Learning React.js - Comprehensive Guide

Table Of Content:

- **Created Under guidance Of Nilesh Prajapati.**
- **Level Of Document : Basic**
- **Outcome: Basic understanding of React js project and able to create a basic project.**

Introduction:

- ➢ Welcome to the Learning React.js comprehensive guide! React.js, developed and maintained by Meta, is a powerful JavaScript library for building user interfaces. Whether you're a beginner looking to get started or an experienced developer aiming to enhance your skills, this guide will take you through essential concepts and resources to master React.js.

Prerequisites:

- ➢ Before delving into React.js, ensure you have a solid understanding of the following:

Basic HTML/CSS:

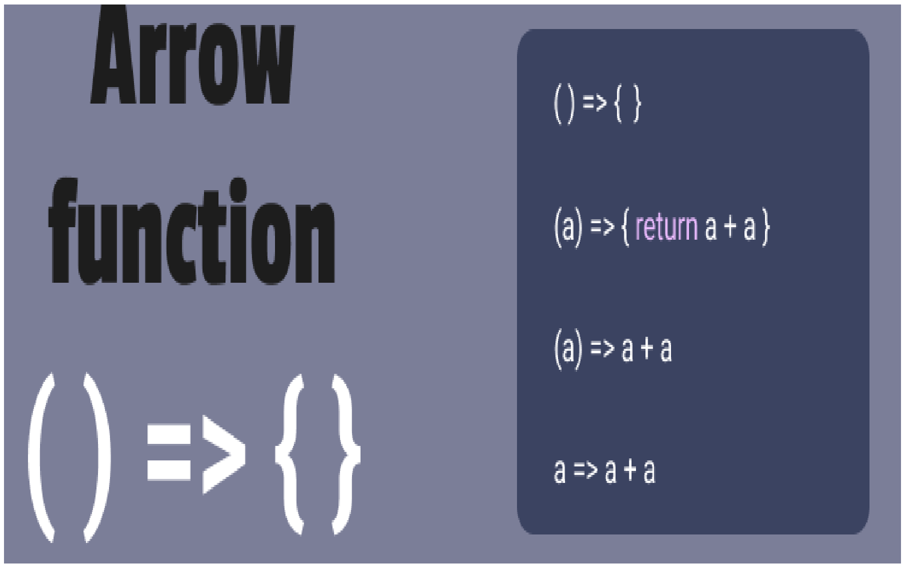
- ✓ Understanding the structure and styling of web pages using HTML and CSS is fundamental for creating React components.

JavaScript Fundamentals:

- ✓ A good grasp of core JavaScript concepts like variables, functions, loops, and conditionals is essential.
- ✓ Also having understanding of filter, reduce , map , (…)spread Operator.

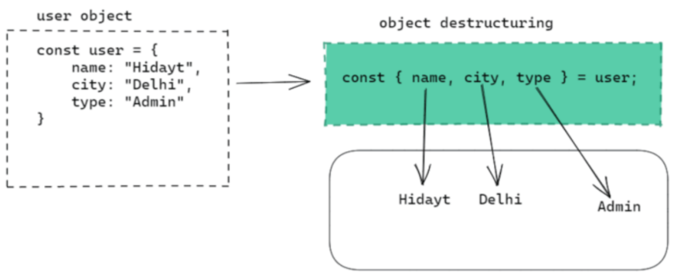
ES6 Features:

- ✓ Familiarity with ES6 features such as
- ✓ Arrow functions



```
( ) => { }
(a) => { return a + a }
(a) => a + a
a => a + a
```

- ✓ Destructuring



- ✓ Classes will be beneficial in writing modern React code.

Node.js and npm:

- ✓ Node.js is a JavaScript runtime, and npm (Node Package Manager) is used for managing dependencies. Make sure to have Node.js and npm installed on your machine before starting.

Getting Started

- ✓ **Installing Node.js and npm**

1. **Download Node.js:**

   - ✓ Visit https://nodejs.org/
   - ✓ Download the recommended version for your operating system.

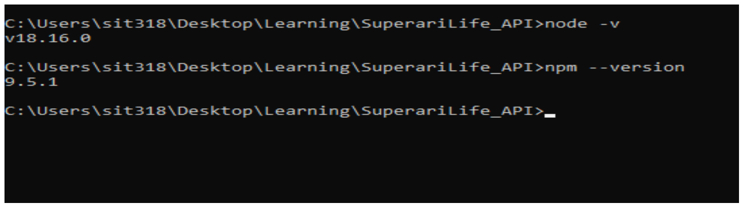
2. **Install Node.js:**

   - ✓ Follow the installation instructions for your operating system. npm will be installed automatically.

3. **Verify Installation:**

Open a terminal or command prompt.

Type `node -v` and `npm -v` to check if Node.js and npm are installed successfully.



```
C:\Users\sit318\Desktop\Learning\SuperariLife_API>node -v
v18.16.0

C:\Users\sit318\Desktop\Learning\SuperariLife_API>npm --version
9.5.1

C:\Users\sit318\Desktop\Learning\SuperariLife_API>_
```

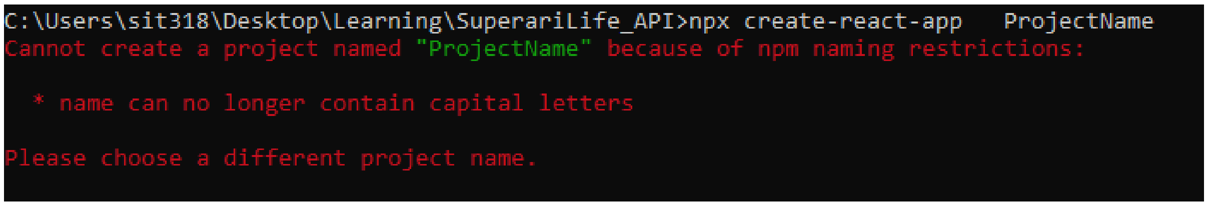
➢ Creating a React App:

1. **Open a Terminal or Command Prompt:**

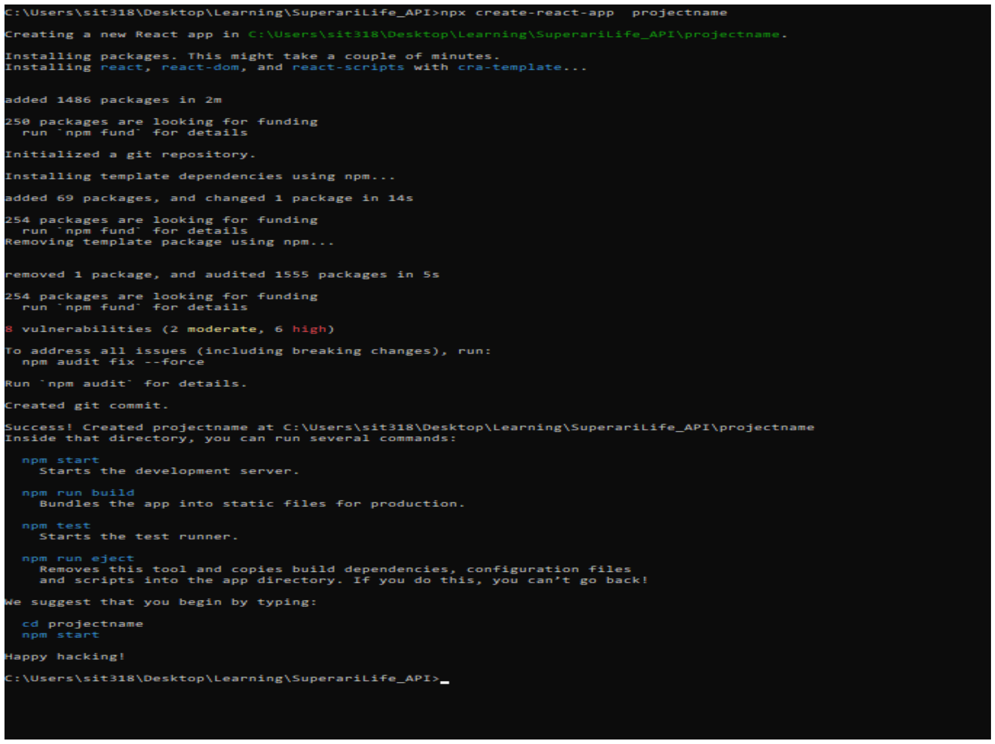
   ✓ Navigate to the directory where you want to create your React app.

2. **Run `create-react-app` Command:**

   ✓ Use `npx create-react-app my-react-app` to create a new React app (replace "my-react-app" with your preferred project name).



```
C:\Users\sit318\Desktop\Learning\SuperariLife_API>npx create-react-app   ProjectName
Cannot create a project named "ProjectName" because of npm naming restrictions:

  * name can no longer contain capital letters

Please choose a different project name.
```

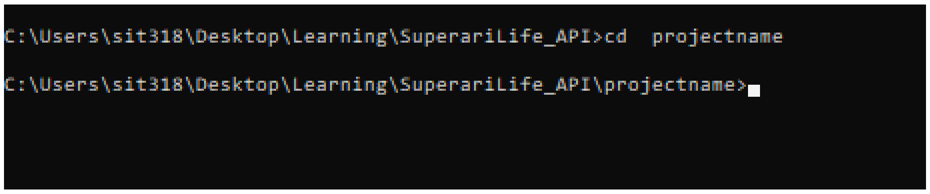


```
C:\Users\sit318\Desktop\Learning\SuperariLife_API>npx create-react-app  projectname

Creating a new React app in C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname.

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


added 1486 packages in 2m

250 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 69 packages, and changed 1 package in 14s

254 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1555 packages in 5s

254 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created projectname at C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd projectname
  npm start

Happy hacking!

C:\Users\sit318\Desktop\Learning\SuperariLife_API>
```

3. Starting the Development Server

1. Navigate into the Project Directory:

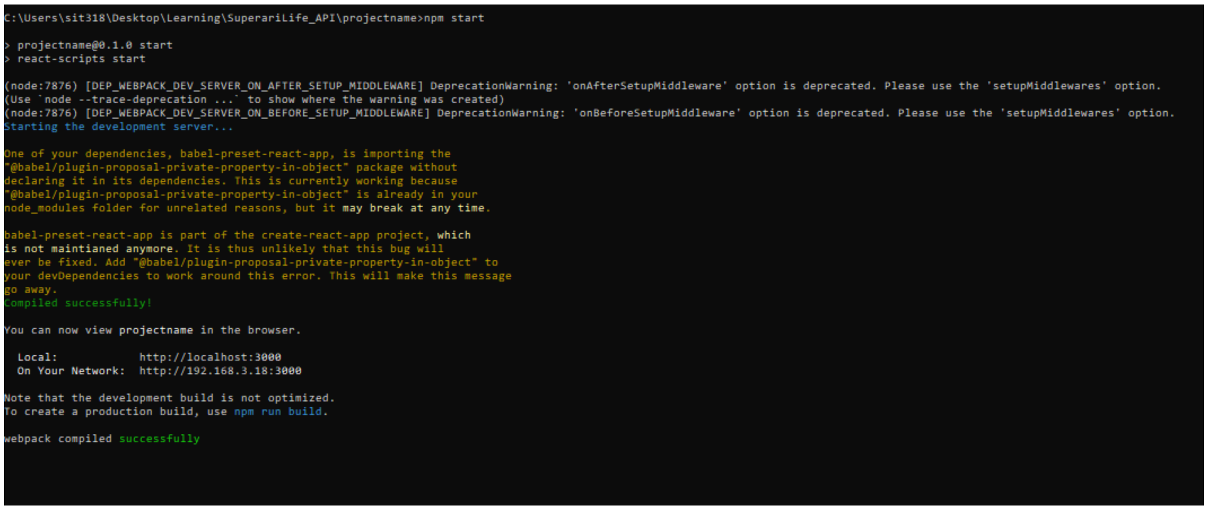
- ✓ Run `cd projectname`.



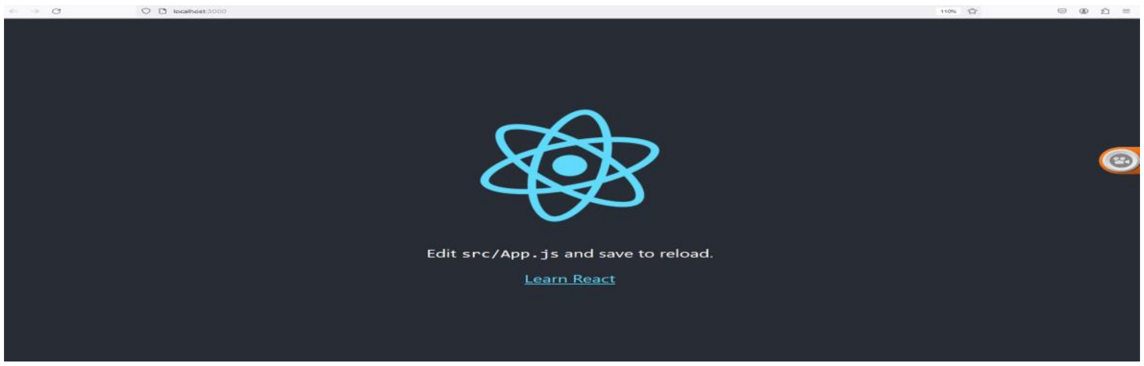
```
C:\Users\sit318\Desktop\Learning\SuperariLife_API>cd  projectname

C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname>
```

2. Start the Development Server:

- ✓ Run `npm start` to open your new React app in your default web browser.

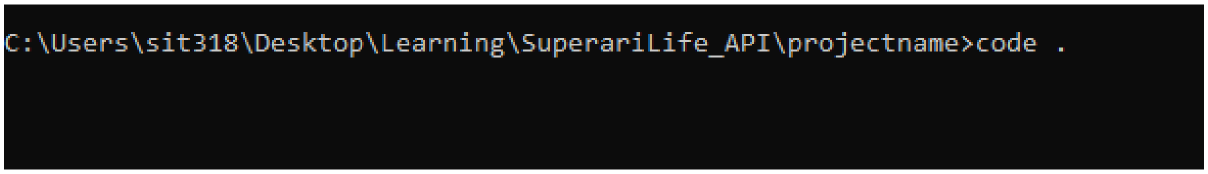


```
C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname>npm start

> projectname@0.1.0 start
> react-scripts start

(node:7876) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:7876) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintianed anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.
Compiled successfully!

You can now view projectname in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.3.18:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



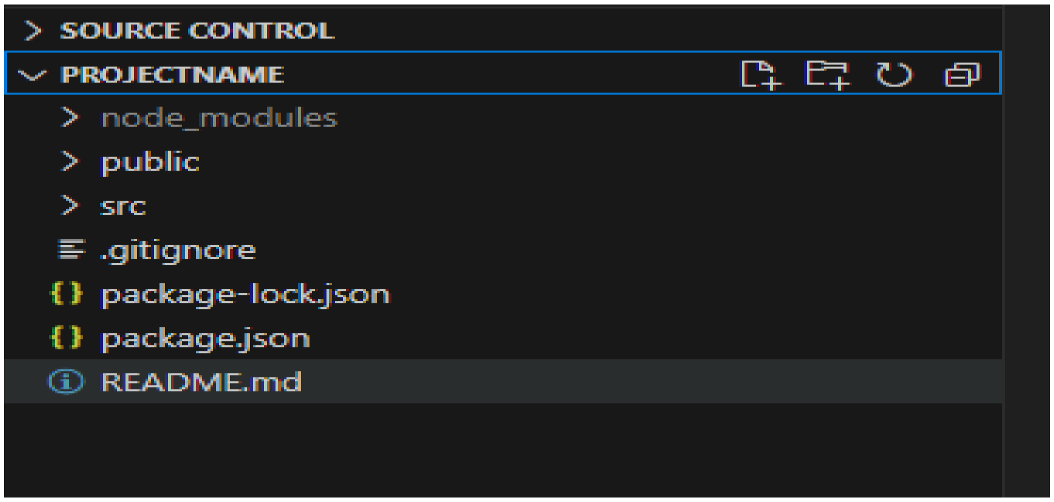
This sets up a basic React project and provides a development server for real-time updates.

3. **Opening Project in Editor (VS code)**

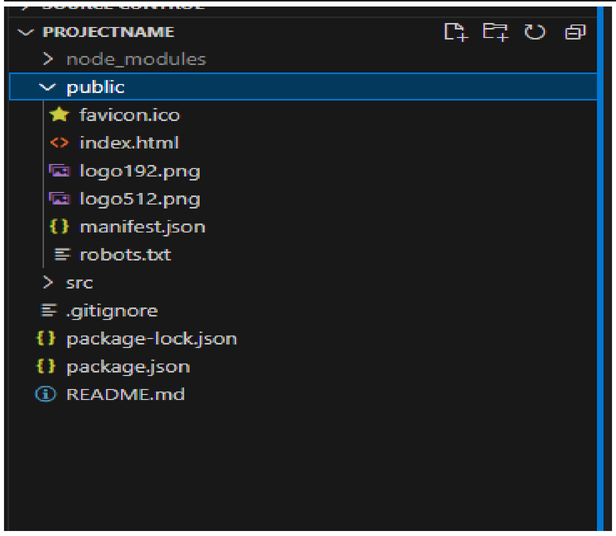


```
C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname>code .
```

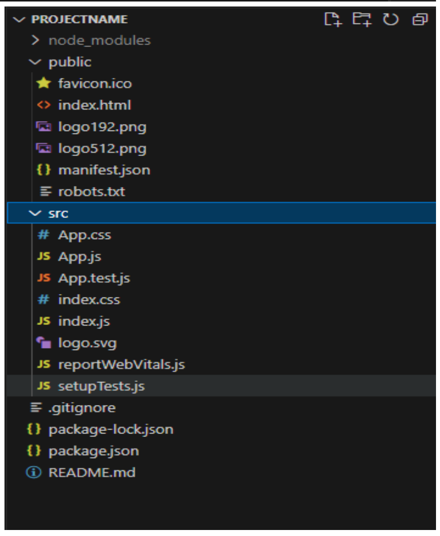
✓ Steps:
Run ‘ code .’ -> Press enter.



```
> SOURCE CONTROL
v PROJECTNAME
  > node_modules
  > public
  > src
    .gitignore
    package-lock.json
    package.json
    README.md
```



```
v PROJECTNAME
  > node_modules
  v public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  > src
    .gitignore
    package-lock.json
    package.json
    README.md
```



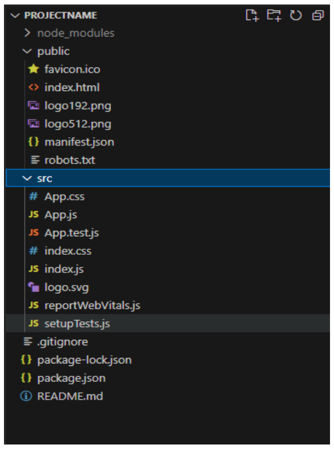
```
v PROJECTNAME
  > node_modules
  v public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  v src
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
```

➢ Folder structure Explanation:

React Application File Structure Guide

Overview

Project Root



```
PROJECTNAME
  > node_modules
  v public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  v src
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
```

- ✓ node_modules/`: Node.js dependencies. (Generated automatically, don't modify manually)
- ✓ `public/`: Static assets and HTML template.
- ✓ `index.html`: The main HTML file where the React app is mounted.
- ✓ `favicon.ico`: Favicon for the application.
- ✓ `src/`: Source code of the React application.
- ✓ `components/`: Reusable UI components.
- ✓ `pages/`: Top-level components representing application pages.
- ✓ `assets/`: Non-code assets such as images, fonts, or other media.
- ✓ `styles/`: Global styles or styles specific to components/pages.
- ✓ `utils/`: Utility functions or helper modules.
- ✓ `services/`: API services or other external services.
- ✓ `context/`: React Context providers and consumers.
- ✓ `hooks/`:Custom React hooks.
- ✓ `App.js`: The main component where the application is initialized.
- ✓ `index.js`: The entry point of the application, where React is rendered into the DOM.
- ✓ `package.json`: Configuration file for Node.js dependencies and scripts.
- ✓ `README.md`: Project documentation.

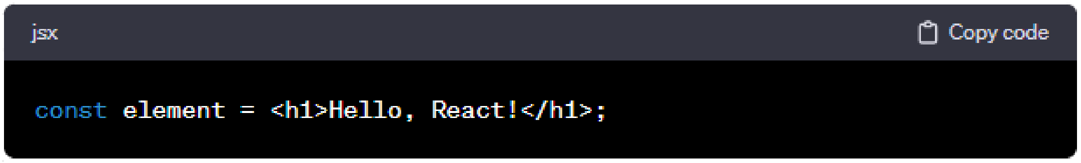
Let's have some understanding of key Concepts before we write a code for our demo project..

➢ Key Concepts:

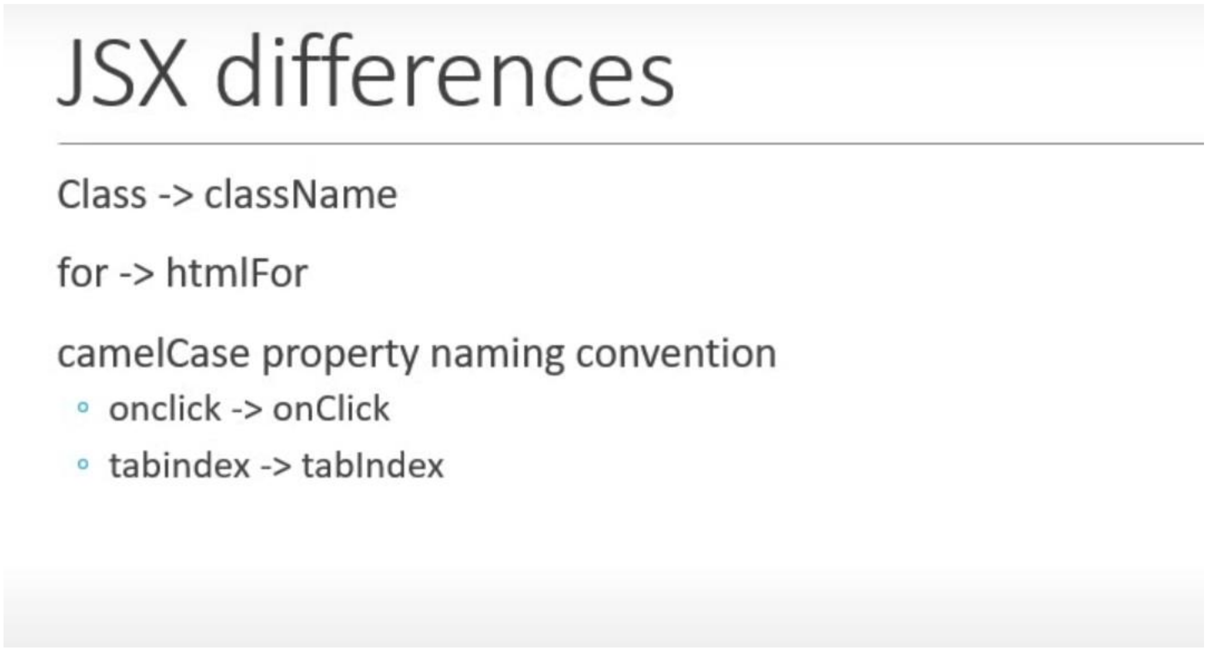
JSX (JavaScript XML)

JSX is a syntax extension for JavaScript recommended by React. It looks similar to XML/HTML and makes React code more readable.

Example:



```jsx
const element = <h1>Hello, React!</h1>;
```



JSX differences

Class -> className

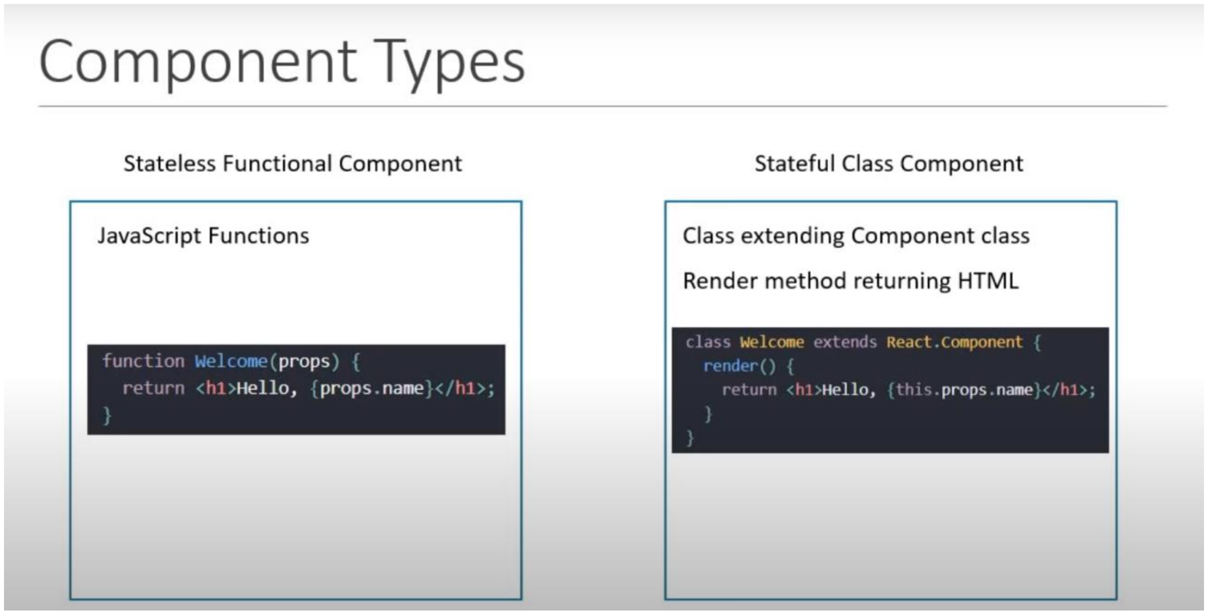
for -> htmlFor

camelCase property naming convention
- onclick -> onClick
- tabindex -> tabIndex

- It is similar to angular html file but here its JSX file and it generally use
- Synatax :React.createElement('h1',null,Hello , React); (Note : This code is for understanding not for use we will use it but for initial level just you should have idea about this.)

Components

Components are the building blocks of React applications. They are reusable, self-contained pieces of code responsible for rendering UI elements.



Component Types

Stateless Functional Component

JavaScript Functions

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Stateful Class Component

Class extending Component class

Render method returning HTML

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

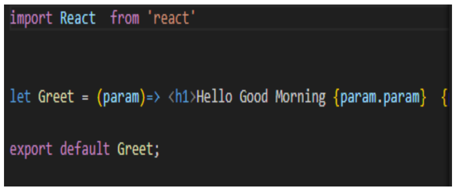
Functional Components

Functional components are stateless and are primarily used for displaying UI elements. They are simple and concise.

```
// Import React
import React from 'react';

// Define a functional component
function MyFunctionalComponent() {
  return (
    <div>
      <h1>Hello, I'm a Functional Component!</h1>
    </div>
  );
}

// Export the component for use in other files
export default MyFunctionalComponent;
```

Using arrow function syntax: (Advance code structure)

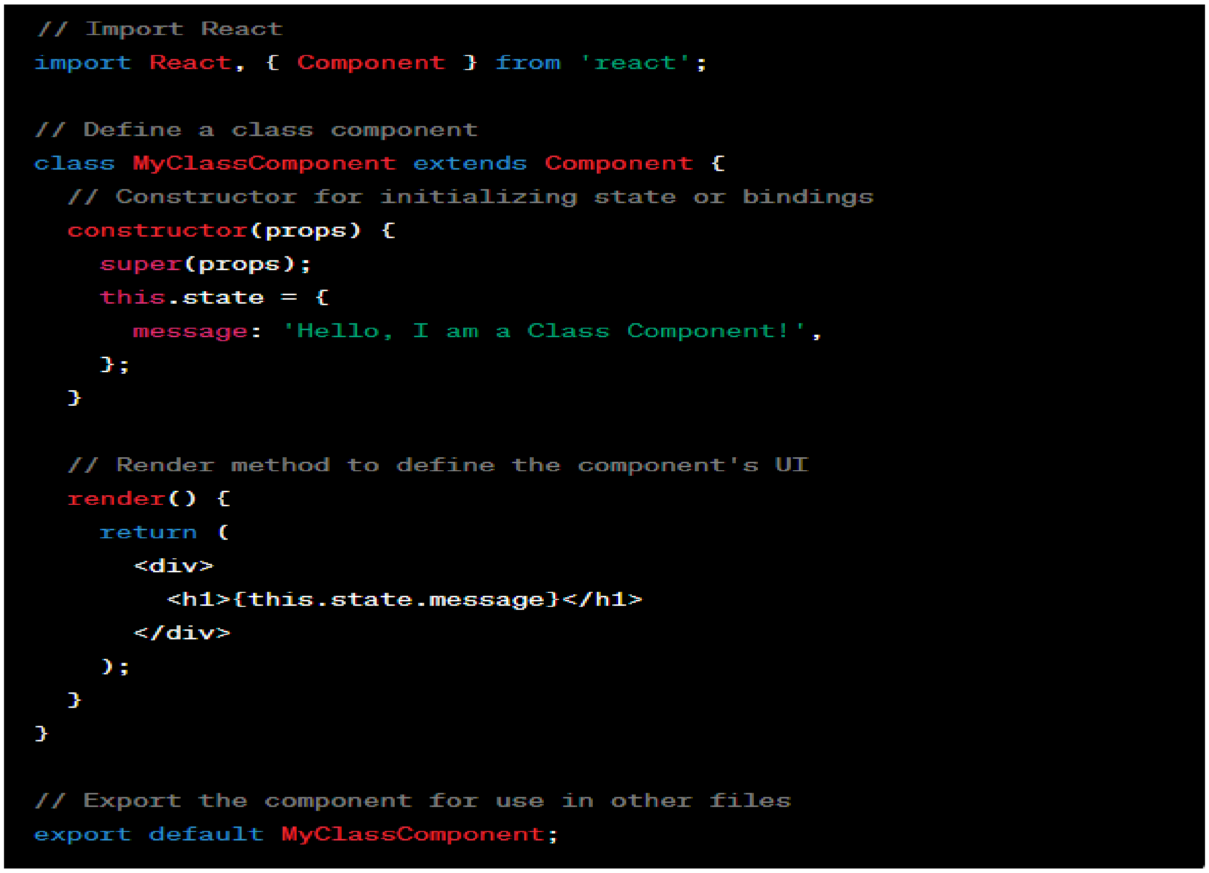
Syntax :
→let MyFunctionaComponent = ()=> return ({ <div> <h1> Hello, I am functional component</h1></div>})



```
import React  from 'react'


let Greet = (param)=> <h1>Hello Good Morning {param.param}  {

export default Greet;
```

Class Components

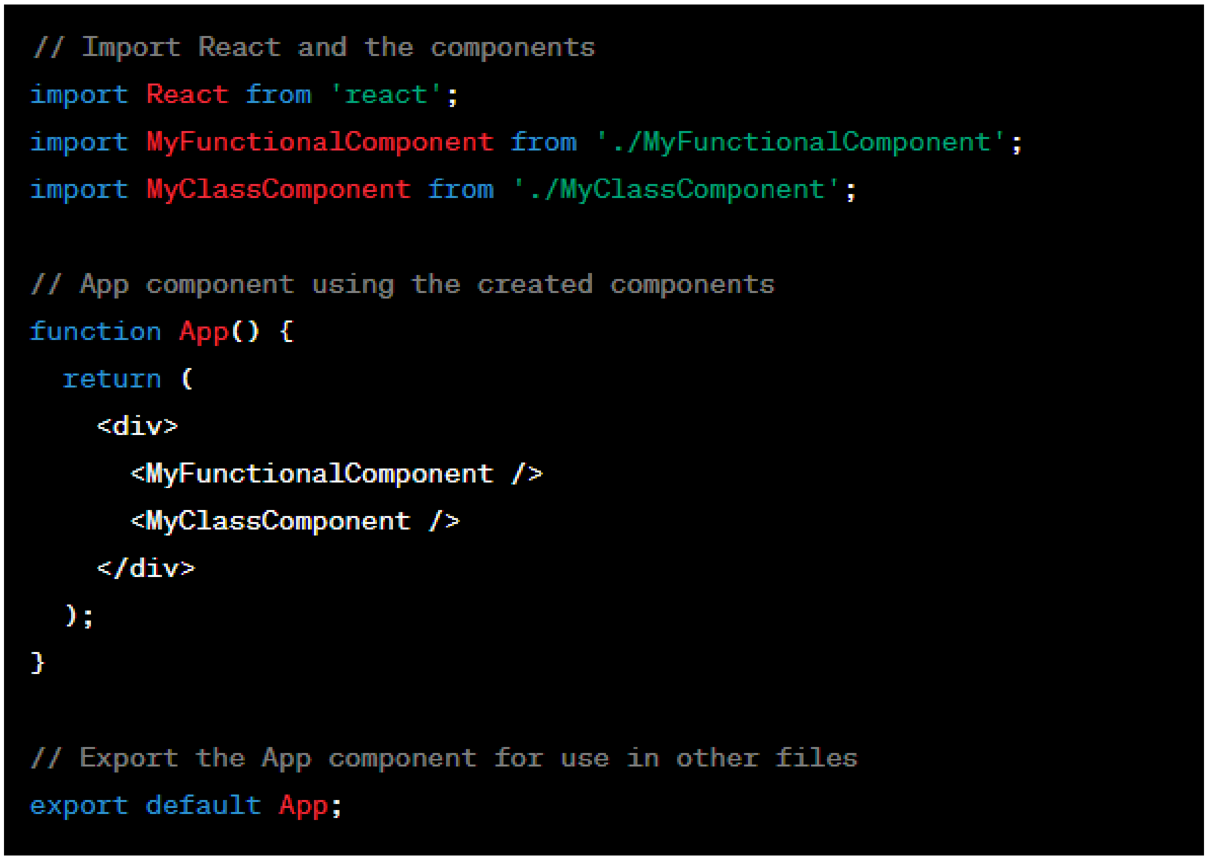
Class components are stateful and can have additional features like lifecycle methods. They are used when you need to manage and manipulate state within a component.



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

// Define a class component
class MyClassComponent extends Component {
  // Constructor for initializing state or bindings
  constructor(props) {
    super(props);
    this.state = {
      message: 'Hello, I am a Class Component!',
    };
  }

  // Render method to define the component's UI
  render() {
    return (
      <div>
        <h1>{this.state.message}</h1>
      </div>
    );
  }
}

// Export the component for use in other files
export default MyClassComponent;
```

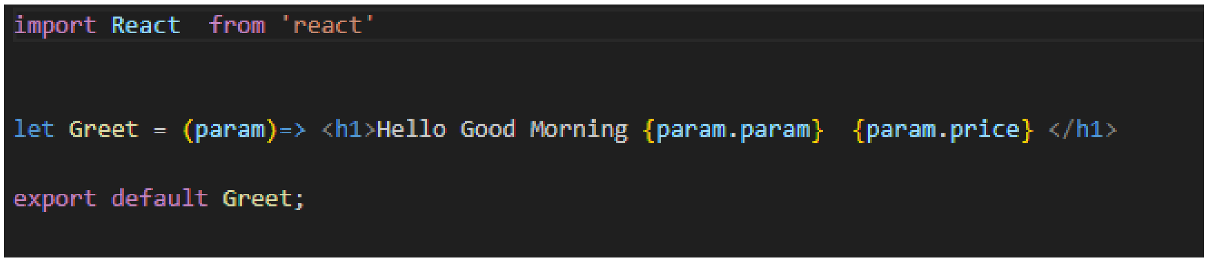
Using Components

Once you've created a component, you can use it in other parts of your application.

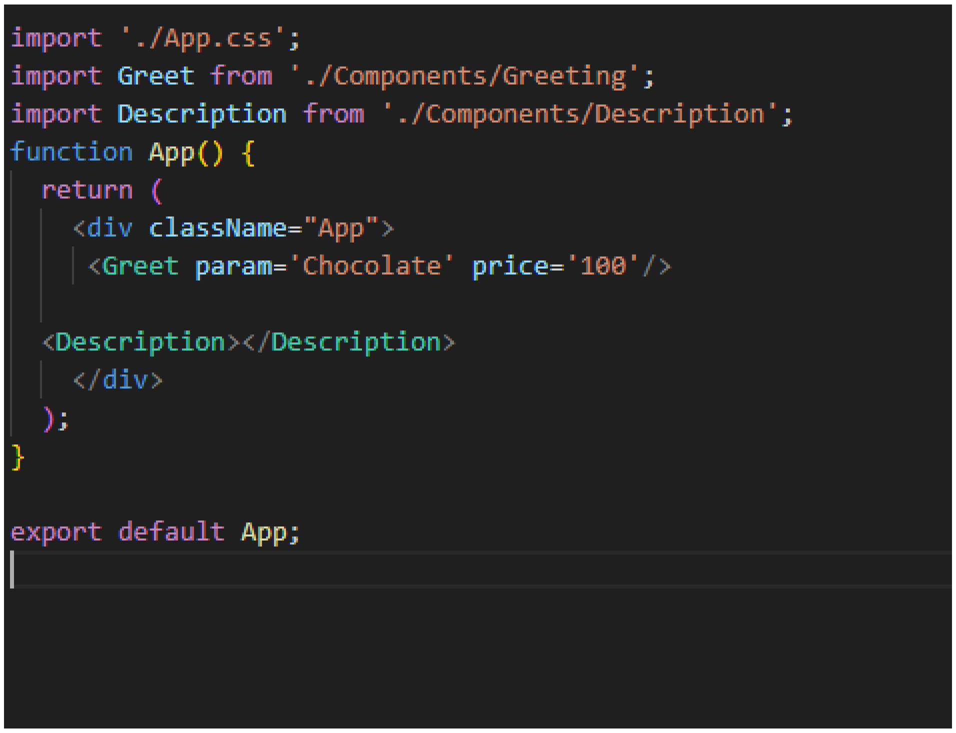


```
// Import React and the components
import React from 'react';
import MyFunctionalComponent from './MyFunctionalComponent';
import MyClassComponent from './MyClassComponent';

// App component using the created components
function App() {
  return (
    <div>
      <MyFunctionalComponent />
      <MyClassComponent />
    </div>
  );
}

// Export the App component for use in other files
export default App;
```

Through Props passing: (Detail props understanding is understand below)



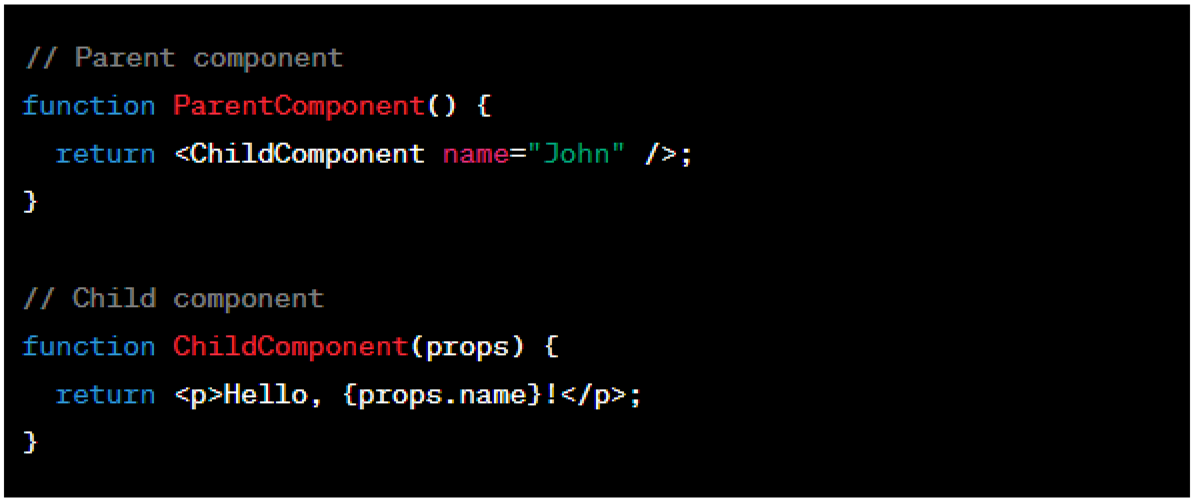
```
import React  from 'react'


let Greet = (param)=> <h1>Hello Good Morning {param.param}  {param.price} </h1>

export default Greet;
```



```
import './App.css';
import Greet from './Components/Greeting';
import Description from './Components/Description';
function App() {
  return (
    <div className="App">
     <Greet param='Chocolate' price='100'/>

  <Description></Description>
    </div>
  );
}

export default App;
```

Props:

- ✓ Props (short for properties) are used to pass data from a parent component to a child component in React. They enable communication between components.

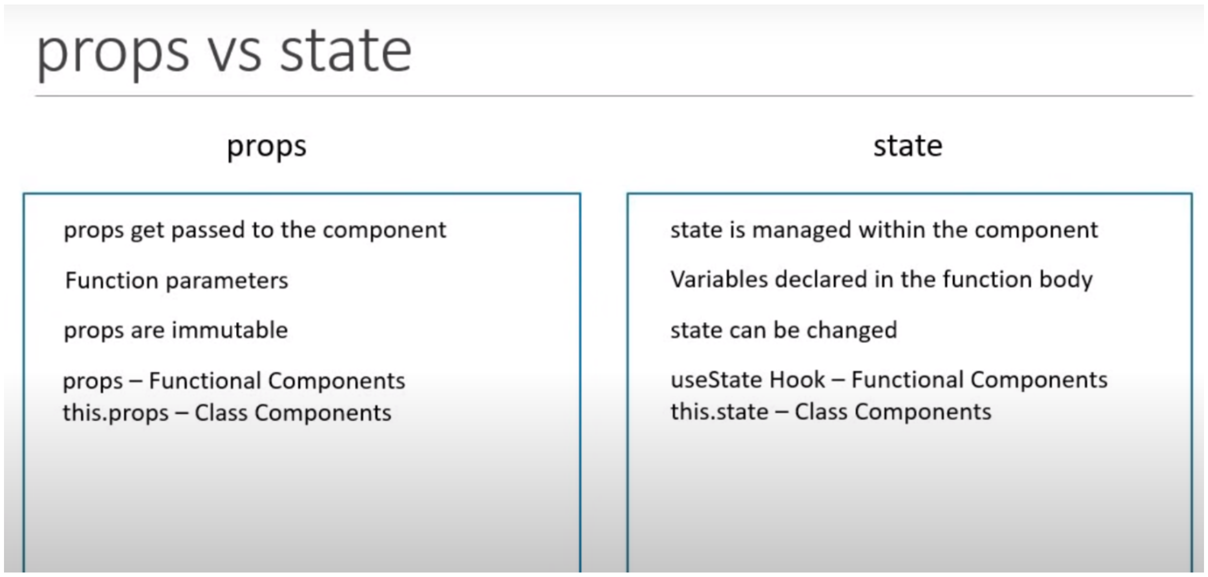


```
// Parent component
function ParentComponent() {
  return <ChildComponent name="John" />;
}

// Child component
function ChildComponent(props) {
  return <p>Hello, {props.name}!</p>;
}
```

State

State is internal data management within a component. It allows components to dynamically update and render UI based on changes.

```
// Class component with state
class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  // Increment count
  incrementCount = () => {
    this.setState({ count: this.state.count + 1 });
  };

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={this.incrementCount}>Increment</button>
      </div>
    );
  }
}
```



props vs state

props	state
props get passed to the component	state is managed within the component
Function parameters	Variables declared in the function body
props are immutable	state can be changed
props – Functional Components this.props – Class Components	useState Hook – Functional Components this.state – Class Components

Lifecycle Methods

- ✓ Class components in React have lifecycle methods, such as `componentDidMount` and `componentWillUnmount`, useful for managing side effects.

Hooks

- ✓ Hooks are functions that enable functional components to have state and lifecycle features.

1. The **useState** hook allows functional components to manage state. It returns an array with two elements: the current state value and a function to update it

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

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

2. The `useEffect` hook is used for side effects in functional components, such as data fetching, subscriptions, or manually changing the DOM. It runs after every render.

```
import React, { useState, useEffect } from 'react';

function ExampleComponent() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // Fetch data or perform other side effects here
    fetchData().then((result) => setData(result));
  }, []); // Empty dependency array means this effect runs once

  return <div>{data && <p>Data: {data}</p>}</div>;
}
```

3. The `useContext` hook enables functional components to consume values from a React context without introducing a context consumer.

```
import React, { useContext } from 'react';
import MyContext from './MyContext';

function MyComponent() {
  const contextValue = useContext(MyContext);

  return <p>Context Value: {contextValue}</p>;
}
```

4. The `useReducer` hook is an alternative to `useState` for managing complex state logic. It takes a reducer function and the initial state.

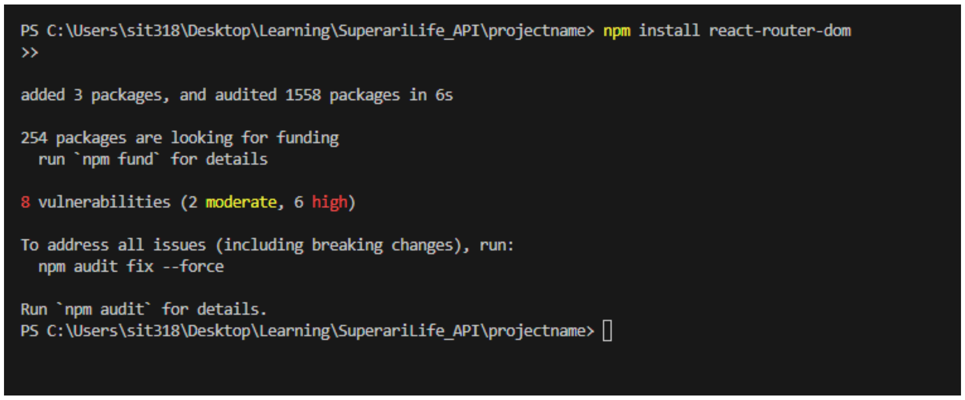
```
import React, { useReducer } from 'react';

function counterReducer(state, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    default:
      return state;
  }
}

function Counter() {
  const [state, dispatch] = useReducer(counterReducer, { count: 0 });

  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'INCREMENT' })}>Incremen
    </div>
  );
}
```

➢ React Router :

React Router is a standard library for routing in React applications. It enables navigation between different components without a page refresh.

- Installation

Run the following command to install React Router:



```
PS C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname> npm install react-router-dom
>>

added 3 packages, and audited 1558 packages in 6s

254 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\sit318\Desktop\Learning\SuperariLife_API\projectname> 
```

Wrap your app with `Browser Router` and use `Route` components to define routes.

Example:

```
import { BrowserRouter as Router, Route } from 'react-router-dom';

function App() {
  return (
    <Router>
      <Route path="/" exact component={Home} />
      <Route path="/about" component={About} />
    </Router>
  );
}
```

Best Practices:

Following best practices is crucial for writing clean and maintainable React code:

- ✓ Component Structure: Keep components small and focused on one task.
- ✓ State Management: Lift state up when needed and use state management tools for complex applications.
- ✓ Prop Types: Use Prop Types to specify the type of each prop a component receives.
- ✓ Functional Components: Prefer functional components over class components, and use hooks for state management.

DEMO Project (Coupon CRUD)

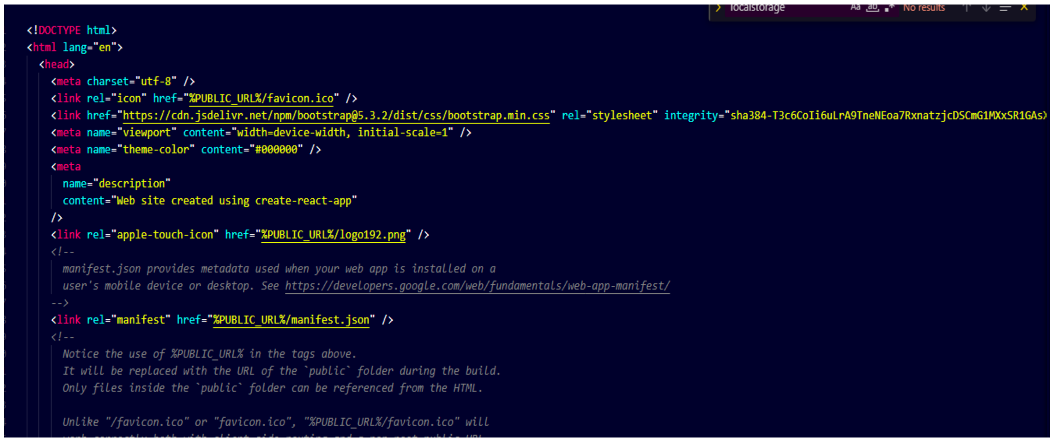
Prerequisites:

- ✓ **API For Call**
- ✓ **Database**

Steps:

I. **Project creation and set up is mentioned above for react.js**

II. **Added CDN link for bootstrap in index.html file.**

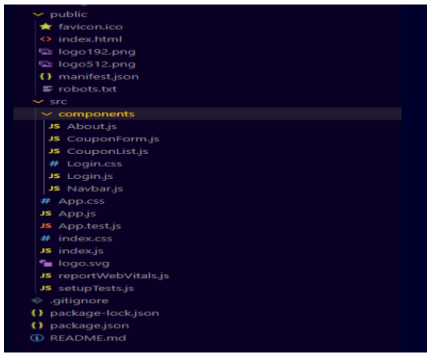


```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAs>
    <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" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      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
```

Link for CDN : Bootstrap

III. **Install react-router-dom (For download check above in key concept).**

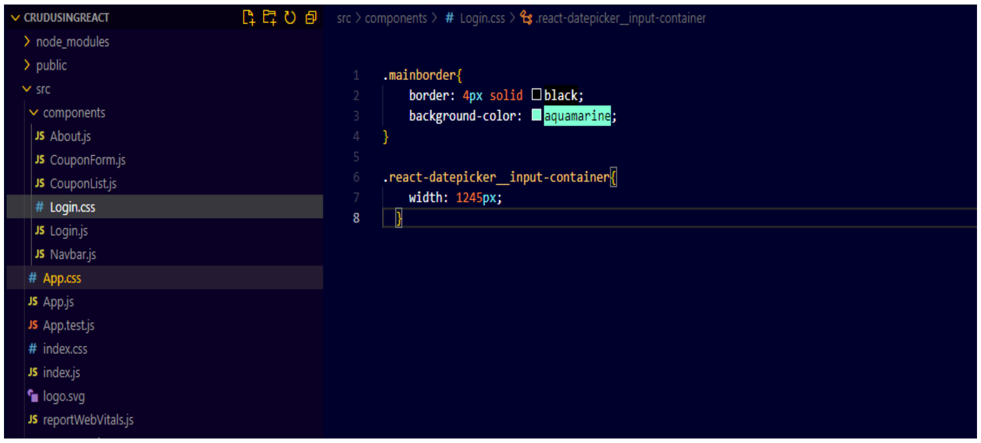
IV. **Project Structure of demo**



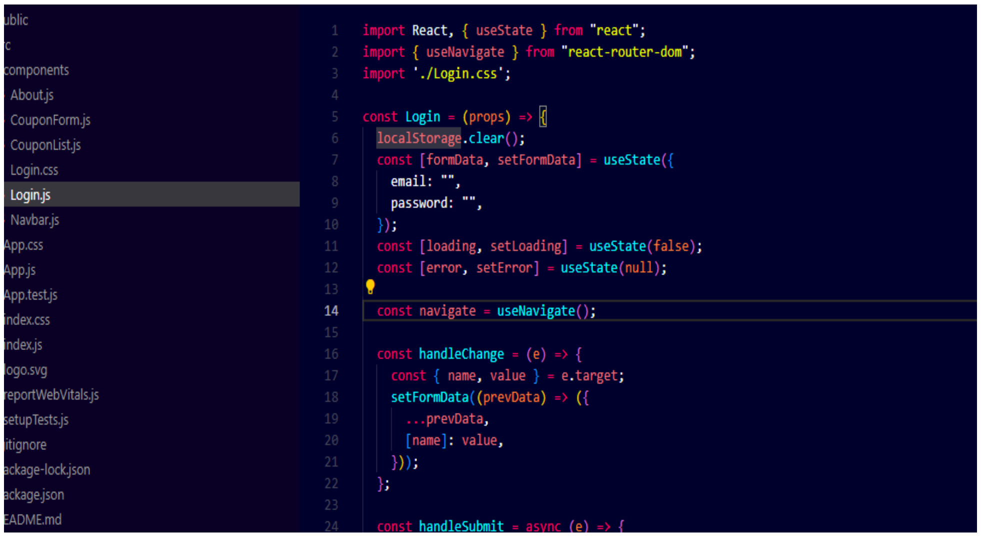
```
public
  favicon.ico
  index.html
  logo192.png
  logo512.png
  manifest.json
  robots.txt
src
  components
    About.js
    CouponForm.js
    CouponList.js
    Login.css
    Login.js
    Navbar.js
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
.gitignore
package-lock.json
package.json
README.md
```

V. **Created Component folder added required .js file by click on add file option in VS code with extension .js.**

VI. For Custom css for particular component you can made xyz.css file for xyz.js file.

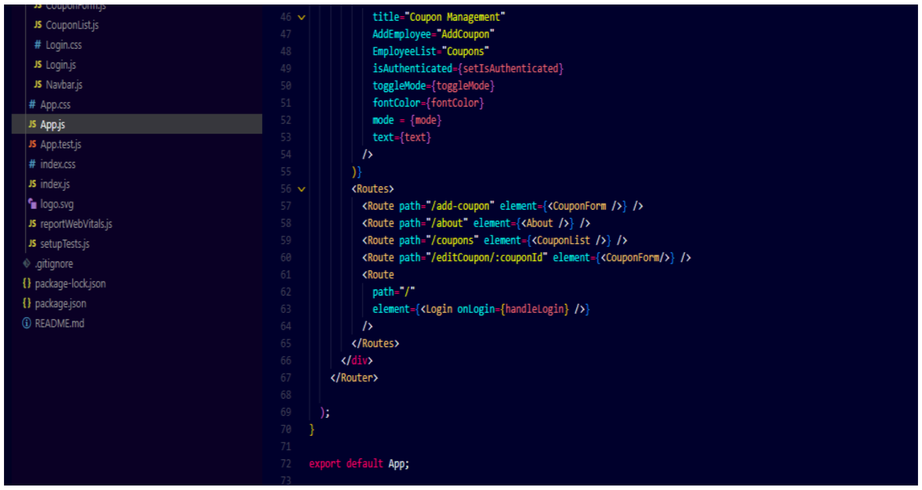


```css
.mainborder{
    border: 4px solid black;
    background-color: aquamarine;
}

.react-datepicker__input-container{
    width: 1245px;
}
```

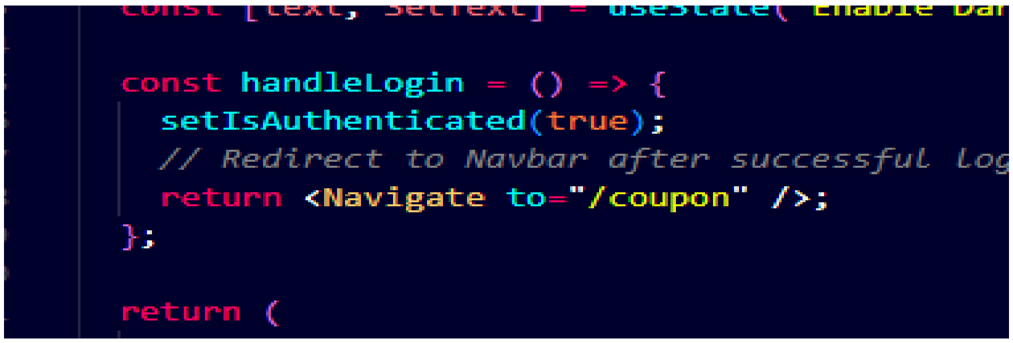


```javascript
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import './Login.css';

const Login = (props) => {
  localStorage.clear();
  const [formData, setFormData] = useState({
    email: "",
    password: "",
  });
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  const navigate = useNavigate();

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({
      ...prevData,
      [name]: value,
    }));
  };

  const handleSubmit = async (e) => {
```

VII. Routing implementation and calling:

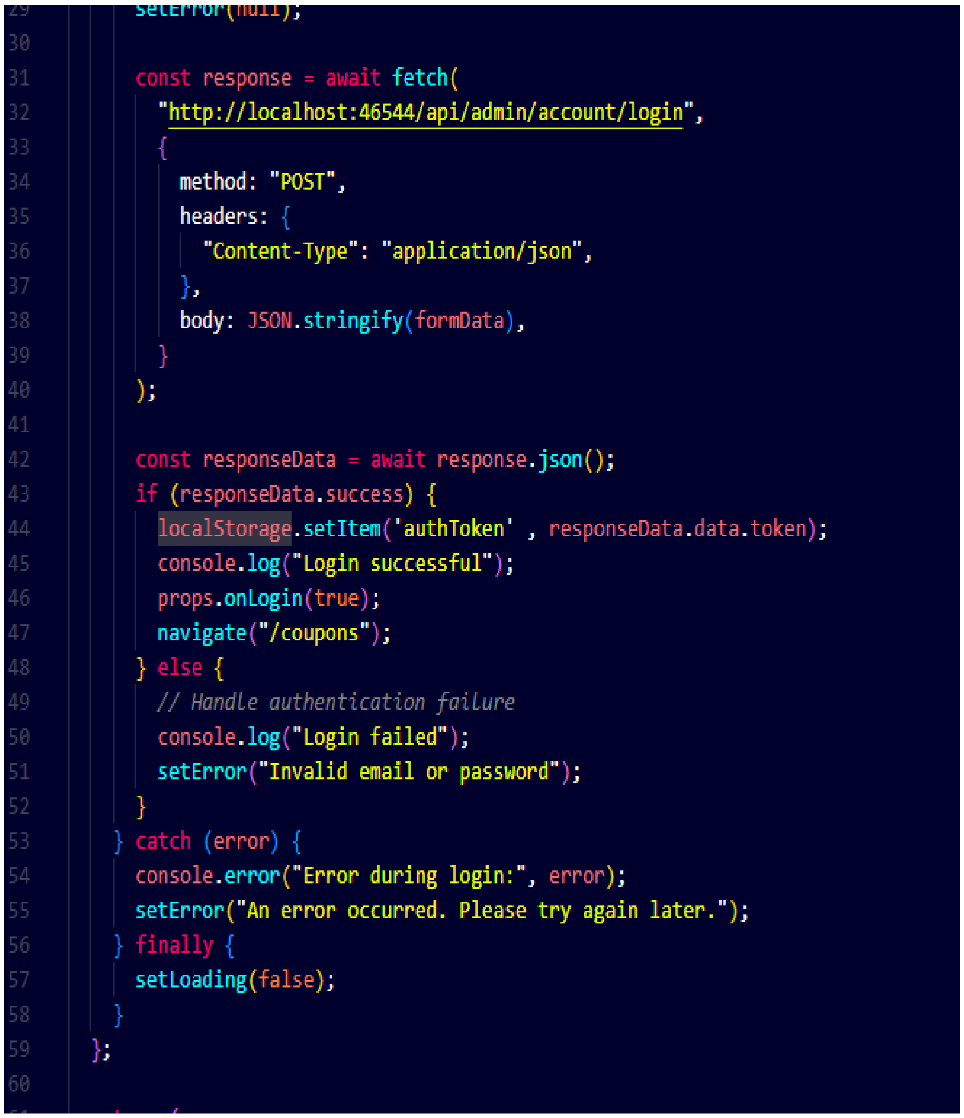


```jsx
            title="Coupon Management"
            AddEmployee="AddCoupon"
            EmployeeList="Coupons"
            isAuthenticated={setIsAuthenticated}
            toggleMode={toggleMode}
            fontColor={fontColor}
            mode = {mode}
            text={text}
          />
        )}
        <Routes>
          <Route path="/add-coupon" element={<CouponForm />} />
          <Route path="/about" element={<About />} />
          <Route path="/coupons" element={<CouponList />} />
          <Route path="/editCoupon/:couponId" element={<CouponForm/>} />
          <Route
            path="/"
            element={<Login onLogin={handleLogin} />}
          />
        </Routes>
      </div>
    </Router>

  );
}

export default App;
```



```
  const handleLogin = () => {
    setIsAuthenticated(true);
    // Redirect to Navbar after successful log
    return <Navigate to="/coupon" />;
  };

  return (
```

Service Call:



```
      setError(null);

      const response = await fetch(
        "http://localhost:46544/api/admin/account/login",
        {
          method: "POST",
          headers: {
            "Content-Type": "application/json",
          },
          body: JSON.stringify(formData),
        }
      );

      const responseData = await response.json();
      if (responseData.success) {
        localStorage.setItem('authToken' , responseData.data.token);
        console.log("Login successful");
        props.onLogin(true);
        navigate("/coupons");
      } else {
        // Handle authentication failure
        console.log("Login failed");
        setError("Invalid email or password");
      }
    } catch (error) {
      console.error("Error during login:", error);
      setError("An error occurred. Please try again later.");
    } finally {
      setLoading(false);
    }
  };
```

For Demo Link Project Link: [Project Demo]

➢ Resources

Explore additional resources to deepen your understanding of React.js:

- React Documentation(https://reactjs.org/docs/getting-started.html)
- FreeCodeCamp React.js Tutorial(https://www.youtube.com/watch?v=DLX62G4lc44)
- React GitHub Repository(https://github.com/facebook/react)
- React Podcast(https://reactpodcast.simplecast.com/)