             <h1>Posts</h1>
8             Year: {year}, Month: {month}
9         </div>
10    );
11}
12
13 export default Posts;
```

The `useNavigate` hook returns a function that lets you navigate programmatically, for example in an effect:

```
1 import { useNavigate } from "react-router-dom";
2
3 function useLogoutTimer() {
4   const userIsInactive = useFakeInactiveUser();
5   const navigate = useNavigate();
6
7   useEffect(() => {
8     if (userIsInactive) {
9       fake.logout();
10      navigate("/session-timed-out");
11    }
12  }, [userIsInactive]);
13}
```

# Programmatic Navigation

- In `productDetails.jsx`

```
› components > productDetails.jsx > ...
import { useNavigate, useParams } from "react-router-dom";

function ProductDetails() {
  let { id } = useParams();
  const navigate = useNavigate();

  const saveProduct = () => {
    navigate("/");
  };

  return (
    <div>
      <h1>Product Details - {id}</h1>
      <button onClick={saveProduct}>Save</button>
    </div>
  );
}

export default ProductDetails;
```

# Exercise – Navbar & Routing

Vidly Movies Customers Rentals

All Genres	Showing 9 movies in the database.			
	Title ^	Genre	Stock	Rate
Action	Airplane	Comedy	7	3.5
Comedy	Die Hard	Action	5	2.5
Thriller	Get Out	Thriller	8	3.5
	Gone Girl	Thriller	7	4.5

1 2 3

# Adding React Router

- Install react router dom [Go to vidly folder and `npm i react-router-dom`]
- In `index.jsx`

```
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { BrowserRouter } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.css";
import "font-awesome/css/font-awesome.css";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

# Creating Nav Components

```
src > components > ✨ customers.jsx > [?] default
1  const Customers = () => {
2    return <h1>Customers</h1>;
3  };
4
5  export default Customers;
```

```
src > components > ✨ movieForm.jsx > [?] default
```

```
1  const MovieForm = () => {
2    return (
3      <div>
4        <h1>Movie Form {} </h1>
5        <button className="btn btn-primary">Save</button>
6      </div>
7    );
8  };
9
10 export default MovieForm;
```

```
src > components > ✨ rentals.jsx > ...
```

```
1  const Rentals = () => {
2    return <h1>Rentals</h1>;
3  };
4
5  export default Rentals;
6  |
```

```
src > components > ✨ notFound.jsx > ...
```

```
1  const NotFound = () => {
2    return <h1>Not Found</h1>;
3  };
4
5  export default NotFound;
6  |
```

# Adding Routes

- In App.jsx

```
import "./App.css";
import { Routes, Route } from "react-router-dom";

import NavBar from "./components/navBar";
import Movies from "./components/movies";
import Customers from "./components/customers";
import Rentals from "./components/rentals";
import MovieForm from "./components/movieForm";
import NotFound from "./components/notFound";

function App() {
  return (
    <div className="App">

      <main className="container">
        <Routes>
          <Route path="/" element={<Movies />} />
          <Route path="/:id" element={<MovieForm />} />
          <Route path="customers" element={<Customers />} />
          <Route path="rentals" element={<Rentals />} />
          <Route path="*" element={<NotFound />} />
        </Routes>
      </main>
    </div>
  );
}

export default App;
```

# Get the Navbar.jsx from

<https://github.com/codewithz/cwz-react-requirements/blob/main/Navbar.jsx>

The screenshot shows a GitHub repository page for 'codewithz / cwz-react-requirements'. The repository is public. The 'Code' tab is selected. A commit from 'codewithz' titled 'Create Navbar.jsx' is highlighted with a red box around the 'Navbar.jsx' file entry in the commit log.

File	Description	Time
Navbar.jsx	Create Navbar.jsx	15 seconds ago
loginForm.html	loginForm Skeleton	2 minutes ago
services.zip	Add files via upload	3 days ago

# Adding the Navbar

- Create navbar.jsx

```
components > navBar.jsx > ...
import { Link, NavLink } from "react-router-dom";

const NavBar = () => {
  return (
    <nav className="navbar navbar-expand-lg navbar-light bg-light">
      <Link className="navbar-brand" to="/">
        Vidly
      </Link>
      <button
        className="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbarNavAltMarkup"
        aria-controls="navbarNavAltMarkup"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span className="navbar-toggler-icon" />
      </button>
      <div className="collapse navbar-collapse" id="navbarNavAltMarkup">
```

# Adding the Navbar

- Create navbar.jsx

```
    <span className="navbar-toggler-icon" />
</button>
<div className="collapse navbar-collapse" id="navbarNavAltMarkup">
  <div className="navbar-nav">
    <NavLink className="nav-item nav-link" to="/">
      Movies
    </NavLink>
    <NavLink className="nav-item nav-link" to="/customers">
      Customers
    </NavLink>
    <NavLink className="nav-item nav-link" to="/rentals">
      Rentals
    </NavLink>
  </div>
</div>
</nav>
);

};

export default NavBar;
```

# Adding the Navbar

- In App.jsx

```
> ⚙ App.jsx > ...
  import "./App.css";
  import { Routes, Route } from "react-router-dom";

  import NavBar from "./components/navBar";
  import Movies from "./components/movies";
  import Customers from "./components/customers";
  import Rentals from "./components/rentals";
  import MovieForm from "./components/movieForm";
  import NotFound from "./components/notFound";

  function App() {
    return (
      <div className="App">
        <NavBar />
        <main className="container">
          <Routes>
            <Route path="/" element={<Movies />} />
            <Route path="/:id" element={<MovieForm />} />
            <Route path="customers" element={<Customers />} />
            <Route path="rentals" element={<Rentals />} />
            <Route path="*" element={<NotFound />} />
          </Routes>
        </main>
      </div>
```

# Linking to the Movie Form

- In `moviesTable.jsx`

```
src > components > moviesTable.jsx > MoviesTable
  1 import { Link } from "react-router-dom";
  2 import Like from "./common/Like";
  3
  4 function MoviesTable(props) {
  5   const { allMovies, onDelete, onLike, onSort } = props;
  6   return [
  7     <table className="table">
  8       <thead>
  9         <tr>
 10           <th onClick={() => onSort("title")}>Title</th>
 11           <th onClick={() => onSort("genre.name")}>Genre</th>
 12           <th onClick={() => onSort("numberInStock")}>Stock</th>
 13           <th onClick={() => onSort("dailyRentalRate")}>Rate</th>
 14           <th></th>
 15           <th></th>
 16         </tr>
 17       </thead>
 18       <tbody>
 19         {allMovies.map((movie) => (
 20           <tr key={movie.id}>
 21             <td><Link to={`/ ${movie._id}`}>{movie.title}</Link></td>
 22             <td>{movie.genre.name}</td>
 23             <td>{movie.numberInStock}</td>
 24             <td>{movie.dailyRentalRate}</td>
 25             <td>
 26               <Like liked={movie.liked} onClick={() => onLike(movie)} />
 27             </td>
 28           </tr>
 29         ))
 30       </tbody>
 31     </table>
 32   ];
 33 }
```

# Linking to the Movie Form

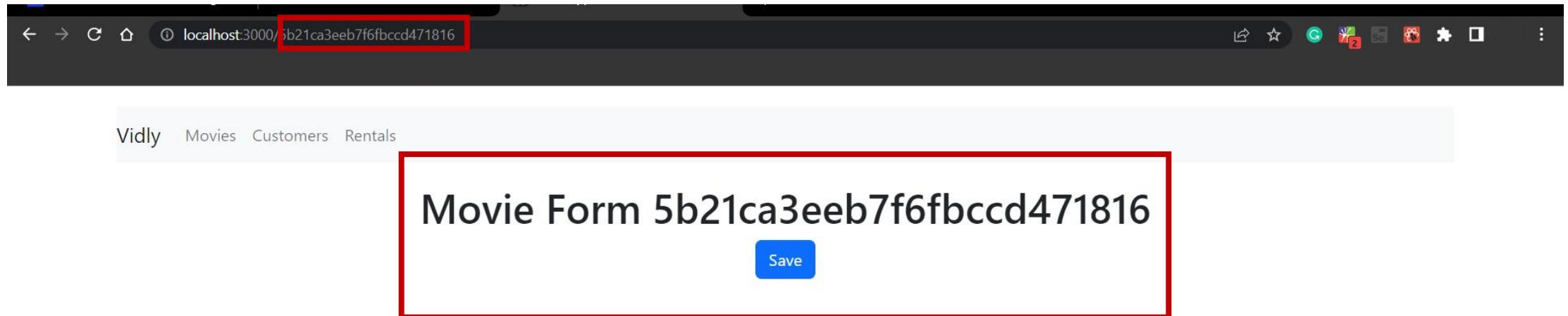
- In `moviesForm.jsx`

```
> components > ✨ movieForm.jsx > ...
import { useNavigate, useParams } from "react-router-dom";

const MovieForm = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  return (
    <div>
      <h1>Movie Form {id}</h1>
      <button className="btn btn-primary" onClick={() => navigate("/")}>
        Save
      </button>
    </div>
  );
};

export default MovieForm;
```

# Linking to the Movie Form Output





# Forms

## Login

Username

Password

[Login](#)

## Register

Username

user@domain.com

Password

\*\*\*\*\*

Name

[Register](#)

## Movie Form

Title

Airplane

Genre

Action

Number in Stock

29

Rate

3.5

[Save](#)

All Genres

Action

Comedy

Thriller

New Movie

Showing 9 movies in the database.

Search...

Title ▲    Genre    Stock    Rate

Airplane    Comedy    7    3.5    [Delete](#)

# Building A Bootstrap Form

- Create **LoginForm** component

```
src > components > LoginForm.jsx > ...
1  function LoginForm() {
2    return <h1>Login</h1>;
3  }
4
5  export default LoginForm;
6  |
```

- In **Navbar** add

```
      </NavLink>
      <NavLink className="nav-item nav-link" to="/login">
        Login
      </NavLink>
    </div>
```

# Building A Bootstrap Form

- In App.js add route

```
import Rentals from "./components/rentals";
import MovieForm from "./components/movieForm";
import LoginForm from "./components/loginForm";
import NotFound from "./components/notFound";

function App() {
  return (
    <div className="App">
      <NavBar />
      <main className="container">
        <Routes>
          <Route path="/" element={<Movies />} />
          <Route path="/:id" element={<MovieForm />} />
          <Route path="customers" element={<Customers />} />
          <Route path="rentals" element={<Rentals />} />
          <Route path="login" element={<LoginForm />} />
          <Route path="*" element={<NotFound />} />
        </Routes>
      </main>
    </div>
  );
}
```

# Get the skeleton code for loginForm from

<https://github.com/codewithz/cwz-react-requirements/blob/main/loginForm.html>

The screenshot shows a GitHub repository page for 'codewithz / cwz-react-requirements'. The 'Code' tab is selected. A red box highlights the file 'loginForm.html' in the list of files.

File	Description	Time Ago
Navbar.jsx	Create Navbar.jsx	3 minutes ago
loginForm.html	loginForm Skeleton	5 minutes ago
services.zip	Add files via upload	3 days ago

At the bottom, there is a call to action: "Help people interested in this repository understand your project by adding a README." with a "Add a README" button.

# Building A Bootstrap Form

- Update `loginForm.jsx`

```
components > loginForm.jsx > ...
function LoginForm() {
  return (
    <div className="container">
      <h1>Login</h1>
      <form className="container w-50 m-auto">
        <div className="form-group mt-3">
          <label htmlFor="username">Username</label>
          <input id="username" type="text" className="form-control mt-2" />
        </div>
        <div className="form-group mt-3">
          <label htmlFor="password">Password</label>
          <input id="password" type="text" className="form-control mt-2" />
        </div>
        <div className="mt-3">
          <button className="btn btn-primary">Login</button>
        </div>
      </form>
    </div>
  );
}

export default LoginForm;
```

# Building A Bootstrap Form

- Login form should look like

The image shows a screenshot of a web application's login page. At the top left, there is a navigation bar with the brand name "Vidly" and links for "Movies", "Customers", "Rentals", and "Login". The main content area is titled "Login". It contains two input fields: one for "Username" with the value "scasc" and another for "Password" with the value "acsacsc". Below the password field is a blue "Login" button. The entire form area is enclosed in a thick red rectangular border.

Vidly Movies Customers Rentals Login

Login

Username

scasc

Password

acsacsc

Login

# Handling Form Submission

loginForm.jsx

```
components > LoginForm.jsx > LoginForm
function LoginForm() {
  const handleSubmit = (e) => {
    e.preventDefault();
  }

  return (
    <div className="container">
      <h1>Login</h1>
      <form onSubmit={handleSubmit} className="container w-50 m-auto">
        <div className="form-group mt-3">
          <label htmlFor="username">Username</label>
          <input id="username" type="text" className="form-control mt-2" />
        </div>
        <div className="form-group mt-3">
          <label htmlFor="password">Password</label>
          <input id="password" type="text" className="form-control mt-2" />
        </div>
      </form>
    </div>
  );
}
```

# Controlled Elements

loginForm.jsx

```
> components > loginForm.jsx > LoginForm
>
> import { useState } from "react";
> function LoginForm() {
>   const [account, setAccount] = useState({ username: "", password: "" });
>
>   const handleChange = (e) => {
>     const { id, value } = e.target;
>     setAccount((prevState) => ({
>       ...prevState,
>       [id]: value,
>     }));
>     console.log(account);
>   };
>
>   const handleSubmit = (e) => {
>     e.preventDefault();
>   };
>
>   return [
>     <div className="container">
>       <h1>Login</h1>
>       <form onSubmit={handleSubmit} className="container w-50 m-auto">
>         <div className="form-group mt-3">
>           <label htmlFor="username">Username</label>
>           <input
>             autoFocus
>             type="text"
>             id="username"
>             onChange={handleChange}
>             value={account.username}
>           />
>         </div>
>         <div className="form-group mt-3">
>           <label htmlFor="password">Password</label>
>           <input
>             type="password"
>             id="password"
>             onChange={handleChange}
>             value={account.password}
>           />
>         </div>
>         <div className="d-grid gap-2 mt-3">
>           <button type="submit" className="btn btn-primary">Login</button>
>         </div>
>       </form>
>     </div>
>   ];
> }
```

# Controlled Elements

loginForm.jsx

```
Components > loginForm.jsx > LoginForm
    <label htmlFor="username">Username</label>
    <input
        autoFocus
        onChange={handleChange}
        id="username"
        type="text"
        className="form-control mt-2"
    />
    </div>
    <div className="form-group mt-3">
        <label htmlFor="password">Password</label>
        <input
            onChange={handleChange}
            id="password"
            type="text"
            className="form-control mt-2"
        />
        </div>
        <div className="mt-3">
            <button className="btn btn-primary">Login</button>
        </div>
    </form>
</div>
);
```

# Validation

loginForm.jsx

```
> components > loginForm.jsx > LoginForm > handleSubmit
1 import { useState } from "react";
2 function LoginForm() {
3   const [account, setAccount] = useState({ username: "", password: "" });
4   const [errors, setErrors] = useState({});  
5
6   const handleChange = (e) => {
7     const { id, value } = e.target;
8     setAccount((prevState) => ({
9       ...prevState,
10      [id]: value,
11    }));
12     console.log(account);
13   };
14
15   const validate = () => {
16     return { username: "Username is Required." };
17   };
18
19   const handleSubmit = (e) => [
20     e.preventDefault();
21     setErrors(validate());
22     if (errors) return;
23   ];
24 }
```

# A Basic Validation Implementation

loginForm.jsx

```
components > loginForm.jsx > LoginForm
  import { useState } from "react";
  function LoginForm() {
    const [account, setAccount] = useState({ username: "", password: "" });
    const [errors, setErrors] = useState({});
```

```
const validate = (values) => {
  const formErrors = {};
  if (values.username.trim() === "") {
    formErrors.username = "Username is required.";
  }
  if (values.password.trim() === "") {
    formErrors.password = "Password is required.";
  }
  return Object.keys(formErrors).length === 0 ? null : formErrors;
};
```

```
const handleSubmit = (e) => {
  e.preventDefault();
  setErrors(validate(account));
  if (errors) return;
};
```

# Displaying Validation Errors

loginForm.jsx

```
const validate = (values) => {
  const formErrors = {};
  if (values.username.trim() === "") {
    formErrors.username = "Username is required.";
  }
  if (values.password.trim() === "") {
    formErrors.password = "Password is required.";
  }
  return formErrors;
};
```

```
<div className="form-group mt-3">
  <label htmlFor="username">Username</label>
  <input
    autoFocus
    onChange={handleChange}
    id="username"
    type="text"
    className="form-control mt-2"
  />
  {errors.username && <div className="alert alert-danger">{errors.username}</div>}
</div>
<div className="form-group mt-3">
  <label htmlFor="password">Password</label>
  <input
    onChange={handleChange}
    id="password"
    type="text"
    className="form-control mt-2"
  />
  {errors.password && <div className="alert alert-danger">{errors.password}</div>}
</div>
<div className="mt-3">
  <button className="btn btn-primary">Login</button>
</div>
```

# Login Page

Vidly   Movies   Customers   Rentals   Login

## Login

Username

Username is required.

Password

Password is required.

Login

# Validation On Change

loginForm.jsx

```
const validateProperty = ({ id, value }) => {
  if (id === "username") {
    if (value.trim() === "") {
      return "Username is required.";
    }
  }
  if (id === "password") {
    if (value.trim() === "") {
      return "Password is required.";
    }
  }
};

const handleChange = (e) => {
```

```
const handleChange = (e) => {
  const { id, value } = e.target;
  setAccount((prevState) => ({
    ...prevState,
    [id]: value,
  }));

  const errorMessage = validateProperty({ id, value });
  if (errorMessage) {
    setErrors((prevState) => ({
      ...prevState,
      [id]: errorMessage,
    }));
  } else delete errors[id];
};
```

# Joi

- npm i joi-browser

```
Components > loginForm.jsx > LoginForm > validate > result
import { useState } from "react";
import Joi from "joi-browser";

function LoginForm() {
  const [account, setAccount] = useState({ username: "", password: "" });
  const [errors, setErrors] = useState({});

  const schema = {
    username: Joi.string().required(),
    password: Joi.string().required(),
  };

  const validate = (values) => [
    const result = Joi.validate(account, schema, {
      abortEarly: false,
    });
    console.log(result);

    const formErrors = {};
    if (values.username.trim() === "") {
```

Result from Joi

[LoginForm.jsx:21](#)

```
:  
  {error: ValidationError: child "Username" fails because ["User  
  ▼name" is not allowed to be empty]  
    at expo..., value: {...}, then: f, catch: f} ⓘ  
  ▼catch: f _catch(reject)  
    length: 1  
    name: "_catch"  
    ►prototype: {constructor: f}  
      arguments: (...)  
      caller: (...)  
      [[FunctionLocation]]: joi-browser.js:1717  
    ►[[Prototype]]: f ()  
    ►[[Scopes]]: Scopes[6]  
    ▼error: ValidationError: child "Username" fails because ["User  
      ►annotate: f (stripColorCodes)  
      ▼details: Array(1)  
        ▼0:  
          ►context: {value: '', invalids: Array(1), key: 'username'  
            message: "\"Username\" is not allowed to be empty"  
            ►path: Array(1)  
              0: "username"  
              length: 1  
            ►[[Prototype]]: Array(0)  
            type: "any.empty"  
            ►[[Prototype]]: Object  
            length: 1  
          ►[[Prototype]]: Array(0)
```

# Validating a Form using Joi

loginForm.jsx

```
const [errors, setErrors] = useState({}),  
  
const schema = {  
  username: Joi.string().required().label("Username"),  
  password: Joi.string().required().label("Password"),  
};  
  
const validateProperty = ({ id, value }) => {  
  if (id === "username") {  
    // Validation logic for Username  
  }  
  if (id === "password") {  
    // Validation logic for Password  
  }  
};
```

```
const validate = (values) => {  
  const result = Joi.validate(account, schema, {  
    abortEarly: false,  
  });  
  if (!result.error) return null;  
  
  const formErrors = {};  
  for (let item of result.error.details) {  
    formErrors[item.path[0]] = item.message;  
  }  
  return formErrors;  
};
```

# Validating a Field using JOI

loginForm.jsx

```
    password: Joi.string().required().label('Password'),  
};
```

```
const validateProperty = ({ id, value }) => {  
  const obj = { [id]: value };  
  const currentSchema = { [id]: schema[id] };  
  const { error } = Joi.validate(obj, currentSchema);  
  return error ? error.details[0].message : null;  
};
```

```
const handleChange = (e) => {  
  const { id, value } = e.target;  
  setAccount((prevState) => ({
```

## Disabling the Submit Button

```
</div>  
<div className="mt-3">  
  <button disabled={validate()} className="btn btn-primary">  
    Login  
  </button>  
</div>
```

# Extracting a Reusable Form Methods

- Create a folder by name `helper` inside `components`
- Create `form.jsx` inside helpers folder and add all the methods from `loginForm.jsx` and export them

```
src > components > helpers > ⚙ form.jsx > [?] validateProperty
  1 import Joi from "joi-browser";
  2
  3 export const validate = (data, schema) => {
  4   const result = Joi.validate(data, schema, {
  5     abortEarly: false,
  6   });
  7   if (!result.error) return null;
  8
  9   const formErrors = {};
 10   for (let item of result.error.details) {
 11     formErrors[item.path[0]] = item.message;
 12   }
 13   return formErrors;
 14 };
 15
 16 export const validateProperty = ({ id, value }, schema) => [
 17   const obj = { [id]: value };
 18   const currentSchema = { [id]: schema[id] };
 19   const { error } = Joi.validate(obj, currentSchema);
 20   return error ? error.details[0].message : null;
 21 ];
 22
```

# Extracting a Reusable Form Methods

- Create `form.jsx` inside helpers folder and add all the methods from `loginForm.jsx` and export them

```
export const handleChange = (e, schema, data, setData, errors, setErrors) => {
  const { id, value } = e.target;
  setData((prevState) => ({
    ...prevState,
    [id]: value,
  }));

  const errorMessage = validateProperty({ id, value }, schema);
  if (errorMessage) {
    setErrors((prevState) => ({
      ...prevState,
      [id]: errorMessage,
    }));
  } else delete errors[id];
};


```

```
export const handleSubmit = (e, data, schema, errors, setErrors, doSubmit) => {
  e.preventDefault();

  setErrors(validate(data, schema) ?? {});

  doSubmit();
};
```

# Extracting a Reusable Form Methods

- Import and use it inside `loginForm.jsx`

```
Components > loginForm.jsx > LoginForm
import { useState } from "react";
import Joi from "joi-browser";

import { validate, handleChange, handleSubmit } from "./helpers/form";

function LoginForm() {
  const [data, setData] = useState({ username: "", password: "" });
  const [errors, setErrors] = useState({});

  const schema = {
    username: Joi.string().required().label("Username"),
    password: Joi.string().required().label("Password"),
  };

  const doSubmit = () => {
    console.log("Submitted");
  };

  return (
    <Form>
      <Input type="text" value={data.username} onChange={handleChange("username")}></Input>
      <Input type="password" value={data.password} onChange={handleChange("password")}></Input>
      <Button onClick={doSubmit}>Submit</Button>
    </Form>
  );
}

export default LoginForm;
```

# Extracting a Reusable Form Methods

- Import and use it inside `loginForm.jsx`

```
return (
  <div className="container">
    <h1>Login</h1>
    <form
      onSubmit={(e) =>
        handleSubmit(e, data, schema, errors, setErrors, doSubmit)
      }
      className="container w-50 m-auto"
    >
      <div className="alert alert-danger">{errors.password}</div>
    </div>
    <div className="mt-3">
      <button disabled={validate(data, schema)} className="btn btn-primary">
        Login
      </button>
    </div>
  </div>
```

```
<div className="form-group mt-3">
  <label htmlFor="username">Username</label>
  <input
    autoFocus
    onChange={(e) =>
      handleChange(e, schema, data, setData, errors, setErrors)
    }
    id="username"
    type="text"
    className="form-control mt-2"
  />
  {errors.username && (
    <div className="alert alert-danger">{errors.username}</div>
  )}
</div>
<div className="form-group mt-3">
  <label htmlFor="password">Password</label>
  <input
    onChange={(e) =>
      handleChange(e, schema, data, setData, errors, setErrors)
    }
    id="password"
    type="text"
    className="form-control mt-2"
  />
```

# Extracting Reusable Form Rendering Methods

```
components > helpers > form.jsx > ...
form.jsx

export const renderInput = (
  label,
  id,
  type,
  schema,
  data,
  setData,
  errors,
  setErrors
) => {
  return (
    <div className="form-group mt-3">
      <label htmlFor={id}>{label}</label>
      <input
        autoFocus
        onChange={(e) =>
          handleChange(e, schema, data, setData, errors, setErrors)
        }
        id={id}
        type={type}
        className="form-control mt-2"
      />
      {errors[id] && <div className="alert alert-danger">{errors[id]}</div>}
    </div>
  );
};
```

# Extracting Reusable Form Rendering Methods

form.jsx

```
export const renderButton = (data, schema) => {
  return (
    <div className="mt-3">
      <button disabled={validate(data, schema)} className="btn btn-primary">
        Login
      </button>
    </div>
  );
}
```

In 83, Col 1 Sr

# Extracting Reusable Form Rendering Methods

- In `loginForm.jsx`

```
    className="container w-50 m-auto">
      >
        {renderInput(
          "Username",
          "username",
          "text",
          schema,
          data,
          setData,
          errors,
          setErrors
        )}
        {renderInput(
          "Password",
          "password",
          "password",
          schema,
          data,
          setData,
          errors,
          setErrors
        )}
        {renderButton(data, schema)}
      </form>
    </div>
```

(“Login”,`data,schema`)

# Register Form

- In `Navbar.jsx`

```
        </NavLink>
        <NavLink className="nav-item nav-link" to="/register">
          Register
        </NavLink>
      /div>
    
```

- Import and create route in `App.js`

```
import MovieForm from "./components/movieForm";
import LoginForm from "./components/loginForm";
import RegisterForm from "./components/registerForm";
import NotFound from "./components/notFound";
```

```
<Route path="rentals" element={<Rentals />} />
<Route path="login" element={<LoginForm />} />
<Route path="register" element={<RegisterForm />} />
<Route path="*" element={<NotFound />} />
```

# Register Form

- Create `registerForm.jsx` in component folder

```
src > components > ✨ registerForm.jsx > 📁 RegisterForm
  1 import { useState } from "react";
  2 import Joi from "joi-browser";
  3 import { handleSubmit, renderInput, renderButton } from "./helpers/form";
  4
  5 function RegisterForm() {
  6   const [data, setData] = useState({ username: "", password: "", name: "" });
  7   const [errors, setErrors] = useState({});
  8
  9   const schema = {
 10     username: Joi.string().required().email().label("Username"),
 11     password: Joi.string().required().min(5).label("Password"),
 12     name: Joi.string().required().label("Name"),
 13   };
 14
 15   const doSubmit = () => {
 16     // call the server
 17     console.log("Submitted");
 18   };
}
```

# Register Form

- Create `registerForm.jsx` in component folder

```
return (
  <div className="container">
    <h1>Register</h1>
    <form
      onSubmit={(e) =>
        handleSubmit(e, data, schema, errors, setErrors, doSubmit)
      }
      className="container w-50 m-auto"
    >
      {renderInput(
        "Username",
        "username",
        "text",
        schema,
        data,
        setData,
        errors,
        setErrors
      )}
      {renderInput(
        "Password",
        "password",
        "password",
        schema,
        data,
        setData,
        errors,
        setErrors
      )}
    
```

```
)}
{renderInput(
  "Password",
  "password",
  "password",
  schema,
  data,
  setData,
  errors,
  setErrors
)}
{renderInput(
  "Name",
  "name",
  "text",
  schema,
  data,
  setData,
  errors,
  setErrors
)}
{renderInput(
  "Name",
  "name",
  "text",
  schema,
  data,
  setData,
  errors,
  setErrors
)}
{renderButton(data, schema)}
</form>
</div>
);

export default RegisterForm;
```

# Register Form

Vidly   Movies   Customers   Rentals   Login   Register

## Register

Username

"Username" is not allowed to be empty

Password

"Password" is not allowed to be empty

Name

"Name" is not allowed to be empty

Login

# Movie Form

- Update App.jsx

```
<Route path="/" element={<Movies />} />
<Route path="movies/:id" element={<MovieForm />} />
<Route path="customers" element={<Customers />} />
<Route path="rentals" element={<Rentals />} />
<Route path="login" element={<LoginForm />} />
<Route path="register" element={<RegisterForm />} />
```

- Update Movies.jsx

```
<div className="col">
  <Link to="movies/new" className="btn btn-primary" style={{ marginBottom: 20 }}>
    New Movie
  </Link>
  <p>Showing {filtered.length} Movies in the Database</p>
  <MoviesTable>
```

- Update MoviesTable.jsx

```
<tr key={movie._id}>
  <td><Link to={`/movies/${movie._id}`}>{movie.title}</Link></td>
  <td>{movie.genre.name}</td>
  <td>{movie.numberInStock}</td>
```

# Movie Form

- Update fakeMovieService.jsx

```
export function saveMovie(movie) {
  let movieInDb = movies.find(m => m._id === movie._id) || {};
  movieInDb.name = movie.title;
  movieInDb.genre = genresAPI.genres.find(g => g._id === movie.genreId);
  movieInDb.numberInStock = movie.numberInStock;
  movieInDb.dailyRentalRate = movie.dailyRentalRate;

  if (!movieInDb.id) {
    movieInDb._id = Date.now().toString();
    movies.push(movieInDb);
  }

  return movieInDb;
}
```

# Movie Form

- Update `form.jsx`

```
export const renderSelect = (
  label,
  id,
  options,
  schema,
  data,
  setData,
  errors,
  setErrors
) => {
  return (
    <div className="form-group mt-3">
      <label htmlFor={id}>{label}</label>
      <select
        onChange={(e) =>
          handleChange(e, schema, data, setData, errors, setErrors)
        }
        value={data.genreId}
        id={id}
        className="form-control mt-2"
      >
        <option value="" />
        {options.map((option) => (
          <option key={option._id} value={option._id}>
            {option.name}
          </option>
        ))}
      </select>
      {errors[id] && <div className="alert alert-danger">{errors[id]}</div>}
    </div>
  );
};
```

# Movie Form

- Update `form.jsx`

```
export const renderButton = (buttonText, data, schema) => {
  return (
    <div className="mt-3">
      <button disabled={validate(data, schema)} className="btn btn-primary">
        {buttonText}
      </button>
    </div>
  );
};
```

# Movie Form

- Update MovieForm.jsx

```
components > 🌐 movieForm.jsx > [🔗] MovieForm
  import { useNavigate, useParams } from "react-router-dom";
  import { useEffect, useState } from "react";
  import Joi from "joi-browser";

  import { getGenres } from "../../services/fakeGenreService";
  import { getMovie, saveMovie } from "../../services/fakeMovieService";
  import {
    handleSubmit,
    renderInput,
    renderSelect,
    renderButton,
  } from "./helpers/form";

  const MovieForm = () => {
    const { id } = useParams();
    const navigate = useNavigate();
```

# Movie Form

- Update MovieForm.jsx

```
const MovieForm = () => {
  const { id } = useParams();
  const navigate = useNavigate();

  const [data, setData] = useState({
    title: "",
    genreId: "",
    numberInStock: "",
    dailyRentalRate: "",
  });
  const [genres, setGenres] = useState([]);
  const [errors, setErrors] = useState({});

  const schema = {
    _id: Joi.string(),
    title: Joi.string().required().label("Title"),
    genreId: Joi.string().required().label("Genre"),
    numberInStock: Joi.number()
  }
```

# Movie Form

- Update MovieForm.jsx

```
const schema = {
  _id: Joi.string(),
  title: Joi.string().required().label("Title"),
  genreId: Joi.string().required().label("Genre"),
  numberInStock: Joi.number()
    .required()
    .min(0)
    .max(10)
    .label("Number in Stock"),
  dailyRentalRate: Joi.number()
    .required()
    .min(0)
    .max(10)
    .label("Daily Rental Rate"),
};

useEffect(() => [
  setGenres(getGenres()),
  if (id === "new") return;
]
```

# Movie Form

- Update MovieForm.jsx

```
useEffect(() => {
  setGenres(getGenres());
  if (id === "new") return;

  const movie = getMovie(id);
  if (!movie) return navigate("/not-found", { replace: true });

  setData(mapToViewModel(movie));
}, [id, navigate]);

const mapToViewModel = (movie) => {
  return {
    _id: movie._id,
    title: movie.title,
    genreId: movie.genre._id,
    numberInStock: movie.numberInStock,
    dailyRentalRate: movie.dailyRentalRate,
  };
};

const doSubmit = () => {
  console.log(data)
  // send data to server
};
```

# Movie Form

- Update MovieForm.jsx

```
const doSubmit = () => {
  console.log(data)
  saveMovie(data);
  navigate("/");
}

return (
  <div className="container">
    <h1>Movie Form</h1>
    <form
      onSubmit={(e) =>
        handleSubmit(e, data, schema, errors, setErrors, doSubmit)
      }
      className="container w-50 m-auto"
    >
      {renderInput(
        "Title",
        "title",
        "text",
        schema,
        data,
        setData,
        errors,
        setErrors
      )}
    
```

# Movie Form

- Update MovieForm.jsx

```
{renderInput(  
  "Title",  
  "title",  
  "text",  
  schema,  
  data,  
  setData,  
  errors,  
  setErrors  
)  
{renderSelect(  
  "Geners",  
  "genreId",  
  genres,  
  schema,  
  data,  
  setData,  
  errors,  
  setErrors  
)  
}  
  
<div>  
  {renderInput(  
    "Number in Stock",  
    "numberInStock",  
    "number",  
    schema,  
    data,  
    setData,  
    errors,  
    setErrors  
)  
  }  
  {renderInput(  
    "Rate",  
    "dailyRentalRate",  
    "number",  
    schema,  
    data,  
    setData,  
    errors,  
    setErrors  
)  
  }  
  {renderButton("Save", data, schema)}  
</form>  
</div>  
  
export default MovieForm;
```

# Search Movies

movies.jsx

- In Movies component add and update

```
const [selectedGenre, setSelectedGenre] = useState(allGenres);
const [pageSize] = useState(4);
const [searchQuery, setSearchQuery] = useState("");
const [currentPage, setCurrentPage] = useState(1);
const [sortColumn, setSortColumn] = useState({ path: "title", order: "asc" });
```

```
let filtered = movies;
if (searchQuery) {
  filtered = movies.filter((m) =>
    m.title.toLowerCase().startsWith(searchQuery.toLowerCase())
  );
} else if (selectedGenre && selectedGenre._id) {
  filtered = movies.filter((m) => m.genre._id === selectedGenre._id);
}
```

```
// const filtered =
//   selectedGenre && selectedGenre._id
//     ? movies.filter((m) => m.genre._id === selectedGenre._id)
//     : movies;
```

OLD

# Search Movies

movies.jsx

- In Movies component add and update

```
const handleGenreSelect = (genre) => {
  setSelectedGenre(genre);
  setSearchQuery("");
  setCurrentPage(1);
};
```

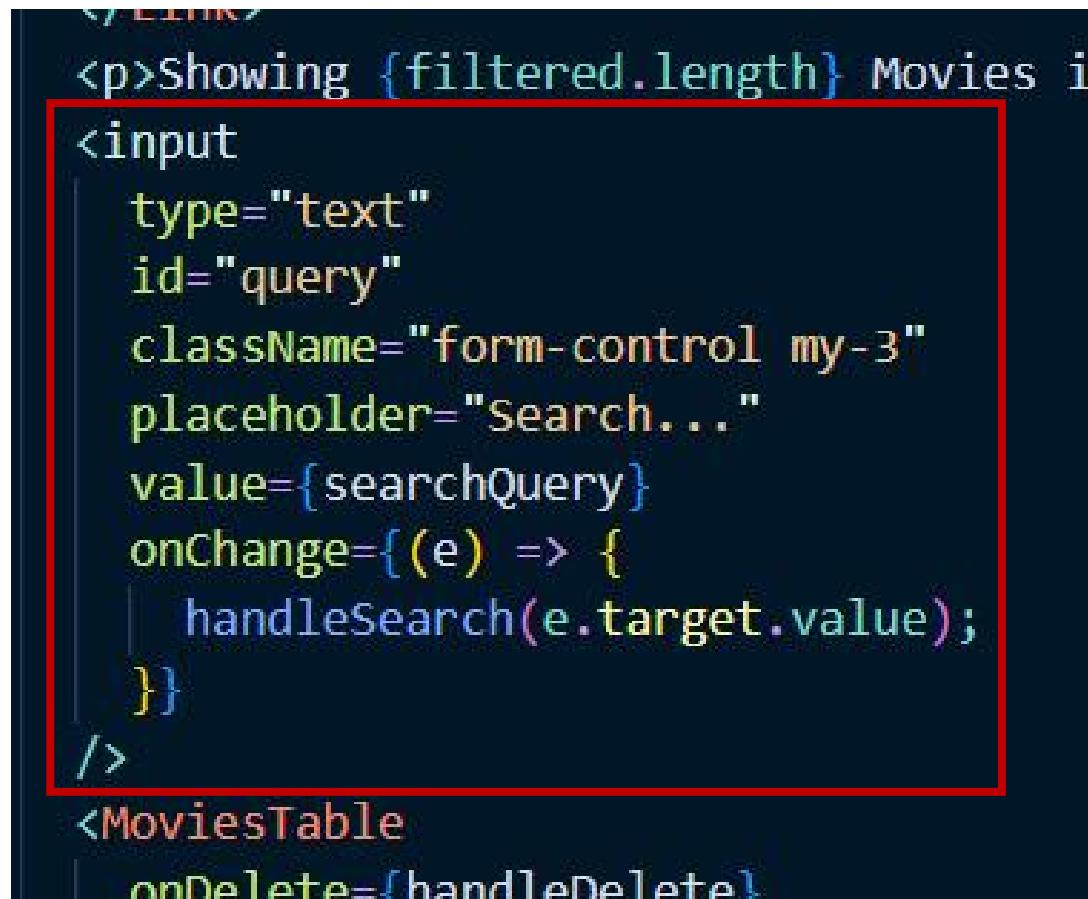
```
const handleSearch = (query) => {
  setSearchQuery(query);
  setSelectedGenre(allGenres);
  setCurrentPage(1);
};
```

```
const handlePageChange = (page) => {
```

# Search Movies

movies.jsx

- In Movies component add and update



```
Showing {filtered.length} Movies in results
```

```
<input
  type="text"
  id="query"
  className="form-control my-3"
  placeholder="Search..."
  value={searchQuery}
  onChange={(e) => {
    handleSearch(e.target.value);
  }}
/>
<MoviesTable
  onDelete={handleDelete}>
```

# Search Movies

- Output

Vidly   Movies   Customers   Rentals   Login   Register

All Genres

Action

Comedy

Thriller

New Movie

Showing 2 Movies in the Database

Title	Genre	Stock	Rate	Heart	Delete
<a href="#">Get Out</a>	Thriller	8	3.5	♥	<a href="#">Delete</a>
<a href="#">Gone Girl</a>	Thriller	7	4.5	♥	<a href="#">Delete</a>



# useEffect

# useEffect

When the core React Hooks were added to the library in 2018 (useState, useEffect, and so on), many developers were confused by the name of this hook: "useEffect".

What exactly is an "effect"?

The word effect refers to a functional programming term called a "side effect".

But to really understand what a side effect is, we first have to grasp the concept of a pure function.

# Pure Function:

- Predictable
- Has no side-effects

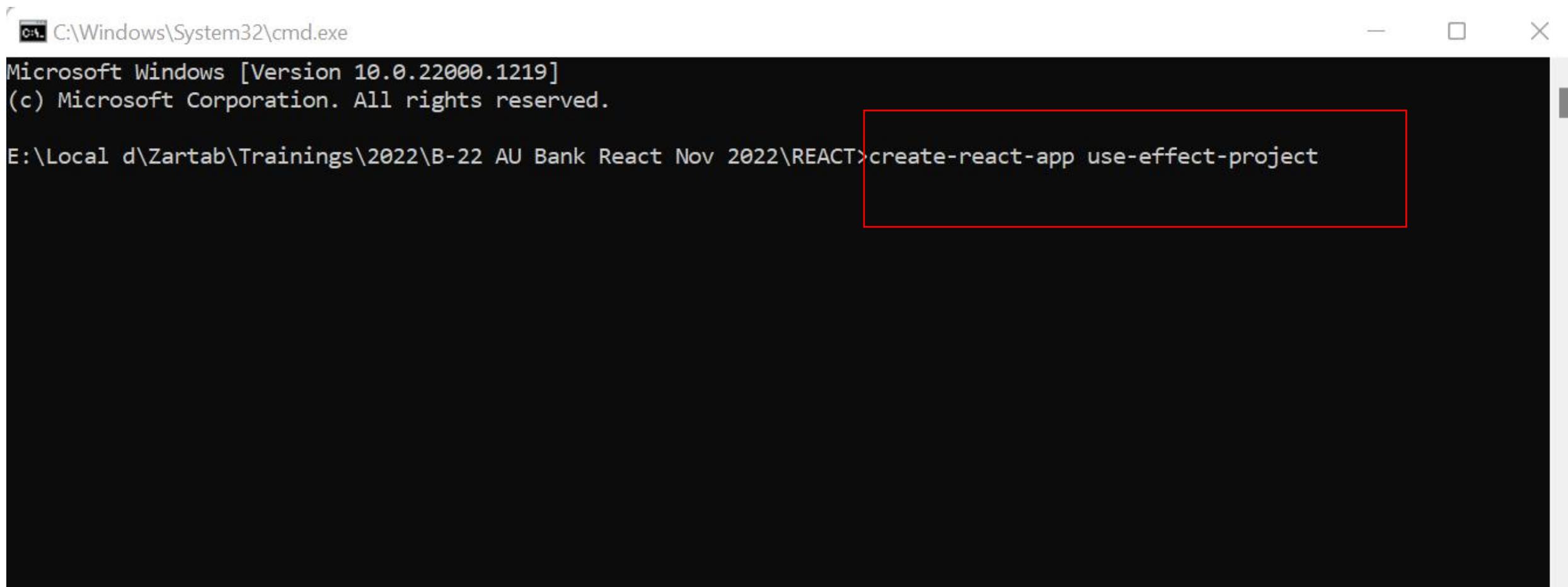
```
function add(a,b){  
    return a+b;  
}  
  
console.log(add(2,2));
```

## Impure Function:

- Unpredictable
- Has side-effects

```
function add(a,b){  
    const randomNumber=Math.random()+10;  
    return a+b+randomNumber;  
}  
  
console.log(add(2,2));
```

# Create a new react project

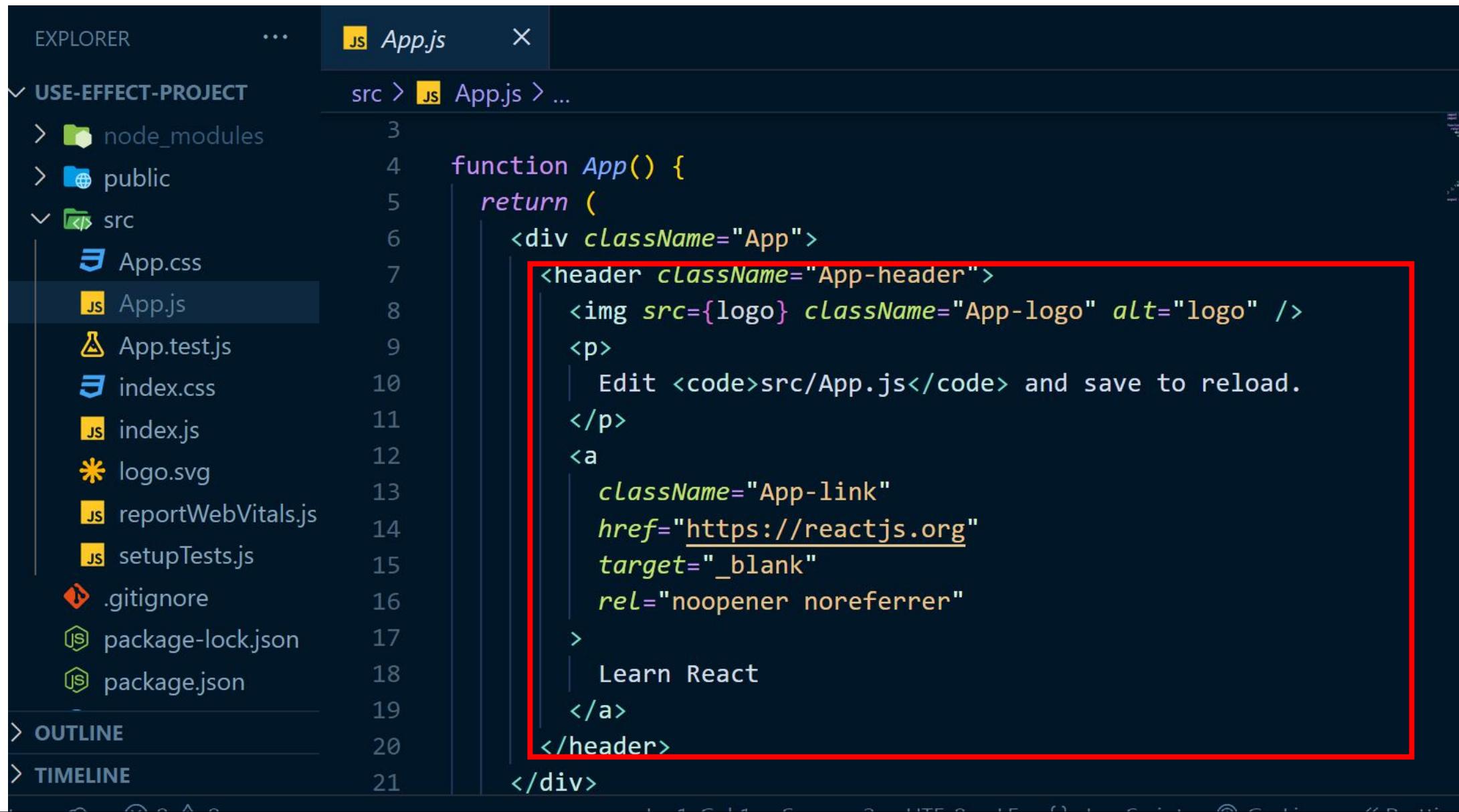


C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.22000.1219]  
(c) Microsoft Corporation. All rights reserved.

E:\Local d\Zartab\Trainings\2022\B-22 AU Bank React Nov 2022\REACT>create-react-app use-effect-project

# Clear unwanted code from App.js



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

- EXPLORER** sidebar: Shows the project structure under "USE-EFFECT-PROJECT". Files listed include node\_modules, public, and src. Inside src, there are App.css, App.js (selected), App.test.js, index.css, index.js, logo.svg, reportWebVitals.js, setupTests.js, .gitignore, package-lock.json, and package.json.
- Editor Tab Bar**: Shows "App.js" is the active file.
- Editor Path Bar**: Shows the current file path: src > App.js > ...
- Editor Content**: The code for the App component. A red box highlights the header section:

```
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
```
- Bottom Status Bar**: Shows various icons for file operations like Save, Undo, Redo, and Find.



EXPLORER

...



App.js M



User.jsx U X



## USE-EFFECT-PROJECT

&gt; node\_modules

&gt; public

&gt; src

App.css

App.js M

App.test.js

index.css

index.js

logo.svg

reportWebVitals.js

setupTests.js

User.jsx U

.gitignore

package-lock.json



&gt; OUTLINE

src &gt; User.jsx &gt; ...

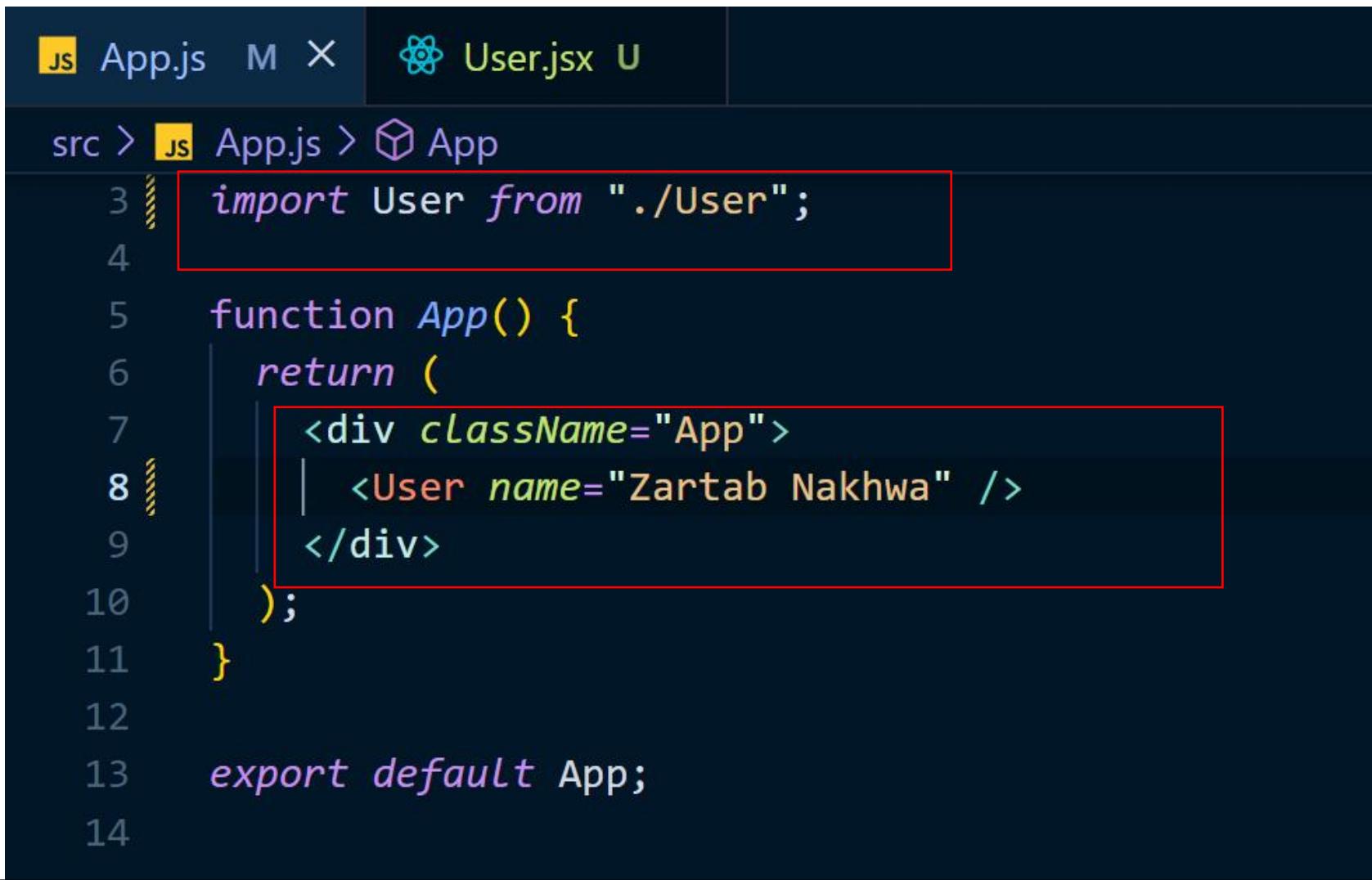
```
1 import React from "react";
2
3 export default function User() {
4   return <div>User</div>;
5 }
6
```

Add a new component  
by name of User.jsx

# Make the changes to `Users.jsx`

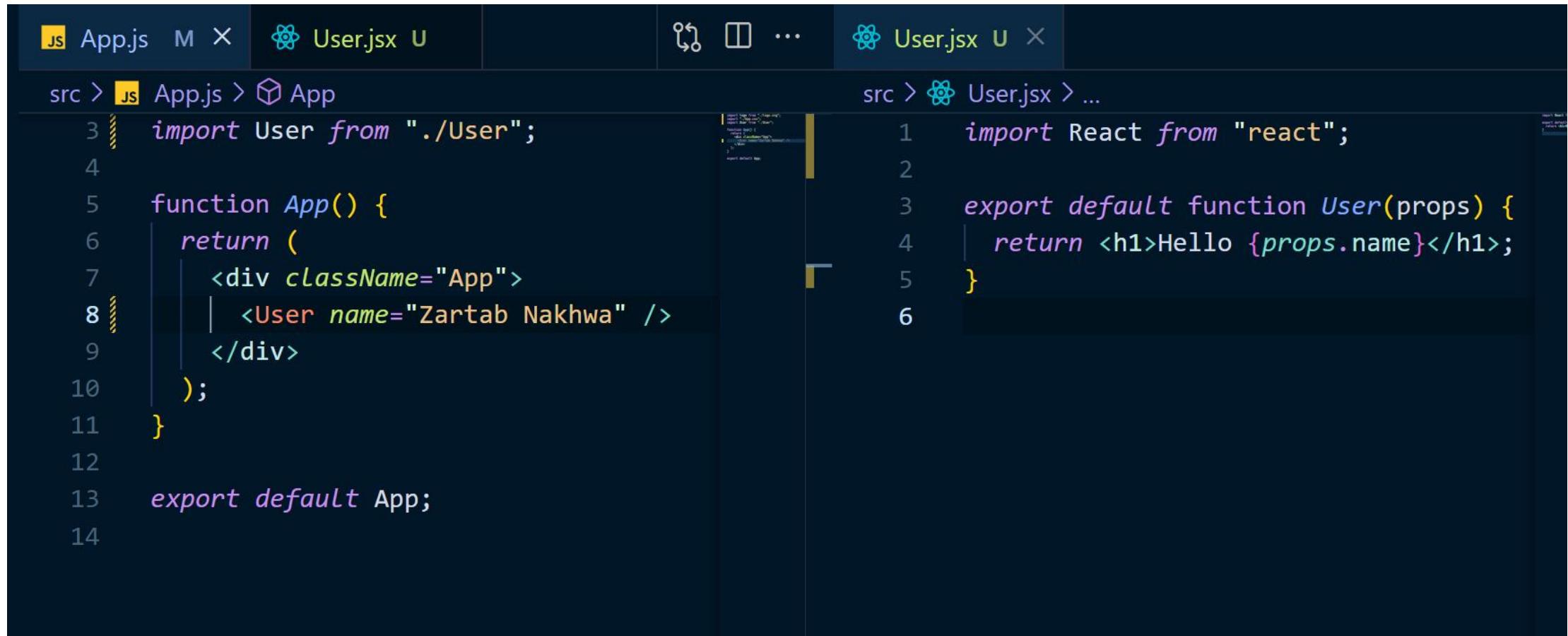
```
src >  User.jsx > ...
1 import React from "react";
2
3 export default function User(props) {
4   return <h1>Hello {props.name}</h1>;
5 }
6
```

# Make the changes in App.js



```
src > JS App.js > ⚙️ App
3 | import User from "./User";
4 |
5 function App() {
6     return (
7         <div className="App">
8             <User name="Zartab Nakhwa" />
9         </div>
10    );
11 }
12
13 export default App;
14
```

Here we have a `User` component that has the prop `name` declared on it. Within `User`, the prop value is displayed in a header element.



The image shows a code editor interface with two tabs: `App.js` and `User.jsx`. The `App.js` file contains the following code:

```
src > JS App.js > ⚡ App
1 import User from "./User";
2
3 function App() {
4   return (
5     <div className="App">
6       <User name="Zartab Nakhwa" />
7     </div>
8   );
9 }
10
11
12
13 export default App;
14
```

The `User.jsx` file contains the following code:

```
src > ⚡ User.jsx > ...
1 import React from "react";
2
3 export default function User(props) {
4   return <h1>Hello {props.name}</h1>;
5 }
6
```

This is pure because, given the same input, it will always return the same output.

If we pass User a name prop with value "Zartab Nakhwa", our output will always be Zartab Nakhwa

You might be saying, "Who cares? Why do we even have a name for this?"

Pure functions have the great benefit of being predictable, reliable, and easy to test.

This is as compared to when we need to perform a side effect in our component.

# What are side effects in React?

- Side effects are not predictable because they are actions which are performed with the "outside world."
- We perform a side effect when we need to reach outside of our React components to do something. Performing a side effect, however, will not give us a predictable result.
- Think about if we were to request data (like blog posts) from a server that has failed and instead of our post data, gives us a 500 status code response.

Virtually all applications rely on side effects to work in one way or another, aside from the simplest applications.

Common side effects include:

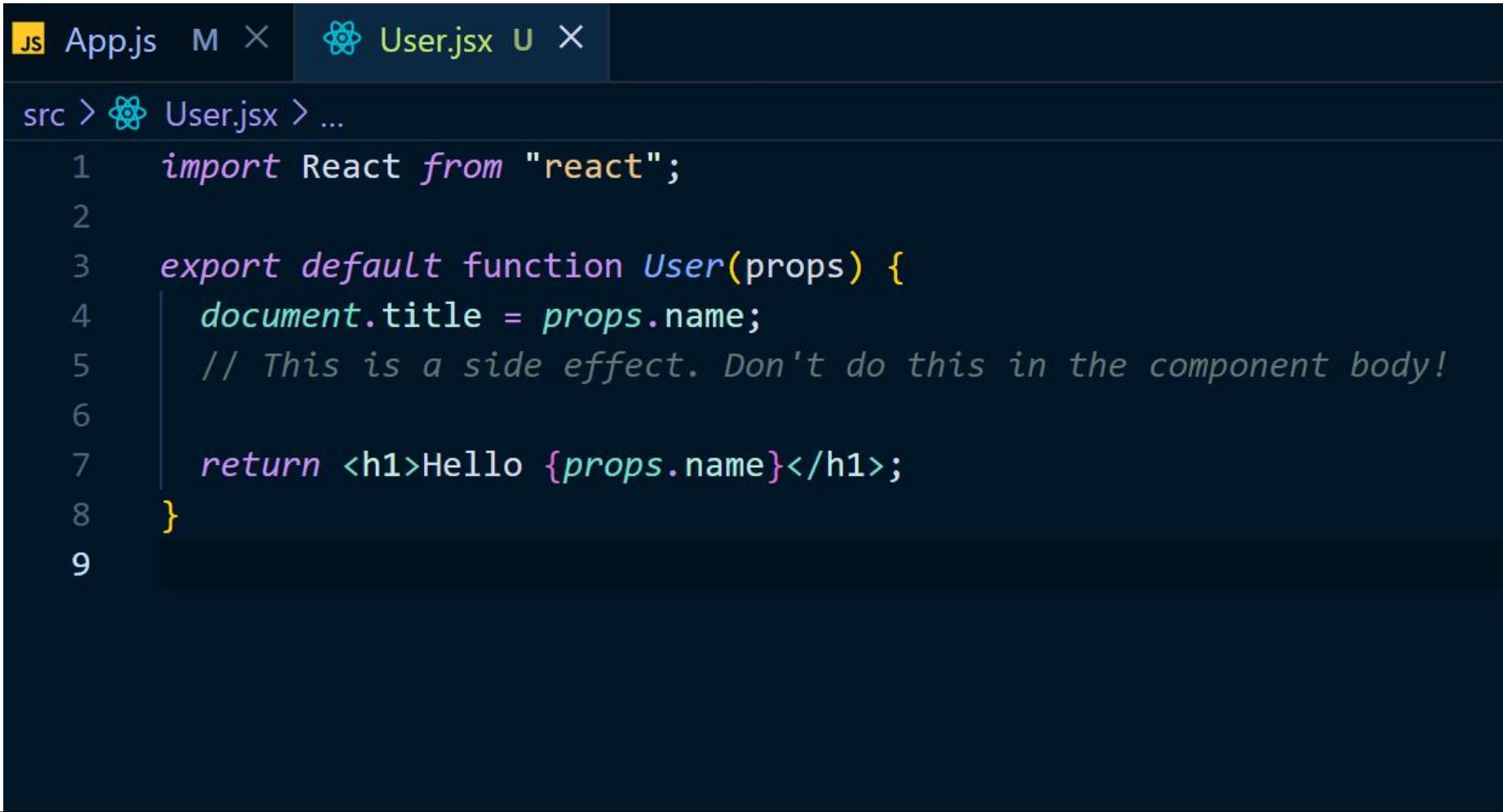
Making a request to an API for data from a backend server

To interact with browser APIs (that is, to use document or window directly)

Using unpredictable timing functions like setTimeout or setInterval

This is why `useEffect` exists: to provide a way to handle performing these side effects in what are otherwise pure React components.

For example, if we wanted to change the title meta tag to display the user's name in their browser tab, we could do it within the component itself, but we shouldn't.



```
JS App.js M X ⚡ User.jsx U X
src > ⚡ User.jsx > ...
1 import React from "react";
2
3 export default function User(props) {
4   document.title = props.name;
5   // This is a side effect. Don't do this in the component body!
6
7   return <h1>Hello {props.name}</h1>;
8 }
9
```

If we perform a side effect directly in our component body, it gets in the way of our React component's rendering.

Side effects should be separated from the rendering process. If we need to perform a side effect, it should strictly be done after our component renders.

**This is what `useEffect` gives us.**

In short, `useEffect` is a tool that lets us interact with the outside world but not affect the rendering or performance of the component that it's in.

# The basic syntax of useEffect is as follows:

```
src > ⚛ MyComponent.jsx > ...
1  // 1. import useEffect
2  import { useEffect } from "react";
3
4  function MyComponent() {
5      // 2. call it above the returned JSX
6      // 3. pass two arguments to it: a function and an array
7      useEffect(() => {}, []);
8
9      // return ...
10 }
11
```

The correct way to perform the side effect in our User component is as follows:

1. We import useEffect from "react"
2. We call it above the returned JSX in our component
3. We pass it two arguments: a function and an array

# Make the changes to User.jsx

```
src > ⚛ User.jsx > ...
1 import React, { useEffect } from "react";
2
3 export default function User({ name }) {
4   useEffect(() => {
5     document.title = name;
6   }, [name]);
7
8   return <h1>Hello {name}</h1>;
9 }
10
```



Zartab Nakhwa



The function passed to `useEffect` is a callback function. This will be called after the component renders.

In this function, we can perform our side effects or multiple side effects if we want.

The second argument is an array, called the dependencies array. This array should include all of the values that our side effect relies upon.

In our example above, since we are changing the title based off of a value in the outer scope, name, we need to include that within the dependencies array.

What this array will do is it will check and see if a value (in this case name) has changed between renders. If so, it will execute our use effect function again.

This makes sense because if the name changes, we want to display that changed name and therefore run our side effect again.

Common mistakes  
we do while working  
with useEffect

Create a new file `colors.jsx` and write following function in it

The screenshot shows a code editor interface with a sidebar on the left displaying a project structure. The project structure includes a `public` folder containing `favicon.ico`, `index.html`, `logo192.png`, `logo512.png`, `manifest.json`, and `robots.txt`. Below this is a `src` folder containing `App.css`, `App.js`, `App.test.js`, and `colors.jsx`. The `colors.jsx` file is highlighted with a red border and is open in the main editor area. The code in `colors.jsx` is:

```
1 export function getColors() {  
2     const p = new Promise((resolve, reject) => {  
3         setTimeout(() => {  
4             const colors = ["Green", "Red", "Blue", "Yellow"];  
5             resolve(colors);  
6         }, 3000);  
7     });  
8     return p;  
9 }  
10
```

# Write a new component MyComponent.jsx

USE-EFFECT-PROJECT

> node\_modules

> public

> src

App.css

App.js M

App.test.js

colors.jsx U

index.css

index.js

logo.svg

MyCompo... U

reportWebVitals.js

setupTests.js

Timer.jsx U

User.jsx U

.gitignore

package-lock.json

src > MyComponent.jsx > MyComponent

```
1 import { useEffect, useState } from "react";
2 import { getColors } from "./colors";
3
4 export function MyComponent() {
5   const [data, setData] = useState([]);
6
7   useEffect(() => {
8     getColors().then(myData) => setData(myData));
9     console.log("I am in Use Effect !!");
10    }, []);
11
12  return (
13    <ul>
14      {data.length > 0 ? (
15        data.map(item) => <li key={item}>{item}</li>)
16      ) : (
17        <h3>Loading....</h3>
18      )}
19    </ul>
20  );
}
```

# Make the changes in App.js

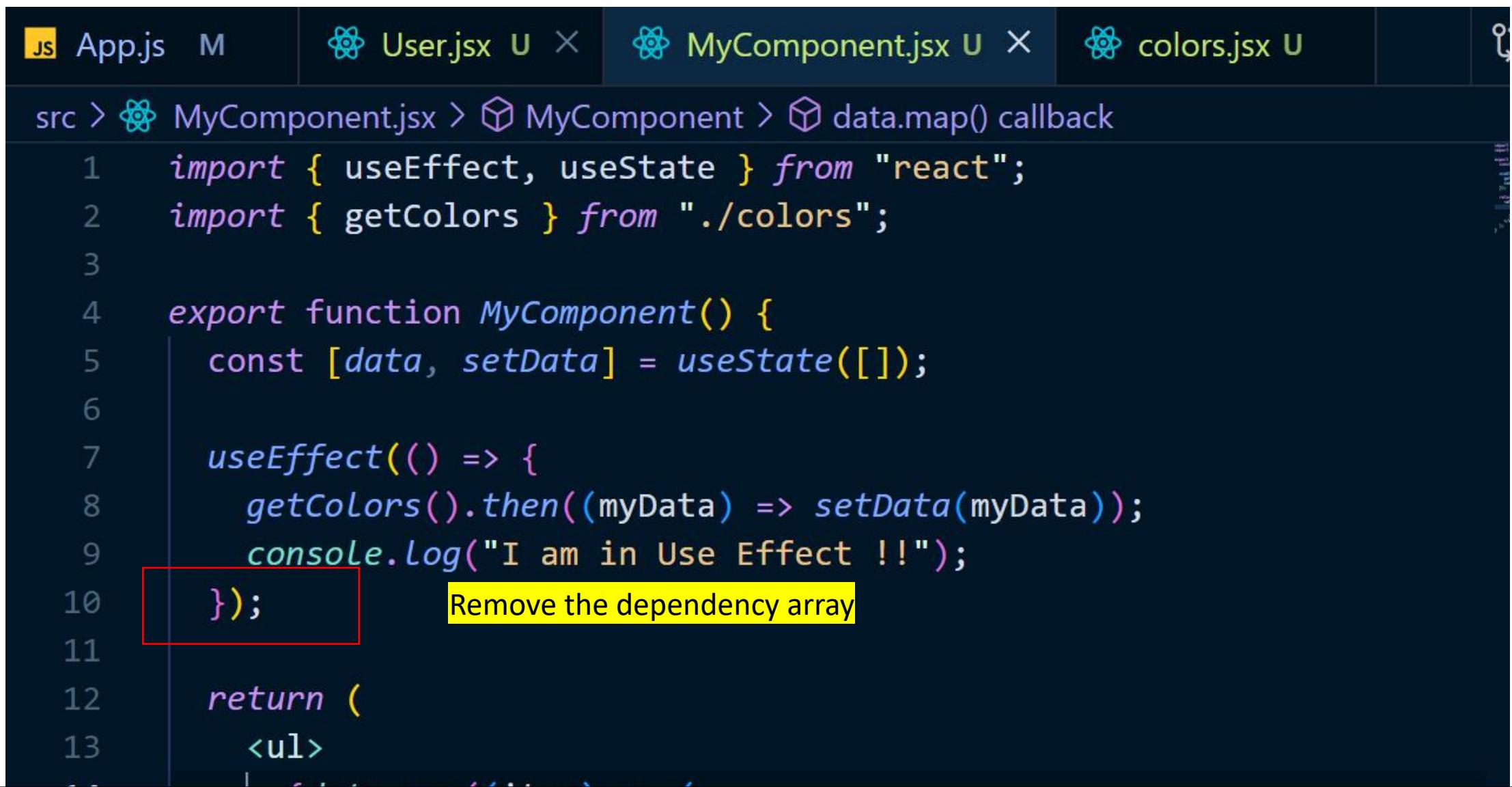
```
src > JS App.js > ⚒ App
  3 | import User from "./User";
  4 | import { MyComponent } from "./MyComponent";
  5 |
  6 function App() {
  7   return (
  8     <div className="App">
  9       <User name="Zartab Nakhwa" />
 10      <br />
 11      <MyComponent />
 12    </div>
 13  );
 14}
 15
 16 export default App;
 17
```

# Hello Zartab Nakhwa

•  
•  
•  
•

Green  
Red  
Blue  
Yellow

# Make one change in MyComponent.jsx



The screenshot shows a code editor with four tabs at the top: App.js, User.jsx, MyComponent.jsx (selected), and colors.jsx. The MyComponent.jsx tab has a yellow background. Below the tabs, the file structure is shown: src > MyComponent.jsx > MyComponent > data.map() callback. The code editor displays the following code:

```
src > MyComponent.jsx > MyComponent > data.map() callback
1 import { useEffect, useState } from "react";
2 import { getColors } from "./colors";
3
4 export function MyComponent() {
5   const [data, setData] = useState([]);
6
7   useEffect(() => {
8     getColors().then((myData) => setData(myData));
9     console.log("I am in Use Effect !!");
10  });
11
12  return (
13    <ul>
14      | <li>{data}</li>
15    </ul>
16  );
17}
```

A red box highlights the closing brace of the useEffect hook on line 10. A yellow box surrounds the entire code block, containing the text "Remove the dependency array".

# Use Effect is running in an infinite loop

The screenshot shows a browser window with a title bar and a developer tools console open. The main content area displays the text "Hello Zartab Nakhwa" and a list of colors: Green, Red, Blue, Yellow. Below this list is a bullet point section that is partially visible. The developer tools console tab is selected, showing the following log entries:

- root:1 Received from "root:1" in "root" record-api.js:169
- Received locators in 'root:1' ► Array(1) record-api.js:66
- FrameLocation updated to ► Object record-api.js:71
- 16 I am in Use Effect !!** MyComponent.jsx:9
- 16 I am in Use Effect !!** MyComponent.jsx:9
- ⚠ DevTools failed to load source map: Could not load content for chrome-extension://mbopgmdnpscbohhpnfglgohlbhfongabi/dist/browser-polyfill.min.js.map: Fetch through target failed: Frame not found; Fallback: HTTP error: status code 404, net::ERR\_UNKNOWN\_URL\_SCHEME

Two specific log entries are highlighted with red boxes: the first entry at line 16 and the second entry at line 2. Both of these entries correspond to the "I am in Use Effect !! " message in the list above, indicating that the effect is running repeatedly.

If you do not provide the dependencies array at all and only provide a function to useEffect, it will run after every render.

This can lead to problems when you're attempting to update state within your useEffect hook.

If you forget to provide your dependencies correctly and you are setting a piece of local state when the state is updated, the default behavior of React is to re-render the component. And therefore, since useEffect runs after every single render without the dependencies array, we will have an infinite loop.

If you are updating state within your `useEffect`, make sure to provide an empty dependencies array.

If you do provide an empty array, which I recommend you do by default for any time that you use `useEffect`, this will cause the effect function to only run once after the component has rendered the first time.

# Add the empty dependency array

The screenshot shows a code editor with several tabs at the top: App.js (selected), User.jsx, MyComponent.jsx (active), and colors.jsx. Below the tabs, a breadcrumb navigation bar indicates the file structure: src > MyComponent.jsx > MyComponent > data.map() callback. The main content area displays the code for MyComponent.jsx. A red box highlights the dependency array [ ] in the useEffect hook at line 10.

```
1 import { useEffect, useState } from "react";
2 import { getColors } from "./colors";
3
4 export function MyComponent() {
5     const [data, setData] = useState([]);
6
7     useEffect(() => {
8         getColors().then(myData) => setData(myData);
9         console.log("I am in Use Effect !!");
10    }, []);
11
12    return (
13        <ul>
14            {data.map(item) => (
15                <li key={item}>{item}</li>
16            ))}
17        </ul>
18    );
19}
```

# Hello Zartab Nakhwa

- Green
- Red
- Blue
- Yellow

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output displays several messages:

- Two messages from 'MyComponent.jsx': "I am in Use Effect !!". These are highlighted with a red box.
- A warning message: "⚠ DevTools failed to load source map: Could not load content for chrome-extension://mbopgmdnpscbohhpnfg1gohlbfongabi/dist/browser-polyfill.min.js.map: Fetch through target failed: Frame not found; Fallback: HTTP error: status code 404, net::ERR\_UNKNOWN\_URL\_SCHEME".
- A warning message: "⚠ DevTools failed to load source map: Could not load content for chrome-extension://mbopgmdnpscbohhpnfg1gohlbfongabi/dist/browser-polyfill.min.js.map: System error: net::ERR\_BLOCKED\_BY\_CLIENT".
- A message: "root:0" followed by the file path "record-api.js:169".
- A warning message: "⚠ DevTools failed to load source map: Could not load content for chrome-extension://mbopgmdnpscbohhpnfg1gohlbfongabi/dist/browser-polyfill.min.js.map: System error: net::ERR\_BLOCKED\_BY\_CLIENT".
- A message: "root" followed by the file path "record-api.js:169".
- A warning message: "⚠ DevTools failed to load source map: Could not load content for chrome-extension://mbopgmdnpscbohhpnfg1gohlbfongabi/dist/browser-polyfill.min.js.map: System error: net::ERR\_BLOCKED\_BY\_CLIENT".
- A message: "CookieManager" followed by the file path "commons.js:2".
- A message: "▶ {hasCookieApiAvailable: false, cookie: f}"
- A message: "Received from "root:0" in "root"" followed by the file path "record-api.js:30".
- A message: "Received locators in 'root:0' ▶ [ 'css=iframe' ]" followed by the file path "record-api.js:66".
- A message: "FrameLocation updated to" followed by the file path "record-api.js:71".
- A message: "▶ {location: 'root:0', locators: Array(1)}"
- A final message: ">"

# What is the cleanup function in useEffect?

The final part of performing side effects properly in React is the effect cleanup function.

Sometimes our side effects need to be shut off. For example, if you have a countdown timer using the setInterval function, that interval will not stop unless we use the clearInterval function.

Another example is to use subscriptions with WebSockets. Subscriptions need to be "turned off" when we are no longer using them, and this is what the cleanup function is for.

If we are setting state using `setInterval` and that side effect is not cleaned up, when our component unmounts and we're no longer using it, the state is destroyed with the component – but the `setInterval` function will keep running.

```
function Timer() {
  const [time, setTime] = useState(0);

  useEffect(() => {
    setInterval(() => setTime(1), 1000);
    // counts up 1 every second
    // we need to stop using setInterval when component unmounts
  }, []);
}
```

The problem with this if the component is destroying, is that `setInterval` will try to update a variable a piece of state time that no longer exists. This is an error called a memory leak.

To use the cleanup function, we need to return a function from within the `useEffect` function.

Within this function we can perform our cleanup, in this case to use `clearInterval` and stop `setInterval`.

 Timer.jsx > ...

```
function Timer() {
  const [time, setTime] = useState(0);

  useEffect(() => {
    let interval = setInterval(() => setTime(1), 1000);

    return () => {
      // clearInterval cleared when component unmounts
      clearInterval(interval);
    };
  }, []);
}
```

The cleanup function will be called when the component is unmounted.

A common example of a component being unmounted is going to a new page or a new route in our application where the component is no longer rendered.

When a component is unmounted, our cleanup function runs, our interval is cleared, and we no longer get an error of attempting to update a state variable that doesn't exist.

Finally, the side effect cleanup is not required in every case. It is only required in a few cases, such as when you need to stop a repeated side effect when your component unmounts.



# Calling Backend Services

**Create a new project**

```
npx create-react-app http-access
```

**Update App.js**

```
> JS App.js > App
1 import { useState } from 'react';
2 import './App.css';
3
4 function App() {
5   const [posts, setPosts] = useState([])
6
7   const handleAdd = async () => {
8     console.log("Added");
9   };
10
11   const handleUpdate = (post) => {
12     console.log("update", post);
13   };
14
15   const handleDelete = (post) => {
16     console.log("Delete", post);
17   };
18   return (
19     <div>
20       <h1>Welcome to React</h1>
21       <p>This is a sample application for calling backend services.</p>
22       <button onClick={handleAdd}>Add</button>
23       <button onClick={handleUpdate}>Update</button>
24       <button onClick={handleDelete}>Delete</button>
25     </div>
26   );
27 }
28
29 export default App;
```

# Calling Backend Services

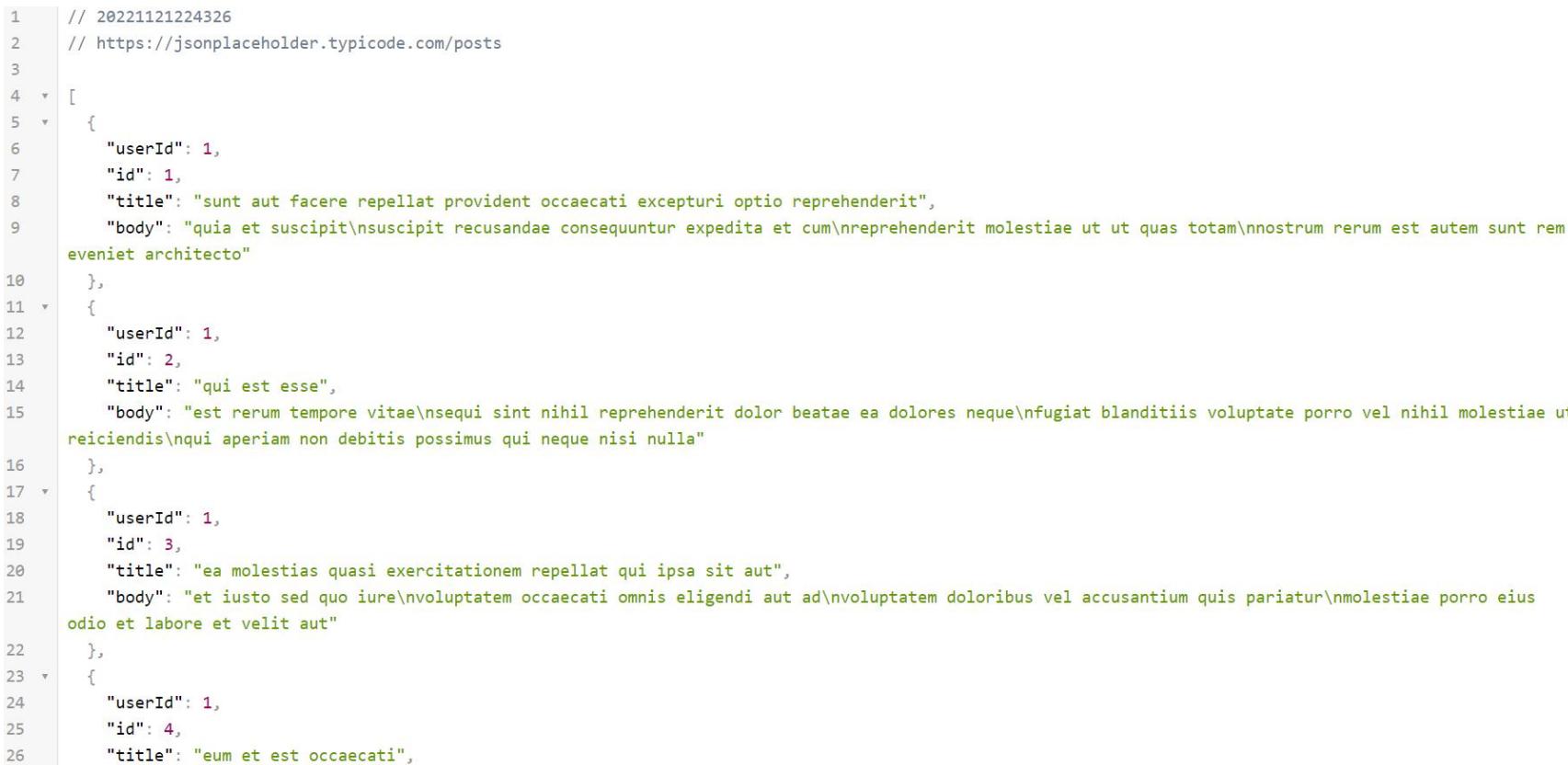
```
js App.js > ⚡ App > ⚡ posts.map() callback
  , return (
    <div className="App">
      <button className="btn btn-primary mt-2" onClick={handleAdd}>
        Add
      </button>
      <table className="table">
        <thead>
          <tr>
            <th>Title</th>
            <th>Update</th>
            <th>Delete</th>
          </tr>
        </thead>
        <tbody>
          {posts.map(post => [
            <tr key={post.id}>
              <td>{post.title}</td>
              <td>
                <button
                  className="btn btn-info btn-sm"
                  onClick={() => handleUpdate(post)}
                >
```

```
> Update
      </button>
    </td>
    <td>
      <button
        className="btn btn-danger btn-sm"
        onClick={() => handleDelete(post)}
      >
        Delete
      </button>
    </td>
  </tr>
)}
</tbody>
</table>
</div>
);

export default App;
```

# Setting Up

- Install Extension JSON Viewer and open  
<https://jsonplaceholder.typicode.com/posts>



```
1 // 20221121224326
2 // https://jsonplaceholder.typicode.com/posts
3
4 [
5   {
6     "userId": 1,
7     "id": 1,
8     "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
9     "body": "quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"
10    },
11   {
12     "userId": 1,
13     "id": 2,
14     "title": "qui est esse",
15     "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla"
16    },
17   {
18     "userId": 1,
19     "id": 3,
20     "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
21     "body": "et iusto sed quo iure\\nvolutatem occaecati omnis eligendi aut ad\\nvolutatem doloribus vel accusantium quis pariatur\\nmolestiae porro eius odio et labore et velit aut"
22    },
23   {
24     "userId": 1,
25     "id": 4,
26     "title": "eum et est occaecati",
```

# Getting Data

- Install Axios -- `npm i axios`
- Update `App.js`

```
App.js > ⚡ App
import { useState, useEffect } from 'react';
import axios from 'axios';
import './App.css';

function App() {
  const [posts, setPosts] = useState([])
  const apiEndpoint = "https://jsonplaceholder.typicode.com/posts"

  async function getResults() {
    const results = await axios(apiEndpoint);
    setPosts(results.data)
  }

  useEffect(() => {
    getResults()
  }, [])
}
```

# Getting Data

<a href="#">Add</a>	<a href="#">Title</a>	<a href="#">Update</a>	<a href="#">Delete</a>
	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	<a href="#">Update</a>	<a href="#">Delete</a>
	qui est esse	<a href="#">Update</a>	<a href="#">Delete</a>
	ea molestias quasi exercitationem repellat qui ipsa sit aut	<a href="#">Update</a>	<a href="#">Delete</a>
	eum et est occaecati	<a href="#">Update</a>	<a href="#">Delete</a>
	nesciunt quas odio	<a href="#">Update</a>	<a href="#">Delete</a>
	dolorem eum magni eos aperiam quia	<a href="#">Update</a>	<a href="#">Delete</a>
	magnam facilis autem	<a href="#">Update</a>	<a href="#">Delete</a>
	dolorem dolore est ipsam	<a href="#">Update</a>	<a href="#">Delete</a>
	nesciunt iure omnis dolorem tempora et accusantium	<a href="#">Update</a>	<a href="#">Delete</a>
	optio molestias id quia eum	<a href="#">Update</a>	<a href="#">Delete</a>
	et ea vero quia laudantium autem	<a href="#">Update</a>	<a href="#">Delete</a>
	in quibusdam tempore odit est dolorem	<a href="#">Update</a>	<a href="#">Delete</a>
	dolorum ut in voluptas mollitia et saepe quo animi	<a href="#">Update</a>	<a href="#">Delete</a>

# Creating Data

```
const handleAdd = async () => {
  const obj = { title: "a title", body: "a body" }
  const results = await axios.post(apiEndpoint, obj);
  setPosts([results.data, ...posts])
};
```

Post List			
Title	Content	Update	Delete
a title	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	<button>Update</button>	<button>Delete</button>
qui est esse		<button>Update</button>	<button>Delete</button>

# Lifecycle of a Request

- OPTIONS : Different Domains
- POST : Created Val

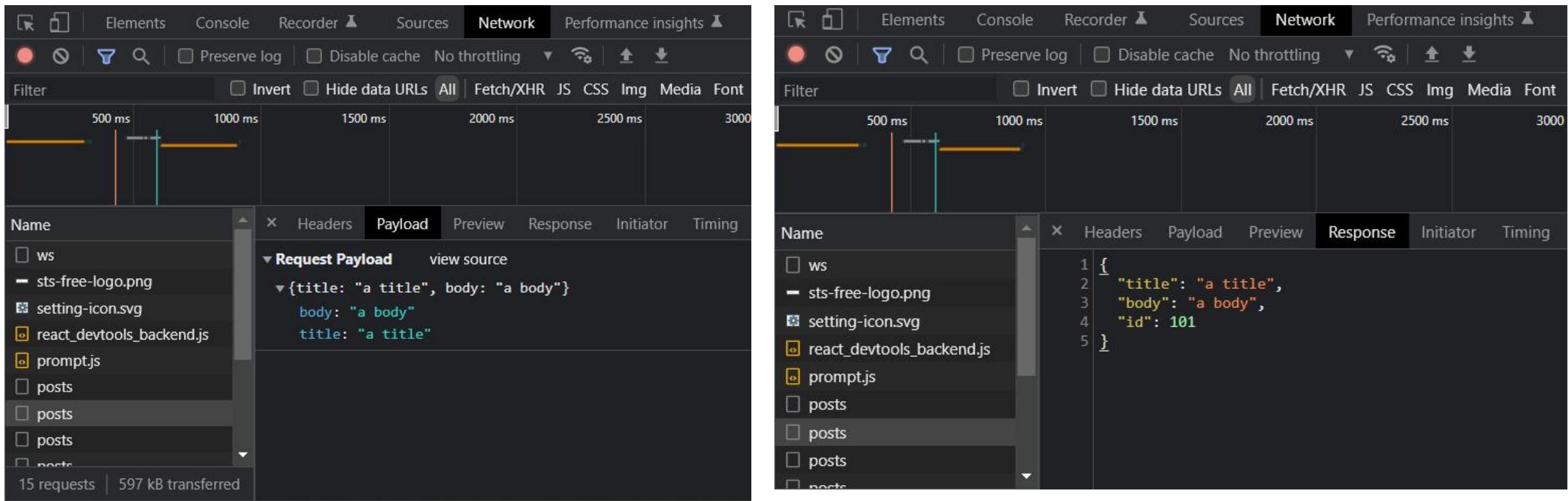
The screenshot shows the Network tab in the Chrome DevTools developer tools. The timeline at the top indicates a request starting at 500 ms and ending at 1000 ms. The list of requests on the left shows several items, with "posts" selected. The main panel displays details for the selected request:

- Name:** posts
- Request URL:** <https://jsonplaceholder.typicode.com/posts>
- Request Method:** OPTIONS (highlighted with a red box)
- Status Code:** 204
- Remote Address:** 172.67.155.76:443
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
  - access-control-allow-credentials: true
  - access-control-allow-headers: content-type

The screenshot shows the Network tab in the Chrome DevTools developer tools. The timeline at the top indicates a request starting at 500 ms and ending at 1000 ms. The list of requests on the left shows several items, with "posts" selected. The main panel displays details for the selected request:

- Name:** posts
- Request URL:** <https://jsonplaceholder.typicode.com/posts>
- Request Method:** POST (highlighted with a red box)
- Status Code:** 201
- Remote Address:** 172.67.155.76:443
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
  - access-control-allow-credentials: true
  - access-control-allow-origin: http://localhost:3000

# Lifecycle of a Request



# Updating Data

```
const handleUpdate = async (post) => {
  post.title = "UPDATED A Title"
  await axios.put(apiEndpoint + "/" + post.id, post);

  const index = posts.indexOf(post)
  posts[index].title = post.title

  setPosts(posts.map(p => {
    return p
  }));
};
```

Add	Title	Update	Delete
	UPDATED A Title	<button>Update</button>	<button>Delete</button>
	qui est esse	<button>Update</button>	<button>Delete</button>
	ea molestias quasi exercitationem repellat qui ipsa sit aut	<button>Update</button>	<button>Delete</button>

# Deleting Data

```
const handleDelete = async (post) => {
  await axios.delete(apiEndpoint + "/" + post.id);
  setPosts(posts.filter((p) => p.id !== post.id));
};
```

# Optimistic vs Pessimistic Updates

- Our Current Approach was Pessimistic, lets make it Optimistic

```
const handleDelete = async (post) => {
  const originalPosts = posts

  setPosts(posts.filter((p) => p.id !== post.id));

  try {
    await axios.delete(apiEndpoint + "/" + post.id);
  } catch (error) {
    alert("Something Failed while deleting a Post!")
    setPosts(originalPosts)
  }
};
```

# Expected vs UnExpected Errors

## Expected errors

- Expected errors are errors you know will happen and can not be prevented. For example, an application is communicating with another application, and somehow, a server is down, or the connection is gone. There are many ways to minimize the chance of this happening, but it will happen.
- Expected errors can be predicted upfront, and therefore can and should be tested. You could, for example, shut down a server, disconnect the WiFi, make deliberate mistakes in config files, etc. and see how the application responds.
- Expected errors can also be outside of your influence, for example when a user has a failing WiFi connection. There is nothing you can do about that, so logging has no use. Giving clear feedback, if you can, might, however.

# Expected vs UnExpected Errors

## Unexpected errors

- Unexpected errors are annoying and hard.
- Suppose you built and tested your application(s), everything is on production, and then an error occurs: The response from the API you called returns JSON in a structure that is not how your application expected it. so parsing fails with an error. Or a user enters data in a form, but the value is too long.
- The error message "An error occurred, please try again" does not apply here. Unexpected errors are caused by bugs (programmer mistakes), so you can press that button as many times as you want, but chances are virtually zero it will somehow work later, without anyone changing anything in the source code of one of the applications involved.

## Expected vs Unexpected Errors

```
const handleDelete = async (post) => {
  const originalPosts = posts

  setPosts(posts.filter((p) => p.id !== post.id));

  try {
    await axios.delete(apiEndpoint + "/" + post.id);
  } catch (error) {
    if (error.response && error.response.status === 404)
      alert("This post has already been deleted.");
    else {
      console.log("Logging the error", error);
      alert("An unexpected error occurred.");
    }
    setPosts(originalPosts)
  };
}
```

# Handling Unexpected Errors Globally

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

axios.interceptors.response.use(null, error => {
  const expectedError =
    error.response &&
    error.response.status >= 400 &&
    error.response.status < 500;

  if (!expectedError) {
    console.log("Logging the error", error);
    alert("An unexpected error occurred.");
  }

  return Promise.reject(error)
});

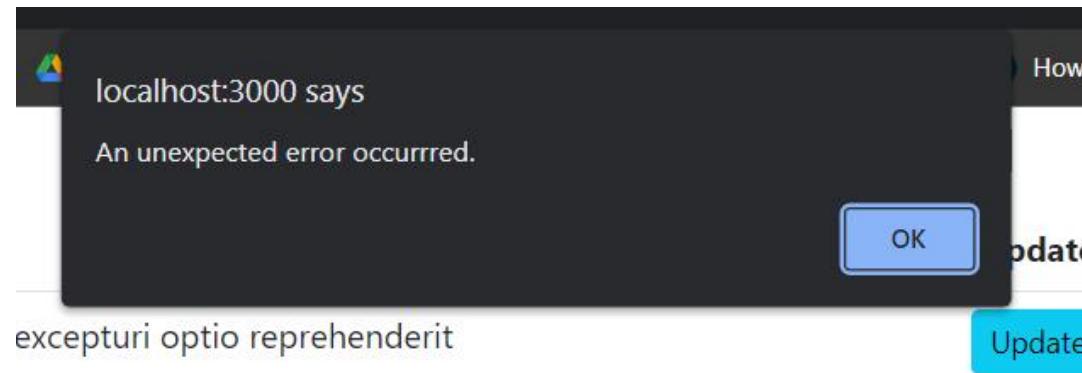
function App() {
```

# Handling Unexpected Errors Globally

```
const handleDelete = async (post) => {
  const originalPosts = posts

  setPosts(posts.filter((p) => p.id !== post.id));

  try {
    await axios.delete(`f${apiEndpoint}/${post.id}`);
  } catch (error) {
    if (error.response && error.response.status === 404)
      alert("This post has already been deleted.");
    setPosts(originalPosts)
  }
}
```



# Extracting a Reusable http Service

- Create new folder ‘services’ and create `httpService.js` inside that.
- Move Interceptor from `App.js` into `httpService.js`

```
services > JS httpService.js > [D] default
  import axios from 'axios';

  axios.interceptors.response.use(null, error => {
    const expectedError =
      error.response &&
      error.response.status >= 400 &&
      error.response.status < 500;

    if (!expectedError) {
      console.log("Logging the error", error);
      alert("An unexpected error occurred.");
    }

    return Promise.reject(error)
  });


```

```
  if (!expectedError) {
    console.log("Logging the error", error);
    alert("An unexpected error occurred.");
  }

  return Promise.reject(error)
});

const axiosMethods = {
  get: axios.get,
  post: axios.post,
  put: axios.put,
  delete: axios.delete
}

export default axiosMethods;
```

# Extracting a Reusable http Service

- Replace all axios occurrence

```
js App.js > 3D App
import { useState, useEffect } from 'react';
import axiosMethods from './services/httpService';
import './App.css';

function App() {
  const [posts, setPosts] = useState([])
  const apiEndpoint = "https://jsonplaceholder.typicode.com/posts"

  async function getResults() {
    const results = await axiosMethods.get(apiEndpoint);
    setPosts(results.data)
  }
}
```

# Extracting a Config Module

- Create a file config.json in src

```
src > {..} config.json > ...
1  {
2    "apiEndpoint": "https://jsonplaceholder.typicode.com/posts"
3  }
4
```

- Replace all endpoint occurrence in app.js with prefix of 'config.'

```
import { useState, useEffect } from 'react';
import axiosMethods from './services/httpService';
import config from './config.json';
import './App.css';

function App() {
  const [posts, setPosts] = useState([])

  async function getResults() {
    const results = await axiosMethods.get(config.apiEndpoint);
    setPosts(results.data)
  }

  useEffect(() => {
    getResults();
  }, [])
}

export default App;
```

# Connecting Movies Page to the Backend

- Switch to vidly
- npm i axios
- npm i react-toastify
- Add httpService.js previously created to services folder
- Add config previously created to src and replace endpoint with vidly endpoint <http://codewithz.centralindia.cloudapp.azure.com:3900/api/>

```
c > {} config.json > ...
1  {
2    "apiEndpoint": "http://codewithz.centralindia.cloudapp.azure.com:3900/api/"
3  }
4
```

# Displaying Toast Notification

- Install ‘npm i react-toastify’ then In App.js import and add

```
c > ⚙ App.jsx > ...
1  import "./App.css";
2  import { Routes, Route } from "react-router-dom";
3
4  import { ToastContainer } from "react-toastify";
5  import 'react-toastify/dist/ReactToastify.css'
6
7  import NavBar from "./components/navBar";
```

```
function App() {
  return (
    <div className="App">
      <ToastContainer />
      <NavBar />
      <main className="container">
        <Routes>
```

# Displaying Toast Notification

- Update `httpService.js`
- Replace '`alert`' with '`toast`' '`error, success....`' in whole application.

```
services > JS httpService.js > [o] axiosMethods > ⚙ delete
import axios from 'axios';
import { toast } from 'react-toastify';

axios.interceptors.response.use(null, error => {
  const expectedError =
    error.response &&
    error.response.status >= 400 &&
    error.response.status < 500;

  if (!expectedError) {
    console.log("Logging the error", error);
    toast.error("An unexpected error occurred.");
  }

  return Promise.reject(error)
});
```

# Displaying Toast Notification

- When Error occurs

The screenshot shows a web-based movie database application named 'Vidly'. The top navigation bar includes links for 'Movies', 'Customers', 'Rentals', 'Login', and 'Register'. On the left, a sidebar titled 'All Genres' lists categories: Action, Comedy, Musical, Romance, and Thriller. A 'New Movie' button is located at the top right. The main content area displays a message 'Showing 9 Movies in the Database' above a search bar. Below the search bar is a table with columns: Title, Genre, Stock, Rate, a heart icon, and a 'Delete' button. The table contains four rows of movie data:

Title	Genre	Stock	Rate	Heart Icon	Delete
<a href="#">Pretty Woman</a>	Romance	15	2	♥	<a href="#">Delete</a>
<a href="#">Terminator</a>	Action	10	2	♥	<a href="#">Delete</a>
<a href="#">The Avengers</a>	Action	15	2	♥	<a href="#">Delete</a>
<a href="#">The Hangover</a>	Comedy	10	2	♥	<a href="#">Delete</a>

At the bottom, there are page navigation buttons labeled 1, 2, and 3.

A red box highlights a toast notification in the top right corner, which contains an exclamation mark icon, the text 'An unexpected error occurred.', and a close button.

# Replacing fakeGenreService

- Create `genreService.jsx` in `service` folder

```
src > services > js genreService.js > ...
1 import axiosMethods from './httpService'
2 import config from './../config.json'
3
4 export function getGenres() {
5   return axiosMethods.get(config.apiEndpoint + 'genres')
6 }
7
```

- Delete `fakeGenreService` and replace import in `movies.jsx`

```
import { getMovies } from "../../services/fakeMovieService";
import { getGenres } from "../services/genreService";
import Pagination from "./common/pagination";
import { Paginate } from "./utils/paginate";
importListGroup from "./common/listGroup";
import MoviesTable from "./moviesTable";
```

# Replacing fakeGenreService

- Update `getGenres` methods in `movies.js`

```
19  const [currentPage, setCurrentPage] = useState(1);
20  const [sortColumn, setSortColumn] = useState({ path: "title", order: "asc" });
21
22  async function getAllGenres() {
23    const { data } = await getGenres();
24    setGenres([allGeneres, ...data]);
25  }
26
27  useEffect(() => {
28    setMovies(getMovies);
29    getAllGenres();
30  }, []);
31
```

# Replacing Fake Movie Service

- Create `movieService.js` in `service` folder

```
services > JS movieService.js > ...
import axiosMethods from './httpService'
import config from '../../config.json'

export function getMovies() {
  return axiosMethods.get(config.apiEndpoint + 'movies')
}

export function deleteMovie(movieId) {
  return axiosMethods.delete(config.apiEndpoint + 'movies/' + movieId)
}
```

- Delete `fakemovieService` and replace import in `movies.jsx`

```
import { getMovies, deleteMovie } from "../../services/movieService";
import { getGenres } from "../../services/genreService";
import Pagination from "./common/pagination";
import { Paginate } from "./utils/paginate";
```

# Replacing Fake Movie Service

- Create `getAllMovies` methods in `movies.jsx`

```
async function getAllMovies() {
  const { data } = await getMovies();
  setMovies([...data]);
}

useEffect(() => {
  getAllMovies();
  getAllGenres();
}, [ ]);
```

```
const handleDelete = async (movie) => {
  const originalMovies = movies;

  setMovies(originalMovies.filter(m => m._id !== movie._id));
  try {
    await deleteMovie(movie._id);
  } catch (error) {
    if (error.response && error.response.status === 404) {
      alert("This Movie has already been deleted!");
      setMovies(originalMovies);
    }
  }
};
```

# Connecting Movie Form to the Backend

- Update MovieForm

```
import Joi from "joi-browser";

import { getGenres } from "../../services/genreService";
import { getMovie, saveMovie } from "../../services/fakeMovieService";
```

```
async function getAllGenres() {
  const { data } = await getGenres();
  setGenres([...data]);
}
```

```
useEffect(() => {
  getAllGenres();
  if (id === "new") return;

  const movie = getMovie(id);
  if (!movie) return navigate("/not-found", { replace: true });

  setData(mapToViewModel(movie));
}, [id, navigate]);
```

# Populating the Form

- Add in Movie Service

```
src > services > movieService.js > ...
1 import axiosMethods from './httpService'
2 import config from './../config.json'
3
4 export function getMovies() {
5   return axiosMethods.get(config.apiEndpoint + 'movies')
6 }
7
8 export function getMovie(movieId) {
9   return axiosMethods.get(config.apiEndpoint + "movies/" + movieId);
10 }
11
12 export function saveMovie(movieId) {
13 }
14
15 export function deleteMovie(movieId) {
16   return axiosMethods.delete(config.apiEndpoint + 'movies/' + movieId)
17 }
18
```

# Populating the Form

- Update in MovieForm

```
    setGenres([...data]);
}

async function getSingleMovie() {
  try {
    const { data } = await getMovie(id);
    setData(mapToViewModel(data));
  } catch (error) {
    if (error.response && error.response.status === 404) {
      navigate("/not-found", { replace: true });
    }
  }
}

useEffect(() => {
  getAllGenres();
  if (id === "new") return;

  getSingleMovie();
}, [id]);

const mapToViewModel = (movie) => {
```

# Saving the Movie

- Update MovieService saveMovie

```
4  export function getMovies() {
5    return axiosMethods.get(config.apiEndpoint + 'movies')
6  }
7
8  export function getMovie(movieId) {
9    return axiosMethods.get(config.apiEndpoint + "movies/" + movieId);
10 }
11
12 export function saveMovie(movie) {
13   if (movie._id) {
14     const body = { ...movie };
15     delete body._id;
16     return axiosMethods.put(config.apiEndpoint + "movies/" + movie._id, body);
17   }
18   return axiosMethods.post(config.apiEndpoint + "movies/", movie);
19 }
20
21
22 export function deleteMovie(movieId) {
23   return axiosMethods.delete(config.apiEndpoint + 'movies/' + movieId)
24 }
```

# Saving the Movie

- Update in MovieForm

```
4     });
5   };
6
7   const doSubmit = async () => {
8     await saveMovie(data);
9     navigate("/");
10    };
11
12   return (
13     <div className="container">
```



# Authentication & Authorization

JSON Web Tokens

Calling Protected APIs

Showing / Hiding Elements

Protecting Routes

# Registering A New User

- Create userService.js

```
src > services > JS userService.js > ...
1 import axiosMethods from './httpService'
2 import config from '../config.json'
3
4 export function register(user) {
5   return axiosMethods.post(config.apiEndpoint + "users", {
6     email: user.username,
7     password: user.password,
8     name: user.name
9   });
10 }
11
```

- Update RegisterForm

```
import { handleSubmit, renderInput, renderButton } from "./helpers/form";
import { register } from "../services/userService";
```

```
const doSubmit = async () => {
  await register(data);
};
```

# Handling Registration Errors

- Update registerForm

```
const doSubmit = async () => {
  try {
    await register(data);
  } catch (error) {
    if (error.response && error.response.status === 400) {
      errors.username = error.response.data;
      setErrors({ username: error.response.data });
    }
  }
};
```

# Submitting the Login Form

- Create authService.js in services

```
src > services > JS authService.js > ...
1 import axiosMethods from './httpService'
2 import config from './../config.json'
3
4 export function login(email, password) {
5     return axiosMethods.post(config.apiEndpoint + "auth", {
6         email: email,
7         password: password
8     });
9 }
10
```

# Submitting the Login Form

- Update in loginForm

```
components > LoginForm.jsx > LoginForm
import { useState } from "react";
import Joi from "joi-browser";

import { handleSubmit, renderInput, renderButton } from "./helpers/form";
import { login } from "../services/authService";  

function LoginForm() {
  const [data, setData] = useState({ username: "", password: "" });
  const [errors, setErrors] = useState({});

  const schema = {
    username: Joi.string().required().label("Username"),
    password: Joi.string().required().label("Password"),
  };

  const doSubmit = async () => {
    await login(data.username, data.password);
    console.log("Submitted");
  };
}
```

# Handling login Errors

```
const doSubmit = async () => {
  try {
    await login(data.username, data.password);
  } catch (error) {
    if (error.response && error.response.status === 400) {
      errors.username = error.response.data;
      setErrors({ username: error.response.data });
    }
  }
};
```

# Storing the JWT

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import Joi from "joi-browser";

import { handleSubmit, renderInput, renderButton } from "./helpers/form";
import { login } from "../services/authService";

function LoginForm() {
  const [data, setData] = useState({ username: "", password: "" });
  const [errors, setErrors] = useState({});
  const navigate = useNavigate();

  const schema = {
    const doSubmit = async () => {
      try {
        const response = await login(data.username, data.password);
        localStorage.setItem("token", response.data);
        navigate("/");
      } catch (error) {
        if (error.response && error.response.status === 400) {
          errors.username = error.response.data;
          setErrors({ username: error.response.data });
        }
      }
    };
  };
}
```

# Logging the user upon Registration

- Change in api users route

```
const token = user.generateAuthToken();
res
  .header("x-auth-token", token)
  .header("access-control-expose-headers", "x-auth-token")
  .send(_.pick(user, ["_id", "name", "email"]));
});
```

- In registerForm

```
import Joi from "joi-browser";
import { useNavigate } from "react-router-dom";
import { handleSubmit, renderInput, renderButton } from "./helpers/form";
import { register } from "../services/userService";

function RegisterForm() {
  const [data, setData] = useState({ username: "", password: "", name: "" });
  const [errors, setErrors] = useState({});
  const navigate = useNavigate()
```

# Logging the user upon Registration

```
const doSubmit = async () => {
  try {
    const response = await register(data);
    localStorage.setItem("token", response.headers["x-auth-token"]);
    navigate("/");
  } catch (error) {
    if (error.response && error.response.status === 400) {
      errors.username = error.response.data;
      setErrors({ username: error.response.data });
    }
  }
};
```

# JSON Web Tokens



Debugger   Libraries   Introduction   Ask

Crafted by  auth0 by Okta

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJf  
aWQiOiI2Mzd1NTJiOTQ4NWViMjE3OWNlYzVkJ  
jAiLCJuYW1lIjoiQWRtaW4gNSIsImVtYWlsIjoi  
YWRtaW41QGdtYWlsLmNvbSIIsImhdCI6MTY20TI  
yMzA5N30.d6Tcs8cHlaASxCe9Zxr5en8CWskJKB  
Tbk1H5cznLGfs|
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "_id": "637e52b9485eb2179cec5d20",  
  "name": "Admin 5",  
  "email": "admin5@gmail.com",  
  "iat": 1669223097  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret
```

# Getting the current user

- Install : npm i jwt-decode
- Import and update App.js

```
src > 🌐 App.jsx > 📄 App > 📄 useEffect() callback
  1 import "./App.css";
  2 import { useEffect, useState } from "react";
  3 import { Routes, Route } from "react-router-dom";
  4
  5 import { ToastContainer } from "react-toastify";
  6 import "react-toastify/dist/ReactToastify.css";
  7
  8 import jwtDecode from "jwt-decode";
  9
 10 import NavBar from "./components/navBar";
 11 import Movies from "./components/movies";
```

# Getting the current user

- Install : npm i jwt-decode
- Import and update App.js

```
function App() {
  const [user, setUser] = useState();

  useEffect(() => {
    try {
      const jwt = localStorage.getItem("token");
      setUser(jwtDecode(jwt));
    } catch (error) {}
  }, []);

  return (
    <div className="App">
      <ToastContainer />
      <NavBar user={user} />
      <main className="container">
```

# Displaying the current user on Navbar

- Update in Navbar

```
import { Link, NavLink } from "react-router-dom"

const NavBar = (props) => {
  return (
    <nav className="container navbar navbar-expand-lg">
      <Link to="/" className="navbar-brand">React Router DOM</Link>
      <button
        type="button"
        className="navbar-toggler"
        data-toggle="collapse"
        data-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent"
        aria-expanded="false"
        aria-label="Toggle navigation">
        <span>Menu</span>
      </button>
      <div
        className="collapse navbar-collapse"
        id="navbarSupportedContent">
        <ul
          className="navbar-nav mr-auto">
          <li><NavLink to="/about">About</NavLink></li>
          <li><NavLink to="/contact">Contact</NavLink></li>
        </ul>
        <ul
          className="navbar-nav">
          <li><NavLink to="/rentals">Rentals</NavLink></li>
          <li><NavLink to="/login">Login</NavLink></li>
          <li><NavLink to="/register">Register</NavLink></li>
        </ul>
      </div>
    </nav>
  )
}

export default NavBar
```

```
; > 🌐 navBar.jsx > [?] NavBar
  <div>
    <NavLink className="nav-item nav-link" to="/rentals">
      Rentals
    </NavLink>
    {!props.user && (
      <>
        <NavLink className="nav-item nav-link" to="/login">
          Login
        </NavLink>
        <NavLink className="nav-item nav-link" to="/register">
          Register
        </NavLink>
      </>
    )}
    {props.user && (
      <>
        <NavLink className="nav-item nav-link" to="/me">
          {props.user.name}
        </NavLink>
        <NavLink className="nav-item nav-link" to="/logout">
          Logout
        </NavLink>
      </>
    )}
  </div>
```

# Displaying the current user on Navbar

- Update in loginForm and registerForm

```
const doSubmit = async () => {
  try {
    const response = await login(data.username, data.password);
    localStorage.setItem("token", response.data);
    window.location = "/";
  } catch (error) {
    if (error.response && error.response.status === 400) {
      errors.username = error.response.data;
      setErrors([username, error.response.data]);
    }
  }
};
```

```
const doSubmit = async () => {
  try {
    const response = await register(data);
    localStorage.setItem("token", response.headers["x-auth-token"]);
    window.location = "/";
  } catch (error) {
    if (error.response && error.response.status === 400) {
      errors.username = error.response.data;
      setErrors([username, error.response.data]);
    }
  }
};
```

# Logging Out a User

- Create logout.jsx in components

```
components > ✨ logout.jsx > ...
import { useEffect } from "react";

function Logout() {
  useEffect(() => {
    localStorage.removeItem("token");
    window.location = "/";
  }, []);
  return null;
}

export default Logout;
```

# Logging Out a User

- Update App.js

```
import Rentals from "./components/rentals";
import MovieForm from "./components/movieForm";
import LoginForm from "./components/loginForm";
import RegisterForm from "./components/registerForm";
import Logout from "./components/logout";
import NotFound from "./components/notFound";
```

```
<Routes>
  <Route path="/" element={<Movies />} />
  <Route path="movies/:id" element={<MovieForm />} />
  <Route path="customers" element={<Customers />} />
  <Route path="rentals" element={<Rentals />} />
  <Route path="login" element={<LoginForm />} />
  <Route path="logout" element={<Logout />} />
  <Route path="register" element={<RegisterForm />} />
  <Route path="*" element={<NotFound />} />
</Routes>
```

# Refactoring

- Update in authService

```
src > services > authService.js > ⚡ getCurrentUser
1 import axiosMethods from './httpService'
2 import config from './../config.json'
3 import jwtDecode from "jwt-decode";
4
5 const tokenKey = "token";
6
7 export async function login(email, password) {
8   const response = await axiosMethods.post(config.apiEndpoint + "auth", {
9     email: email,
10    password: password
11  });
12  localStorage.setItem(tokenKey, response.data);
13}
14
15 export function loginWithJwt(jwt) {
16  localStorage.setItem(tokenKey, jwt);
17}
18
19 export function logout() {
20  localStorage.removeItem(tokenKey);
21}
```

# Refactoring

- Update in authService

```
rc > services > authService.js > ⚡ getCurrentUser
14
15  export function loginWithJwt(jwt) {
16    localStorage.setItem(tokenKey, jwt);
17  }
18
19  export function logout() {
20    localStorage.removeItem(tokenKey);
21  }
22
23  export function getCurrentUser() {
24    try {
25      const jwt = localStorage.getItem(tokenKey);
26      return jwtDecode(jwt);
27    } catch (error) {
28      return null;
29    }
30  }
31
32  const auth = {
33    login,
34    loginWithJwt,
35    logout,
36    getCurrentUser
37  }
```

# Refactoring

- Update in loginForm , remove all unused const and imports

```
import { handleSubmit, renderInput, renderButton } from "./helpers/form";
import auth from "../services/authService";

function LoginForm() {
  const [data, setData] = useState({ username: "", password: "" });
  ...
```

```
const doSubmit = async () => {
  try {
    await auth.login(data.username, data.password);
    window.location = "/";
  } catch (error) {
    if (error.response && error.response.status === 400) {
      errors.username = error.response.data;
      setErrors({ username: error.response.data });
    }
  }
};
```

# Refactoring

- Update in App.jsx , remove all unused const and imports

```
import jwtDecode from "jwt-decode";  
import auth from "./services/authService";  
import NavBar from "./components/navBar";
```

```
function App() {  
  const [user, setUser] = useState();  
  
  useEffect(() => {  
    | setUser(auth.getCurrentUser())  
  }, []);  
  
  return (  
    <div>
```

# Refactoring

- Update in Logout

```
> components > logout.jsx > ...
1 import { useEffect } from "react";
2 import auth from './../../../services/authService'
3
4 function Logout() {
5   useEffect(() => {
6     auth.logout()
7     window.location = "/";
8   }, []);
9
10  return null;
11}
12
13 export default Logout;
14|
```

# Refactoring

- Update in registerForm

```
import { register } from "../services/userService";
import auth from "../services/authService";

function RegisterForm() {

  const doSubmit = async () => {
    try {
      const response = await register(data);
      auth.loginWithJwt(response.headers["x-auth-token"]);
      window.location = "/";
    } catch (error) {
      if (error.response && error.response.status === 400) {
        errors.username = error.response.data;
        setErrors({ username: error.response.data });
      }
    }
  }
}
```

# Calling Protected API Endpoints

- Turn on authentication in backend
- Update in authService & in httpService

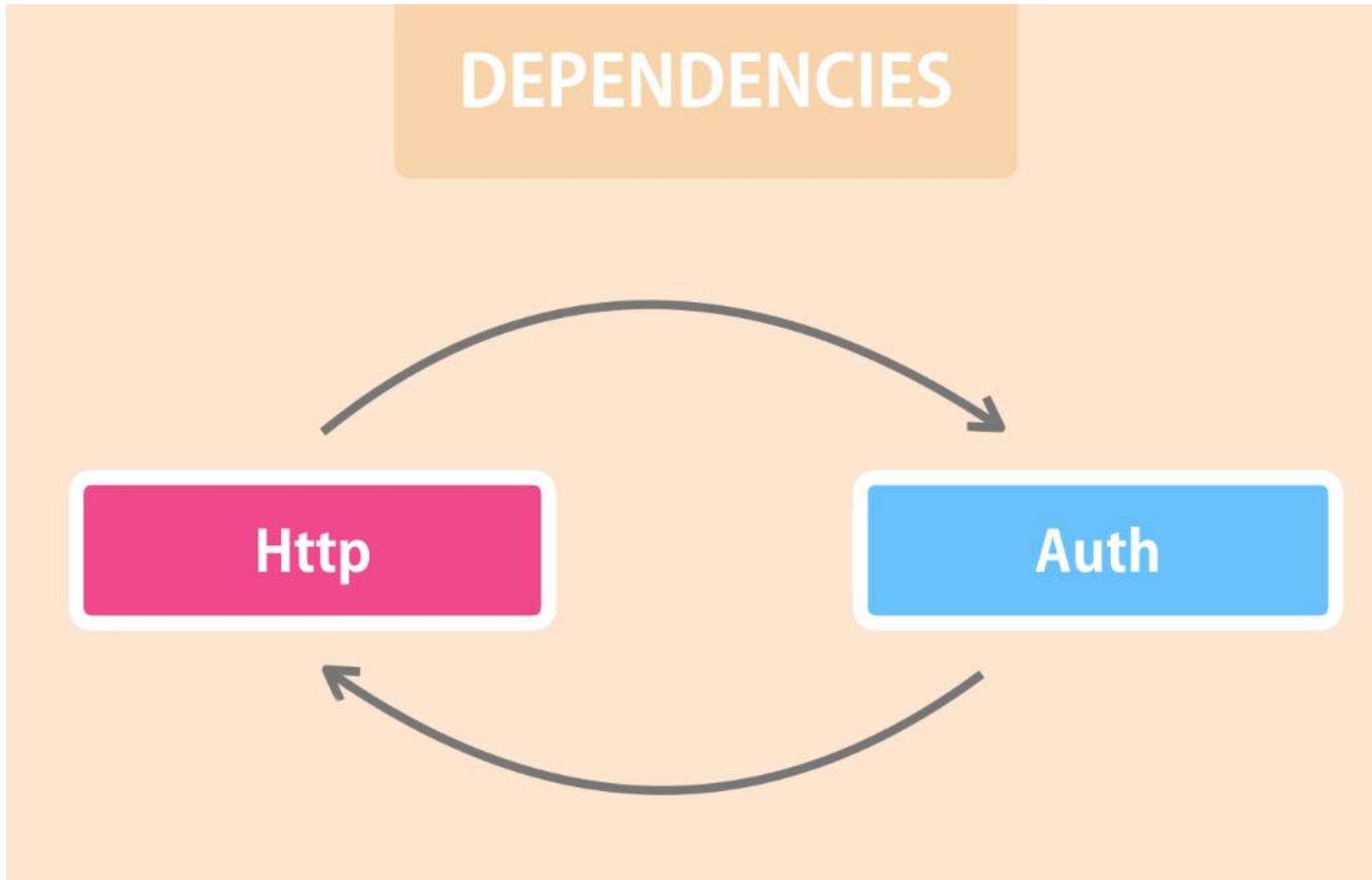
```
export function getJwt() {
  return localStorage.getItem(tokenKey)
}
```

```
const auth = {
  login,
  loginWithJwt,
  logout,
  getCurrentUser,
  getJwt
}
```

```
config > {} default.json > ...
1  {
2    "jwtPrivateKey": "unsecureKey",
3    "db": "mongodb://127.0.0.1:27017/vidly",
4    "port": "3900",
5    "requiresAuth": true
6  }
```

```
src > services > JS httpService.js > ⓘ axios.interceptors.response.use() callback
1  import axios from 'axios';
2  import auth from './authService';
3  import { toast } from 'react-toastify';
4
5  axios.defaults.headers.common['x-auth-token'] = auth.getJwt()
6
7  axios.interceptors.response.use(null, error => {
```

# Fixing Bi-directional Dependencies



# Fixing Bi-directional Dependencies

- Update in httpService -- remove

```
> services > JS httpService.js > ...
1 import axios from 'axios';
2 import auth from './authService';
3 import { toast } from 'react-toastify';
4
5 axios.defaults.headers.common['x-auth-token'] = auth.getJwt();
6
7 axios.interceptors.response.use(null, error => {
8     const expectedError =
9         error.response &&
10        error.response.status >= 400 &&
```

# Fixing Bi-directional Dependencies

- Update in httpService -- add

```
    return Promise.reject(error)
});

function setJwt(jwt) {
  axios.defaults.headers.common["x-auth-token"] = jwt
}

const axiosMethods = {
  get: axios.get,
  post: axios.post,
  put: axios.put,
  delete: axios.delete,
  setJwt
}

export default axiosMethods;
```

# Fixing Bi-directional Dependencies

- Update in authService -- add

```
import config from './config.json';
import jwtDecode from "jwt-decode";

const tokenKey = "token";

axiosMethods.setJwt(getJwt())

export async function login(email, password)
  const response = await axiosMethods.post('/
```

# Authorization

- Delete requires user to be Admin so, Update in Database

```
_id: ObjectId('637d1a57f2b07003689ac6d0')
name: "Admin"
email: "admin@gmail.com"
password: "$2b$10$qojYgOyE3N0BqhxFud.Y..6BZsIelesXhJmRZyonI75Vmvyqf09S"
__v: 0
isAdmin: true
```

```
_id: ObjectId('637e5024eb58030d880e1d23')
... . . . .
```

# Showing or Hiding elements based on the user

- Update in App.js

```
<main className="container">
  <Routes>
    <Route path="/" element={<Movies user={user} />} />
    <Route path="movies/:id" element={<MovieForm />} />
    <Route path="customers" element={<Customers />} />
```

- Update in movies.jsx

```
<div className="col">
  {props.user && <Link
    to="movies/new"
    className="btn btn-primary"
    style={{ marginBottom: 20 }}
  >
    New Movie
  </Link>}
  <p>Showing {filtered.length} Mo
```

# Protecting Routes

- Update In App.js

```
<main className="container">
  <Routes>
    <Route path="/" element={<Movies user={user} />} />
    <Route path="movies/:id" element={<MovieForm user={user} />} />
    <Route path="customers" element={<Customers />} />
```

- Update in MoviesForm

```
useEffect(() => {
  if (!props.user) {
    navigate("/login");
  }
  getAllGenres();
  if (id === "new") return;

  getSingleMovie(),
  [id]);
```

# Extracting the Protected Route

- Create ProtectedRoute component in common

```
> components > common > ✨ ProtectedRoute.jsx > ...
import { useEffect } from "react";
import { useNavigate } from "react-router-dom";

export const ProtectedRoute = (props) => {
  const navigate = useNavigate();
  useEffect(() => {
    if (!props.auth) {
      navigate("/login");
    }
  }, [navigate, props.auth]);
  return props.children;
};
```

# Extracting the Protected Route

- Update in App.js

```
import RegisterForm from './components/RegisterForm';
import Logout from "./components/logout";
import { ProtectedRoute } from "./components/common/ProtectedRoute";
import NotFound from "./components/notFound";
```

```
<Routes>
  <Route path="/" element={<Movies user={user} />} />
  <Route
    path="movies/:id"
    element={
      <ProtectedRoute auth={user}>
        <MovieForm />
      </ProtectedRoute>
    }
  />
  <Route path="customers" element={<Customers />} />
```