Introduction to React JS | Cheat Sheet

Concepts in Focus

- React JS
- Why React JS?
- Advantages of React JS
- Running JavaSript in HTML
- Creating Elements using React JS
- React CDN
- React.createElement()
- ReactDOM.render()
- JSX
- Babel
- Embedding Variables and Expressions in JSX
- Nesting JSX elements

1. React JS

React JS is an open-source JavaScript library used to build user interfaces. It was developed by Facebook.

1.1 Why React JS?

- Performant websites
- Fewer lines of code
- Improves readability of code
- Less time consuming
- Open Source
- Reusable code

1.2 Advantages of React JS

- Easy to Learn
- Large Community
- Developer Toolset

2. Running JavaScript in HTML

We can run JavaScript in HTML using the HTML script element. It is used to include JavaScript in HTML.

```
<body>
  <div id="root"></div>
  <script type="text/javascript">
```

```
    const rootElement = document.getElementById("root");
    const element = document.createElement("h1");
    element.textContent = "Hello World!";
    element.classList.add("greeting");
    rootElement.appendChild(element);
  </script>
</body>
```

Here the type attribute specifies the type of the script.

To include an external JavaScript file, we can use the HTML script element with the attribute src. The src attribute specifies the path of an external JS file.

```
<script type="text/javascript" src="PATH_TO_JS_FILE.js"></script>
```

Note:
When the browser comes across a script element while loading the HTML, it must wait for the script to download, execute it, and only then can it process the rest of the page.

So, we need to put a script element at the bottom of the page. Then the browser can see elements above it and doesn't block the page content from showing.

If more than one script elements are in the HTML, the script elements will be executed in the order they appear.

3. Creating Elements using React JS

3.1 React CDN

```
<script
src="https://unpkg.com/react@17.0.0/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@17.0.0/umd/react-
dom.development.js"></script>
<script   src="https://unpkg.com/@babel/standalone@7.12.4/babel.js"></script>
```

3.2 React.createElement()

The React.createElement() method is used to create an element using React JS. It is similar to the document.createElement() method in regular JavaScript.
Syntax:

React.createElement(type, props);

type - Tag names like div, h1 and p, etc.
props - Properties like className, onClick and id, etc.

```
<script type="module">
```

```
  const elementProps = { className: "greeting", children: "Hello world!" };
  const elementType = "h1";
  const element = React.createElement(elementType, elementProps);
</script>
```

Note:

The `type` attribute value of the HTML `script` element should be `module` to ru React JS.

3.3 ReactDOM.render()

The ReactDOM.render() method is used to display the React element.

Syntax:

```
ReactDOM.render(reactElement, container);
```

reactElement - What to render
container - Where to render

```
<body>
  <div id="root"></div>
  <script type="module">
    const elementProps = { className: "greeting", children: "Hello world!" };
    const elementType = "h1";
    const element = React.createElement(elementType, elementProps);
    ReactDOM.render(element, document.getElementById("root"));
  </script>
</body>
```

4. JSX

React JS introduced a new HTML like syntax named JSX to create elements.

```
const element = <h1 className="greeting">Hello World</h1>;
```

The above JSX element compiles to,

```
const elementProps = { className: "greeting", children: "Hello world!" };

const element = React.createElement("h1", elementProps);
```

Warning:

In JSX, HTML tags always need to be closed. For example, `<br />`, `<img />`.

4.1 Babel

JSX is not JavaScript. We have to convert it to JavaScript using a code compiler. Babel is one such tool.

It is a JavaScript compiler that translates JSX into regular JavaScript.

```
<script type="text/babel">
  const elementProps = { className: "greeting", children: "Hello world!" };
  const element = React.createElement("h1", elementProps);
  const element = <h1 className="greeting">Hello World</h1>;
  ReactDOM.render(element, document.getElementById("root"));
</script>
```

Note:

For JSX, the type attribute value of the HTML script element should be text/babel

For providing class names in JSX, the attribute name should be className

Differences between HTML and JSX:

HTML	JSX
class	className
for	htmlFor

4.2 Embedding Variables and Expressions in JSX

We can embed the variables and expressions using the flower brackets
{}
.

Embedding variables in JSX:

```
<body>
  <div id="root"></div>
  <script type="text/babel">
    const name = "Rahul";
    const className = "greeting";
    const element = <h1 className="greeting">Hello World</h1>;
    const element = <h1 className={className}>Hello {name}!</h1>;
    ReactDOM.render(element, document.getElementById("root"));
  </script>
</body>
```

Embedding Expressions in JSX:

```
<body>
  <div id="root"></div>
  <script type="text/babel">
    const fullName = (user) => user.firstName + " " + user.lastName;
    const user = { firstName: "Rahul ", lastName: "Attuluri" };
    const element = <h1 className="greeting"> Hello, {fullName(user)}!</h1>;
    ReactDOM.render(element, document.getElementById("root"));
  </script>
</body>
```

4.3 Nesting JSX elements

```
<body>
  <script type="text/babel">
    const element = (
      <div>
        <h1 className="greeting">Hello!</h1>
        <p>Good to see you here.</p>
      </div>
    );
    ReactDOM.render(element, document.getElementById("root"));
  </script>
</body>
```

Components & Props

Concepts in Focus

- Component
  - Properties (Props)
  - Component is Reusable
  - Component is Composable
- Third-party Packages
  - create-react-app
  - Pre-Configured tools

1. Component

A Component is a JS function that returns a JSX element.

```
const Welcome = () => <h1 className="message">Hello, User</h1>;
```

The component name should always start with a capital letter as react treats the components starting with lowercase letters as HTML elements.

We can call the function with self-closing tags as shown above <Welcome />.

1.1 Properties (Props)

React allows us to pass information to a component using props.

1.1.1 Passing Props

We can pass props to any component as we declare attributes for any HTML element.

Syntax:

```
<Component propName1="propValue1" propName2="propValue2" />
```

```
const Welcome = () => <h1 className="message">Hello, User</h1>;
ReactDOM.render(
  <Welcome name="Rahul" greeting="Hello" />,
  document.getElementById("root")
);
```

1.1.2 Accessing Props

The components accept props as parameters and can be accessed directly.

Syntax:

```
const Component = (props) => {
  // We can access props here
};
const Welcome = (props) => {
  const { name, greeting } = props;
  return (
    <h1 className="message">
      {greeting}, {name}
    </h1>
  );
};

ReactDOM.render(
```

```
  <Welcome name="Rahul" greeting="Hello" />,
  document.getElementById("root")
);
```

1.2 Component is Reusable

A Component is a piece of reusable code that can be used in various parts of an application.

Example:

```
const Welcome = (props) => {
  const { name, greeting } = props;
  return (
    <h1 className="message">
      {greeting}, {name}
    </h1>
  );
};

ReactDOM.render(
  <div>
    <Welcome name="Rahul" greeting="Hello" />
    <Welcome name="Ram" greeting="Hi" />
  </div>,
  document.getElementById("root")
);
```

1.3 Component is Composable

We can include a component inside another component.

```
const Welcome = (props) => {
  const { name, greeting } = props;
  return (
    <h1 className="message">
      {greeting}, {name}
    </h1>
  );
```

```
};

const Greetings = () => (
  <div>
    <Welcome name="Rahul" greeting="Hello" />
    <Welcome name="Ram" greeting="Hi" />
  </div>
);


ReactDOM.render(<Greetings />, document.getElementById("root"));
```

2. Third-party Packages

Creating a real-world app involves lot of setup because a large number of components need to be organised.

Facebook has created a third-party package,
create-react-app
, to generate a ready-made React application setup.

2.1 create-react-app

Installation Command:

npm install -g create-react-app

It installs create-react-app globally in our environment.

2.1.1 Creating a React Application

create-react-app myapp –use-npm

2.1.2 React Application Folder Structure

- public/folder: Where we will keep assets like images, icons, videos etc
- src/folder: Where we will do the majority of our work. All of our React components will placed here.
- node_modules
- package-lock.json

node_modules:

This directory contains dependencies and sub-dependencies of packages used by the current react app, as specified by package.json.

package-lock.json:

This file contains the exact dependency tree installed in node_modules. This provides a way to ensure every team member have the same version of dependencies and sub-dependencies.

The index.js in the path src/folder/ is a starting point to the application. App.js, App.css are imported in this file.

2.1.3 Starting a React Application

Run the below command from the React application directory.

npm start

Note:

All the ES6 Modules should be named with .js extension.

2.2 Pre-Configured tools

The create-react-app comes pre-configured with:

- **Live editing**: Allows React components to be live reloaded.
- **ESLint**: Analyzes source code to report programming errors, bugs, and syntax errors.
- **Prettier**: Enforces a consistent style for indentation, spacing, semicolons and quotes, etc.
- **Babel**: Compiles JSX into Regular JavaScript
- **Webpack**: Stitches together a group of modules into a single file (or group of files). This process is called Bundling.

Lists & Keys:

Concepts in Focus

1. Keys

Keys help React to identify which items have changed, added, or removed. They should be given to the elements inside the array for a stable identity.
The best way to pick a key is to use a string that uniquely identifies a list item among its siblings. Most often, we would use IDs (uniqueNo) from our data as keys.
Example:

```
const userDetails = [
  {
    uniqueNo: 1,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/esther-howard-img.png',
    name: 'Esther Howard',
    role: 'Software Developer'
  }
]
```

File: src/App.js

```
import UserProfile from './components/UserProfile/index'
const userDetailsList = [
  {
    uniqueNo: 1,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/esther-howard-img.png',
    name: 'Esther Howard',
    role: 'Software Developer'
  },
  {
    uniqueNo: 2,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/floyd-miles-img.png',
    name: 'Floyd Miles',
    role: 'Software Developer'
  },
  {
    uniqueNo: 3,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/jacob-jones-img.png',
    name: 'Jacob Jones',
    role: 'Software Developer'
  },
  {
    uniqueNo: 4,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/esther-devon-lane.png',
    name: 'Devon Lane',
    role: 'Software Developer'
  }
]

const App = () => (
  <div className="list-container">
    <h1 className="title">Users List</h1>
    <ul>
      {userDetailsList.map((eachItem) => (
        <UserProfile userDetails={eachItem} />
      ))}
    </ul>
  </div>
)
export default App
```

The index.js file in the folder is considered to be the main file in it. So, a path like ./components/UserProfile can be used instead of ./components/UserProfile/index

Note:

Keys used within arrays should be unique among their siblings. However, they don't need to be unique in the entire application.

1.1 Keys as Props

Keys don't get passed as a prop to the components.

```
const UserProfile = props => {
  const {userDetails} = props
  const {imageUrl, name, role, key} = userDetails
  console.log(key)  // undefined

  return (
    <li className="user-card-container">
      <img src={imageUrl} className="avatar" alt="avatar" />
      <div className="user-details-container">
        <h1 className="user-name"> {name} </h1>
        <p className="user-designation"> {role} </p>
      </div>
    </li>
  )
}
export default UserProfile
```

If we need the same value in the component, pass it explicitly as a prop with a different name.

Example:

```
const UsersList = userDetailsList.map((userDetails) =>
  <UserProfile
    key={userDetails.uniqueNo}
    uniqueNo={userDetails.uniqueNo}
    name={userDetails.name} />
);
```

2. Users List Application

File: src/App.js

```
import UserProfile from './components/UserProfile/index'

const userDetailsList = [
  {
    uniqueNo: 1,
    imageUrl:
```

```
      'https://assets.ccbp.in/frontend/react-js/esther-howard-img.png',
    name: 'Esther Howard',
    role: 'Software Developer'
  },
  {
    uniqueNo: 2,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/floyd-miles-img.png',
    name: 'Floyd Miles',
    role: 'Software Developer'
  },
  {
    uniqueNo: 3,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/jacob-jones-img.png',
    name: 'Jacob Jones',
    role: 'Software Developer'
  },
  {
    uniqueNo: 4,
    imageUrl:
      'https://assets.ccbp.in/frontend/react-js/esther-devon-lane.png',
    name: 'Devon Lane',
    role: 'Software Developer'
  }
]

const App = () => (
  <div className="list-container">
    <h1 className="title">Users List</h1>
    <ul>
      {userDetailsList.map((eachItem) => (
```

```
        <UserProfile userDetails={eachItem} />
      ))}
    </ul>
  </div>
)
export default App
```

File: src/components/UserProfile/index.js

```
import './index.css'

const UserProfile = props => {
  const {userDetails} = props
  const {imageUrl, name, role} = userDetails

  return (
    <li className="user-card-container">
      <img src={imageUrl} className="avatar" alt="avatar" />
      <div className="user-details-container">
        <h1 className="user-name"> {name} </h1>
        <p className="user-designation"> {role} </p>
      </div>
    </li>
  )
}
export default UserProfile.
```

Note:

React Error - ENOSPC: System limit for the number of file watchers reached

Since **create-react-app** live-reloads and recompiles files on save, it needs to keep track of all project files.

To fix the error, run the below commands in the terminal:

1. Insert the new value into the system config

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p
```

2. Check that the new value was applied

cat /proc/sys/fs/inotify/max_user_watches

3. Config variable name (not runnable)

fs.inotify.max_user_watches=524288

Class Component and State

Concepts in Focus

1. Components

There are two ways to write React Components.

They are:

- Functional Components
- Class Components

1.1 Functional Components

These are JavaScript functions that take props as a parameter if necessary and return react element (JSX).

```
const Welcome = () => <h1>Hello, User</h1>;

export default Welcome;
```

1.2 Class Components

These components are built using an ES6 class.

To define a React Class Component,

- Create an ES6 class that extends React.Component
  .
- Add a single empty method to it called render()
  .

1.2.1 extends

The extends keyword is used to inherit methods and properties from the React.Component
.

1.2.2 render()

The render() method is the only required method in a class component. It returns the JSX element.

Syntax:

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


class MyComponent extends Component {
  render() {
    return JSX;
  }
}
```

Use this.props in the render() body to access the props in a class component.

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

Note: The component name should always be in the pascal case.

2. React Events

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

1. React events are named using **camelCase**, rather than **lowercase**.

Example:

HTML	JSX
onclick	onClick
onblur	onBlur
onchange	onChange

2. With JSX, you pass a **function** as the event handler rather than a **string**.

Example:

```
<button onclick="activateLasers()">Activate Lasers</button>
```

```
<button onClick={activateLasers}>Activate Lasers</button>
```

We should not call the function when we add an event in JSX.

```
class MyComponent extends Component {
  handleClick = () => {
    console.log("clicked")
  }
  render() {
    return <button onClick={this.handleClick()}>Click Me</button>
  }
}
```

In the above function, the handleClick is called instead of passed as a reference.

```
class MyComponent extends Component {
  handleClick = () => {
    console.log("clicked")
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>
  }
}
```

In the above function, the handleClick is passed as a reference. So, the function is not being called every time the component renders.

Providing Arrow Functions

To not change the context of this, we have to pass an arrow function to the event.

```
class MyComponent extends Component {
  handleClick() {
    console.log(this)  // undefined
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>
  }
}
```

```
class MyComponent extends Component {
  handleClick = () => {
    console.log(this)  // MyComponent {...}
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>
  }
}
```

3. State

The state is a JS object in which we store the component's data that changes over time.

When the state object changes, the component re-renders.

Intialising State:

```
class Counter extends Component {
  state = { count: 0 }
  render() {
    const { count } = this.state;
    return <p className="count">{count}</p>;
  }
}
```

3.1 Updating State

We can update the state by using
setState()
. We can provide function/object as an argument to set the state.

Providing Function as an Argument:

Syntax:

this.setState(prevState => ({... }))

Here the previous state is sent as a parameter to the callback function.

```
onIncrement = () => {
  this.setState((prevState) =>
    console.log(`previous state value ${prevState.count}`)
  )
}
```

3.2 State Updates are Merged

State updates are merged. It means that when you update only one key-value pair in the state object, it will not affect the other key-value pairs in the state object.

```
// For example let's say your state is as followed:
state = { key1 : value1, key2 : value2 }

// if you use this.setState such as :
this.setState((prevState) => ({ prevState.key1 : value3 }))

// your new state will be :
state = { key1 : value3, key2 : value2 }
```

3.3 Functional Components vs Class Components

Functional Components	Class Components
Renders the UI based on props	Renders the UI based on props and state

Use Class Components whenever the state is required. Otherwise, use the Functional components.

4. Counter Application

File: src/App.js

```
import Counter from "./components/Counter";

const App = () => {
  return <Counter />
}

export default App;
```

File: src/components/Counter/index.js

```
import { Component } from "react"

import "./index.css"

class Counter extends Component {
  state = { count: 0 }
  onIncrement = () => {
    this.setState((prevState) => ({ count: prevState.count + 1 }))
  }
  onDecrement = () => {
    this.setState((prevState) => ({ count: prevState.count - 1 }))
  }
  render() {
    const { count } = this.state
    return (
      <div className="container">
        <h1 className="count">Count {count}</h1>
        <button className="button" onClick={this.onIncrement}>
          Increase
        </button>
        <button className="button" onClick={this.onDecrement}>
          Decrease
        </button>
      </div>
```

```
    )
  }
}

export default Counter
```

Conditional Rendering:

Concepts in Focus

- Conditional Rendering
    - Using an If...Else Statement
    - Using Element Variables
    - Using Ternary Operators
    - Using Logical && Operator
- Default Props

1 . Conditional Rendering

Conditional Rendering allows us to render different elements or components based on a condition.

Different ways to implement **Conditional Rendering** are:

- Using an If...Else Statement
- Using Element Variables
- Using Ternary Operators
- Using Logical && Operator

1.1 Using an If...Else Statement

```
import { Component } from "react"
import './App.css'

class App extends Component {
  state = { isLoggedIn: true }

   renderAuthButton = () => {
    const {isLoggedIn} = this.state
    if (isLoggedIn === true) {
      return <button>Logout</button>
    }
    return <button>Login</button>
  }

  render() {
    return (
     <div className="container">
```

```
        {this.renderAuthButton()}
      </div>
    )
  }
}

export default App
```

1.2 Using Element Variables

```
import { Component } from "react"
import './App.css'


class App extends Component {
  state = { isLoggedIn: true }


  render() {
    const { isLoggedIn } = this.state
    let authButton
    if (isLoggedIn) {
      authButton = <button>Logout</button>
    } else {
      authButton = <button>Login</button>
    }
    return (
      <div className="container">
        <h1>React JS</h1>
        {authButton}
      </div>
    )
  }
}
```

```
export default App
```

1.3 Using Ternary Operators

```
import { Component } from "react"
import './App.css'

class App extends Component {

  render() {
    const { isLoggedIn } = this.state
    return (
      <div className="container">
        {isLoggedIn ? <button>Logout</button> : <button>Login</button>}
      </div>
    )
  }
}

export default App
```

1.4 Using Logical && Operator

```
import { Component } from "react"

import './App.css'


class App extends Component {


  render() {

    const { isLoggedIn } = this.state

    return (

      <div className="container">

        {isLoggedIn && <button>Logout</button>}

        {!isLoggedIn && <button>Login</button>}

      </div>

    )
```

```
  }
}

export default App
```

Note: Conditional Rendering can be achieved using inline styles or adding classes with CSS `display` property with value `none`. However, it is not preferable.

2. Default Props

defaultProps is a property in React Component used to set default values for the props. This is similar to adding default parameters to the function.

Syntax:

```
// Component Definition

ComponentName.defaultProps = {
  propName1: "propValue1",
  propName2: "propValue2"
}

// Exporting Component
```

Example:

File: src/Welcome/index.js

```
const Welcome = (props) => {
  const { name, greeting } = props;
  return (
    <h1 className="message">
      {greeting}, {name}
    </h1>
  );
};
```

```
Welcome.defaultProps = {
  name: "Rahul",
  greeting: "Hello"
};

export default Welcome;
```

File: src/App.js

```
import { Component } from "react";
import Welcome from "./Welcome";

class App extends Component {
  state = { isLoggedIn: true };
  render() {
    const { isLoggedIn } = this.state;
    return (
      <div className="container">
        <Welcome greeting="Hello" />
      </div>
    );
  }
}

export default App;
```

Note: While accessing the props, the correct prop name should be given.

Class Component and State | Part 2

Concepts in Focus

- setState()
  Object Syntax

1. setState() Object Syntax

The setState() object syntax can be used while updating the state to the value that is independent of the previous state.

Syntax:

```
this.setState(
  {propertyName1: propertyValue1},
  {propertyName2: propertyValue2}
  // and many more...
);
```

1.1 Callback vs Object

Callback:

```
this.setState(prevState => {
  return { count: prevState.count + 1 };
});
```

It is used while updating the state to a value, which is computed based on the previous state.

Object:

```
this.setState({ quantity: 2 });
```

It is used while updating the state to a static value.

2. Sending Function as Callback

We can pass functions as props to child components.

Syntax:

```
<ComponentName functionName={this.functionName} />
```

3. Input Element

In React, the Input Element
value
can be handled in two ways:

- Controlled Input
- Uncontrolled Input

3.1 Controlled Input

If the Input Element value is handled by a React State then it is called **Controlled Input**. Controlled Inputs are the React Suggested way to handle Input Element value.
Example:

```
import {Component} from 'react'


class App extends Component {
  state = {
    searchInput: '',
  }


  onChangeSearchInput = event => {
    this.setState({
      searchInput: event.target.value,
    })
  }


  render() {
    const {searchInput} = this.state
    return (
      <input
        type="text"
        onChange={this.onChangeSearchInput}
        value={searchInput}
      />
    )
  }
}
```

export default App

3.2 Uncontrolled Input

If the Input Element value is handled by the browser itself then it is called **Uncontrolled Input**. **Uncontrolled inputs** are like traditional HTML form inputs. Its value can only be set by a user, but not programmatically. However, in controlled input value is programmatically handled using React State.

Example:

```
<input type="text" />
```

4. Searchable Users List Application

File: src/App.js

```
import {Component} from 'react'
import UserProfile from './components/UserProfile'

import './App.css'

const initialUserDetailsList = [
  {
    uniqueNo: 1,
    imageUrl: 'https://assets.ccbp.in/frontend/react-js/esther-howard-img.png',
    name: 'Esther Howard',
    role: 'Software Developer'
  },
  {
    uniqueNo: 2,
    imageUrl: 'https://assets.ccbp.in/frontend/react-js/floyd-miles-img.png',
    name: 'Floyd Miles',
    role: 'Software Developer'
  },
  {
    uniqueNo: 3,
    imageUrl: 'https://assets.ccbp.in/frontend/react-js/jacob-jones-img.png',
    name: 'Jacob Jones',
    role: 'Software Developer'
  },
  {
    uniqueNo: 4,
    imageUrl: 'https://assets.ccbp.in/frontend/react-js/devon-lane-img.png',
    name: 'Devon Lane',
    role: 'Software Developer'
```

```
  }
]

class App extends Component {
  state = {
    searchInput: '',
    usersDetailsList: initialUserDetailsList
  }

  onChangeSearchInput = event => {
    this.setState({
      searchInput: event.target.value
    })
  }

  deleteUser = uniqueNo => {
    const {usersDetailsList} = this.state
    const filteredUsersData = usersDetailsList.filter(
      each => each.uniqueNo !== uniqueNo
    )
    this.setState({
      usersDetailsList: filteredUsersData
    })
  }

  render() {
    const {searchInput, usersDetailsList} = this.state
    const searchResults = usersDetailsList.filter(eachUser =>
      eachUser.name.includes(searchInput)
    )
    return (
      <div className="app-container">
        <h1 className="title">Users List</h1>
        <input
          type="search"
          onChange={this.onChangeSearchInput}
          value={searchInput}
        />
        <ul className="list-container">
          {searchResults.map(eachUser => (
            <UserProfile
              userDetails={eachUser}
              key={eachUser.uniqueNo}
              deleteUser={this.deleteUser}
            />
          ))}
```

```
      </ul>
    </div>
  )
 }
}

export default App
```

File: src/components/UserProfile/index.js

```
import './index.css'

const UserProfile = props => {
 const {userDetails, deleteUser} = props
 const {imageUrl, name, role, uniqueNo} = userDetails
 const onDelete = () => {
  deleteUser(uniqueNo)
 }
 return (
  <li className="user-card-container">
   <img src={imageUrl} className="profile-pic" alt="profile-pic" />
   <div className="user-details-container">
    <h1 className="user-name"> {name} </h1>
    <p className="user-designation"> {role} </p>
   </div>
   <button className="delete-button" onClick={onDelete}>
    <img
     src="https://assets.ccbp.in/frontend/react-js/cross-img.png"
     alt="cross"
     className="delete-img"
    />
   </button>
  </li>
 )
}

export default UserProfile
```

Common Mistakes:

Concepts in Focus

- Missing Export Statement
- Missing Import Statement

1. Missing Export Statement

Mistake:

File: src/App.js

```
import Counter from './components/Counter'

const App = () => {
  return <Counter />
}

export default App
```

File: src/components/Counter/index.js

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  render() {
    return (
```

```
      <p className="counter">Counter</p>
    )
  }
}
```

2. Missing Import Statement

Mistake:

File: src/App.js

```
const App = () => {
  return <Counter />
}
export default App
```

File: src/components/Counter/index.js

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  render() {
    return (
      <p className="counter">Counter</p>
    )
  }
}
export default Counter
```

3. Missing Extending the React Component Class

Mistake:

File: src/App.js

```
import Counter from './components/Counter'
```

```
const App = () => {
  return <Counter />
}
export default App
```

File: src/components/Counter/index.js

```
import './index.css'

class Counter {
  render() {
    return (
      <p className="counter">Counter</p>
    )
  }
}
export default Counter
```

4. class vs className

Mistake:

File: src/components/Counter/index.js

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  render() {
    return (
      <p class="counter">Counter</p>
    )
  }
}
export default Counter
```

5. onclick vs onClick

Mistake:

File: src/components/Counter/index.js

```
import { Component } from 'react'
```

```
import './index.css'


class Counter extends Component {
  onIncrement = () => {
    console.log("on increment called")
  }
  render() {
    return (
      <p className="counter">Counter</p>
      <button onclick={this.onIncrement}>Increase</button>
    )
  }
}
export default Counter
```

6. Event Listeners inside Class Component

Mistake:

File: src/components/Counter/index.js

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  state = { count: 0 }
  onIncrement() {
    const {count} = this.state
    console.log(count)
  }
  render() {
    return (
      <p className="counter">Counter</p>
      <button onClick={this.onIncrement}>Increase</button>
```

```
    )
  }
}
export default Counter
```

Solution:

```
onIncrement = () => {
  const {count} = this.state;
  console.log(count);
};
```

In Arrow functions, this refers to the context in which the code is defined.

7. Passing Event Handler

Mistake:

File: src/components/Counter/index.js

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  state = {count: 0}
  onIncrement = () => {
    this.setState(prevState => ({
      count: prevState.count + 1
    }))
  }
  render() {
    return <button onClick={onIncrement}>Increase</button>
  }
}
export default Counter
```

Solution:

```
<button onClick={this.onIncrement}>Increase</button>
```

8. Modifying the State Directly

Modifying the state directly won't trigger the render() method.

Mistake:

```
onIncrement = () => {
  this.state.count = this.state.count + 1;
```

```
};
```

Solution:

```
onIncrement = () => {
  this.setState(prevState => ({
    count: prevState.count + 1
  }));
};
```

Updating Object:

Mistake:

```
state = {
  person: {name: "Rahul", age: 30}
}

onUpdateAge = () => {
  const {person} = this.state
  person.age = 29
}
```

Solution:

```
state = {
  person: {name: "Rahul", age: 30}
}

onUpdateAge = () => {
  const {person} = this.state
  const newPerson = {...person, age: 29}
  this.setState({person: newPerson})
}
```

Updating List:

Mistake:

```
state = { numbers: [1, 2, 3] }

onUpdateNumbers = () => {
  const {numbers} = this.state
  numbers.push(4)
}
```

Solution:

```
state = {numbers: [1, 2, 3]}

onUpdateNumbers = () => {
  const {numbers} = this.state
  const updatedNumbers = [...numbers, 4]
  this.setState({numbers: updatedNumbers})
}
```

9. Calling setState() from render()

Mistake:

```
render() {
  this.setState({ count: 0 })
  return ({...})
}
```

Specifying a new state with setState() function causes a new render. If we call setState from the render() function, this will cause an infinite loop.

When we call setState from the render() function, it will cause an infinite loop.

10. Invoking Event Handler

Mistake:

```
import { Component } from 'react'


import './index.css'


class Counter extends Component {
  state = {count: 0}
  onIncrement = () => {
    this.setState(prevState => ({
      count: prevState.count + 1
    }))
  }
  render() {
    return (
      <p className="counter">Counter</p>
```

```
      <button onClick={this.onIncrement()}>Increase</button>
    )
  }
}
export default Counter
```

Solution:

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  state = {count: 0}
  onIncrement = () => {
    this.setState(prevState => ({
      count: prevState.count + 1
    }))
  }
  render() {
    return (
      <p className="counter">Counter</p>
      <button onClick={this.onIncrement}>Increase</button>
    )
  }
}

export default Counter
```

11. setState() is Asynchronous setState() does not update state immediately.

Mistake:

```
import { Component } from 'react'

import './index.css'

class Counter extends Component {
  state = {count: 0}
  onIncrement = () => {
    const {count} = this.state;
    this.setState(prevState => ({
      count: prevState.count + 1
    }))
    console.log(count)
```

```
  }
  render() {
    return (
      <p className="counter">Counter</p>
      <button onClick={this.onIncrement}>Increase</button>
    )
  }
}
export default Counter
```

12. React Component should return a single JSX element

Mistake:

```
const Welcome = () => (
  <h1>Hello, User</h1>
  <p>You are learning React</p>
)
export default Welcome

import Welcome from'./components/Welcome'
const App = () => <Welcome />
export default App
```

12.1 Fragments

The fragment is an alternate way to return a single JSX element. It groups a list of children without adding extra nodes to the DOM.

```
import {Fragment} from 'react'

const Welcome = () => (
  <Fragment>
    <h1>Hello, User</h1>
    <p>You are learning React</p>
  </Fragment>
)
export default Welcome
```

Short Syntax:

```
const Welcome = () => (
   <>
     <h1>Hello, User</h1>
     <p>You are learning React</p>
   </>
```

)

export default Welcome

13. Props are Read-only

Mistake:

File: src/App.js

```
import Welcome from './components/Welcome'
const App = () => <Welcome name="Rahul" />
export default App
```

File: **src/components/Welcome/index.js**

```
const Welcome = props => {
  props.name = 'Ramu'
  const {name} = props
  return (
    <>
      <h1>Hello, {name}</h1>
      <p>You are learning React</p>
    </>
  )
}

export default Welcome
```

Debugging using Developer tools:

Debugging

Debugging is the process of finding & fixing the bugs in the code.

We can debug using:

- Browser Developer Tools
- React Developer Tools

Browser Developer Tools

These are the tools provided by the browsers to debug the application loaded in the web browser.

Using Browser Developer Tools, we can:

- View the source code (HTML, CSS, and JS)
- View and change CSS

- View logged messages in the console
- Run JavaScript in the console
- Check the responsiveness of an application, etc.

React Developer Tools

For React Developer Tools, we can install the React Developer Tools Extension for Google Chrome.

On Demand Session:

Concepts in Focus

- Projects Example

1. Projects Example

File: src/App.js

```
import {Component} from 'react'

import TabItem from './components/TabItem'
import ProjectItem from './components/ProjectItem'
import Header from './components/Header'

import './App.css'

const tabsList = [
  {tabId: 'STATIC', displayText: 'Static'},
  {tabId: 'RESPONSIVE', displayText: 'Responsive'},
  {tabId: 'DYNAMIC', displayText: 'Dynamic'},
]

const projectsList = [
  {
    projectId: 0,
    category: 'STATIC',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-s3-img.png',
    title: 'Music Page',
    description:
      'The music page enables the users to browse through the images of all-time favorite music albums.',
  },
  {
```

```
    projectId: 1,
    category: 'STATIC',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-s4-img.png',
    title: 'Tourism Website',
    description:
      'A tourism website enables the user to browse through the images of popular
destinations.',
  },
  {
    projectId: 2,
    category: 'STATIC',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-s1-img.png',
    title: 'Advanced Technologies',
    description:
      'A website that gives you a basic understanding of Advanced Technologies.',
  },
     {
       projectId: 3,
       category: 'STATIC',
       imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-s2-img.png',
       title: 'Happy Meals',
       description: 'Discover the best foods in over 1,000 restaurants.',
     },
  {
    projectId: 4,
    category: 'RESPONSIVE',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-r4-img.png',
    title: 'VR Website',
    description:
      'VR Website enables users to explore AR and VR Products and Industry
happenings.',
  },
  {
    projectId: 5,
    category: 'RESPONSIVE',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-r2-img.png',
    title: 'Food Munch',
    description: 'Food Much Website is a user-centric food tech website.',
  },
```

```
  {
    projectId: 6,
    category: 'RESPONSIVE',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-r3-img.png',
    title: 'Portfolio',
    description:
      'A portfolio is the best alternative for a resume to showcase your skills to the
digital world.',
  },
 {
   projectId: 7,
   category: 'RESPONSIVE',
   imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-r1-img.png',
   title: 'Design',
   description:
     'A website to showcase the best features and give more information about the
Design tool.',
 },
  {
    projectId: 8,
    category: 'DYNAMIC',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-d3-img.png',
    title: 'Speed Typing Test',
    description:
      'Speed Typing Test Application is capable of calculating the time to type the
randomly generated quote.',
  },
  {
    projectId: 9,
    category: 'DYNAMIC',
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-d1-img.png',
    title: 'Random Joke Page',
    description:
      'Random Joke Page is an API-based dynamic Web Application that generates a
new joke.',
  },
  {
    projectId: 10,
    category: 'DYNAMIC',
```

```
    imageURL: 'https://assets.ccbp.in/frontend/react-js/projects-d2-img.png',
    title: 'Sizing An Image',
    description:
      'This is a dynamic web application capable of adjusting the size of an element
using DOM manipulations.',
  },
]

class App extends Component {
  state = {
    activeTabId: tabsList[0].tabId,
  }

  clickTabItem = tabValue => {
    this.setState({activeTabId: tabValue})
  }

  getFilteredProjects = () => {
    const {activeTabId} = this.state
    const filteredProjects = projectsList.filter(
      eachprojectDetails => eachprojectDetails.category === activeTabId,
    )
    return filteredProjects
  }

  render() {
    const {activeTabId} = this.state
    const filteredProjects = this.getFilteredProjects()
    return (
      <div className="app-container">
        <Header />
        <h1 className="title">Projects</h1>
        <p className="description">
          Your skills and achievements showcase your strengths and abilities.
          Speak about any new skills or software you learnt to perform the
          project responsibilities.
        </p>

        <ul className="tabs-container">
```

```
        {tabsList.map(tabDetails => (
          <TabItem
            key={tabDetails.tabId}
            tabDetails={tabDetails}
            clickTabItem={this.clickTabItem}
            isActive={activeTabId === tabDetails.tabId}
          />
        ))}
      </ul>

      <ul className="project-list-container">
        {filteredProjects.map(projectDetails => (
          <ProjectItem
            key={projectDetails.projectId}
            projectDetails={projectDetails}
          />
        ))}
      </ul>
    </div>
  )
 }
}
export default App
```

File: src/components/TabItem/index.js

```
import './index.css'

const TabItem = props => {
 const {tabDetails, clickTabItem, isActive} = props
 const {tabId, displayText} = tabDetails
 const onClickTabItem = () => {
  clickTabItem(tabId)
 }

 const activeTabBtnClassName = isActive ? 'active-tab-btn' : ''

 return (
  <li className="tab-item-container ">
   <button
```

```
        type="button"
        className={`tab-btn ${activeTabBtnClassName}`}
        onClick={onClickTabItem}
      >
        {displayText}
      </button>
    </li>
  )
}
export default TabItem
```

Note: In the above code, multiple class names can be added using template literals.

File: src/components/Header/index.js

```
import './index.css'

const Header = () => (
  <nav className="nav-header">
    <div className="nav-content">
      <p className="website-logo">RA</p>
      <ul className="nav-menu">
        <li>
          <img
            className="social-network-img"
            src="https://image.flaticon.com/icons/png/512/174/174857.png"
            alt="Linked In"
          />
        </li>
        <li>
          <img
            className="social-network-img"
            src="https://image.flaticon.com/icons/png/512/25/25231.png"
            alt="Git Hub"
          />
        </li>
        <li>
          <img
            className="social-network-img"
```

```
          src="https://res.cloudinary.com/dmwrugc6z/image/upload/v1621681890/
Background_4x_1_bzyjew.png"
          alt="Twitter"
        />
       </li>
     </ul>
   </div>
  </nav>
)
export default Header
```

File: src/components/ProjectItem/index.js

```
import './index.css'

const ProjectItem = props => {
  const {projectDetails} = props
  const {projectId, imageURL, description, title} = projectDetails
  return (
    <>
      <li className="project-item-container">
        <img
          className="project-item-image"
          src={imageURL}
          alt={`project-item${projectId}`}
        />
        <div className="project-item-details-container">
          <h1 className="project-item-title">{title}</h1>
          <p className="project-item-description">{description}</p>
        </div>
      </li>
    </>
  )
}
export default ProjectItem
```

On-Demand Session | Part 2

Concepts in Focus

- Third-Party Packages

1. Third-Party Packages

- UUID (Universally Unique IDentifier)
- Using the UUID package, we can generate a unique id

1.1 Installing UUID

npm install uuid

1.2 Importing of UUID

UUID package provides **uuidv4()**, which returns a unique id whenever it is called.

import {v4 as uuidv4} from 'uuid'

2. Best Practice

- The **state** should be **immutable**. We shouldn't update the array/object directly.
- The best practice is to create a new array/object from the array/object in the previous state using the **spread** operator.

this.state.contactsList = initialContactsList

this.state.contactsList.push(...)

this.setState(prevState => ({

  contactsList: [...prevState.contactsList, newContact],

}))

2.1 Updating a Property of an Item inside List

We should not update the property of a list item directly. To update the property of a list item, we should create a new object and return it to the list.

Syntax:

{...object, newItem}

Exmaple:

File: src/App.js

import {Component} from 'react'

```
import {v4 as uuidv4} from 'uuid'

import ContactItem from './components/ContactItem'

import './App.css'

const initialContactsList = [
  {
    id: uuidv4(),
    name: 'Ram',
    mobileNo: 9999988888,
    isFavorite: false,
  },
  {
    id: uuidv4(),
    name: 'Pavan',
    mobileNo: 8888866666,
    isFavorite: true,
  },
  {
    id: uuidv4(),
    name: 'Nikhil',
    mobileNo: 9999955555,
    isFavorite: false,
  },
]

class App extends Component {
```

```
state = {
  contactsList: initialContactsList,
  name: '',
  mobileNo: '',
}

toggleIsFavorite = id => {
  this.setState(prevState => ({
    contactsList: prevState.contactsList.map(eachContact => {
      if (id === eachContact.id) {
        //   eachContact.isFavorite = !eachContact.isFavorite
        return {...eachContact, isFavorite: !eachContact.isFavorite}
      }
      return eachContact
    }),
  }))
}

onAddContact = event => {
  event.preventDefault()
  const {name, mobileNo} = this.state
  const newContact = {
    id: uuidv4(),
    name,
    mobileNo,
    isFavorite: false,
  }
```

```
  this.setState(prevState => ({
    contactsList: [...prevState.contactsList, newContact],
    name: '',
    mobileNo: '',
  }))
}

onChangeMobileNo = event => {
  this.setState({mobileNo: event.target.value})
}

onChangeName = event => {
  this.setState({name: event.target.value})
}

render() {
  const {name, mobileNo, contactsList} = this.state
  return (
    <div className="app-container">
      <div className="responsive-container">
        <h1 className="heading">Contacts</h1>
        <form className="contact-form-container" onSubmit={this.onAddContact}>
          <input
            value={name}
            onChange={this.onChangeName}
            className="input"
            placeholder="Name"
          />
```

```
          <input
            className="input"
            value={mobileNo}
            onChange={this.onChangeMobileNo}
            placeholder="Mobile Number"
          />
          <button type="submit" className="button">
            Add Contact
          </button>
        </form>
        <ul className="contacts-table">
          <li className="table-header">
            <p className="table-header-cell name-column">Name</p>
            <hr className="separator" />
            <p className="table-header-cell">Mobile Number</p>
          </li>
          {contactsList.map(eachContact => (
            <ContactItem
              key={eachContact.id}
              contactDetails={eachContact}
              toggleIsFavorite={this.toggleIsFavorite}
            />
          ))}
        </ul>
      </div>
    </div>
  )
}
```

```
}
export default App
```

3. Contacts App Final Code

File: src/App.js

```
import {Component} from 'react'
import {v4 as uuidv4} from 'uuid'

import ContactItem from './components/ContactItem'

import './App.css'

const initialContactsList = [
  {
    id: uuidv4(),
    name: 'Ram',
    mobileNo: 9999988888,
    isFavorite: false,
  },
  {
    id: uuidv4(),
    name: 'Pavan',
    mobileNo: 8888866666,
    isFavorite: true,
  },
  {
    id: uuidv4(),
    name: 'Nikhil',
    mobileNo: 9999955555,
    isFavorite: false,
  },
]

class App extends Component {
  state = {
    contactsList: initialContactsList,
    name: '',
```

```
  mobileNo: '',
}

toggleIsFavorite = id => {
  this.setState(prevState => ({
    contactsList: prevState.contactsList.map(eachContact => {
      if (id === eachContact.id) {
        return {...eachContact, isFavorite: !eachContact.isFavorite}
      }
      return eachContact
    }),
  }))
}

onAddContact = event => {
  event.preventDefault()
  const {name, mobileNo} = this.state
  const newContact = {
    id: uuidv4(),
    name,
    mobileNo,
    isFavorite: false,
  }

  this.setState(prevState => ({
    contactsList: [...prevState.contactsList, newContact],
    name: '',
    mobileNo: '',
  }))
}

onChangeMobileNo = event => {
  this.setState({mobileNo: event.target.value})
}

onChangeName = event => {
  this.setState({name: event.target.value})
}
```

```
render() {
  const {name, mobileNo, contactsList} = this.state
  return (
    <div className="app-container">
      <div className="responsive-container">
        <h1 className="heading">Contacts</h1>
        <form className="contact-form-container" onSubmit={this.onAddContact}>
          <input
            value={name}
            onChange={this.onChangeName}
            className="input"
            placeholder="Name"
          />
          <input
            className="input"
            value={mobileNo}
            onChange={this.onChangeMobileNo}
            placeholder="Mobile Number"
          />
          <button type="submit" className="button">
            Add Contact
          </button>
        </form>
        <ul className="contacts-table">
          <li className="table-header">
            <p className="table-header-cell name-column">Name</p>
            <hr className="separator" />
            <p className="table-header-cell">Mobile Number</p>
          </li>
          {contactsList.map(eachContact => (
            <ContactItem
              key={eachContact.id}
              contactDetails={eachContact}
              toggleIsFavorite={this.toggleIsFavorite}
            />
          ))}
        </ul>
      </div>
    </div>
```

```
    )
  }
}
export default App
```

File: src/components/ContactItem/index.js

```
import './index.css'

const ContactItem = props => {
  const {contactDetails, toggleIsFavorite} = props
  const {name, mobileNo, isFavorite, id} = contactDetails

  const starImgUrl = isFavorite
    ? 'https://assets.ccbp.in/frontend/react-js/star-filled-img.png'
    : 'https://assets.ccbp.in/frontend/react-js/star-outline-img.png'

  const onClickFavoriteIcon = () => {
    toggleIsFavorite(id)
  }

  return (
    <li className="table-row">
      <div className="table-cell name-column">
        <p>{name}</p>
      </div>
      <hr className="separator" />
      <div className="table-cell mobile-no-column">
        <p className="mobile-no-value">{mobileNo}</p>
        <button
          type="button"
```

```
          className="favorite-icon-container"
          onClick={onClickFavoriteIcon}
        >
          <img src={starImgUrl} className="favorite-icon" alt="star" />
        </button>
      </div>
    </li>
  )
}
export default ContactItem
```

Component Life Cycle

Concepts in Focus

1. Component Life Cycle Phases

Every React Component goes through three phases throughout its lifetime:

- Mounting Phase
- Updating Phase
- Unmounting phase

2. Mounting Phase

In this phase, the instance of a component is created and inserted into the DOM.

2.1 Methods

We mainly use the three methods. The three methods are called in the given order:

- constructor()
- render()

- componentDidMount()

2.1.1 constructor()

The
constructor()
method is used to set up the initial
state
and class variables.

Syntax:

```
constructor(props) {
  super(props)
  //state and class variables
}
```

We must call the super(props) method before any other statement. Calling super(props) makes sure that constructor() of the React.Component gets called and initializes the instance.

Initialising State through props

```
constructor(props) {
  super(props)
  this.state = { date: props.date }
}
```

2.1.2 render()

The render() method is used to return the JSX that is displayed in the UI.

2.1.3 componentDidMount()

The componentDidMount() method is used to run statements that require that the component is already placed in the DOM.
Example: set timers, initiate API calls, etc.

3. Updating Phase

In this phase, the component is updated whenever there is a change in the component's state.

3.1 Methods

3.1.1 render()

The render() method is called whenever there is a change in the component's state.

4. Unmounting Phase

In this phase, the component instance is removed from the DOM.

4.1 Methods

4.1.1 componentWillUnmount()

The componentWillUnmount() method is used to cleanup activities performed.
Example: clearing timers, cancelling API calls, etc.

5. Clock Example

File: src/App.js

```
import {Component} from 'react'
import Clock from './components/Clock'

import './App.css'

class App extends Component {
  state = {
    showClock: false
  }
  onToggleClock = () => {
    this.setState( prevState => {
      const { showClock } = prevState
      return {
        showClock: !showClock
      }
    })
  }
  render() {
    const { showClock } = this.state
    return (
      <div className="app-container">
        <button onClick={ this.onToggleClock } type="button" className="toggle-button">
          {showClock ? 'Show Clock' : 'Hide Clock'}
        </button>
        {showClock && <Clock />}
      </div>
    )
  }
}
export default App
```

Note:
The logical && operator can be handy for conditionally including an element or component.

File: src/components/Clock/index.js

```
import {Component} from 'react'
```

```
import './index.css'

class Clock extends Component {
  constructor(props) {
    super(props)
    this.state = { date: new Date() }
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000)
  }
  componentWillUnmount() {
    clearInterval(this.timerID)
  }
  tick = () => {
    this.setState({
      date: new Date()
    })
  }
  render() {
    const { date } = this.state
    return (
      <div className="clock-container">
        <h1 className="heading">Clock</h1>
        <p className="time">{date.toLocaleTimeString()}</p>
      </div>
    )
  }
}
```

export default Clock

Routing using React Router:

Concepts in Focus

- Web Apps
- React Router
- Routing Example

1. Web Apps

Web Apps are of two types, based on how we get content:

- Multi-page application (MPA)
- Single-page application (SPA)

1.1 Multi-page application (MPA)

- Every URL is associated with corresponding resources (HTML, CSS, JS).
- The browser downloads these resources when you access them or navigate between URLs.

1.2 Single-page application (SPA)

- All URLs are associated with a single HTML page.
- On navigating we only get the additional content (Component - HTML, CSS, JS).

1.2.1 Advantages of using Single-page application (SPA)

- Faster Page loading - since they load only necessary Component (HTML, CSS, JS) resources on subsequent requests.
- React is mainly used to build Single-page applications.

2. React Router

In React, we build Single-page applications using **React Router**.

To implement routing, React Router provides various components:

- BrowserRouter
- Link
- Route
- Switch

2.1 BrowserRouter

To add routing wrap all the Components with
BrowserRouter
.

Syntax:

<BrowserRouter>

  <Component 1>

  <Component 2>

  ...

</BrowserRouter>

2.2 Link

Link Component creates hyperlinks that allows to navigate around in application.

Syntax:

<Link to="Path"> Display Text</Link>

The **to** prop specifies absolute path.

2.3 Route

The Route component renders specific UI component when path matches current URL.

<Route path="Path" component={Component} />

2.3.1 Exact

Renders the route if path matches exactly the current url

<Route exact path="Path1" component={Component1} />

Note: If user enters undefined Path, the Component won't be rendered

2.3 Switch

The Switch component will only render the first route that matches the path. If no path matches, it renders the Not Found component.

```
<Switch>
  <Route path="Path1" component={Component1} />
  <Route path="Path2" component={Component2} />
 <Route component={NotFound} />
</Switch>
```

3. Routing Example

File: src/App.js

```
import { BrowserRouter, Route, Switch } from "react-router-dom"
import NotFound from "./components/NotFound";
...
```

```
const App = () => (
  <BrowserRouter>
    <Header />
      <Switch>
        <Route exact path="/" component={Home} />
        <Route exact path="/about" component={About} />
        <Route exact path="/contact" component={Contact} />
        <Route component={NotFound} />
      </Switch>
 </BrowserRouter>
)

export default App
```

File: src/components/Header/index.js

```
import { Link } from "react-router-dom";
...
const Header = () => (
  ...
    <ul className="nav-menu">
      <li>
        <Link className="nav-link" to="/">Home</Link>
      </li>
      <li>
        <Link className="nav-link" to="/about">About</Link>
      </li>
      <li>
        <Link className="nav-link" to="/contact">Contact</Link>
      </li>
    </ul>
  ...
)
```

Routing using React Router | Part 2 & 3

Concepts in Focus

- API Calls
    - Fetch
- Route Props
    - Match
- BlogsList Example

1 . API Calls

In General we make API Calls inside componentDidMount(). So that it doesn't block render().

1.1 Fetch

Fetch is a promise-based API which returns a response object. In the backend, we use **snake_case** for naming conventions.

2. Route Props

When a component is rendered by the Route, some additional props are passed

- match
- location
- history

2.1 Match

The match object contains the information about the path from which the component is rendered.

3. BlogsList Example

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'


import Header from './components/Header'

import About from './components/About'

import Contact from './components/Contact'

import BlogsList from './components/BlogsList'

import BlogItemDetails from './components/BlogItemDetails'

import NotFound from './components/NotFound'


import './App.css'


const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
```

```
  </BrowserRouter>
)
export default App
```

File: src/components/Header/index.js

```
import { Link } from 'react-router-dom'

import './index.css'

const Header = () => (
  <nav className="nav-header">
    <div className="blog-container">
      <h1 className="blog-title">Dev Blog</h1>
      <ul className="nav-menu">
        <Link className="nav-link" to="/">
          <li>Home</li>
        </Link>
        <Link className="nav-link" to="/about">
          <li>About</li>
        </Link>
        <Link className="nav-link" to="/contact">
          <li>Contact</li>
        </Link>
      </ul>
    </div>
  </nav>
)
export default Header
```

File: src/components/BlogsList/index.js

```
import { Component } from 'react'
import Loader from 'react-loader-spinner'

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css'
```

```
import BlogItem from '../BlogItem'

import './index.css'

class BlogsList extends Component {
  state = { isLoading: true, blogsData: [] }

  componentDidMount() {
    this.getBlogsData()
  }

  getBlogsData = async () => {
    const response = await fetch('https://apis.ccbp.in/blogs')
    const statusCode = await response.statusCode
    console.log(statusCode)
    const data = await response.json()

    const formattedData = data.map(eachItem => ({
      id: eachItem.id,
      title: eachItem.title,
      imageUrl: eachItem.image_url,
      avatarUrl: eachItem.avatar_url,
      author: eachItem.author,
      topic: eachItem.topic,
    }))
    this.setState({ blogsData: formattedData, isLoading: false })
  }

  render() {
    const { blogsData, isLoading } = this.state
    console.log(isLoading)
```

```
    return (
      <div className="blog-list-container">
        {isLoading ? (
          <Loader type="TailSpin" color="#00BFFF" height={50} width={50} />
        ) : (
          blogsData.map(item => <BlogItem blogData={item} key={item.id} />)
        )}
      </div>
    )
  }
}
export default BlogsList
```

File: src/components/BlogItem/index.js

```
import { Link } from 'react-router-dom'

import './index.css'

const BlogItem = props => {
  const { blogData } = props
  const { id, imageUrl, topic, title, avatarUrl, author } = blogData

  return (
    <Link to={`/blogs/${id}`} className="item-link">
      <div className="item-container">
        <img className="item-image" src={imageUrl} alt={`item${id}`} />
        <div className="item-info">
          <p className="item-topic">{topic}</p>
          <p className="item-title">{title}</p>
          <div className="author-info">
            <img className="avatar" src={avatarUrl} alt={`avatar${id}`} />
            <p className="author-name">{author}</p>
```

```
        </div>
      </div>
    </div>
  </Link>
 )
}
export default BlogItem
```

File: src/components/About/index.js

```
import './index.css'

const About = () => (
 <div className="about-container">
  <img
   src="https://assets.ccbp.in/frontend/react-js/about-blog-img.png"
   alt="about"
   className="about-img"
  />
  <h1 className="about-heading">About</h1>
  <p className="about-paragraph">All about Blogs of frontend developers</p>
 </div>
)
export default About
```

File: src/components/Contact/index.js

```
import './index.css'

const Contact = () => (
 <div className="contact-container">
  <img
   src="https://assets.ccbp.in/frontend/react-js/contact-blog-img.png"
   alt="contact"
   className="contact-img"
  />
```

```
    <h1 className="contact-heading">Contact</h1>
  </div>
)
export default Contact
```

File: src/components/BlogItemDetails/index.js

```
import { Component } from 'react'
import Loader from 'react-loader-spinner'

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css'
import './index.css'

class BlogItemDetails extends Component {
  state = { blogData: {}, isLoading: true }

  componentDidMount() {
    this.getBlogItemData()
  }

  getBlogItemData = async () => {
    const { match } = this.props
    const { params } = match
    const { id } = params

    const response = await fetch(`https://apis.ccbp.in/blogs/${id}`)
    const data = await response.json()

    const updatedData = {
      title: data.title,
      imageUrl: data.image_url,
      content: data.content,
      avatarUrl: data.avatar_url,
      author: data.author,
```

```
  }
  this.setState({ blogData: updatedData, isLoading: false })
}

renderBlogItemDetails = () => {
  const { blogData } = this.state
  const { title, imageUrl, content, avatarUrl, author } = blogData

  return (
    <div className="blog-info">
      <h2 className="blog-details-title">{title}</h2>
      <div className="author-details">
        <img className="author-pic" src={avatarUrl} alt={author} />
        <p className="details-author-name">{author}</p>
      </div>
      <img className="blog-image" src={imageUrl} alt={title} />
      <p className="blog-content">{content}</p>
    </div>
  )
}

render() {
  const { isLoading } = this.state

  return (
    <div className="blog-container">
      {isLoading ? (
        <Loader type="TailSpin" color="#00BFFF" height={50} width={50} />
      ) : (
        this.renderBlogItemDetails()
      )}
    </div>
```

```
    )
  }
}
export default BlogItemDetails
```

File: src/components/NotFound/index.js

```
import './index.css'

const NotFound = () => (
  <div className="not-found-container">
    <img
      src="https://assets.ccbp.in/frontend/react-js/not-found-blog-img.png"
      alt="not-found"
      className="not-found-img"
    />
  </div>
)
export default NotFound
```

Common Mistakes | Part 2

Concepts in Focus

- Rendering Promise Object
- Using Link and Route Components without BrowserRouter
- Providing Wrong Route Path
- Missing exact keyword
- Missing Switch
- Placing Common component inside Switch
- Providing the same PATH for multiple Routes
- Missing Colon(:) while providing Path Parameters
- Accessing Promise Object
- Ordering of Routes inside Switch

1. Rendering Promise Object

Mistake:

- Async Function returns a Promise Object But, React cannot render this Promise Object.
- The best way to make API call is in the componentDidMount()

.

File: src/components/BlogsList/index.js

```
import { Component } from 'react'
import Loader from 'react-loader-spinner'

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css'

import BlogItem from '../BlogItem'

import './index.css'

class BlogsList extends Component {
  state = { isLoading: true, blogsData: [] }

  getBlogsData = async () => {
    const response = await fetch('https://apis.ccbp.in/blogs')
    const statusCode = await response.statusCode
    console.log(statusCode)
    const data = await response.json()

    const formattedData = data.map(eachItem => ({
      id: eachItem.id,
      title: eachItem.title,
      imageUrl: eachItem.image_url,
      avatarUrl: eachItem.avatar_url,
      author: eachItem.author,
      topic: eachItem.topic,
    }))
  }
```

```
  render() {
    const { blogsData, isLoading } = this.state
    console.log(isLoading)
    return (
      <div className="blog-list-container">
       {this.getBlogsData()}
      </div>
    )
  }
}
export default BlogsList
```

Solution:

File: src/components/BlogsList/index.js

```
import { Component } from 'react'
import Loader from 'react-loader-spinner'
import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css'
import BlogItem from '../BlogItem'
import './index.css'
class BlogsList extends Component {
  state = { isLoading: true, blogsData: [] }
  componentDidMount() {
    this.getBlogsData()
  }

  getBlogsData = async () => {
    const response = await fetch('https://apis.ccbp.in/blogs')
    const statusCode = await response.statusCode
    console.log(statusCode)
```

```
    const data = await response.json()
    const formattedData = data.map(eachItem => ({
      id: eachItem.id,
      title: eachItem.title,
      imageUrl: eachItem.image_url,
      avatarUrl: eachItem.avatar_url,
      author: eachItem.author,
      topic: eachItem.topic,
    }))
    this.setState({ blogsData: formattedData, isLoading: false })
  }
  render() {
    const { blogsData, isLoading } = this.state
    console.log(isLoading)
    return (
      <div className="blog-list-container">
        {isLoading ? (
          <Loader type="TailSpin" color="#00BFFF" height={50} width={50} />
        ) : (
          blogsData.map(item => <BlogItem blogData={item} key={item.id} />)
        )}
      </div>
    )
  }
}
export default BlogsList
```

2. Using Link and Route Components without BrowserRouter

Mistake:

File: src/App.js

```
import {Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
   <Header />
   <Switch>
     <Route exact path="/" component={BlogsList} />
     <Route path="/about" component={About} />
     <Route path="/contact" component={Contact} />
     <Route path="/blogs/:id" component={BlogItemDetails} />
     <Route component={NotFound} />
   </Switch>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
```

```
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

File: src/components/Header/index.js

```
import { Link } from 'react-router-dom'

import './index.css'

const Header = () => (
  <nav className="nav-header">
    <div className="blog-container">
      <h1 className="blog-title">Dev Blog</h1>
      <ul className="nav-menu">
        <Link className="nav-link" to="/">
          <li>Home</li>
        </Link>
```

```
        <Link className="nav-link" to="/about">
          <li>About</li>
        </Link>
        <Link className="nav-link" to="/contact">
          <li>Contact</li>
        </Link>
      </ul>
    </div>
  </nav>
)
export default Header
```

3. Providing Wrong Route Path

Mistake:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/abot" component={About} />
```

```
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
```

```
export default App
```

File: src/components/Header/index.js

```
import { Link } from 'react-router-dom'

import './index.css'

const Header = () => (
  <nav className="nav-header">
    <div className="blog-container">
      <h1 className="blog-title">Dev Blog</h1>
      <ul className="nav-menu">
        <Link className="nav-link" to="/">
          <li>Home</li>
        </Link>
        <Link className="nav-link" to="/about">
          <li>About</li>
        </Link>
        <Link className="nav-link" to="/contact">
          <li>Contact</li>
        </Link>
      </ul>
    </div>
  </nav>
)
export default Header
```

4. Missing exact keyword

Mistake:

- Switch renders a Route that matches the path.
- If Switch is missed, Browser will render Multiple Routes in the single path.

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'
```

```
import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route  path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'
```

```
import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route exact path="/about" component={About} />
      <Route exact path="/contact" component={Contact} />
      <Route exact path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

5. Missing Switch

Mistake:

File: src/App.js

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

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
```

```
  <BrowserRouter>
      <Header />
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
  </BrowserRouter>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
      <Header />
      <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
```

```
      </Switch>
  </BrowserRouter>
)
export default App
```

6. Placing Common component inside Switch

Mistake:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
     <Switch>
     <Header />
     <Route exact path="/" component={BlogsList} />
     <Route path="/about" component={About} />
     <Route path="/contact" component={Contact} />
     <Route path="/blogs/:id" component={BlogItemDetails} />
     <Route component={NotFound} />
      </Switch>
  </BrowserRouter>
)
```

```
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
     <Header />
     <Switch>
     <Route exact path="/" component={BlogsList} />
     <Route path="/about" component={About} />
     <Route path="/contact" component={Contact} />
     <Route path="/blogs/:id" component={BlogItemDetails} />
     <Route component={NotFound} />
      </Switch>
  </BrowserRouter>
)
export default App
```

7. Providing the same PATH for multiple Routes

Mistake:

- If same path is given for multiple Routes, the Switch renders the first matched Component.

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
      <Header />
      <Switch>
      <Route exact path="/about" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
       </Switch>
  </BrowserRouter>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
```

```
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
      <Header />
      <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
       </Switch>
  </BrowserRouter>
)
export default App
```

8. Missing Colon(:) while providing Path Parameters

Mistake:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'
```

```
import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
```

```
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

9. Accessing Promise Object

Mistake:

- In the Promise Object response, we got an error message with key error_msg but in the below code snippet, we are trying to access the promise object with the wrong key errorText so undefined will be logged in the console.

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }
```

```
onChangePassword = event => {
  this.setState({password: event.target.value})
}

onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.errorText)
```

```
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
```

```
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
```

```
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
export default LoginForm
```

Solution:

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
```

```
}

onChangeUsername = event => {
  this.setState({username: event.target.value})
}

onChangePassword = event => {
  this.setState({password: event.target.value})
}

onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
```

```
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
```

```
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
```

```
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
          className="login-image"
          alt="website login"
        />
        <form className="form-container" onSubmit={this.submitForm}>
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            className="login-website-logo-desktop-image"
            alt="website logo"
          />
          <div className="input-container">{this.renderUsernameField()}</div>
          <div className="input-container">{this.renderPasswordField()}</div>
          <button type="submit" className="login-button">
            Login
          </button>
          {showSubmitError && <p className="error-message">{errorMsg}</p>}
        </form>
      </div>
    )
  }
}
export default LoginForm
```

10. Ordering of Routes inside Switch

Mistake:

- In the below code snippet NotFound Component is placed first in the Switch Component .
- If the path is not mentioned and route (NotFound Route) is placed first inside Switch, Browser will render it in all the paths.

File: src/App.js

```
import {Route, Switch } from 'react-router-dom'


import Header from './components/Header'
```

```
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
   <Header />
   <Switch>
     <Route exact path="/" component={BlogsList} />
     <Route path="/about" component={About} />
     <Route path="/contact" component={Contact} />
     <Route path="/blogs/:id" component={BlogItemDetails} />
     <Route component={NotFound} />
   </Switch>
)
export default App
```

Solution:

File: src/App.js

```
import { BrowserRouter, Route, Switch } from 'react-router-dom'

import Header from './components/Header'
import About from './components/About'
import Contact from './components/Contact'
import BlogsList from './components/BlogsList'
import BlogItemDetails from './components/BlogItemDetails'
import NotFound from './components/NotFound'

import './App.css'
```

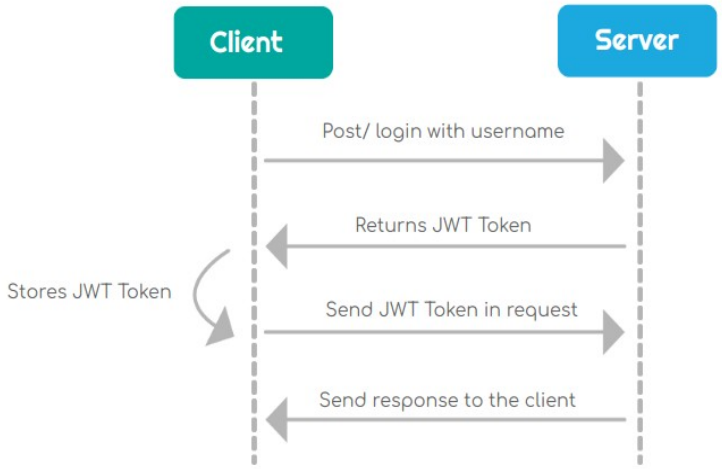
```
const App = () => (
  <BrowserRouter>
    <Header />
    <Switch>
      <Route exact path="/" component={BlogsList} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route path="/blogs/:id" component={BlogItemDetails} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

Authentication & Authorization

Concepts in Focus

- Client-Server Communication
  - Authentication
  - Authorization
- Authentication Flow
- Route Parameters
  - History
- E-Commerce Application

1. Client-Server Communication



1.1 Authentication

Authentication is the process of verifying a user's identity.

1.2 Authorization

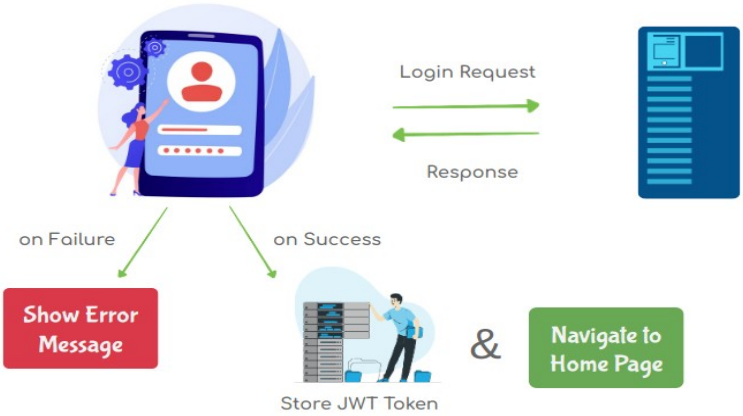
Authorization is the process of verifying whether the user is authenticated and permitted to perform some actions like accessing resources, etc.

Example:

After successful authentication, employees are only allowed to access certain resources based on their roles.

- Admin can Read, Create, Delete, and Update the Resources
- User can only Read and Create the Resources

2. Authentication Flow



3. Route Parameters

When a component is rendered by the Route, some additional props are passed.

They are:

- match
- history
- location

3.1 History

The history object has some methods to control the navigation in the browser, and it also maintains the history of the routes we navigated.
It has the following methods to control the navigation in the browser:

- push()
- replace()
- go()
- goBack()
- goForward()
  , etc.

The history.push() and history.replace() methods are used to navigate to other routes programmatically.

3.1.1 history.push()

With the history.push() method, the user can go forward and backwards in the browser, and the URL will change.
Syntax:

```
history.push("PATH");
```

3.1.2 history.replace()

The history.replace() method replaces the current URL with new one. The user can't go backwards to the previous URL.
Syntax:

```
history.replace("PATH");
```

4. E-Commerce Application

- **Make an Authentication Request to Login API**
- **Handle Login API Response**
  - **On Login Success**
  - On Login Failure
- Store the JWT Token

Authenticated Credentials:

Username: henry

password: henry_the_developer

Username: david

password: the_miller@23

Username: robert

password: WilsonRobert45

Username: mosh

password: DevMosh22

Username: rahul

password: rahul@2021

Username: praneetha

password: praneetha@2021

File: src/App.js

```
import {BrowserRouter, Route, Switch} from 'react-router-dom'

import LoginForm from './components/LoginForm'
import Home from './components/Home'
import Products from './components/Products'
import Cart from './components/Cart'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Switch>
      <Route exact path="/login" component={LoginForm} />
      <Route exact path="/" component={Home} />
      <Route exact path="/products" component={Products} />
      <Route exact path="/cart" component={Cart} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
export default App
```

File: src/components/LoginForm/index.js

```
import {Component} from 'react'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
  }

  onSubmitSuccess = () => {
    const {history} = this.props
    history.replace('/')
  }

  submitForm = async event => {
    event.preventDefault()
    const {username, password} = this.state
    const userDetails = {username, password}
    const url = 'https://apis.ccbp.in/login'
    const options = {
      method: 'POST',
      body: JSON.stringify(userDetails),
    }
    const response = await fetch(url, options)
    const data = await response.json()
    if (response.ok === true) {
      this.onSubmitSuccess()
    }
  }

  onChangeUsername = event => {
```

```
  this.setState({username: event.target.value})
}

onChangePassword = event => {
  this.setState({password: event.target.value})
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
```

```
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
      />
    </>
  )
}

render() {
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
```

```
        <button type="submit" className="login-button">
          Login
        </button>
      </form>
    </div>
  )
 }
}
export default LoginForm
```

Note:

- If the Response status code is 2XX, then response.ok will be true else it is false.
- Whenever the route changes, the switch in the App.js will trigger again, and the corresponding component will render.

File: src/components/Home/index.js

```
import Header from '../Header'
import './index.css'

const Home = () => (
 <>
   <Header />
   <div className="home-container">
     <div className="home-content">
       <h1 className="home-heading">Clothes That Get YOU Noticed</h1>
       <img
         src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
         alt="dresses to be noticed"
         className="home-mobile-img"
       />
       <p className="home-description">
         Fashion is part of the daily air and it does not quite help that it
         changes all the time. Clothes have always been a marker of the era and
```

```
          we are in a revolution. Your fashion makes you been seen and heard
          that way you are. So, celebrate the seasons new and exciting fashion
          in your own way.
        </p>
        <button type="button" className="shop-now-button">
          Shop Now
        </button>
      </div>
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
        alt="dresses to be noticed"
        className="home-desktop-img"
      />
    </div>
  </>
)
export default Home
```

File: src/components/Cart/index.js

```
import Header from '../Header'
import './index.css'

const Cart = () => (
  <>
    <Header />
    <div className="cart-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-img.png"
        alt="cart"
        className="cart-img"
      />
    </div>
  </>
```

)

export default Cart

File: src/components/Header/index.js

```
import {Link} from 'react-router-dom'
import './index.css'

const Header = () => (
  <nav className="nav-header">
    <div className="nav-content">
      <img
        className="website-logo"
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        alt="website logo"
      />
      <ul className="nav-menu">
        <Link to="/" className="nav-link">
          <li>Home</li>
        </Link>
        <Link to="/products" className="nav-link">
          <li>Products</li>
        </Link>
        <Link to="/cart" className="nav-link">
          <li>Cart</li>
        </Link>
      </ul>
      <button type="button" className="logout-desktop-btn">
        Logout
      </button>
      <button type="button" className="logout-mobile-btn">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
          alt="logout icon"
```

```
          className="logout-icon"
        />
      </button>
    </div>
  </nav>
)
export default Header
```

File: src/components/Products/index.js

```
import Header from '../Header'

import './index.css'

const Products = () => (
  <>
    <Header />
    <div className="products-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-img.png"
        alt="products"
        className="products-img"
      />
    </div>
  </>
)
export default Products
```

File: src/components/NotFound/index.js

```
import "./index.css";

const NotFound = () => (
  <div className="not-found-container">
    <img
      src="https://assets.ccbp.in/frontend/react-js/not-found-blog-img.png"
```

```
      alt="not-found"
      className="not-found-img"
    />
  </div>
);
export default NotFound;
```

Authentication & Authorization | Part 2

Concepts in Focus

- JWT Token
- Storage Mechanisms
- Cookies
    - Why Cookies?
    - Cookies vs Local Storage
    - Third Party Package js-cookie
- Redirect Component
- withRouter
- E-Commerce Application

1. JWT Token

JSON Web Token is a standard used to create Access Tokens. These access tokens are also called JWT Tokens.

The client uses these access tokens on every subsequent request to communicate with the Server.

Note: While making HTTP Request, we have to send an access token in the HTTP Headers with the key Authorization.

Example:

Authorization: Bearer jwt_token

1.1 Storing JWT Token in State

When we store the JWT Token in the state,

- On page refresh, the JWT token won't be available
- It is difficult to pass state information to every component

2. Storage Mechanisms

- Client-Side Data Storage
    - Storing Data on the Client
- Server-Side Data Storage
    - Storing Data on the Server using some kind of Database

Different types of Client-Side data storage mechanisms are:

- Local Storage
- **Cookies**
- Session Storage
- IndexedDB, etc.

3. Cookies

A cookie is a piece of data that is stored on the user's computer by the web browser.

A cookie is made up of:

- Name & Value
- Expires − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- Domain − The domain name of your site.
- Path − The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
- Secure − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists, etc.

3.1 Why Cookies?

With cookies, we can set the expiry duration.

Examples:

- Banking Applications - Cookies get expired in minutes
- Facebook - Cookies get expired in months or years

3.2 Cookies vs Local Storage

Cookies	Local Storage
We can set an expiration for Cookies	Local storage data never expires
Cookies can store up to 4KB of data	Local Storage can store up to 5 to 10 MB of data

3.3 Third Party Package - js-cookie

JavaScript can read, create, modify, and delete the cookies.

NPM contains a **js-cookie**, a third-party package to manipulate cookies easily.

Installation Command:

npm install js-cookie –save
js-cookie methods are:

- Cookies.set()
  - It is used to set the cookie
- Cookies.get()
  - It is used to get the cookie
- Cookies.remove()
  - It is used to remove the cookie

3.3.1 Cookies.set()

Syntax:

Cookies.set('CookieName', 'CookieValue', {expires: DAYS});

Example:

Cookies.set('ACCESS_TOKEN', 'Us1L90PXl...', {expires: 1});

3.3.2 **Cookies.get()**

It returns undefined if the cookie expires or does not exist.

Syntax:

Cookies.get('CookieName');

Example:

Cookies.get('ACCESS_TOKEN');

3.3.3 **Cookies.remove()**

Syntax:

Cookies.remove('CookieName');

Example:

Cookies.remove('ACCESS_TOKEN');

4. Redirect Component

The **react-router-dom** provides the **Redirect** component. It can be used whenever we want to redirect to another path.

Syntax:

<Redirect to="PATH" />

Example:

<Redirect to="/login" />

4.1 Redirect Component vs history Methods

- Use the **Redirect** Component when you have to stop displaying UI and navigate to a route. Ex: Inside Class Component - render()
- In all other cases, use history.push() or history.replace() syntax Ex: onSubmit, onClick event callback functions

Note:

The **Redirect** component uses the history push and replace methods behinds the scene.

5. withRouter

The history prop will be available for only components which are directly given for Route.

To provide history prop to other components, we can wrap it with the withRouter function while exporting it.
Example:

```
import { withRouter} from 'react-router-dom'
...
export default withRouter(ComponentName)
```

6. E-Commerce Application

- Make an Authentication Request to Login API
- Handle Login API Response
    - On Login Success
    - **On Login Failure**
- **Store the JWT Token**

File: src/App.js

```
import {BrowserRouter, Route, Switch} from 'react-router-dom'

import LoginForm from './components/LoginForm'
import Home from './components/Home'
import Products from './components/Products'
import Cart from './components/Cart'
import NotFound from './components/NotFound'

import './App.css'

const App = () => (
  <BrowserRouter>
    <Switch>
      <Route exact path="/login" component={LoginForm} />
      <Route exact path="/" component={Home} />
      <Route exact path="/products" component={Products} />
      <Route exact path="/cart" component={Cart} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)

export default App
```

File: src/components/Cart/index.js

```
import Header from '../Header'
import './index.css'

const Cart = () => (
  <>
    <Header />
    <div className="cart-container">
      <img
```

```
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-img.png"
        alt="cart"
        className="cart-img"
      />
    </div>
  </>
)

export default Cart
```

File: src/components/Header/index.js

```
import {Link, withRouter} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'

const Header = props => {
  const {history} = props
  const onClickLogout = () => {
    Cookies.remove('jwt_token')
    history.replace('/login')
  }
  return (
    <nav className="nav-header">
      <div className="nav-content">
        <img
          className="website-logo"
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          alt="website logo"
        />
        <ul className="nav-menu">
          <Link to="/" className="nav-link">
            <li>Home</li>
          </Link>
          <Link to="/products" className="nav-link">
            <li>Products</li>
          </Link>
          <Link to="/cart" className="nav-link">
            <li>Cart</li>
          </Link>
        </ul>
        <button
          type="button"
          className="logout-desktop-btn"
          onClick={onClickLogout}
        >
          Logout
        </button>
        <button
          type="button"
```

```
        className="logout-mobile-btn"
        onClick={onClickLogout}
      >
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
          alt="logout icon"
          className="logout-icon"
        />
      </button>
    </div>
  </nav>
 )
}
export default withRouter(Header)
```

File: src/components/Home/index.js

```
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'
import Header from '../Header'

import './index.css'

const Home = () => {
 const jwtToken = Cookies.get('jwt_token')
 if (jwtToken === undefined) {
  return <Redirect to="/login" />
 }

 return (
  <>
    <Header />
    <div className="home-container">
      <div className="home-content">
        <h1 className="home-heading">Clothes That Get YOU Noticed</h1>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
          alt="dresses to be noticed"
          className="home-mobile-img"
        />
        <p className="home-description">
          Fashion is part of the daily air and it does not quite help that it
          changes all the time. Clothes have always been a marker of the era
          and we are in a revolution. Your fashion makes you been seen and
          heard that way you are. So, celebrate the seasons new and exciting
          fashion in your own way.
        </p>
        <button type="button" className="shop-now-button">
          Shop Now
        </button>
      </div>
      <img
```

```
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
          alt="dresses to be noticed"
          className="home-desktop-img"
        />
      </div>
    </>
  )
}

export default Home
```

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }

  onSubmitSuccess = jwtToken => {
    const {history} = this.props

    Cookies.set('jwt_token', jwtToken, {expires: 30})
    history.replace('/')
  }

  onSubmitFailure = errorMsg => {
    this.setState({showSubmitError: true, errorMsg})
  }

  submitForm = async event => {
    event.preventDefault()
    const {username, password} = this.state
    const userDetails = {username, password}
    const url = 'https://apis.ccbp.in/login'
    const options = {
      method: 'POST',
      body: JSON.stringify(userDetails),
```

```
  }
  const response = await fetch(url, options)
  const data = await response.json()
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
```

```
          className="login-website-logo-mobile-image"
          alt="website logo"
        />
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
          className="login-image"
          alt="website login"
        />
        <form className="form-container" onSubmit={this.submitForm}>
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            className="login-website-logo-desktop-image"
            alt="website logo"
          />
          <div className="input-container">{this.renderUsernameField()}</div>
          <div className="input-container">{this.renderPasswordField()}</div>
          <button type="submit" className="login-button">
            Login
          </button>
          {showSubmitError && <p className="error-message">*{errorMsg}</p>}
        </form>
      </div>
    )
  }
}

export default LoginForm
```

File: src/components/NotFound/index.js

```
import './index.css'

const NotFound = () => (
  <div className="not-found-container">
    <img
      src="https://assets.ccbp.in/frontend/react-js/not-found-blog-img.png"
      alt="not-found"
      className="not-found-img"
    />
  </div>
)

export default NotFound
```

File: src/components/Products/index.js

```
import Header from '../Header'

import './index.css'

const Products = () => (
  <>
```

```
    <Header />
    <div className="products-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-img.png"
        alt="products"
        className="products-img"
      />
    </div>
  </>
)

export default Products
```

Authentication & Authorization | Part 3

Concepts in Focus

- Router Switch
- Wrapper Component
    - Protected Route
- E-Commerce Application

1.Router Switch

Switch can have any React Component inside it

- Route
- User defined Component
- Redirect

2. Wrapper Component

Redirection Logic can be reused by separating out into a React Component called **Wrapper Component**. Each route will be wrapped with it.

2.1 Protected Route

ProtectedRoute is the Wrapper Component which returns Home Route Component.

File: src/components/ProtectedRoute/index.js

```
import { Route, Redirect } from "react-router-dom";

import Cookies from "js-cookie";


const ProtectedRoute = (props) => {

  const token = Cookies.get("jwt_token");

  if (token === undefined) {
```

```
    return <Redirect to="/login" />;
  }
  return <Route {...props} />;
};
export default ProtectedRoute;
```

3. E-Commerce Application

File: src/App.js

```
import { BrowserRouter, Route, Switch, Redirect } from "react-router-dom";

import LoginForm from "./components/LoginForm";
import Home from "./components/Home";
import Products from "./components/Products";
import Cart from "./components/Cart";
import NotFound from "./components/NotFound";
import ProtectedRoute from "./components/ProtectedRoute";

import "./App.css";

const App = () => (
  <BrowserRouter>
    <Switch>
      <Route exact path="/login" component={LoginForm} />
      <ProtectedRoute exact path="/" component={Home} />
      <ProtectedRoute exact path="/products" component={Products} />
      <ProtectedRoute exact path="/cart" component={Cart} />
      <Route path="/not-found" component={NotFound} />
      <Redirect to="not-found" />
    </Switch>
  </BrowserRouter>
);
export default App;
```

File: src/components/AllProductsSection/index.js

```
import { Component } from "react";
import Cookies from "js-cookie";

import ProductCard from "../ProductCard";
import "./index.css";

class AllProductsSection extends Component {
  state = {
    productsList: [],
  };

  componentDidMount() {
```

```
    this.getProducts();
  }

  getProducts = async () => {
    const apiUrl = "https://apis.ccbp.in/products";
    const jwtToken = Cookies.get("jwt_token");
    const options = {
      headers: {
        Authorization: `Bearer ${jwtToken}`,
      },
      method: "GET",
    };
    const response = await fetch(apiUrl, options);
    if (response.ok === true) {
      const fetchedData = await response.json();
      const updatedData = fetchedData.products.map((product) => ({
        title: product.title,
        brand: product.brand,
        price: product.price,
        id: product.id,
        imageUrl: product.image_url,
        rating: product.rating,
      }));
      this.setState({
        productsList: updatedData,
      });
    }
  };

  renderProductsList = () => {
    const { productsList } = this.state;
    return (
      <div>
        <h1 className="products-list-heading">All Products</h1>
        <ul className="products-list">
          {productsList.map((product) => (
            <ProductCard productData={product} key={product.id} />
          ))}
        </ul>
      </div>
    );
  };

  render() {
    return <>{this.renderProductsList()}</>;
  }
}

export default AllProductsSection;
```

File: src/components/Cart/index.js

```
import Header from "../Header";
import "./index.css";

const Cart = () => (
  <>
    <Header />
    <div className="cart-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-img.png"
        alt="cart"
        className="cart-img"
      />
    </div>
  </>
);

export default Cart;
```

File: src/components/Header/index.js

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

import Cookie from "js-cookie";

import "./index.css";

const Header = (props) => {
  const onClickLogout = () => {
    Cookie.remove("jwt_token");
    const { history } = props;
    history.replace("/login");
  };
  return (
    <nav className="nav-header">
      <div className="nav-content">
        <img
          className="website-logo"
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          alt="website logo"
        />
        <ul className="nav-menu">
          <Link to="/" className="nav-link">
            <li>Home</li>
          </Link>
          <Link to="/products" className="nav-link">
            <li>Products</li>
          </Link>
          <Link to="/cart" className="nav-link">
            <li>Cart</li>
          </Link>
        </ul>
        <button
```

```
          type="button"
          className="logout-desktop-btn"
          onClick={onClickLogout}
        >
          Logout
        </button>
        <button
          type="button"
          className="logout-mobile-btn"
          onClick={onClickLogout}
        >
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
            alt="logout icon"
            className="logout-icon"
          />
        </button>
      </div>
    </nav>
  );
};
export default withRouter(Header);
```

File: src/components/Home/index.js

```
import Header from "../Header";
import "./index.css";

const Home = () => (
  <>
    <Header />
    <div className="home-container">
      <div className="home-content">
        <h1 className="home-heading">Clothes That Get YOU Noticed</h1>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
          alt="dresses to be noticed"
          className="home-mobile-img"
        />
        <p className="home-description">
          Fashion is part of the daily air and it does not quite help that it
          changes all the time. Clothes have always been a marker of the era and
          we are in a revolution. Your fashion makes you been seen and heard
          that way you are. So, celebrate the seasons new and exciting fashion
          in your own way.
        </p>
        <button type="button" className="shop-now-button">
          Shop Now
        </button>
      </div>
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
```

```
        alt="dresses to be noticed"
        className="home-desktop-img"
      />
    </div>
  </>
);

export default Home;
```

File: src/components/LoginForm/index.js

```
import { Component } from "react";
import Cookies from "js-cookie";
import { Redirect } from "react-router-dom";

import "./index.css";

class LoginForm extends Component {
  state = {
    username: "",
    password: "",
    showSubmitError: false,
    errorMsg: "",
  };

  onChangeUsername = (event) => {
    this.setState({ username: event.target.value });
  };

  onChangePassword = (event) => {
    this.setState({ password: event.target.value });
  };

  onSubmitSuccess = (jwtTkoken) => {
    const { history } = this.props;

    Cookies.set("jwt_token", jwtTkoken, {
      expires: 30,
      path: "/",
    });
    history.replace("/");
  };

  onSubmitFailure = (errorMsg) => {
    console.log(errorMsg);
    this.setState({ showSubmitError: true, errorMsg });
  };

  submitForm = async (event) => {
    event.preventDefault();
    const { username, password } = this.state;
```

```
  const userDetails = { username, password };
  const url = "https://apis.ccbp.in/login";
  const options = {
    method: "POST",
    body: JSON.stringify(userDetails),
  };
  const response = await fetch(url, options);
  const data = await response.json();
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token);
  } else {
    this.onSubmitFailure(data.error_msg);
  }
};

renderPasswordField = () => {
  const { password } = this.state;
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
      />
    </>
  );
};

renderUsernameField = () => {
  const { username } = this.state;
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
      />
    </>
  );
};

render() {
```

```
    const { showSubmitError, errorMsg } = this.state;
    const jwtToken = Cookies.get("jwt_token");
    if (jwtToken !== undefined) {
      return <Redirect to="/" />;
    }
    return (
      <div className="login-form-container">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-mobile-image"
          alt="website logo"
        />
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
          className="login-image"
          alt="website login"
        />
        <form className="form-container" onSubmit={this.submitForm}>
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            className="login-website-logo-desktop-image"
            alt="website logo"
          />
          <div className="input-container">{this.renderUsernameField()}</div>
          <div className="input-container">{this.renderPasswordField()}</div>
          <button type="submit" className="login-button">
            Login
          </button>
          {showSubmitError && <p className="error-message">*{errorMsg}</p>}
        </form>
      </div>
    );
  }
}

export default LoginForm;
```

File: src/components/NotFound/index.js

```
import "./index.css";

const NotFound = () => (
  <div className="not-found-container">
    <img
      src="https://assets.ccbp.in/frontend/react-js/not-found-blog-img.png"
      alt="not-found"
      className="not-found-img"
    />
  </div>
);

export default NotFound;
```

File: src/components/ProductCard/index.js

```
import "./index.css";

const ProductCard = (props) => {
  const { productData } = props;
  const { title, brand, imageUrl, rating, price } = productData;

  return (
    <li className="product-item">
      <img src={imageUrl} alt="product" className="thumbnail" />
      <h1 className="title">{title}</h1>
      <p className="brand">by {brand}</p>
      <div className="product-details">
        <p className="price">Rs {price}/-</p>
        <div className="rating-container">
          <p className="rating">{rating}</p>
          <img
            src="https://assets.ccbp.in/frontend/react-js/star-img.png"
            alt="star"
            className="star"
          />
        </div>
      </div>
    </li>
  );
};
export default ProductCard;
```

File: src/components/Products/index.js

```
import AllProductsSection from "../AllProductsSection";

import Header from "../Header";

import "./index.css";

const Products = () => (
  <>
    <Header />
    <div className="product-sections">
      <AllProductsSection />
    </div>
  </>
);

export default Products;
```

File: src/components/ProtectedRoute/index.js

```
import { Route, Redirect } from "react-router-dom";
import Cookies from "js-cookie";
```

```
const ProtectedRoute = (props) => {
  const token = Cookies.get("jwt_token");
  if (token === undefined) {
    return <Redirect to="/login" />;
  }
  return <Route {...props} />;
};

export default ProtectedRoute;
```

Common Mistakes | Part 3

Concepts in Focus

- User defined functions should be in the arrow function syntax
- Using Redirect Component in Callbacks or Event Listeners
- Using history.replace() in render() instead of Redirect Component
- Missing withRouter() to have history prop
- When request object keys are incorrect
- Not Updating the State when required

1. User defined functions should be in arrow function syntax

Mistake

In the below code snippet, we have defined the onChangeUsername function as a function definition instead of an arrow function.
File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername(event){
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }

  onSubmitSuccess = jwtToken => {
    const {history} = this.props
```

```
  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
```

```
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
}
```

}

export default LoginForm

Explanation:

- In Class Component, the function definition will execute in the browser context, so the value of this will be null.
- Component Life Cycle Methods(componentDidMount,render,constructor,componentWillUnmount) are extended from React Component. As they run in React Context, there is no need to write them in arrow functions.

Solution:

File: src/components/LoginForm/index.js

```
import {Component} from 'react'

import Cookies from 'js-cookie'

import {Redirect} from 'react-router-dom'


import './index.css'


class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }
```

```
onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
```

```
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
```

```
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
```

```
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
export default LoginForm
```

2. Using Redirect Component in Callbacks or Event Listeners

Mistake:

As the Redirect Component returns JSX, it should not be used in event handlers.

File: src/components/Header/index.js

```
import {Link, withRouter} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'

const Header = props => {
  const onClickLogout = () => {
    Cookies.remove('jwt_token')
    return <Redirect to="/login" />
  }

  return (
    <nav className="nav-header">
      <div className="nav-content">
        <div className="nav-bar-mobile-logo-container">
```

```
  <img
    className="website-logo"
    src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
    alt="website logo"
  />

  <button type="button" className="nav-mobile-btn">
    <img
      src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
      alt="nav logout"
      className="nav-bar-image"
      onClick={onClickLogout}
    />
  </button>
</div>

<div className="nav-bar-large-container">
  <img
    className="website-logo"
    src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
    alt="website logo"
  />
  <ul className="nav-menu">
    <li className="nav-menu-item">
      <Link to="/" className="nav-link">
        Home
      </Link>
    </li>

    <li className="nav-menu-item">
      <Link to="/products" className="nav-link">
        Products
```

```
        </Link>
      </li>

      <li className="nav-menu-item">
        <Link to="/cart" className="nav-link">
          Cart
        </Link>
      </li>
    </ul>
    <button
      type="button"
      className="logout-desktop-btn"
      onClick={onClickLogout}
    >
      Logout
    </button>
  </div>
</div>
<div className="nav-menu-mobile">
  <ul className="nav-menu-list-mobile">
    <li className="nav-menu-item-mobile">
      <Link to="/" className="nav-link">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-icon.png"
          alt="nav home"
          className="nav-bar-image"
        />
      </Link>
    </li>

    <li className="nav-menu-item-mobile">
      <Link to="/products" className="nav-link">
```

```
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-icon.png"
              alt="nav products"
              className="nav-bar-image"
            />
          </Link>
        </li>
        <li className="nav-menu-item-mobile">
          <Link to="/cart" className="nav-link">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-icon.png"
              alt="nav cart"
              className="nav-bar-image"
            />
          </Link>
        </li>
      </ul>
    </div>
  </nav>
 )
}
export default withRouter(Header)
```

Explanation:

- Redirect Component returns JSX and it is used to render the UI in the specific Route.
- To navigate to the specific Routes in event handlers or callbacks used the history object.

Solution:

File: src/components/Header/index.js

```
import {Link, withRouter} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'
```

```
const Header = props => {
  const onClickLogout = () => {
    const {history} = props
    Cookies.remove('jwt_token')
    history.replace('/login')
  }

  return (
    <nav className="nav-header">
      <div className="nav-content">
        <div className="nav-bar-mobile-logo-container">
          <img
            className="website-logo"
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            alt="website logo"
          />

          <button type="button" className="nav-mobile-btn">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
              alt="nav logout"
              className="nav-bar-image"
              onClick={onClickLogout}
            />
          </button>
        </div>

        <div className="nav-bar-large-container">
          <img
            className="website-logo"
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
```

```
      alt="website logo"
    />
    <ul className="nav-menu">
      <li className="nav-menu-item">
        <Link to="/" className="nav-link">
          Home
        </Link>
      </li>

      <li className="nav-menu-item">
        <Link to="/products" className="nav-link">
          Products
        </Link>
      </li>

      <li className="nav-menu-item">
        <Link to="/cart" className="nav-link">
          Cart
        </Link>
      </li>
    </ul>
    <button
      type="button"
      className="logout-desktop-btn"
      onClick={onClickLogout}
    >
      Logout
    </button>
  </div>
</div>
<div className="nav-menu-mobile">
  <ul className="nav-menu-list-mobile">
```

```
        <li className="nav-menu-item-mobile">
          <Link to="/" className="nav-link">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-icon.png"
              alt="nav home"
              className="nav-bar-image"
            />
          </Link>
        </li>

        <li className="nav-menu-item-mobile">
          <Link to="/products" className="nav-link">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-icon.png"
              alt="nav products"
              className="nav-bar-image"
            />
          </Link>
        </li>
        <li className="nav-menu-item-mobile">
          <Link to="/cart" className="nav-link">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-icon.png"
              alt="nav cart"
              className="nav-bar-image"
            />
          </Link>
        </li>
      </ul>
    </div>
  </nav>
)
```

```
}

export default withRouter(Header)
```

3. Using history.replace() in render() instead of Redirect Component

Mistake:

In the render method, we have used history.replace() instead of Redirect Component and history prop is used in event handlers or callbacks to navigate to specific routes.

File: src/components/LoginForm/index.js

```
import {Component} from 'react'

import Cookies from 'js-cookie'

import {Redirect} from 'react-router-dom'


import './index.css'


class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }

  onSubmitSuccess = jwtToken => {
    const {history} = this.props
```

```
  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
```

```
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
```

```
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  const {history} = this.props
  if (jwtToken !== undefined) {
    history.replace('/')
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
```

```
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
export default LoginForm
```

Explanation:

In render() JSX is displayed and Redirect Component returns JSX. So it's a best practice to use Redirect Component while navigating in render().

Solution:

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }
```

```
onChangePassword = event => {
  this.setState({password: event.target.value})
}

onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
```

```
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
```

```
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
   return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
```

```
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
export default LoginForm
```

4. Missing withRouter() to have history prop

Mistake:

The history prop will be available for only Route Components. To use the history prop in other components, it should be wrapped with withRouter()

.

File: src/components/Header/index.js

```
import {Link} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'

const Header = props => {
  const onClickLogout = () => {
    const {history} = props
```

```
  Cookies.remove('jwt_token')
  history.replace('/login')
}

return (
  <nav className="nav-header">
    <div className="nav-content">
      <div className="nav-bar-mobile-logo-container">
        <img
          className="website-logo"
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          alt="website logo"
        />

        <button type="button" className="nav-mobile-btn">
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
            alt="nav logout"
            className="nav-bar-image"
            onClick={onClickLogout}
          />
        </button>
      </div>

      <div className="nav-bar-large-container">
        <img
          className="website-logo"
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          alt="website logo"
        />
        <ul className="nav-menu">
          <li className="nav-menu-item">
```

```
          <Link to="/" className="nav-link">
            Home
          </Link>
        </li>

        <li className="nav-menu-item">
          <Link to="/products" className="nav-link">
            Products
          </Link>
        </li>

        <li className="nav-menu-item">
          <Link to="/cart" className="nav-link">
            Cart
          </Link>
        </li>
      </ul>
      <button
        type="button"
        className="logout-desktop-btn"
        onClick={onClickLogout}
      >
        Logout
      </button>
    </div>
  </div>
  <div className="nav-menu-mobile">
    <ul className="nav-menu-list-mobile">
      <li className="nav-menu-item-mobile">
        <Link to="/" className="nav-link">
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-icon.png"
```

```
            alt="nav home"
            className="nav-bar-image"
          />
        </Link>
      </li>

      <li className="nav-menu-item-mobile">
        <Link to="/products" className="nav-link">
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-icon.png"
            alt="nav products"
            className="nav-bar-image"
          />
        </Link>
      </li>
      <li className="nav-menu-item-mobile">
        <Link to="/cart" className="nav-link">
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-icon.png"
            alt="nav cart"
            className="nav-bar-image"
          />
        </Link>
      </li>
    </ul>
  </div>
</nav>
 )
}
export default Header
```

Solution:

File: src/components/Header/index.js

```
import {Link, withRouter} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'

const Header = props => {
  const onClickLogout = () => {
    const {history} = props
    Cookies.remove('jwt_token')
    history.replace('/login')
  }

  return (
    <nav className="nav-header">
      <div className="nav-content">
        <div className="nav-bar-mobile-logo-container">
          <img
            className="website-logo"
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            alt="website logo"
          />

          <button type="button" className="nav-mobile-btn">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
              alt="nav logout"
              className="nav-bar-image"
              onClick={onClickLogout}
            />
          </button>
```

```
</div>

<div className="nav-bar-large-container">
  <img
    className="website-logo"
    src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
    alt="website logo"
  />
  <ul className="nav-menu">
    <li className="nav-menu-item">
      <Link to="/" className="nav-link">
        Home
      </Link>
    </li>

    <li className="nav-menu-item">
      <Link to="/products" className="nav-link">
        Products
      </Link>
    </li>

    <li className="nav-menu-item">
      <Link to="/cart" className="nav-link">
        Cart
      </Link>
    </li>
  </ul>
  <button
    type="button"
    className="logout-desktop-btn"
    onClick={onClickLogout}
  >
```

```
      Logout
    </button>
  </div>
</div>
<div className="nav-menu-mobile">
  <ul className="nav-menu-list-mobile">
    <li className="nav-menu-item-mobile">
      <Link to="/" className="nav-link">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-icon.png"
          alt="nav home"
          className="nav-bar-image"
        />
      </Link>
    </li>

    <li className="nav-menu-item-mobile">
      <Link to="/products" className="nav-link">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-icon.png"
          alt="nav products"
          className="nav-bar-image"
        />
      </Link>
    </li>
    <li className="nav-menu-item-mobile">
      <Link to="/cart" className="nav-link">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-icon.png"
          alt="nav cart"
          className="nav-bar-image"
        />
```

```
          </Link>
        </li>
      </ul>
    </div>
  </nav>
 )
}
export default withRouter(Header)
```

5. When request object keys are incorrect

Mistake:

For example below in the request object, we need to send username and password but userName is sent instead of the username

.

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
 state = {
   userName: '',
   password: '',
   showSubmitError: false,
   errorMsg: '',
 }

 onChangeUsername = event => {
   this.setState({username: event.target.value})
 }
```

```
onChangePassword = event => {
  this.setState({password: event.target.value})
}

onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {userName, password} = this.state
  const userDetails = {userName, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
```

```
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
```

```
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
```

```
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
export default LoginForm
```

Solution:

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
 state = {
  username: '',
  password: '',
  showSubmitError: false,
  errorMsg: '',
 }
```

```
onChangeUsername = event => {
  this.setState({username: event.target.value})
}

onChangePassword = event => {
  this.setState({password: event.target.value})
}

onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
```

```
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
```

```
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
```

```
          className="login-image"
          alt="website login"
        />
        <form className="form-container" onSubmit={this.submitForm}>
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            className="login-website-logo-desktop-image"
            alt="website logo"
          />
          <div className="input-container">{this.renderUsernameField()}</div>
          <div className="input-container">{this.renderPasswordField()}</div>
          <button type="submit" className="login-button">
            Login
          </button>
          {showSubmitError && <p className="error-message">{errorMsg}</p>}
        </form>
      </div>
    )
  }
}
export default LoginForm
```

6. Not Updating the State when required

Mistake:

On login failure, the showSubmitError state should be updated to display the error message

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
```

```
state = {
  userName: '',
  password: '',
  showSubmitError: false,
  errorMsg: '',
}

onChangeUsername = event => {
  this.setState({username: event.target.value})
}

onChangePassword = event => {
  this.setState({password: event.target.value})
}

onSubmitSuccess = jwtToken => {
  const {history} = this.props


  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
  })

  history.replace('/')
}

onSubmitFailure = errorMsg => {
  this.setState({errorMsg})
}

submitForm = async event => {
  event.preventDefault()
```

```
  const {userName, password} = this.state
  const userDetails = {userName, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
```

```
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
```

```
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-mobile-image"
          alt="website logo"
        />
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
          className="login-image"
          alt="website login"
        />
        <form className="form-container" onSubmit={this.submitForm}>
          <img
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            className="login-website-logo-desktop-image"
            alt="website logo"
          />
          <div className="input-container">{this.renderUsernameField()}</div>
          <div className="input-container">{this.renderPasswordField()}</div>
          <button type="submit" className="login-button">
            Login
          </button>
          {showSubmitError && <p className="error-message">{errorMsg}</p>}
        </form>
      </div>
    )
  }
}
export default LoginForm
```

Solution:

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
```

```
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }

  onSubmitSuccess = jwtToken => {
    const {history} = this.props


    Cookies.set('jwt_token', jwtToken, {
      expires: 30,
    })

    history.replace('/')
  }

  onSubmitFailure = errorMsg => {
```

```
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  console.log(data.error_msg)
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
```

```
        className="password-input-filed"
        value={password}
        onChange={this.onChangePassword}
        placeholder="Password"
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-filed"
        value={username}
        onChange={this.onChangeUsername}
        placeholder="Username"
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
```

```
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
```

export default LoginForm

Authentication & Authorisation | Part 4

Concepts in Focus Get Exclusive Prime Deals

1. Integrating APIs

1.1 Get Exclusive Prime Deals

- Exclusive Prime deals are for Prime Users.
- All Products are for both Prime and Non-Prime users.

2. API Call Possible Views

2.1 Success View

When the Prime User is logged in and accessed Prime Deals Section then we should show the Exclusive Prime Deals section.

2.2 Failure View

When the Non-prime User is logged in and accessed Prime Deals Section then we should show the Get Exclusive Deals section.

Reasons for API Call Failure :

- Sending UnAuthorized User credentials
- Not specifying Authorization header
- Using the wrong HTTP method

2.3 Loading View

When the Prime or Non-prime User is logged in and accessed Prime Deals Section, we should show the Loading view until data is in progress.

3. E-Commerce Application

File: src/App.js

```
import {BrowserRouter, Route, Switch, Redirect} from 'react-router-dom'


import LoginForm from './components/LoginForm'
import Home from './components/Home'
import Products from './components/Products'
import Cart from './components/Cart'
import NotFound from './components/NotFound'
import ProtectedRoute from './components/ProtectedRoute'


import './App.css'


const App = () => (
  <BrowserRouter>
    <Switch>
      <Route exact path="/login" component={LoginForm} />
      <ProtectedRoute exact path="/" component={Home} />
      <ProtectedRoute exact path="/products" component={Products} />
      <ProtectedRoute exact path="/cart" component={Cart} />
      <Route path="/not-found" component={NotFound} />
      <Redirect to="not-found" />
    </Switch>
  </BrowserRouter>
)
export default App
```

File: src/components/AllProductsSection/index.js

```
import {Component} from 'react'
import Loader from 'react-loader-spinner'
import Cookies from 'js-cookie'


import ProductCard from '../ProductCard'
```

```
import './index.css'

class AllProductsSection extends Component {
  state = {
    productsList: [],
    isLoading: false,
  }

  componentDidMount() {
    this.getProducts()
  }

  getProducts = async () => {
    this.setState({
      isLoading: true,
    })
    const jwtToken = Cookies.get('jwt_token')
    const apiUrl = 'https://apis.ccbp.in/products'
    const options = {
      headers: {
        Authorization: `Bearer ${jwtToken}`,
      },
      method: 'GET',
    }
    const response = await fetch(apiUrl, options)
    if (response.ok) {
      const fetchedData = await response.json()
      const updatedData = fetchedData.products.map(product => ({
        title: product.title,
        brand: product.brand,
        price: product.price,
        id: product.id,
```

```
      imageUrl: product.image_url,
      rating: product.rating,
    }))
    this.setState({
      productsList: updatedData,
      isLoading: false,
    })
  }
}

renderProductsList = () => {
  const {productsList} = this.state
  return (
    <>
      <h1 className="products-list-heading">All Products</h1>
      <ul className="products-list">
        {productsList.map(product => (
          <ProductCard productData={product} key={product.id} />
        ))}
      </ul>
    </>
  )
}

renderLoader = () => (
  <div className="products-loader-container">
    <Loader type="ThreeDots" color="#0b69ff" height="50" width="50" />
  </div>
)

render() {
  const {isLoading} = this.state
```

```
    return isLoading ? this.renderLoader() : this.renderProductsList()
  }
}
export default AllProductsSection
```

File: src/components/Cart/index.js

```
import Header from '../Header'
import './index.css'

const Cart = () => (
  <>
    <Header />
    <div className="cart-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-img.png"
        alt="cart"
        className="cart-img"
      />
    </div>
  </>
)
export default Cart
```

File: src/components/Header/index.js

```
import {Link, withRouter} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'

const Header = props => {
  const onClickLogout = () => {
    Cookies.remove('jwt_token')
    const {history} = props
    history.replace('/login')
  }
  return (
    <nav className="nav-header">
      <div className="nav-content">
        <Link to="/">
          <img
            className="website-logo"
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
```

```
      alt="website logo"
    />
  </Link>
  <ul className="nav-menu">
    <Link to="/" className="nav-link">
      <li>Home</li>
    </Link>
    <Link to="/products" className="nav-link">
      <li>Products</li>
    </Link>
    <Link to="/cart" className="nav-link">
      <li>Cart</li>
    </Link>
  </ul>
  <button
    type="button"
    className="logout-desktop-btn"
    onClick={onClickLogout}
  >
    Logout
  </button>
  <button
    type="button"
    className="logout-mobile-btn"
    onClick={onClickLogout}
  >
    <img
      src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
      alt="logout icon"
      className="logout-icon"
    />
  </button>
</div>
<div className="nav-menu-mobile">
  <ul className="nav-menu-list-mobile">
    <Link to="/">
      <li className="nav-menu-item-mobile">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-icon.png"
          alt="nav home"
          className="nav-bar-image"
        />
      </li>
    </Link>
    <Link to="/products">
      <li className="nav-menu-item-mobile">
```

```
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-icon.png"
              alt="nav products"
              className="nav-bar-image"
            />
          </li>
        </Link>
        <Link to="/cart">
          <li className="nav-menu-item-mobile">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-icon.png"
              alt="nav cart"
              className="nav-bar-image"
            />
          </li>
        </Link>
      </ul>
    </div>
  </nav>
 )
}
export default withRouter(Header)
```

File: src/components/Home/index.js

```
import Cookies from 'js-cookie'
import {Redirect, Link} from 'react-router-dom'

import Header from '../Header'
import './index.css'

const Home = () => {
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken === undefined) {
    return <Redirect to="/login" />
  }

  return (
```

```
<>
  <Header />
  <div className="home-container">
    <div className="home-content">
      <h1 className="home-heading">Clothes That Get YOU Noticed</h1>
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
        alt="clothes to be noticed"
        className="home-mobile-img"
      />
      <p className="home-description">
        Fashion is part of the daily air and it does not quite help that it
        changes all the time. Clothes have always been a marker of the era
        and we are in a revolution. Your fashion makes you been seen and
        heard that way you are. So, celebrate the seasons new and exciting
        fashion in your own way.
      </p>
      <Link to="/products">
        <button type="button" className="shop-now-button">
          Shop Now
        </button>
      </Link>
    </div>
    <img
      src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
      alt="dresses to be noticed"
      className="home-desktop-img"
    />
```

```
      </div>
    </>
  )
}

export default Home
```

File: src/components/LoginForm/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }

  onSubmitSuccess = jwtToken => {
    const {history} = this.props
```

```
  Cookies.set('jwt_token', jwtToken, {
    expires: 30,
    path: '/',
  })
  history.replace('/')
}

onSubmitFailure = errorMsg => {
  console.log(errorMsg)
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
```

```
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
      </label>
      <input
        type="password"
        id="password"
        className="password-input-field"
        value={password}
        onChange={this.onChangePassword}
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-field"
        value={username}
        onChange={this.onChangeUsername}
      />
    </>
```

```
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
```

```
        {showSubmitError && <p className="error-message">*{errorMsg}</p>}
      </form>
    </div>
  )
 }
}
export default LoginForm
```

File: src/components/NotFound/index.js

```
import './index.css'

const NotFound = () => (
 <div className="not-found-container">
  <img
   src="https://assets.ccbp.in/frontend/react-js/not-found-blog-img.png"
   alt="not-found"
   className="not-found-img"
  />
 </div>
)
export default NotFound
```

File: src/components/PrimeDealsSection/index.js

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import Loader from 'react-loader-spinner'

import ProductCard from '../ProductCard'
import './index.css'

const apiStatusConstants = {
 initial: 'INITIAL',
 success: 'SUCCESS',
```

```
  failure: 'FAILURE',
  inProgress: 'IN_PROGRESS',
}

class PrimeDealsSection extends Component {
  state = {
    primeDeals: [],
    apiStatus: apiStatusConstants.initial,
  }

  componentDidMount() {
    this.getPrimeDeals()
  }

  getPrimeDeals = async () => {
    this.setState({
      apiStatus: apiStatusConstants.inProgress,
    })

    const jwtToken = Cookies.get('jwt_token')

    const apiUrl = 'https://apis.ccbp.in/prime-deals'
    const options = {
      headers: {
        Authorization: `Bearer ${jwtToken}`,
      },
      method: 'GET',
    }
    const response = await fetch(apiUrl, options)
    if (response.ok === true) {
      const fetchedData = await response.json()
      const updatedData = fetchedData.prime_deals.map(product => ({
```

```
      title: product.title,
      brand: product.brand,
      price: product.price,
      id: product.id,
      imageUrl: product.image_url,
      rating: product.rating,
    }))
    this.setState({
      primeDeals: updatedData,
      apiStatus: apiStatusConstants.success,
    })
  }
  if (response.status === 401) {
    this.setState({
      apiStatus: apiStatusConstants.failure,
    })
  }
}

renderPrimeDealsList = () => {
  const {primeDeals} = this.state
  return (
    <div>
      <h1 className="primedeals-list-heading">Exclusive Prime Deals</h1>
      <ul className="products-list">
        {primeDeals.map(product => (
          <ProductCard productData={product} key={product.id} />
        ))}
      </ul>
    </div>
  )
}
```

```
  renderPrimeDealsFailureView = () => (
    <img
      src="https://assets.ccbp.in/frontend/react-js/exclusive-deals-banner-img.png"
      alt="Register Prime"
      className="register-prime-image"
    />
  )

  renderLoadingView = () => (
    <div className="products-loader-container">
      <Loader type="ThreeDots" color="#0b69ff" height="50" width="50" />
    </div>
  )

  render() {
    const {apiStatus} = this.state
    switch (apiStatus) {
      case apiStatusConstants.success:
        return this.renderPrimeDealsList()
      case apiStatusConstants.failure:
        return this.renderPrimeDealsFailureView()
      case apiStatusConstants.inProgress:
        return this.renderLoadingView()
      default:
        return null
    }
  }
}
export default PrimeDealsSection
```

File: src/components/ProductCard/index.js

```
import './index.css'
```

```
const ProductCard = props => {
  const {productData} = props
  const {title, brand, imageUrl, rating, price} = productData

  return (
    <li className="product-item">
      <img src={imageUrl} alt="product" className="thumbnail" />
      <h1 className="title">{title}</h1>
      <p className="brand">by {brand}</p>
      <div className="product-details">
        <p className="price">Rs {price}/-</p>
        <div className="rating-container">
          <p className="rating">{rating}</p>
          <img
            src="https://assets.ccbp.in/frontend/react-js/star-img.png"
            alt="star"
            className="star"
          />
        </div>
      </div>
    </li>
  )
}
export default ProductCard
```

File: src/components/Products/index.js

```
import AllProductsSection from '../AllProductsSection'
import PrimeDealsSection from '../PrimeDealsSection'

import Header from '../Header'

import './index.css'
```

```
const Products = () => (
  <>
    <Header />
    <div className="product-sections">
      <PrimeDealsSection />
      <AllProductsSection />
    </div>
  </>
)
export default Products
```

File: src/components/ProtectedRoute/index.js

```
import {Redirect, Route} from 'react-router-dom'
import Cookie from 'js-cookie'

const ProtectedRoute = props => {
  const token = Cookie.get('jwt_token')
  if (token === undefined) {
    return <Redirect to="/login" />
  }
  return <Route {...props} />
}
export default ProtectedRoute
```

4. Best Practices

4.1 State Variable-isLoading

isLoading is used to handle Success View, Loading View only.

```
render() {
   const {isLoading} = this.state
   return <>
   { isLoading ? this.renderLoader() : this.renderProductsList() }
```

```
  </>
 }
```

4.2 State Variable-apiStatus

apiStatus is used to handle Success, Failure and Loading Views.

```
switch (apiStatus) {
  case apiStatusConstants.success:
    return this.renderPrimeDealsList()
  ...
  ...
  default:
    return null
}
```

4.3 Adding Initial State

Instead of adding empty strings in initial state we can add initial in **apiStatusConstants**.

File: src/components/PrimeDealsSection/index.js

```
...
const apiStatusConstants = {
  initial: "INITIAL",
  inProgress: "IN_PROGRESS",
  success: "SUCCESS",
  failure: "FAILURE",
};
class PrimeDealsSection extends Component {
  state = {
    primeDeals: [],
    apiStatus: apiStatusConstants.initial,
  }

  renderPrimeDeals = () => {
    const { apiStatus } = this.state;
    switch (apiStatus) {
```

```
      case apiStatusConstants.success:
        return this.renderPrimeDealsSuccessView();
      case apiStatusConstants.failure:
        return this.renderPrimeDealsFailureView();
      case apiStatusConstants.inProgress:
        return this.renderLoader();
      default:
        return null;
    }
  };

  render() {
    return <>{this.renderPrimeDeals()}</>
  }
}
export default PrimeDealsSection
```

Sorting Products

Concepts in Focus

- setState() Callback Function
- React Icons
    - Installing React Icons
    - Searching React Icons
    - Importing React Icons
- Sorting Products

1. setState() - Callback Function

- The setState() is asynchronous, it takes an optional callback parameter that can be used to make updates after the state is changed.

Syntax

```
this.setState({property1: value1,...}, callbackFunction)
```

2. React Icons

- React-icons is a third-party package contains bundle of icons like bootstrap, font awesome, material icons etc..,
- Check react-icons website here .

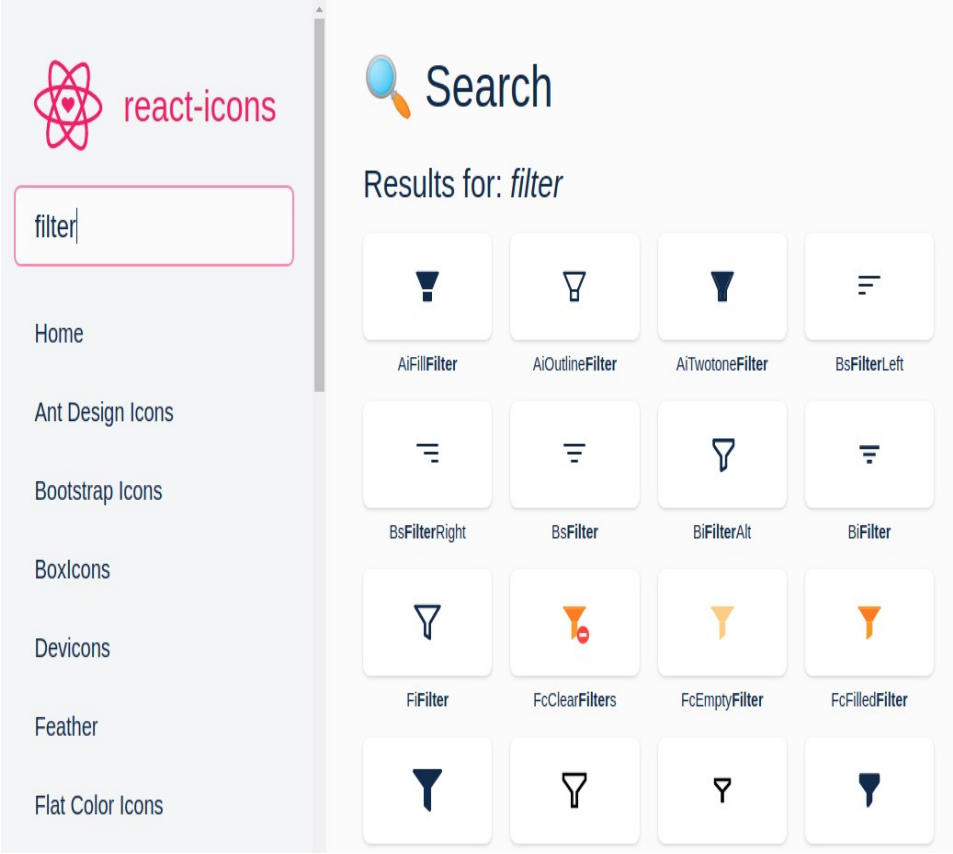
2.1 Installing React Icons

- This below command installs all the react-icons library in your React project.

npm install react-icons

2.2 Searching React Icons

- Click on the Icon to copy the **Icon Name**.



2.3 Importing React Icons

- The First letters of the icon indicates the **category** of the Icon.
- Each category of **Icons** have import statements **separately**, go to the category and **copy** the import statement.

Example:

```
import { BsFilterRight } from "react-icons/bs";
import { FaFacebookF } from "react-icons/fa";
import { MdDelete } from "react-icons/md";

const ReactIcon = (props) => {
  return (
    <div>
```

```
      <BsFilterRight />
      <FaFacebookF />
      <MdDelete />
    </div>
  );
};
export default ReactIcon;
```

3. Sorting Products

File: src/components/AllProductsSection

```
import {Component} from 'react'
import Loader from 'react-loader-spinner'
import Cookies from 'js-cookie'

import ProductCard from '../ProductCard'
import ProductsHeader from '../ProductsHeader'
import './index.css'

const sortbyOptions = [
  {
    optionId: 'PRICE_HIGH',
    displayText: 'Price (High-Low)',
  },
  {
    optionId: 'PRICE_LOW',
    displayText: 'Price (Low-High)',
  },
]

class AllProductsSection extends Component {
  state = {
    productsList: [],
```

```
  isLoading: false,
  activeOptionId: sortbyOptions[0].optionId,
}

componentDidMount() {
  this.getProducts()
}

getProducts = async () => {
  this.setState({
    isLoading: true,
  })
  const jwtToken = Cookies.get('jwt_token')
  const {activeOptionId} = this.state
  const apiUrl = `https://apis.ccbp.in/products?sort_by=${activeOptionId}`
  const options = {
    headers: {
      Authorization: `Bearer ${jwtToken}`,
    },
    method: 'GET',
  }
  const response = await fetch(apiUrl, options)
  if (response.ok) {
    const fetchedData = await response.json()
    const updatedData = fetchedData.products.map(product => ({
      title: product.title,
      brand: product.brand,
      price: product.price,
      id: product.id,
      imageUrl: product.image_url,
      rating: product.rating,
    }))
```

```
    this.setState({
      productsList: updatedData,
      isLoading: false,
    })
  }
}

updateActiveOptionId = activeOptionId => {
  this.setState({activeOptionId}, this.getProducts)
}

renderProductsList = () => {
  const {productsList, activeOptionId} = this.state
  return (
    <>
      <ProductsHeader
        activeOptionId={activeOptionId}
        sortbyOptions={sortbyOptions}
        updateActiveOptionId={this.updateActiveOptionId}
      />
      <ul className="products-list">
        {productsList.map(product => (
          <ProductCard productData={product} key={product.id} />
        ))}
      </ul>
    </>
  )
}

renderLoader = () => (
  <div className="products-loader-container">
    <Loader type="ThreeDots" color="#0b69ff" height="50" width="50" />
```

```
    </div>
  )

  render() {
    const {isLoading} = this.state
    return isLoading ? this.renderLoader() : this.renderProductsList()
  }
}
export default AllProductsSection
```

File: src/components/Cart

```
import Header from '../Header'
import './index.css'

const Cart = () => (
  <>
    <Header />
    <div className="cart-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-img.png"
        alt="cart"
        className="cart-img"
      />
    </div>
  </>
)
export default Cart
```

File: src/components/Header

```
import {Link, withRouter} from 'react-router-dom'

import Cookies from 'js-cookie'

import './index.css'
```

```
const Header = props => {
  const onClickLogout = () => {
    Cookies.remove('jwt_token')
    const {history} = props
    history.replace('/login')
  }
  return (
    <nav className="nav-header">
      <div className="nav-content">
        <Link to="/">
          <img
            className="website-logo"
            src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
            alt="website logo"
          />
        </Link>
        <ul className="nav-menu">
          <Link to="/" className="nav-link">
            <li>Home</li>
          </Link>
          <Link to="/products" className="nav-link">
            <li>Products</li>
          </Link>
          <Link to="/cart" className="nav-link">
            <li>Cart</li>
          </Link>
        </ul>
        <button
          type="button"
          className="logout-desktop-btn"
          onClick={onClickLogout}
```

```
  >
    Logout
  </button>
  <button
    type="button"
    className="logout-mobile-btn"
    onClick={onClickLogout}
  >
    <img
      src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-log-out-img.png"
      alt="logout icon"
      className="logout-icon"
    />
  </button>
</div>
<div className="nav-menu-mobile">
  <ul className="nav-menu-list-mobile">
    <Link to="/">
      <li className="nav-menu-item-mobile">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-icon.png"
          alt="nav home"
          className="nav-bar-image"
        />
      </li>
    </Link>
    <Link to="/products">
      <li className="nav-menu-item-mobile">
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-products-icon.png"
          alt="nav products"
          className="nav-bar-image"
```

```
            />
          </li>
        </Link>
        <Link to="/cart">
          <li className="nav-menu-item-mobile">
            <img
              src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-cart-icon.png"
              alt="nav cart"
              className="nav-bar-image"
            />
          </li>
        </Link>
      </ul>
    </div>
  </nav>
 )
}
export default withRouter(Header)
```

File: src/components/Home

```
import Cookies from 'js-cookie'
import {Redirect, Link} from 'react-router-dom'

import Header from '../Header'
import './index.css'

const Home = () => {
 const jwtToken = Cookies.get('jwt_token')
 if (jwtToken === undefined) {
  return <Redirect to="/login" />
 }

 return (
```

```
  <>
    <Header />
    <div className="home-container">
      <div className="home-content">
        <h1 className="home-heading">Clothes That Get YOU Noticed</h1>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
          alt="dresses to be noticed"
          className="home-mobile-img"
        />
        <p className="home-description">
          Fashion is part of the daily air and it does not quite help that it
          changes all the time. Clothes have always been a marker of the era
          and we are in a revolution. Your fashion makes you been seen and
          heard that way you are. So, celebrate the seasons new and exciting
          fashion in your own way.
        </p>

        <Link to="/products">
          <button type="button" className="shop-now-button">
            Shop Now
          </button>
        </Link>
      </div>
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-home-img.png"
        alt="dresses to be noticed"
        className="home-desktop-img"
      />
    </div>
  </>
)
```

```
}
export default Home
```

File: src/components/LoginForm

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import {Redirect} from 'react-router-dom'

import './index.css'

class LoginForm extends Component {
  state = {
    username: '',
    password: '',
    showSubmitError: false,
    errorMsg: '',
  }

  onChangeUsername = event => {
    this.setState({username: event.target.value})
  }

  onChangePassword = event => {
    this.setState({password: event.target.value})
  }

  onSubmitSuccess = jwtToken => {
    const {history} = this.props

    Cookies.set('jwt_token', jwtToken, {
      expires: 30,
      path: '/',
    })
```

```
  history.replace('/')
}

onSubmitFailure = errorMsg => {
  console.log(errorMsg)
  this.setState({showSubmitError: true, errorMsg})
}

submitForm = async event => {
  event.preventDefault()
  const {username, password} = this.state
  const userDetails = {username, password}
  const url = 'https://apis.ccbp.in/login'
  const options = {
    method: 'POST',
    body: JSON.stringify(userDetails),
  }
  const response = await fetch(url, options)
  const data = await response.json()
  if (response.ok === true) {
    this.onSubmitSuccess(data.jwt_token)
  } else {
    this.onSubmitFailure(data.error_msg)
  }
}

renderPasswordField = () => {
  const {password} = this.state
  return (
    <>
      <label className="input-label" htmlFor="password">
        PASSWORD
```

```
      </label>
      <input
        type="password"
        id="password"
        className="password-input-field"
        value={password}
        onChange={this.onChangePassword}
      />
    </>
  )
}

renderUsernameField = () => {
  const {username} = this.state
  return (
    <>
      <label className="input-label" htmlFor="username">
        USERNAME
      </label>
      <input
        type="text"
        id="username"
        className="username-input-field"
        value={username}
        onChange={this.onChangeUsername}
      />
    </>
  )
}

render() {
  const {showSubmitError, errorMsg} = this.state
```

```
  const jwtToken = Cookies.get('jwt_token')
  if (jwtToken !== undefined) {
    return <Redirect to="/" />
  }
  return (
    <div className="login-form-container">
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
        className="login-website-logo-mobile-image"
        alt="website logo"
      />
      <img
        src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-login-img.png"
        className="login-image"
        alt="website login"
      />
      <form className="form-container" onSubmit={this.submitForm}>
        <img
          src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-logo-img.png"
          className="login-website-logo-desktop-image"
          alt="website logo"
        />
        <div className="input-container">{this.renderUsernameField()}</div>
        <div className="input-container">{this.renderPasswordField()}</div>
        <button type="submit" className="login-button">
          Login
        </button>
        {showSubmitError && <p className="error-message">*{errorMsg}</p>}
      </form>
    </div>
  )
}
```

```
}
export default LoginForm
```

File: src/components/NotFound

```
import './index.css'

const NotFound = () => (
  <div className="not-found-container">
    <img
      src="https://assets.ccbp.in/frontend/react-js/not-found-blog-img.png"
      alt="not-found"
      className="not-found-img"
    />
  </div>
)
export default NotFound
```

File: src/components/PrimeDealsSection

```
import {Component} from 'react'
import Cookies from 'js-cookie'
import Loader from 'react-loader-spinner'

import ProductCard from '../ProductCard'
import './index.css'

const apiStatusConstants = {
  initial: 'INITIAL',
  success: 'SUCCESS',
  failure: 'FAILURE',
  inProgress: 'IN_PROGRESS',
}

class PrimeDealsSection extends Component {
  state = {
```

```
  primeDeals: [],
  apiStatus: apiStatusConstants.initial,
}

componentDidMount() {
  this.getPrimeDeals()
}

getPrimeDeals = async () => {
  this.setState({
    apiStatus: apiStatusConstants.inProgress,
  })

  const jwtToken = Cookies.get('jwt_token')

  const apiUrl = 'https://apis.ccbp.in/prime-deals'
  const options = {
    headers: {
      Authorization: `Bearer ${jwtToken}`,
    },
    method: 'GET',
  }
  const response = await fetch(apiUrl, options)
  if (response.ok === true) {
    const fetchedData = await response.json()
    const updatedData = fetchedData.prime_deals.map(product => ({
      title: product.title,
      brand: product.brand,
      price: product.price,
      id: product.id,
      imageUrl: product.image_url,
      rating: product.rating,
```

```
    }))
    this.setState({
      primeDeals: updatedData,
      apiStatus: apiStatusConstants.success,
    })
  } else if (response.status === 401) {
    this.setState({
      apiStatus: apiStatusConstants.failure,
    })
  }
}

renderPrimeDealsList = () => {
  const {primeDeals} = this.state
  return (
    <div>
      <h1 className="primedeals-list-heading">Exclusive Prime Deals</h1>
      <ul className="products-list">
        {primeDeals.map(product => (
          <ProductCard productData={product} key={product.id} />
        ))}
      </ul>
    </div>
  )
}

renderPrimeDealsFailureView = () => (
  <img
    src="https://assets.ccbp.in/frontend/react-js/exclusive-deals-banner-img.png"
    alt="Register Prime"
    className="register-prime-image"
  />
```

```
  )

  renderLoadingView = () => (
    <div className="products-loader-container">
      <Loader type="ThreeDots" color="#0b69ff" height="50" width="50" />
    </div>
  )

  render() {
    const {apiStatus} = this.state
    switch (apiStatus) {
      case apiStatusConstants.success:
        return this.renderPrimeDealsList()
      case apiStatusConstants.failure:
        return this.renderPrimeDealsFailureView()
      case apiStatusConstants.inProgress:
        return this.renderLoadingView()
      default:
        return null
    }
  }
}
export default PrimeDealsSection
```

File: src/components/ProductCard

```
import './index.css'

const ProductCard = props => {
  const {productData} = props
  const {title, brand, imageUrl, rating, price} = productData

  return (
    <li className="product-item">
```

```
      <img src={imageUrl} alt="product" className="thumbnail" />
      <h1 className="title">{title}</h1>
      <p className="brand">by {brand}</p>
      <div className="product-details">
        <p className="price">Rs {price}/-</p>
        <div className="rating-container">
          <p className="rating">{rating}</p>
          <img
            src="https://assets.ccbp.in/frontend/react-js/star-img.png"
            alt="star"
            className="star"
          />
        </div>
      </div>
    </li>
  )
}
export default ProductCard
```

File: src/components/Products

```
import AllProductsSection from '../AllProductsSection'
import PrimeDealsSection from '../PrimeDealsSection'

import Header from '../Header'

import './index.css'

const Products = () => (
  <>
    <Header />
    <div className="product-sections">
      <PrimeDealsSection />
      <AllProductsSection />
```

```
    </div>
  </>
)
export default Products
```

File: src/components/ProductsHeader

```
import {BsFilterRight} from 'react-icons/bs'

import './index.css'

const ProductsHeader = props => {
  const {sortbyOptions, activeOptionId, updateActiveOptionId} = props
  const onChangeSortby = event => {
    updateActiveOptionId(event.target.value)
  }

  return (
    <div className="products-header">
      <h1 className="products-list-heading">All Products</h1>
      <div className="sort-by-container">
        <BsFilterRight className="sort-by-icon" />
        <h1 className="sort-by">Sort by</h1>
        <select
          className="sort-by-select"
          value={activeOptionId}
          onChange={onChangeSortby}
        >
          {sortbyOptions.map(eachOption => (
            <option
              key={eachOption.optionId}
              value={eachOption.optionId}
              className="select-option"
```

```
          >
            {eachOption.displayText}
          </option>
        ))}
      </select>
    </div>
  </div>
 )
}
export default ProductsHeader
```

File: src/components/ProtectedRoute

```
import {Redirect, Route} from 'react-router-dom'
import Cookie from 'js-cookie'

const ProtectedRoute = props => {
 const token = Cookie.get('jwt_token')
 if (token === undefined) {
  return <Redirect to="/login" />
 }
 return <Route {...props} />
}
export default ProtectedRoute
```

File: src/App.js

```
import {BrowserRouter, Route, Switch, Redirect} from 'react-router-dom'

import LoginForm from './components/LoginForm'
import Home from './components/Home'
import Products from './components/Products'
import Cart from './components/Cart'
import NotFound from './components/NotFound'
import ProtectedRoute from './components/ProtectedRoute'
```

```
import './App.css'


const App = () => (
  <BrowserRouter>
    <Switch>
      <Route exact path="/login" component={LoginForm} />
      <ProtectedRoute exact path="/" component={Home} />
      <ProtectedRoute exact path="/products" component={Products} />
      <ProtectedRoute exact path="/cart" component={Cart} />
      <Route path="/not-found" component={NotFound} />
      <Redirect to="not-found" />
    </Switch>
  </BrowserRouter>
)
export default App
```

More React Concepts

Concepts in Focus

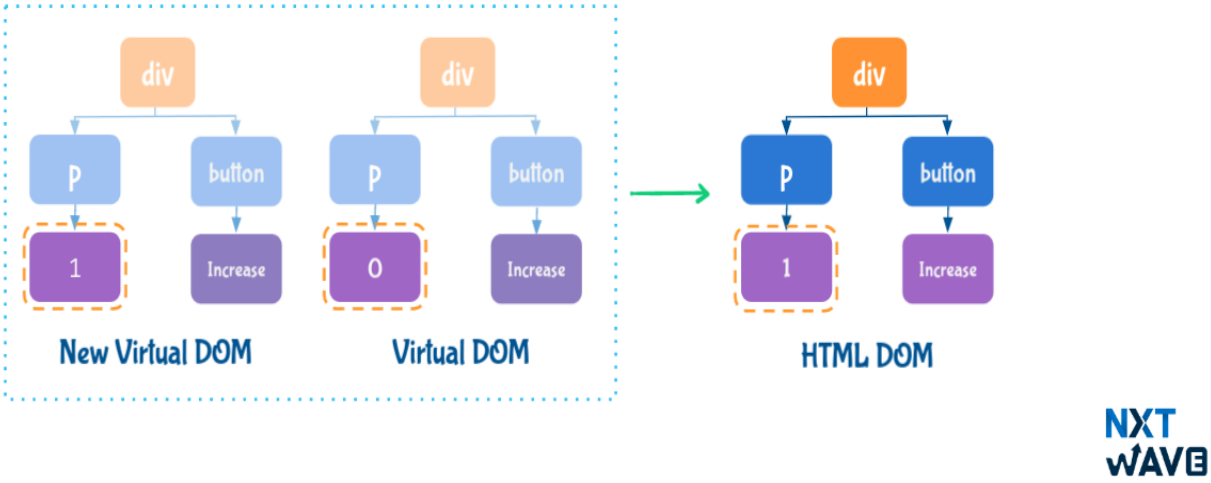
- Reconciliation
    - Virtual DOM
- React Batch Updating
- setState() Syntax
    - Object Syntax
    - Callback Syntax
    - Object Syntax vs Callback Syntax
- Children Prop
    - Passing Text as Children
    - Accessing Children
- Controlled vs Uncontrolled Input
    - Controlled Input
    - Uncontrolled Input
- Props vs State
- State should be minimal
- Keys
    - Keys in Lists

1. Reconciliation

1.1 Virtual DOM

A Virtual DOM is a JS object, which is the virtual representation of an HTML DOM.

Whenever new elements are added to the UI, a virtual DOM is created.

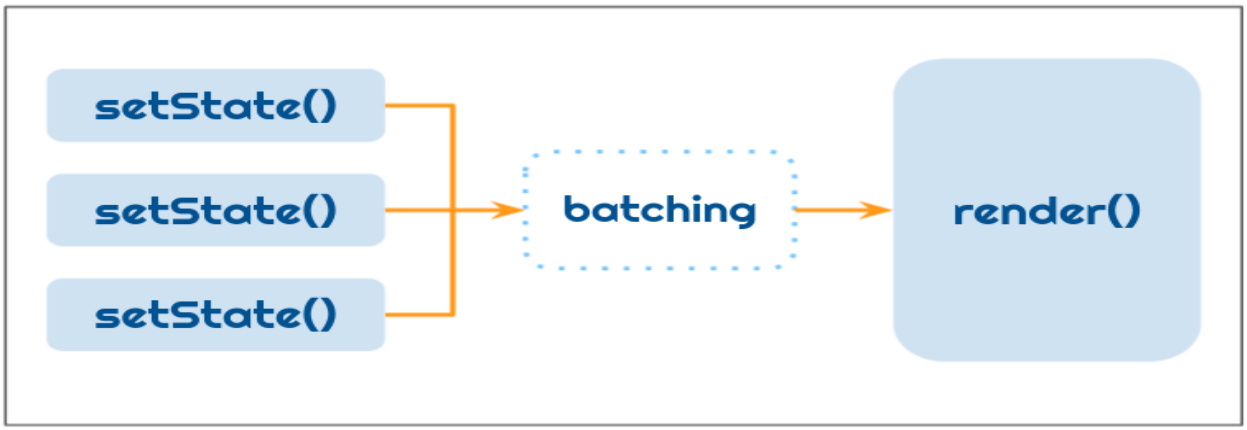


React compares new virtual DOM with current virtual DOM, and the difference will be efficiently updated to HTML DOM. So, the Virtual DOM helps to render the UI more performantly.

React only updates what's necessary. This process is known as **Reconciliation**.

2. React Batch Updating

React combines multiple setState() calls into single update.



Example:

import { Component } from "react"

```
class Counter extends Component {
  state = { count: 0 }

  onIncrement = () => {
    this.setState((prevState) => ({ count: prevState.count + 1 }))
    this.setState((prevState) => ({ count: prevState.count + 1 }))
    this.setState((prevState) => ({ count: prevState.count + 1 }))
  }

  render() {
    const { count } = this.state
    console.log("render() called")
    return (
      <>
        <p className="count">Count {count}</p>
        <button onClick={this.onIncrement}>Increase</button>
      </>
    )
  }
}
export default Counter
```

In the above example, the render is called only once as React combines multiple setState() calls into a single update.

3. setState() Syntax

3.1 Object Syntax

Object Syntax is used while updating the state to a value.

```
this.setState({
  count: 5,
})
```

3.2 Callback Syntax

Callback Syntax is used while updating the state to a value that is computed based on the previous state.

```
this.setState((prevState) => ({ count: prevState.count + 1 }))
```

3.3 Object Syntax vs Callback Syntax

Object Syntax:

```
import { Component } from "react"

class Counter extends Component {
  state = { count: 0 }

  onIncrement = () => {
    this.setState({ count: this.state.count + 1 })
    this.setState({ count: this.state.count + 1 })
    this.setState({ count: this.state.count + 1 })
  }

  render() {
    const { count } = this.state
    return (
      <>
        <p className="count">Count {count}</p>
        <button onClick={this.onIncrement}>Increase</button>
      </>
    )
  }
}

export default Counter
```

When the HTML button element is clicked, **the value of the state variable count is 1**.

When the Object Syntax is used, this.state.count is same for every setState statement as setState is asynchronous.

Callback Syntax:

```
import { Component } from "react"

class Counter extends Component {
  state = { count: 0 }

  onIncrement = () => {
    this.setState((prevState) => ({ count: prevState.count + 1 }))
    this.setState((prevState) => ({ count: prevState.count + 1 }))
    this.setState((prevState) => ({ count: prevState.count + 1 }))
  }

  render() {
    const { count } = this.state
    return (
      <>
        <p className="count">Count {count}</p>
        <button onClick={this.onIncrement}>Increase</button>
```

```
      </>
    )
  }
}

export default Counter
```

When the HTML button element is clicked, **the value of the state variable count is 3**.

The setState callback function receive prevState as an argument. So, it will receive an updated state value when it is executed.

4. Children Prop

The **children** is a special prop that is automatically passed to every React Component.

The JSX Element/Text included in between the opening and closing tags are considered children.

4.1 Passing Text as Children

File: src/components/Post/index.js

```
import "./index.css"
import SocialButton from "./SocialButton"

const Post = () => (
  <div className="post-container">
    <p className="paragraph">Hooks are a new addition</p>
    <SocialButton>Like</SocialButton>
  </div>
)

export default Post
```

4.2 Accessing Children

File: src/components/SocialButton/index.js

```
import "./index.css"

const SocialButton = (props) => (
  <button className="social-button">{props.children}</button>
)

export default SocialButton
```

5. Controlled vs Uncontrolled Input

In React, the Input Element value can be handled in two ways:

- Controlled Input
- Uncontrolled Input

5.1 Controlled Input

If the Input Element
value
is handled by a React State, then it is called **Controlled Input**.

It is the React Suggested way to handle the Input Element value.

Example:

```
import { Component } from "react"

class App extends Component {
  state = {
    searchInput: "",
  }

  onChangeSearchInput = (event) => {
    this.setState({
      searchInput: event.target.value,
    })
  }

  render() {
    const { searchInput } = this.state
    return (
      <>
        <input
          type="text"
          onChange={this.onChangeSearchInput}
          value={searchInput}
        />
        <p>{searchInput}</p>
      </>
    )
  }
}

export default App
```

5.2 Uncontrolled Input

If the Input Element value is handled by the browser itself, then it is called **Uncontrolled Input**.
Example:

```
import { Component } from "react"

class App extends Component {
  state = {
    searchInput: "",
  }

  onChangeSearchInput = (event) => {
    this.setState({
```

```
      searchInput: event.target.value,
    })
  }

  render() {
    const { searchInput } = this.state
    return (
      <>
        <input type="text" onChange={this.onChangeSearchInput} />
        <p>{searchInput}</p>
      </>
    )
  }
}

export default App
```

6. Props vs State

- The props and state are plain JS objects.
- The props and state changes trigger the render method.

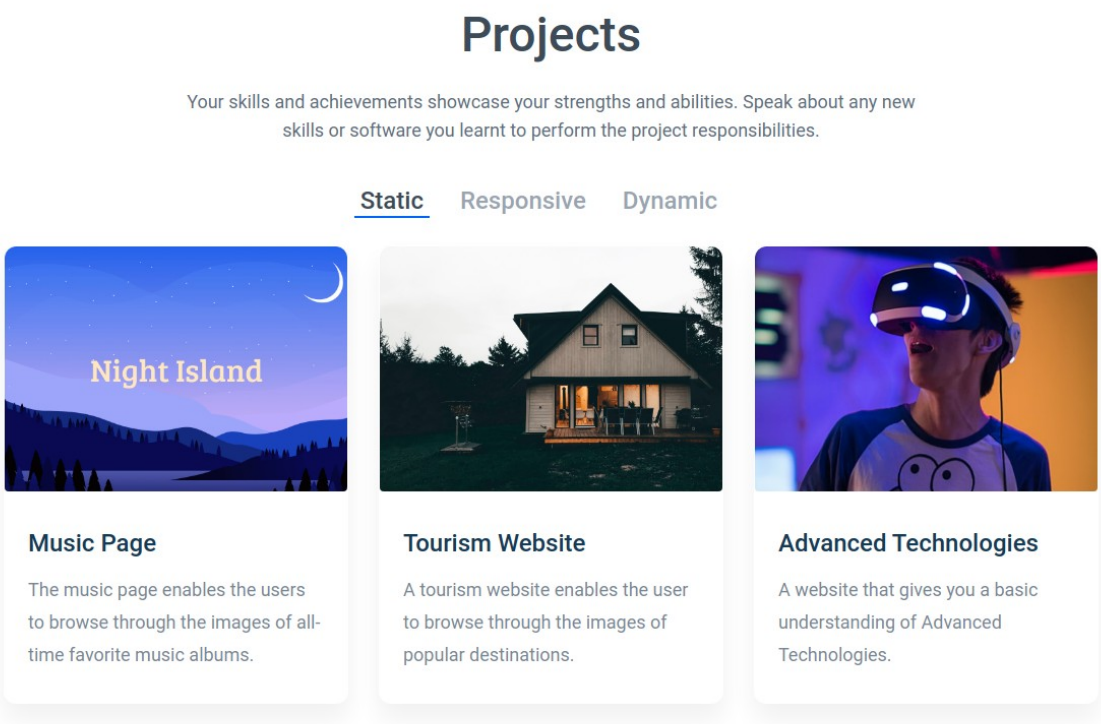
Props	State
Props get passed to the component, similar to function parameters	State is created and managed within the component, similar to a variable declared within the function
Props are used to pass data and event handlers down to the child components	State is used to store the component's data that changes over time
Props are immutable. A component cannot change the props	State should be immutable

7. State should be minimal

We need the state to make the applications interactive.

To build the application correctly, you first need to think of the minimal set of the state that the application needs.

Let's take an example application, **Projects**.



Think of all the pieces of data in the Projects Application. We have:

- The original list of projects
- The active tab item
    - The text decoration of the text - Underline
    - The color of the text
- The filtered list of projects

Let's go through each one and figure out which one is state. Ask three questions about each piece of data:

- Is it passed in from a parent via props? If so, it probably isn't state.
- Does it remain unchanged over time? If so, it probably isn't state.
- Can it be computed based on any other state or props in the component? If so, it isn't state.

Let's figure out which one is state in our application:

Data	Is State?	Reason
The original list of projects	No	It is passed as props
The active tab item	Yes	It changes over time and can't be computed from anything
The filtered list of projects	No	It can be computed by combining the original list of projects with the active tab item
The text decoration of the text - Underline	No	It can be derived from the active tab item
The The color of the text	No	It can be derived from the active tab item

So finally, the state is:

- The active tab item

```
state = {
  activeTabId: tabsList[0].tabId,
}
```

8. Keys

Keys help React identify which items have been changed, added, or removed.

Keys should be given to the elements inside the array to give a stable identity.

```
const usersList = [
  {
    uniqueNo: 1,
    name: "Rahul",
  },
  {
    uniqueNo: 2,
    name: "Praneetha",
  },
  {
    uniqueNo: 3,
    name: "Varakumar",
  },
]
const listItems = usersList.map((user) => (
  <li key={user.uniqueNo}>{user.name}</li>
))
```

8.1 Keys in Lists

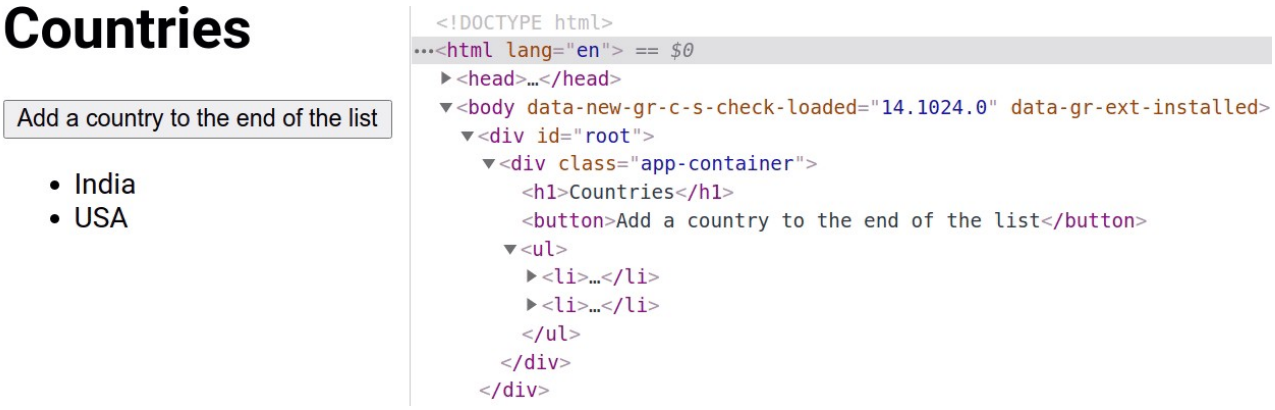
When the state of a component changes, React updates the virtual DOM tree. Once the virtual DOM has been updated, React then compares the new virtual DOM with the current virtual DOM.

Once React knows which virtual DOM objects have changed, then React updates only those objects, in the HTML DOM.

React compares the virtual DOMs if there is a change in the node, node type, or the attributes passed to the node. But there is a problematic case by only looking at the node or it's attributes. i.e. Lists.

Lists will have the same node types and attributes. Hence the list items can't be uniquely identified if they have been changed, added, or removed.

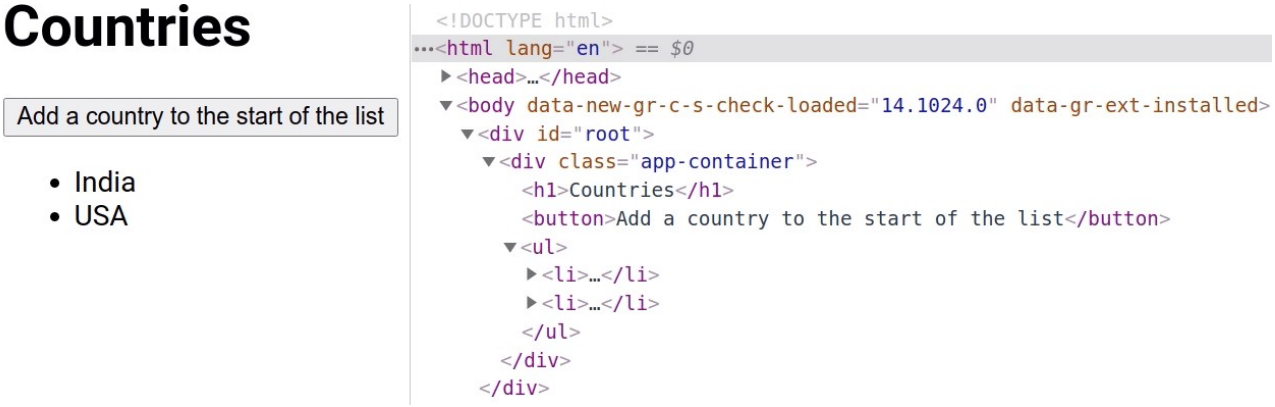
Example 1:



When a new list item is added to the end of the list,

- React creates a new virtual DOM with the three list items.
- React compares the first two list items in the current virtual DOM and the new virtual DOM.
- Since the two list items in both the previous and current virtual DOMs are matched, React creates only the last list item at the end in the HTML DOM.

Example-2:

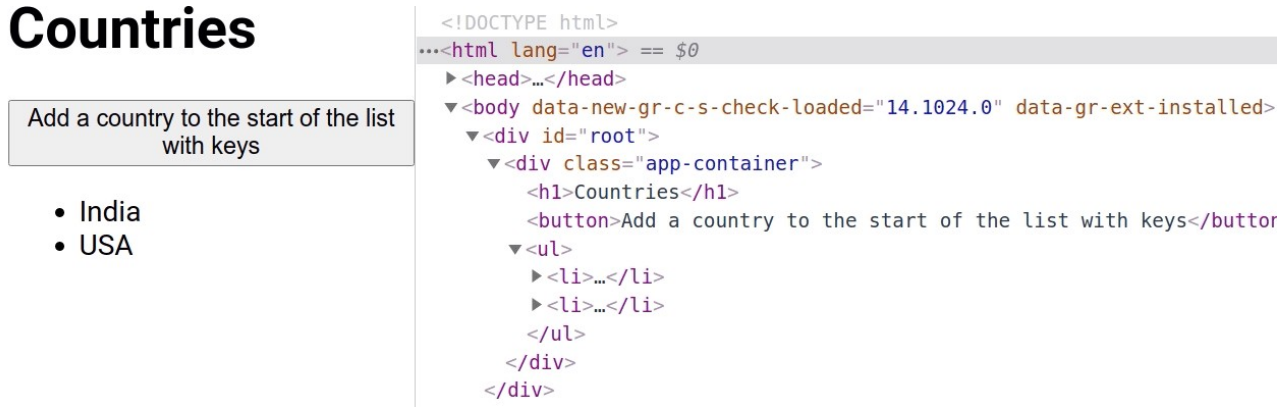


When a new list item is added to the start of the list,

- React creates a new virtual DOM with the three list items.
- React compares the first list item in the current virtual DOM and the new virtual DOM.
- Since the first list item in both the previous and current virtual DOMs is different, React treats as the whole list is changed in the current virtual DOM.
- React creates all the three list items in the HTML DOM.

So, React will recreate every element, instead of reusing the two elements that remained the same between the previous and current virtual DOMs. It leads to bad performance.

That's where keys are important. Using keys, every item in a list will have a unique identifier (ID).



So, React can easily detect what needs to be changed or not, re-rendering only the ones with changes.

Third-Party Packages

Concepts in Focus

1. Third-Party Packages

A Third-Party Package is a reusable code developed to perform a specific functionality

1.1 Advantages

- Easy integration
- Saves Time
- More productivity with fewer lines of code
- Better Error Handling
- More Customisation Options ... many more.

We have used many third-party packages like React Router, React Loader, React Icons, JWT, etc.

Node Package Manager contains Third-Party Packages for React JS, Node JS, Angular JS, and many more libraries and frameworks. To know more about npm you can refer to **this**.

1.2 Selecting a Third-Party Package

1.2.1 Things to Consider

- Users Satisfaction

- Popularity (number of stars)
- Maintenance
- Documentation
- Number of unresolved issues

Compare the packages in the above categories and pick one for the application.

2. Third-Party Package - react-player

NPM contains a **react-player**, a third-party package that provides a React component for playing a variety of URLs, including file paths, YouTube, Facebook, etc.

Installation Command:

```
npm install react-player
```

2.1 React Player Props

We can provide different props to **ReactPlayer**. Below are some of the most commonly used props.

Prop	Description	Default Value
playing	Set to true or false to pause or play the media	false
controls	Set to true to display native player controls.	false
light	Set to true to show the video thumbnail, which loads the full player on click. Pass in an image URL to override the preview image	false
width	Set the width of the player	640px
height	Set the height of the player	360px
className	Add the custom styles to ReactPlayer	-

Example:

File: src/App.js

```
import VideoPlayer from './components/VideoPlayer'

import './App.css'

const App = () => <VideoPlayer />

export default App
```

File: src/components/VideoPlayer/index.js

```
import ReactPlayer from 'react-player'

import './index.css'

const videoURL = 'https://youtu.be/YE7VzlLtp-4'
const VideoPlayer = () => (
```

```
  <div className="video-container">
    <h1 className="heading">Video Player</h1>
    <div className="responsive-container">
      <ReactPlayer url={videoURL} />
    </div>
  </div>
)

export default VideoPlayer
```

ReactPlayer Supports different types of URLs.

2.1.1 controls Prop

File: src/components/VideoPlayer/index.js

```
const VideoPlayer = () => (
  <div className="video-container">
    <h1 className="heading">Video Player</h1>
    <div className="responsive-container">
      <ReactPlayer url={videoURL} controls />
    </div>
  </div>
)
export default VideoPlayer
```

2.1.2 playing Prop

File: src/components/VideoPlayer/index.js

```
import {Component} from 'react'

import ReactPlayer from 'react-player'

import './index.css'

const videoURL = 'https://youtu.be/YE7VzlLtp-4'

class VideoPlayer extends Component {
  state = {
    isPlaying: false,
  }

  onClickPlay = () => {
    this.setState(prevState => ({isPlaying: !prevState.isPlaying}))
  }

  render() {
    const {isPlaying} = this.state
    const btnText = isPlaying ? 'Pause' : 'Play'

    return (
      <div className="video-container">
```

```
        <h1 className="heading">Video Player</h1>
        <div className="responsive-container">
          <ReactPlayer url={videoURL} playing={isPlaying} />
        </div>
        <button type="button" className="button" onClick={this.onClickPlay}>
          {btnText}
        </button>
      </div>
    )
  }
}

export default VideoPlayer
```

Reference

To know more about react-player , you can refer to this.

Recharts | Reading Material

Concepts in Focus

1. Third-Party Package recharts

NPM contains **recharts**, a third-party package to display charts in your application.

It supports different types of charts like:

- Bar Chart
- Pie Chart
- Area Chart
- Composed Chart, etc.

It supports different types of visualization methods like:

Cartesian:

- Area
- Bar
- Line, etc.

Polar:

- Pie
- Radar
- Radial Bar

Installation Command:

npm install recharts

1.1 Advantages

- Responsive
- Built for React, from scratch
- Customizable

2. Bar Chart

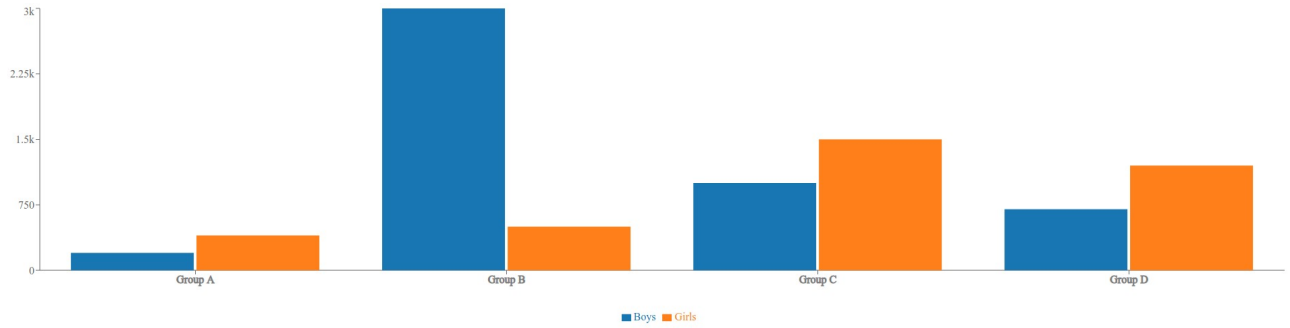
The **BarChart** Component represents the container of the Bar Chart.

Example:

```
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  Legend,
  ResponsiveContainer,
} from "recharts"

const data = [
  {
    group_name: "Group A",
    boys: 200,
    girls: 400,
  },
  {
    group_name: "Group B",
    boys: 3000,
    girls: 500,
  },
  {
    group_name: "Group C",
    boys: 1000,
    girls: 1500,
  },
  {
    group_name: "Group D",
```

```
    boys: 700,
    girls: 1200,
  },
]

const App = () => {
  const DataFormatter = (number) => {
    if (number > 1000) {
      return `${(number / 1000).toString()}k`
    }
    return number.toString()
  }

  return (
    <ResponsiveContainer width="100%" height={500}>
      <BarChart
        data={data}
        margin={{
          top: 5,
        }}
      >
        <XAxis
          dataKey="group_name"
          tick={{
            stroke: "gray",
            strokeWidth: 1,
          }}
        />
        <YAxis
          tickFormatter={DataFormatter}
          tick={{
            stroke: "gray",
            strokeWidth: 0,
          }}
        />
        <Legend
          wrapperStyle={{
            padding: 30,
          }}
        />
        <Bar dataKey="boys" name="Boys" fill="#1f77b4" barSize="20%" />
        <Bar dataKey="girls" name="Girls" fill="#fd7f0e" barSize="20%" />
      </BarChart>
    </ResponsiveContainer>
  )
}

export default App
```

Output :



3. Components in Bar Chart

The recharts supports different Components for the Bar Chart. Below are some of the most commonly used Components.

3.1 ResponsiveContainer

It is a container Component to make charts adapt to the size of the parent container.

Props:

We can provide different props to the ReactJS **ResponsiveContainer** Component. Below are some of the most commonly used props.

Prop	Default Value
width	'100%' (value can be percentage string or number)
height	'100%' (value can be percentage string or number)

Note:

One of the props width and height should be a percentage string in the ResponsiveContainer Component.

3.2 XAxis

The **XAxis** Component represents the X-Axis of a Chart.

Props:

We can provide different props to the ReactJS **XAxis** Component. Below are some of the most commonly used props.

Prop	Description	Default Value
dataKey	The key of the object in data that we want to display it's value on the axis	No default value (value can be string or number)
tick	Represents a tick	No default value. If false - No ticks will be drawn, object - Configuration of ticks, React element - Custom react element for drawing ticks (value can be boolean, object or React element)
tickFormatter	The formatter function of tick	No default value (Function)

Example - tickFormatter:

If we want to show the thousands in the form of k on the tick, the formatter function would be:

```
const DataFormatter = (number) => {
  if (number > 1000) {
    return `${(number / 1000).toString()}k`
  }
  return number.toString()
}
```

3.3 Yaxis

The **YAxis** Component represents the Y-Axis of a Chart.

The Props of the **YAxis** Component are similar to the **XAxis** Component.

3.4 Legend

The **Legend** Component represents the legend of a Chart.

By default, the content of the legend is generated by the name of Line,Bar, Area, etc. If no name has been set, the prop dataKeyis used to generate the content of the legend.

Props:

We can provide different props to the ReactJS **Legend** Component. Below are some of the most commonly used props.

Prop	Description	Default Value
iconType	The type of icon in the legend item	No default value (value can be 'line', 'plainline', 'square', 'rect', 'circle', 'cross', 'diamond','star', 'triangle', or 'wye')
layout	The layout of legend items	'horizontal' (value can be 'horizontal' or 'vertical')
verticalAlign	The alignment of legend items in vertical direction	'middle' (value can be 'top', 'middle', or 'bottom')
align	The alignment of legend items in horizontal direction	'center' (value can be 'left', 'center', or 'right')
wrapperStyle	The style of the legend container	No default value (value can be React Inline styles)

3.5 Bar

The **Bar** Component represents a bar in the Chart.

Props:

We can provide different props to the ReactJS **Bar** Component. Below are some of the most commonly used props.

Prop	Description	Default Value
dataKey	The key of the object in data	No default value (value can be string or number)

Prop	Description	Default Value
	that we want to display it's value	
name	The name of the bar	No default value (value can be string or number)
fill	The color to fill the rectangle in a bar	(value can be given color in hexCode or string format)
barSize	The width or height of the bar	No default value (value can be number)

Note:
The value of the prop name is used in tooltip and legend to represent a bar/pie. If no value was set, the value of dataKey will be used alternatively.

4. PieChart

The **PieChart** Component represents the container of the Pie Chart.

Example:

```
import { PieChart, Pie, Legend, Cell, ResponsiveContainer } from "recharts"

const data = [
  {
    count: 809680,
    language: "Telugu",
  },
  {
    count: 4555697,
    language: "Hindi",
  },
  {
    count: 12345657,
    language: "English",
  },
]

const App = () => {
  return (
    <ResponsiveContainer width="100%" height={300}>
      <PieChart>
        <Pie
          cx="70%"
          cy="40%"
          data={data}
          startAngle={0}
          endAngle={360}
          innerRadius="40%"
          outerRadius="70%"
          dataKey="count"
        >
          <Cell name="Telugu" fill="#fecba6" />
          <Cell name="Hindi" fill="#b3d23f" />
          <Cell name="English" fill="#a44c9e" />
        </Pie>
```

```
        <Legend
          iconType="circle"
          layout="vertical"
          verticalAlign="middle"
          align="right"
        />
      </PieChart>
    </ResponsiveContainer>
  )
}
export default App
```

5. Components in Pie Chart

The recharts supports different Components for the Bar Chart. Below are some of the most commonly used Components.

5.1 Pie

The **Pie** Component represents a pie in the Chart.

Props:

We can provide different props to the ReactJS **Pie** Component. Below are some of the most commonly used props.

Prop	Description	Default Value
cx	The x-axis coordinate of a center point	'50%'. If set a percentage, the final value is obtained by multiplying the percentage of container width (value can be percentage string or number)
cy	The y-axis coordinate of a center point	'50%'. If set a percentage, the final value is obtained by multiplying the percentage of container height (value can be percentage string or number)
data	The source data in which each element is an object	No default value (value can be Array)
startAngle	The start angle of the first sector	0 (value can be number)
endAngle	The end angle of the last sector, which should be unequal to startAngle	360 (value can be number)

Prop	Description	Default Value
innerRadius	The inner radius of all the sectors	0 (value can be percentage or number)
OuterRadius	The outer radius of all the sectors	0 (value can be percentage or number)
dataKey	The key of the object in data that we want to display it's value on the sector	No default value

Note:

If a percentage is set to the props innerRadius or outerRadius, the final value is obtained by multiplying the percentage of maxRadius which is calculated by the width, height, cx , and cy .

5.2 Cell
The **Cell** Component represents the cell of a Chart.

It can be wrapped by a Pie , Bar, or RadialBarComponents to specify the attributes of each child.
Props:

We can provide different props to the ReactJS **Cell** Component. Below are some of the most commonly used props.

Prop	Description	Default Value
name	The name of the cell	No default value (can be a string. This value can be taken as the content of the legend)
fill	The color to fill the cell	(value can be any color in hexCode or string format)

Note:

The ResponsiveContainer and Legend Components in Pie Chart are similar to the Components in Bar Chart.

6. Reference

To know more about the recharts , you can refer to this.

React Chrono | Reading Material

Concepts in Focus

- Third-Party Package react-chrono
    - Advantages
- Chrono Props
    - mode

1. Third-Party Package react-chrono

NPM contains a **react-chrono**, a third-party package to display the timeline in your application.

It provides a React component
Chrono
to build a timeline that displays events chronologically.

Installation Command:

npm install react-chrono

1.1 Advantages

- Can render the timelines in three different modes (horizontal, vertical, and tree).
- Can autoplay the timeline with slideshow mode.
- Can navigate the timeline via keyboard.

2. Chrono Props

We can provide different props to the ReactJS **Chrono** component. Below are some of the most commonly used props.

Prop	Description	Default Value
mode	Sets the mode of the component	'HORIZONTAL' (value can be 'HORIZONTAL', 'VERTICAL' or 'VERTICAL_ALTERNATING')
items	Collection of Timeline Item Model	[]
theme	Customises the colors	No default value

2.1 mode

Example:

<Chrono mode="VERTICAL" />

2.2 items

The items prop is used to build the timeline. It is a collection of the **Timeline Item Model**.
Timeline Item Model is an object consist of the given properties:

Name	Description	Type
title	title of the timeline item	String
cardTitle	title of the timeline card	String

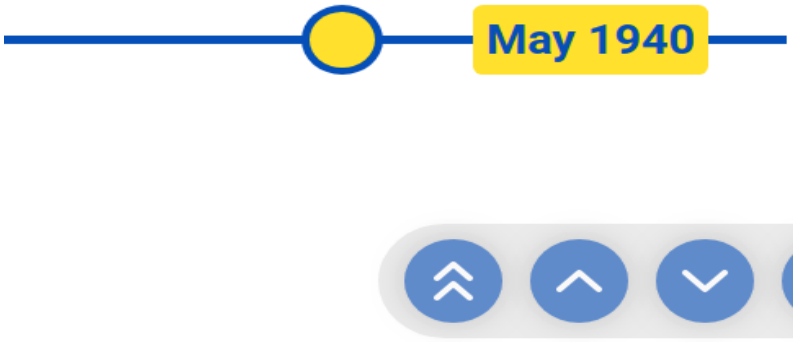
Name	Description	Type
cardSubtitle	text of the timeline card	String
cardDetailedText	detailed text of the timeline card	String or String[]
media	media object to set image or video	Object

Example:

```
.chrono-container {
  width: 500px;
  height: 400px;
}
import {Chrono} from 'react-chrono'


const items = [
  {
    title: 'May 1940',
    cardTitle: 'Dunkirk',
    cardSubtitle: 'Men of the British Expeditionary Force.',
    cardDetailedText:
      'On 10 May 1940, Hitler began his long-awaited offensive in the west by invading neutral Holland and attacking northern France.',
  },
]


const App = () => (
  <div className="chrono-container">
    <Chrono items={items} />
  </div>
)
export default App
```

Output:

A single timeline item is created with the values of the **Timeline Item Model** in the items prop.



Warning:

If any property misses in the **Timeline Item Model**, the respective value won't be displayed in the timeline item.

Note:

The Chrono Component should be wrapped in a container that has a width and height.

2.3 theme

The colors can be customized with the theme prop.
Example:

```
<Chrono
  theme={{
    primary: "red",
    secondary: "blue",
```

```
    cardBgColor: "yellow",
    cardForeColor: "violet",
    titleColor: "red",
  }}
/>
```

3. Rendering Custom Content

The custom content can be inserted by just passing the elements between the Chrono tags.
Example:

The below example will create two timeline items.

```
.chrono-container {
  width: 400px;
  height: 600px;
}

.image {
  width: 200px;
  height: 200px;
}
import {Chrono} from 'react-chrono'

const App = () => (
  <div className="chrono-container">
    <Chrono mode="VERTICAL">
      <div>
        <img
          src="https://assets.ccbp.in/frontend/react-js/csk-logo-img.png"
          className="image"
          alt="chennai-super-kings"
        />
      </div>
      <div>
        <h1>Mumbai Indians</h1>
```

```
        <p>IPL Team winner for the year 2019 is Mumbai Indians.</p>
      </div>
    </Chrono>
  </div>
)
export default App
```

Output:

Each HTML div element is automatically converted into a timeline item and inserted into the timeline card.

Mumbai Indians

IPL Team winner for the year 2019 is Mumbai Indians.

Note:

The items prop is **optional** and custom rendering is supported on all three modes.

Example:

The below example will set the title for the custom contents.

```
.chrono-container {
  width: 400px;
  height: 600px;
}

.image {
  width: 200px;
  height: 200px;
}

import {Chrono} from 'react-chrono'

const items = [{title: '2018'}, {title: '2019'}]

const App = () => (
  <div className="chrono-container">
    <Chrono mode="VERTICAL" items={items}>
      <img
        src="https://assets.ccbp.in/frontend/react-js/csk-logo-img.png"
        className="image"
        alt="chennai-super-kings"
      />
      <div>
        <h1>Mumbai Indians</h1>
        <p>IPL Team winner for the year 2019 is Mumbai Indians.</p>
      </div>
    </Chrono>
  </div>
)

export default App
```

4. Reference

To know more about the react-chrono , you can refer to this.

React Slick | Reading Material

Concepts in Focus

1. Third-Party Package - react-slick

NPM contains a **react-slick**, a third-party package that provides a React component **Slider** to add a carousel to your application.

Installation Command:

npm install react-slick

Also, install **slick-carousel** for CSS and font

npm install slick-carousel

1.1 Advantages

- Easy to use
- Highly customizable
- More performant

2. Slider Props

We can provide different props to the **Slider** component. Below are the most commonly used props.

Prop	Description	Default Value
slidesToShow	Specifies the number of slides to show in one frame	1
dots	If set to false, no dots will display under the carousel	true
slidesToScroll	Specifies the number of slides to scroll at once	1
dotsClass	CSS class for styling dots	"slick-dots"

Example:

File: src/components/ReactSlick/index.js

```
import Slider from 'react-slick'


import 'slick-carousel/slick/slick.css'
```

```
import 'slick-carousel/slick/slick-theme.css'

import './index.css'

const ReactSlick = () => {
  const settings = {
    dots: true,
    slidesToShow: 1,
    slidesToScroll: 1,
  }
  return (
    <div className="slider-container">
      <Slider {...settings}>
        <div>
          <h3>1</h3>
        </div>
        <div>
          <h3>2</h3>
        </div>
        <div>
          <h3>3</h3>
        </div>
        <div>
          <h3>4</h3>
        </div>
      </Slider>
    </div>
  )
}
export default ReactSlick
```

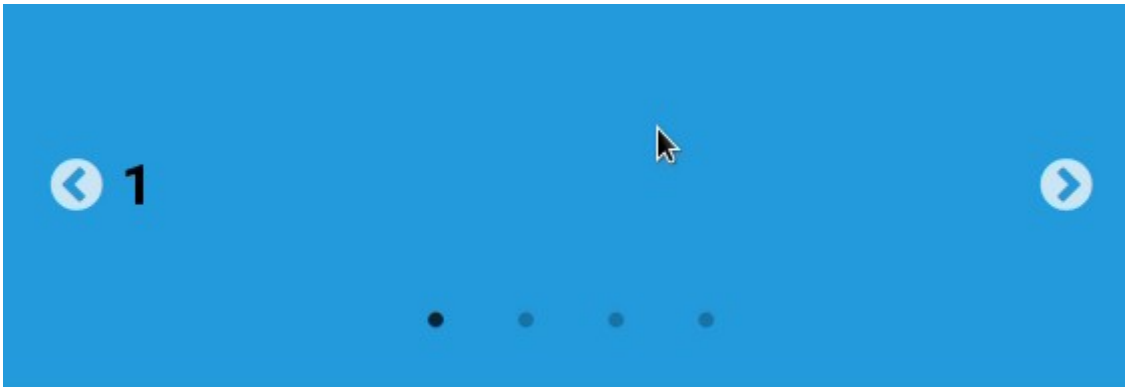
File: src/components/ReactSlick/index.css

```
.slider-container {
```

```
    background-color: #419be0;
    padding: 40px;
}
```

File: src/App.js

```
import ReactSlick from './components/ReactSlick'

const App = () => {
  return <ReactSlick />
}
export default App
```



Reference

To know more about the react-slick , you can refer to this.

React Popup | Reading Material

Concepts in Focus

1. Third-Party Package reactjs-popup

NPM contains a **reactjs-popup**, a third-party package to display popup in your application.

It provides a React component that helps you create simple and complex **Modals**, **Tooltips**, and **Menus** for your application.

Installation Command:

npm install reactjs-popup

1.1 Advantages

- ReactJS provides Modal, Tooltip, and Menu
- Provides Support for controlled Modals & Tooltips

2. ReactJS Popup Props

We can provide different props to the ReactJS **Popup** component. Below are some of the most commonly used props.

Prop	Description	Default Value
trigger	Represents the element to be rendered in-place where the Popup is defined	JSX element
modal	When set to true, Popup content will be displayed as a modal on the trigger of Popup. If not tooltip will be displayed	false
position	Defines the position of the tooltip. It can be one of 'top left', 'top right', 'bottom right', 'bottom left'	bottom center
open	Defines the state of Popup. Whether it is opened or not	false
overlayStyle	Custom overlay style	object

2.1 trigger Prop

File: src/components/ReactPopup/index.js

```
import Popup from 'reactjs-popup'


import 'reactjs-popup/dist/index.css'


import './index.css'


const ReactPopUp = () => (
 <div className="popup-container">
   <Popup
    trigger={
      <button className="trigger-button" type="button">
        Trigger
      </button>
    }
```

```
  >
    <div>
      <p>React is a popular and widely used programming language</p>
    </div>
  </Popup>
 </div>
)
export default ReactPopUp
```

File: src/components/ReactPopup/index.css

```
.trigger-button {
 font-size: 16px;
 font-weight: 400;
 font-family: 'Roboto';
 color: white;
 padding: 8px 15px 8px 15px;
 margin: 8px;
 background-color: #7c69e9;
 border: none;
 border-radius: 4px;
 outline: none;
}
.popup-container {
 display: flex;
 align-items: center;
 justify-content: center;
 height: 100vh;
}
```

File: src/App.js

```
import ReactPopup from './components/ReactPopup'

const App = () => {
```

```
 return <ReactPopup />
}
export default App
```

Note:

The value of the **trigger** prop should be a JSX element.

2.2 modal Prop

File: src/components/ReactPopup/index.js

```
const ReactPopUp = () => (
 <div className="popup-container">
   <Popup
     modal
     trigger={
       <button type="button" className="trigger-button">
         Trigger
       </button>
     }
   >
     <p>React is a popular and widely used programming language</p>
   </Popup>
 </div>
)
export default ReactPopUp
```

2.3 modal Prop with close button

File: src/components/ReactPopup/index.js

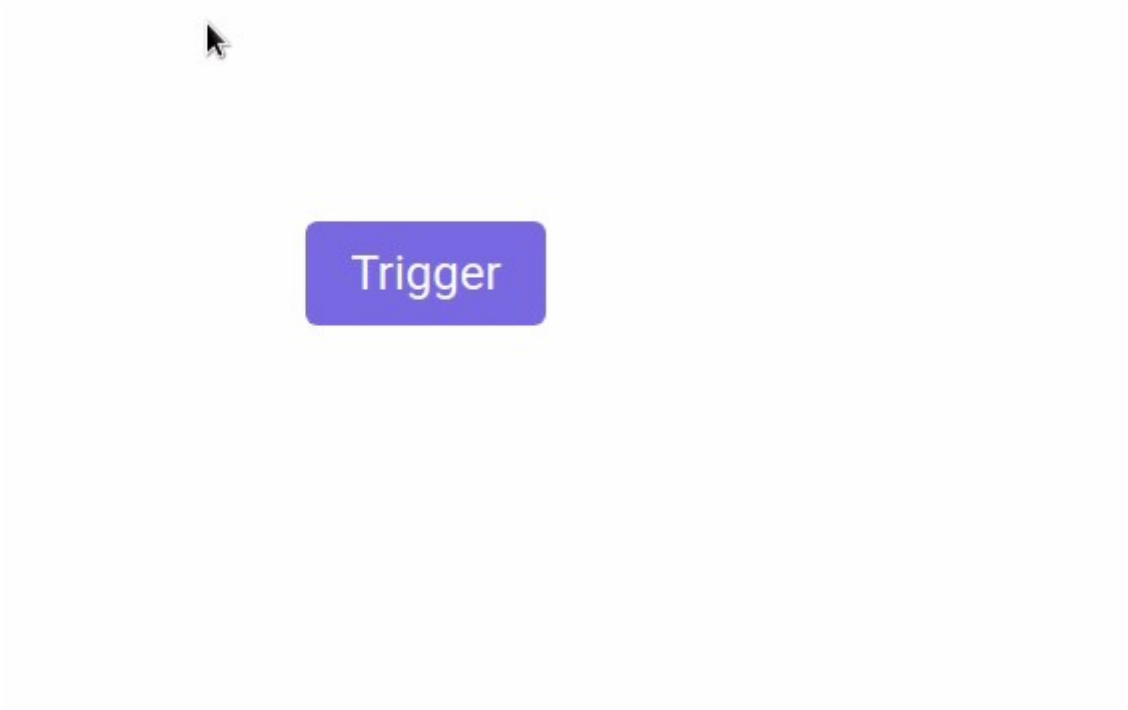
```
const ReactPopUp = () => (
 <div className="popup-container">
   <Popup
     modal
     trigger={
       <button type="button" className="trigger-button">
         Trigger
```

```
      </button>
    }
  >
    {close => (
      <>
        <div>
          <p>React is a popular and widely used programming language</p>
        </div>
        <button
          type="button"
          className="trigger-button"
          onClick={() => close()}
        >
          Close
        </button>
      </>
    )}
  </Popup>
 </div>
)
export default ReactPopUp
```

2.4 position Prop

File: src/components/ReactPopup/index.js

```
const ReactPopUp = () => (
 <div className="popup-container">
   <Popup
    trigger={
      <button type="button" className="trigger-button">
        Trigger
      </button>
    }
    position="bottom left"
   >
    <p>React is a popular and widely used programming language</p>
   </Popup>
 </div>
)
export default ReactPopUp
```



2.5 overlayStyle Prop

File: src/components/ReactPopup/index.js

```
const overlayStyles = {
 backgroundColor: '#ffff',
}
const ReactPopUp = () => (
 <div className="popup-container">
   <Popup
    modal
    trigger={
      <button type="button" className="trigger-button">
        Trigger
      </button>
    }
    overlayStyle={overlayStyles}
   >
    <p>React is a popular and widely used programming language</p>
   </Popup>
 </div>
```

)

export default ReactPopUp

Reference

To know more about the reactjs-popup , you can refer to this.

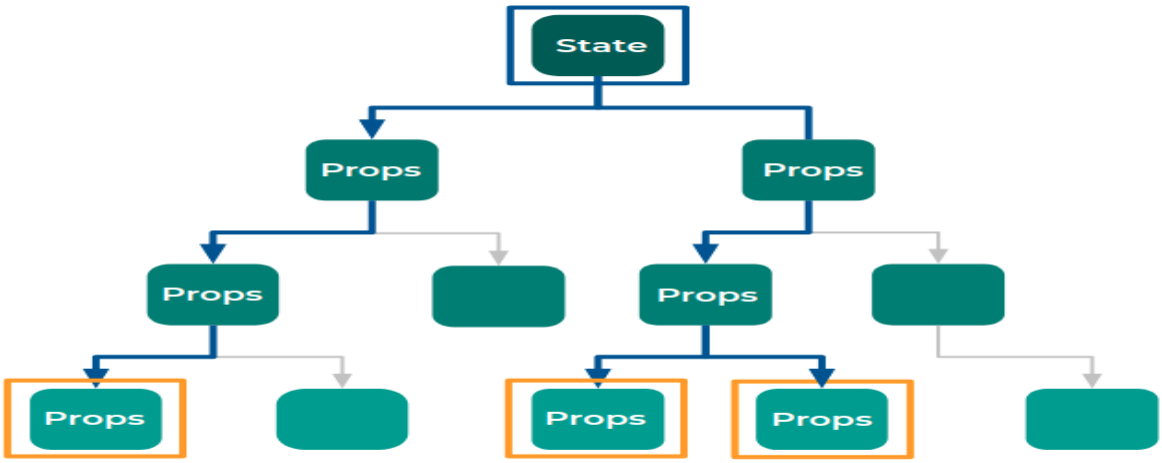
React Context

Concepts in Focus

- Prop Drilling
- React Context
    - Creating Context
    - Context Properties
    - How to Access the Value?
- Consumer Component
    - Understanding Syntax
- Windows App Example
    - Initial Code
    - Final Code
- Google Translator

1. Prop Drilling

Props are passed from one Component to another Component that does not need the data but only helps in passing it through the tree is called **Prop Drilling**.



2. React Context

Context is a mechanism that provides different Components and allows us to pass data without doing prop drilling.

2.1 Creating Context

Syntax:

React.createContext(INITIAL_VALUE)

It returns an object with different properties to update and access values from the context.

2.2 Context Properties

The **Context** object provides two properties.

1. Consumer
2. Provider

2.3 How to Access the Value?

We can access the value in the **Context** using Consumer Component provided by the Context Object.

3. Consumer Component

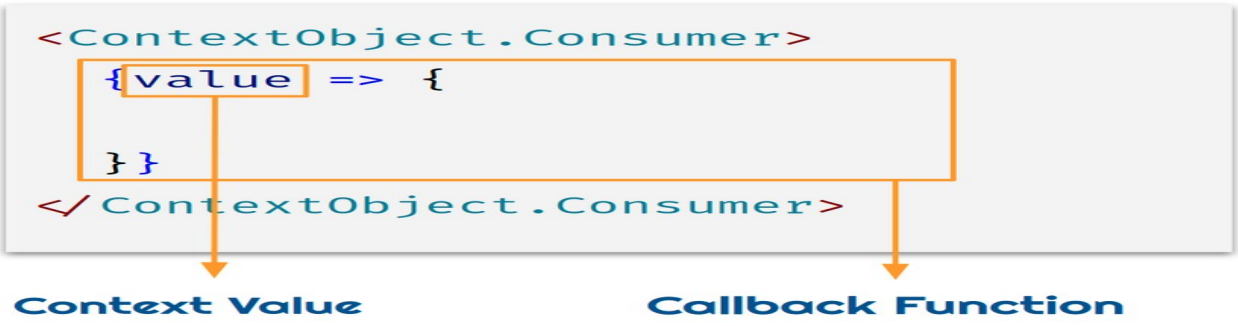
Syntax:

```
<ContextObject.Consumer>
  {/*callback function*/}
</ContextObject.Consumer>
```

- We access the Consumer component using dot notation from the context object .
- We will give a callback function as children for Consumer Component.

3.1 Understanding Syntax

The callback function receives the current **context value** as an argument and returns a **component** or a **JSX element**.



4. Windows App Example

4.1 Initial Code

Run the below command in your IDE to download the initial code.

ccbp start RJSIVWKZP2

4.2 Final Code

Run the below command in your IDE to download the final code.

ccbp start RJSIVM06DJ

5. Google Translator

- In the **Windows App example**, we are using three different languages to display the text.
- You can translate one language to another language using **google translator**.
- Check google translator website here .

React Context | Part 2

Concepts in Focus

- Provider
    - Syntax
- Context Flow For Windows App
- Best Practice
- Final Code

1. Provider

- **Provider** Component updates the value of Context.
- When the provider updates context, all the nested consumers of that provider will be **re-rendered**.
- Updated context value can only be accessed by the consumers nested within the provider.

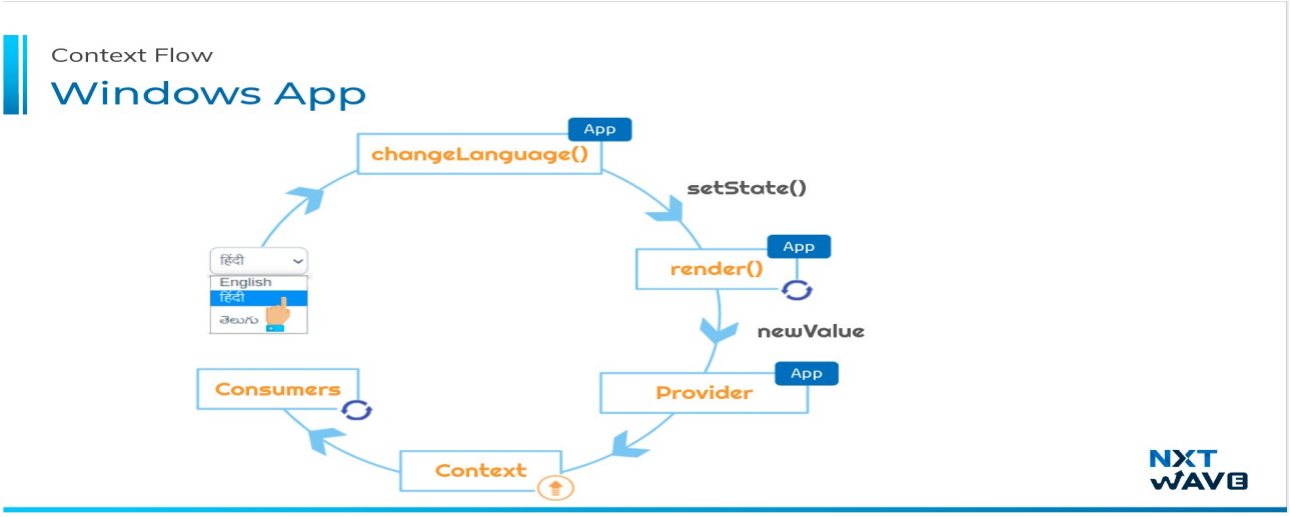
<ContextObject.Provider value={/* some value */}>

...

<ContextObject.Provider/>

To update context value, we have to pass it as a prop to the **Provider** omponent.

2. Context Flow For Windows App



3. Best Practice

- The context value should have the data which the consumers need.
- It should also contain the **required methods** to update that data.

4.Final Code

File: src/App.js

```
import {Component} from 'react'


import Header from './components/Header'

import LandingSection from './components/LandingSection'

import FeaturesSection from './components/FeaturesSection'


import LanguageContext from './context/LanguageContext'
```

```
import './App.css'

class App extends Component {
  state = {activeLanguage: 'EN'}

  changeLanguage = activeLanguage => {
    this.setState({activeLanguage})
  }

  render() {
    const {activeLanguage} = this.state
    return (
      <LanguageContext.Provider
        value={{activeLanguage, changeLanguage: this.changeLanguage}}
      >
        <Header />
        <LandingSection />
        <FeaturesSection />
      </LanguageContext.Provider>
    )
  }
}
export default App
```

File: src/context/LanguageContext.jsx

```
import React from 'react'

const LanguageContext = React.createContext({
  activeLanguage: 'EN',
  changeLanguage: () => {},
})
```

```
export default LanguageContext
```

File: src/components/FeaturesSection/index.js

```
import Playtime from '../Playtime'
import NewWaysToConnect from '../NewWaysToConnect'

import './index.css'

const FeaturesSection = () => (
  <div className="features-section-container">
    <Playtime />
    <NewWaysToConnect />
  </div>
)
export default FeaturesSection
```

File: src/components/Header/index.js

```
import LanguageContext from '../../context/LanguageContext'

import './index.css'

const langOptions = [
  {id: 1, value: 'EN', language: 'English'},
  {id: 2, value: 'HI', language: 'हिंदी'},
  {id: 3, value: 'TE', language: 'తెలుగు'},
]

const Header = () => (
  <LanguageContext.Consumer>
    {value => {
      const {activeLanguage, changeLanguage} = value
      const onChangeLanguage = event => {
        changeLanguage(event.target.value)
```

```
      }

      return (
        <nav className="nav-header">
          <img
            className="website-logo"
            src="https://assets.ccbp.in/frontend/react-js/windows-logo-img.png"
            alt="website logo"
          />
          <select
            className="language-dropdown"
            value={activeLanguage}
            onChange={onChangeLanguage}
          >
            {langOptions.map(eachOption => (
              <option key={eachOption.id} value={eachOption.value}>
                {eachOption.language}
              </option>
            ))}
          </select>
        </nav>
      )
    }}
  </LanguageContext.Consumer>
)
export default Header
```

File: src/components/LandingSection/index.js

```
import LanguageContext from '../../context/LanguageContext'

import './index.css'

const landingSectionContent = {
```

```
  EN: {
    heading: 'Windows 11',
    description:
      'Windows 11 provides a calm and creative space where you can pursue your passions through a fresh experience. From a rejuvenated Start menu to new ways to connect to your favorite people, news, games, and content. Windows  is the place to think, express, and create in a natural way.',
  },
  HI: {
    heading: 'विंडोज 11',
    description:
      'विंडोज 11 एक शांत और रचनात्मक स्थान प्रदान करता है जहां आप एक नए अनुभव के माध्यम से अपने जुनून को आगे बढ़ा सकते हैं। अपने पसंदीदा लोगों, समाचारों, गेमों और सामग्री से जुड़ने के नए तरीकों से एक नए सिरे से शुरू किए गए स्टार्ट मेनू से विंडोज एक प्राकृतिक तरीके से सोचने, व्यक्त करने और बनाने का स्थान है।',
  },
  TE: {
    heading: 'విండోస్ 11',
    description:
      'విండోస్ 11 ప్రశాంతమైన మరియు సృజనాత్మక స్థలాన్ని అందిస్తుంది, ఇక్కడ మీరు మీ అభిరుచులను తాజా అనుభవం ద్వారా కొనసాగించవచ్చు. పునరుజ్జీవింపబడిన ప్రారంభ మెను నుండి మీకు ఇష్టమైన వ్యక్తులు, వార్తలు, ఆటలు మరియు కంటెంట్‌తో కనెక్ట్ అవ్వడానికి కొత్త మార్గాల వరకు విండోస్ అనేది సహజంగా ఆలోచించే, వ్యక్తీకరించే మరియు సృష్టించే ప్రదేశం.',
  },
}

const LandingSection = () => {
  const getLandingSectionData = activeLanguage => {
    switch (activeLanguage) {
      case 'EN':
        return landingSectionContent.EN
      case 'HI':
        return landingSectionContent.HI
```

```
      case 'TE':
        return landingSectionContent.TE
      default:
        return null
    }
  }
  return (
    <LanguageContext.Consumer>
      {value => {
        const {activeLanguage} = value
        const {heading, description} = getLandingSectionData(activeLanguage)
        return (
          <div className="bg-container">
            <div className="responsive-container">
              <div className="description-container">
                <h1 className="heading">{heading}</h1>
                <p className="description">{description}</p>
              </div>
              <img
                className="logo-white"
                src="https://assets.ccbp.in/frontend/react-js/windows-logo-white-img.png"
                alt="windows logo"
              />
            </div>
          </div>
        )
      }}
    </LanguageContext.Consumer>
  )
}
export default LandingSection
```

File: src/components/NewWaysToConnect/index.js

```
import LanguageContext from '../../context/LanguageContext'

import './index.css'

const newWaysToConnectContent = {
  EN: {
    heading: 'New ways to connect',
    description:
      'Connect instantly to the people you care about right from your desktop with Microsoft Teams. Call or chat for free, no matter what device they’re on.',
  },
  HI: {
    heading: 'कनेक्ट करने के नए तरीके',
    description:
      'माइक्रोसॉफ्ट टीमों के साथ सीधे अपने डेस्कटॉप से उन लोगों से तुरंत कनेक्ट हों जिनकी आप परवाह करते हैं। कॉल करें या निःशुल्क चैट करें—चाहे वे किसी भी डिवाइस पर हों।',
  },
  TE: {
    heading: 'కనెక్ట్ చేయడానికి కొత్త మార్గాలు',
    description:
      'మైక్రోసాఫ్ట్ బృందాలతో మీ డెస్క్‌టాప్ నుండి మీరు శ్రద్ధ వహించే వ్యక్తులతో తక్షణమే కనెక్ట్ అవ్వండి. అవి ఏ పరికరంలో ఉన్నా సరే ఉచితంగా కాల్ చేయండి లేదా చాట్ చేయండి',
  },
}

const NewWaysToConnect = () => {
  const getNewWaysToConnectData = activeLanguage => {
    switch (activeLanguage) {
      case 'EN':
        return newWaysToConnectContent.EN
      case 'HI':
```

```
      return newWaysToConnectContent.HI
    case 'TE':
      return newWaysToConnectContent.TE
    default:
      return null
  }
 }
 return (
  <LanguageContext.Consumer>
    {value => {
      const {activeLanguage} = value
      const {heading, description} = getNewWaysToConnectData(activeLanguage)
      return (
        <div className="new-ways-to-connect-container">
          <h1 className="new-ways-to-content-heading">{heading}</h1>
          <p className="new-ways-to-content-description">{description}</p>
        </div>
      )
    }}
  </LanguageContext.Consumer>
 )
}
export default NewWaysToConnect
```

File: src/components/Playtime/index.js

```
import LanguageContext from '../../context/LanguageContext'

import './index.css'

const playtimeContent = {
 EN: {
  heading: 'Playtime. Anytime',
```

```
    description:
      'Windows takes gaming to a whole new level with graphic capabilities that rival reality. Discover your next favorite game with Xbox GamePass, giving you access to over 100 high-quality games.',
  },
  HI: {
    heading: 'विश्राम का समय। किसी भी समय',
    description:
      'विंडोज़ गेमिंग को ग्राफ़िक क्षमताओं के साथ एक नए स्तर पर ले जाता है जो वास्तविकता को टक्कर देता है। एक्सबॉक्स गेम पास के साथ अपना अगला पसंदीदा गेम खोजें, जिससे आपको 100 से अधिक उच्च-गुणवत्ता वाले गेम तक पहुंच मिलती है।',
  },
  TE: {
    heading: 'ఆడూకునే సమయం. ఎప్పుడైనా',
    description:
      'రియాలిటీకి ప్రత్యర్థిగా ఉండే గ్రాఫిక్ సామర్థ్యాలతో విండోస్ గేమింగ్‌ను సరికొత్త స్థాయికి తీసుకువెళుతుంది . మీ తదుపరి ఇష్టమైన ఆటను ఎక్స్‌బాక్స్ గేమ్‌పాస్‌తో కనుగొనండి, మీకు 100 కి పైగా అధిక-నాణ్యత ఆటలకు ప్రాప్యతను ఇస్తుంది.',
  },
}

const Playtime = () => {
  const getPlaytimeData = activeLanguage => {
    switch (activeLanguage) {
      case 'EN':
        return playtimeContent.EN
      case 'HI':
        return playtimeContent.HI
      case 'TE':
        return playtimeContent.TE
      default:
        return null
```

```
    }
  }
  return (
    <LanguageContext.Consumer>
      {value => {
        const {activeLanguage} = value
        const {heading, description} = getPlaytimeData(activeLanguage)
        return (
          <div className="playtime-container">
            <h1 className="playtime-heading">{heading}</h1>
            <div className="playtime-description-container">
              <p className="playtime-description">{description}</p>
              <img
                className="playtime-games-image"
                src="https://assets.ccbp.in/frontend/react-js/gaming-pad-img.png"
                alt="gaming pad"
              />
            </div>
          </div>
        )
      }}
    </LanguageContext.Consumer>
  )
}
export default Playtime
```

React Context | Part 3

Concepts in Focus

- Context
- Using Context in methods
- Final Code

1. Context

Context can also be used:

- When data has to be available between different **Routes**.
- When data has to be available for more number of **Components**

2. Using Context in methods

- We can add **consumer** or **provider** in the class method when it returns **JSX**.
- In the below code snippet, we have used **Consumer** Component in renderProductDetailsView
  because this method is returning **JSX**.

Example:

File: src/components/ProductItemDetails

```
import { Component } from "react";
import Cookies from "js-cookie";
import Loader from "react-loader-spinner";
import { BsPlusSquare, BsDashSquare } from "react-icons/bs";

import CartContext from "../../context/CartContext";

import Header from "../Header";
import SimilarProductItem from "../SimilarProductItem";

import "./index.css";

const apiStatusConstants = {
  initial: "INITIAL",
  success: "SUCCESS",
  failure: "FAILURE",
  inProgress: "IN_PROGRESS",
};

class ProductItemDetails extends Component {
  state = {
    productData: {},
    similarProductsData: [],
```

```
  apiStatus: apiStatusConstants.initial,
  quantity: 1,
};

componentDidMount() {
  this.getProductData();
}

getFormattedData = (data) => ({
  availability: data.availability,
  brand: data.brand,
  description: data.description,
  id: data.id,
  imageUrl: data.image_url,
  price: data.price,
  rating: data.rating,
  title: data.title,
  totalReviews: data.total_reviews,
});

getProductData = async () => {
  const { match } = this.props;
  const { params } = match;
  const { id } = params;

  this.setState({
    apiStatus: apiStatusConstants.inProgress,
  });
  const jwtToken = Cookies.get("jwt_token");
  const apiUrl = `https://apis.ccbp.in/products/${id}`;
  const options = {
    headers: {
```

```
      Authorization: `Bearer ${jwtToken}`,
    },
    method: "GET",
  };
  const response = await fetch(apiUrl, options);
  if (response.ok) {
    const fetchedData = await response.json();
    const updatedData = this.getFormattedData(fetchedData);
    const updatedSimilarProductsData = fetchedData.similar_products.map(
      (eachSimilarProduct) => this.getFormattedData(eachSimilarProduct)
    );
    this.setState({
      productData: updatedData,
      similarProductsData: updatedSimilarProductsData,
      apiStatus: apiStatusConstants.success,
    });
  }
  if (response.status === 404) {
    this.setState({
      apiStatus: apiStatusConstants.failure,
    });
  }
};

renderLoadingView = () => (
  <div className="products-details-loader-container">
    <Loader type="ThreeDots" color="#0b69ff" height="50" width="50" />
  </div>
);

renderFailureView = () => (
  <div className="product-details-error-view-container">
```

```
    <img
      alt="error view"
      src="https://assets.ccbp.in/frontend/react-js/nxt-trendz-error-view-img.png"
      className="error-view-image"
    />
    <h1 className="product-not-found-heading">Product Not Found</h1>
    <button type="button" className="button">
      Continue Shopping
    </button>
  </div>
);

onDecrementQuantity = () => {
  const { quantity } = this.state;
  if (quantity > 1) {
    this.setState((prevState) => ({ quantity: prevState.quantity - 1 }));
  }
};

onIncrementQuantity = () => {
  this.setState((prevState) => ({ quantity: prevState.quantity + 1 }));
};

renderProductDetailsView = () => (
  <CartContext.Consumer>
    {(value) => {
      const { productData, quantity, similarProductsData } = this.state;
      const {
        availability,
        brand,
        description,
        imageUrl,
```

```
  price,
  rating,
  title,
  totalReviews,
} = productData;
const { addCartItem } = value;
const onClickAddToCart = () => {
  addCartItem({ ...productData, quantity });
};

return (
  <div className="product-details-success-view">
    <div className="product-details-container">
      <img src={imageUrl} alt="product" className="product-image" />
      <div className="product">
        <h1 className="product-name">{title}</h1>
        <p className="price-details">Rs {price}/-</p>
        <div className="rating-and-reviews-count">
          <div className="rating-container">
            <p className="rating">{rating}</p>
            <img
              src="https://assets.ccbp.in/frontend/react-js/star-img.png"
              alt="star"
              className="star"
            />
          </div>
          <p className="reviews-count">{totalReviews} Reviews</p>
        </div>
        <p className="product-description">{description}</p>
        <div className="label-value-container">
          <p className="label">Available:</p>
          <p className="value">{availability}</p>
```

```
    </div>
    <div className="label-value-container">
      <p className="label">Brand:</p>
      <p className="value">{brand}</p>
    </div>
    <hr className="horizontal-line" />
    <div className="quantity-container">
      <button
        type="button"
        className="quantity-controller-button"
        onClick={this.onDecrementQuantity}
      >
        <BsDashSquare className="quantity-controller-icon" />
      </button>
      <p className="quantity">{quantity}</p>
      <button
        type="button"
        className="quantity-controller-button"
        onClick={this.onIncrementQuantity}
      >
        <BsPlusSquare className="quantity-controller-icon" />
      </button>
    </div>
    <button
      type="button"
      className="button add-to-cart-btn"
      onClick={onClickAddToCart}
    >
      ADD TO CART
    </button>
  </div>
</div>
```

```
          <h1 className="similar-products-heading">Similar Products</h1>
          <ul className="similar-products-list">
            {similarProductsData.map((eachSimilarProduct) => (
              <SimilarProductItem
                productDetails={eachSimilarProduct}
                key={eachSimilarProduct.id}
              />
            ))}
          </ul>
        </div>
      );
    }}
  </CartContext.Consumer>
);

renderProductDetails = () => {
  const { apiStatus } = this.state;

  switch (apiStatus) {
    case apiStatusConstants.success:
      return this.renderProductDetailsView();
    case apiStatusConstants.failure:
      return this.renderFailureView();
    case apiStatusConstants.inProgress:
      return this.renderLoadingView();
    default:
      return null;
  }
};

render() {
  return (
```

```
      <>
        <Header />
        <div className="product-item-details-container">
          {this.renderProductDetails()}
        </div>
      </>
    );
  }
}
export default ProductItemDetails;
```

3. Final code

Run the below command in your IDE to download the final code.

ccbp start RJSIVIJ1YR

React Context | Part 4

Concepts in Focus

- Deploying the Ecommerce Application Code
- Ecommerce Application-Final Code

1. Deploying the Ecommerce Application Code

Run the below command in your IDE to publish the Ecommerce Application Code.

ccbp publish RJSIV4YSTX domain.ccbp.tech

Here the **domain** is the domain name to be published.

Quick Tip:

To look good in your resume, use your name and **nxttrendz** as the domain name in the domain URL. For example, rahulnxttrendz.ccbp.tech.

Warning:

An error will be thrown if you are trying to publish with the already existing URL.

Note:

A maximum of only 15 characters is allowed in the domain name.

2. Ecommerce Application - Final Code

Run the below command in your IDE to download the final code of an Ecommerce Application.

ccbp start RJSIVPP7N7

Styled Components

Concepts in Focus

- Styling React Components
    - Using CSS
    - CSS-in-JS
- Styled Components
    - Installation
    - Syntax
    - Importing styled
    - Creating and Exporting Styled Component
    - Importing and Rendering Styled Component
    - Adapting based on props
    - Modularity of Code
    - Conditional Styling using Props
- Advantages of Styled Components
    - Unique Class Names
    - Elimination of dead styles
    - Dynamic Styling
    - Can differentiate between the types of props they receive
    - Easier Debugging
    - Global Styling
- Extending Styles
- The "as" prop
- Boolean Attribute

1. Styling React Components

React Components can be styled in different ways:

- Using CSS
- CSS-in-JS
- SASS & SCSS
    - many more...

1.1 Using CSS

Writing CSS in external CSS files for each component and passing class name as a value to the **className** attribute.

File: src/App.js

import "./App.css";

```
const App = () => <h1 className="heading">Hello World</h1>;

export default App;
```

File: src/App.css

```
.heading {
  color: #0070c1;
  font-family: 'Roboto';
}
```

1.2 CSS-in-JS

CSS-in-JS is a styling technique where JavaScript is used to style React Components. It can be implemented using the below third party packages.

- Styled Components
- Emotion
- Radium many more...

2. Styled Components

Styled Components are one of the new ways to use CSS in modern JavaScript. Components are used to reuse code, similarly Styled Components are used to reuse styles.

2.1 Installation

```
npm install styled-components
```

2.2 Syntax

```
const StyledComponentName = styled.tagName`
  property1: value1;
  property2: value2;
  ...
`;
```

Within the template literals **()**, we define the styles.

Note:

StyledComponentName should always start with a **Capital** letter.

2.3 Importing styled

In general, styled components are placed inside the **styledComponents.js** file.

File: src/styledComponents.js

```
import styled from "styled-components";
```

styled is an internal utility method that transforms the styling from JavaScript into actual CSS.

2.4 Creating and Exporting Styled Component

File: src/styledComponents.js

```
import styled from "styled-components";

export const Heading = styled.h1`
  color: #0070c1;
  font-family: "Roboto";
`;
```

2.5 Importing and Rendering Styled Component

File: src/App.js

```
import "./App.css";
import { Heading } from "./styledComponents";
const App = () => <Heading>Hello World</Heading>;
export default App;
```

2.6 Adapting based on props

- With styled components we can re-use the styles created.

With styled components we can re-use the styles created.

```
<StyledComponent propName="propValue">...</StyledComponent>
```

Syntax:

```
const StyledComponentName = styled.tagName`
  property1 : value1;
  property2: ${props => /* access prop value */ };
  ...
`
```

Example:

File: src/App.js

```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" color="#ffffff" bgColor="#0070c1">Click</CustomButton>
    <CustomButton type="button" color="#0070c1" bgColor="#ffffff">Click</CustomButton>
  </>
);
export default App;
```

File: src/styledComponents.js

```
import styled from "styled-components";
```

```
export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => props.color};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => props.bgColor};
`;
```

Value passed as prop can be accessed as **props.propName**.

2.7 Modularity of Code

Example:

File: src/App.js

```
<CustomButton type="button" outline={false}>Click</CustomButton>
```

The **color** and **bgColor** are the part of styling, instead of passing them as props, we can passthe type of button to handle styles in **styled components**.

2.8 Conditional Styling using Props

Condition ? Expression If True : Expression If False;

```
const StyledComponentName = styled.tagName`
  property1 : value1;
  property2: ${(props) => (props.name === someValue ? value2 : value3)};
  ...
`;
```

Example:

File: src/App.js

```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" outline={false}>Click</CustomButton>
    <CustomButton type="button" outline={true}>Click</CustomButton>
  </>
);

export default App;
```

File: src/styledComponents.js

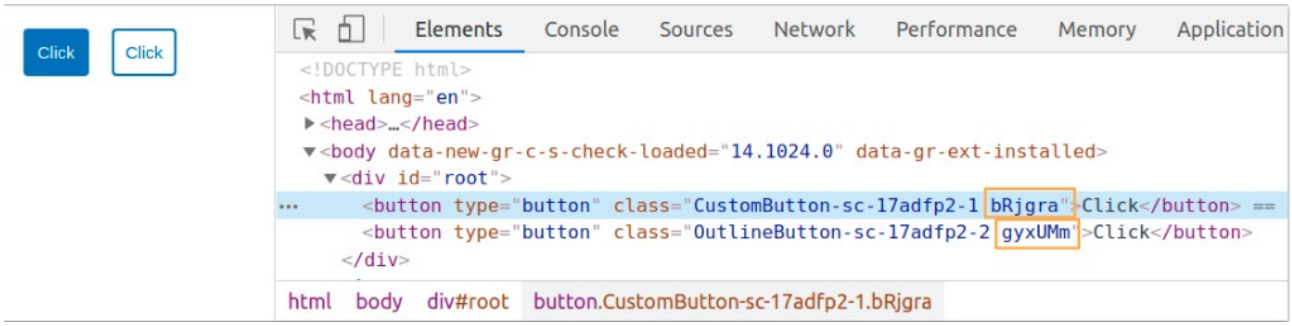
```
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => (props.outline ? "#0070c1" : "#ffffff")};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => (props.outline ? "#ffffff" : "#0070c1")};
`;
```

3.Advantages of Styled Components

3.1 Unique Class Names

Styled Components generate a **unique** class name(s) for your styles.

Example:



3.2 Elimination of dead styles

Styled components remove unused styles, even if they’re declared in your code.

3.3 Dynamic Styling

- Let’s assume we have two types of buttons on our page, one with a **white** background, and the other **navy blue**.
- We do not have to create two styled-components for them. We can adapt their styling based on their props.

File:src/styledComponents.js

```
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => (props.outline ? "#0070c1" : "#ffffff")};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => (props.outline ? "#ffffff" : "#0070c1")};
`;
```

File:src/App.js

```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" outline={false}>Click</CustomButton>
    <CustomButton type="button" outline={true}>Click</CustomButton>
  </>
);

export default App;
```

3.4 Can differentiate between the types of props they receive

File:src/App.js

```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton outline={false}>Click</CustomButton>
    <CustomButton outline={true}>Click</CustomButton>
  </>
);

export default App;
```

You'll notice, though, that we haven't given our button a type in the above code snippet. Let's do that:

File:src/App.js

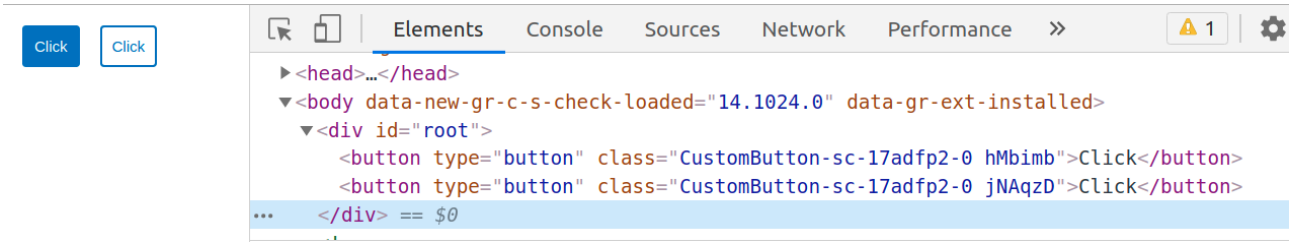
```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button" outline={false}>Click</CustomButton>
    <CustomButton type="submit" onClick={() => alert("clicked")} outline={true}
>Click</CustomButton>
  </>;
);

export default App;
```

Styled components can differentiate between the types of props they receive. They know that type is an HTML attribute, so they render

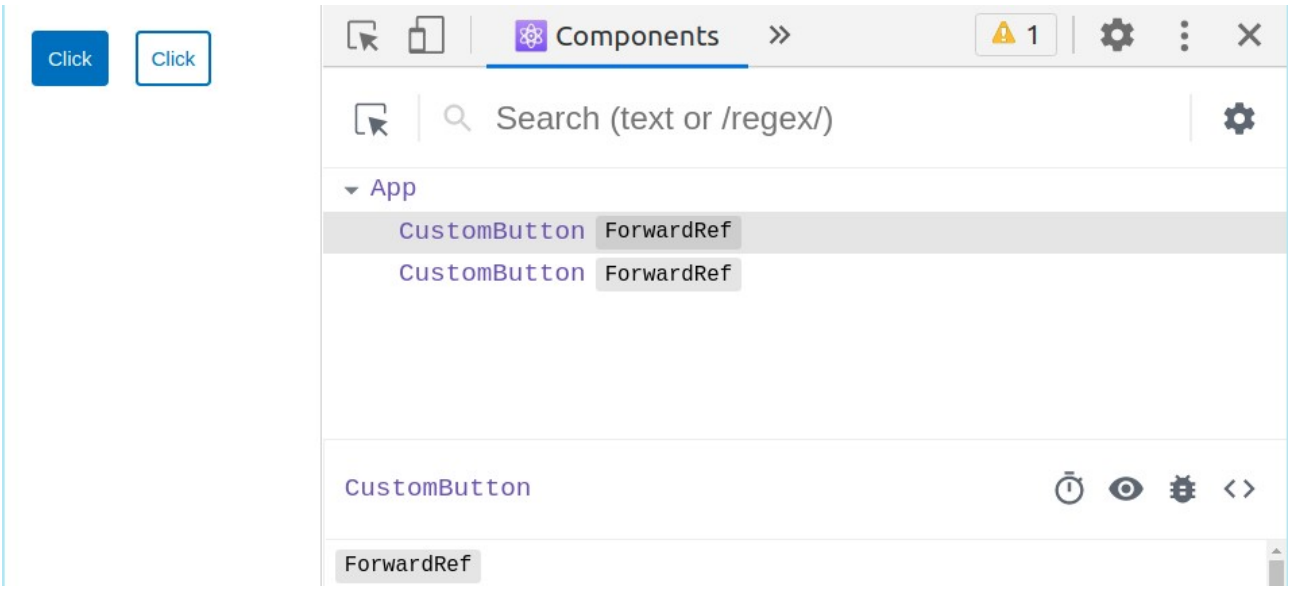
<button type="button">Click</button>, while using the **outline** prop in their own processing. Notice how we attached an event handler, too?

3.5 Easier Debugging

- babel-plugin-styled-components plugin adds support for a nicer debugging experience.
- This option enhances the attached CSS class name on each component with richer output to help identify your components in the DOM without React DevTools. In your page source you'll see:
  <button type="button" class="CustomButton-asdfxxx asdfxx" /> instead of just <button type="button" class="asdfxxx" />
  .



It also allows you to see the component's displayName in React DevTools. For example, consider writing a styled component that renders a button element, called **CustomButton**. It will normally show up in DevTools as styled.button, but with the displayName option enabled, it has the name you gave it **CustomButton**.



This makes it easier to find your components and to figure out where they live in your app.

- For more info, you can go through this link here.

Note:

- In order to configure create-react-app with the babel-plugin-styled-components, we have to use craco - Create React App Configuration Override is an easy and comprehensible configuration layer for create-react-app.
- We'll need to install craco with npm, and then create a craco.config.js at the root of our application, with the content:

module.exports = {

```
  babel: {
    plugins: [
      [
        "babel-plugin-styled-components",
        {
          fileName: false,
        },
      ],
    ],
  },
};
```

It is already configured.

3.6 Global Styling

- We will write **global** styles in the styledComponents.js file. Inside the GlobalStyle variable is where we define all global styles.
- We will import GlobalStyle component and place it at the top of React tree. In many react applications, that's typically the App.js file.
- CreateGlobalStyle is a helper function to generate a special StyledComponent that handles global styles.
- Normally, styled-components are automatically scoped to a local CSS class and therefore isolated from other components. In the case of **createGlobalStyle**, this limitation is removed.

Example:

File: src/styledComponents.js

```
import { createGlobalStyle } from "styled-components";

export const GlobalStyle = createGlobalStyle`
 body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
`;
```

File: src/App.js

```
import { CustomButton } from "./styledComponents";
import { GlobalStyle } from "./styledComponents";

const App = () => (
  <>
    <GlobalStyle />
    <CustomButton type="button">Click</CustomButton>
```

```
    <CustomButton type="button" outline>Click</CustomButton>
  </>
);

export default App;
```

To know more, you can go through this link.

4. Extending Styles

- Frequently you might want to use a component, but change it slightly for a single case. Now, you could pass in a function and change them based on some props, but that's quite a lot of effort for overriding the styles once.
- To easily make a new component that inherits the styling of another, just wrap it in the **styled()**. Here we use the **CustomButton** and create a special one, extending it with some color-related styling.

```
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: #ffffff;
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: #0070c1;
`;
```

Now we need to create another StyledComponent with similar properties of **CustomButton** except **color** and **border-color** property.

```
import styled from "styled-components";

export const OutlineButton = styled(CustomButton)`
  color: #0070c1;
  background-color: #ffffff;
  font-family: "Open sans";
`;
```

- It’s more or less like how the **spread** operator works.
- To know more, you can go through this link.

Note:

You will get a HTML button element rendered for OutlineButton.

5. The "as" prop

- You can pass the **as** prop to your styled component with the value of your preferred element.
- If you want to keep all the styling you've applied to a component but just switch out what's being ultimately rendered (be it a different HTML tag or a different custom component), you can use the **"as"** prop to do this at runtime.

Example:

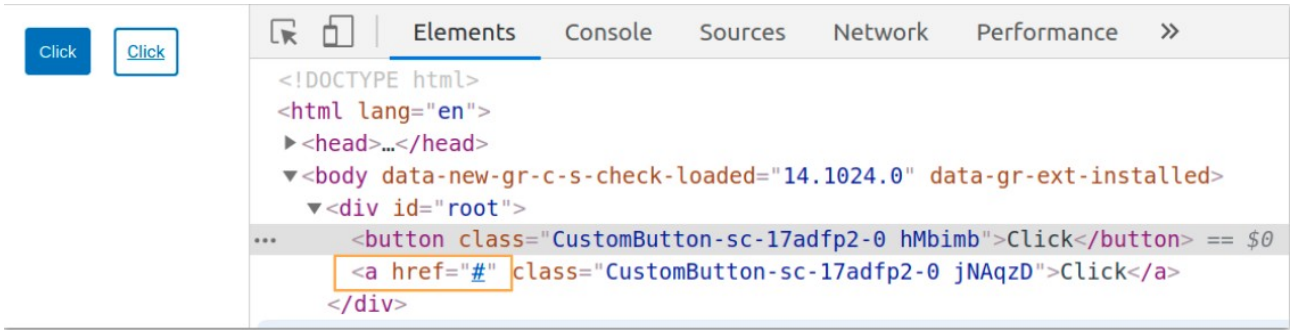
File:src/App.js

```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton outline={false}>Click</CustomButton>
    <CustomButton as="a" href="#" outline={true}>Click</CustomButton>
  </>
);

export default App;
```

File:src/styledComponents.js

```
import styled from "styled-components";

export const CustomButton = styled.button`
  padding: 10px;
  margin-right: 20px;
  font-size: 15px;
  color: ${(props) => (props.outline ? "#0070c1" : "#ffffff")};
  border-radius: 4px;
  border: 2px solid #0070c1;
  background-color: ${(props) => (props.outline ? "#ffffff" : "#0070c1")};
`;
```

- Consider the above code snippet, here the **OutlineButton** styled component is rendered as a button element, but you would prefer an **anchor** to a button for **OutlineButton**, you can pass the as a prop to your styled component with the value of your preferred element.



- This sort of thing is very useful in use cases like a navigation bar where some of the items should be links and some just buttons, but all be styled the same way.

6. Boolean Attribute

- The Presence of boolean attribute is taken as true no need to specify the value again.
- The absence of boolean attribute is taken as false.

```
import "./App.css";
import { CustomButton } from "./styledComponents";

const App = () => (
  <>
    <CustomButton type="button">Click</CustomButton>
    <CustomButton type="button" outline>Click</CustomButton>
  </>
);

export default App;
```