React JS Notes

-By Venisha Parmar

Date: 22/05/2023

What is React JS ?

- ReactJS is an open-source **JavaScript library** that is used for building **user interfaces** in an efficient way. Also we can create Single Page Applications.

What is a Single Page Application ?

→ While using any website when we are moving from one page to another page at that time the page will not reload it will simply direct to that page without reloading.

- It is **component-based front-end library** responsible for the view layer of and MVC architecture.

History of React JS

- Developed By Meta Engineer name Jordan Walke in 2011 for Facebook's News Feed feature. Then later on Instagram in 2012.
- Early prototype of React was FaxJs.

Why use React JS ?

- While building client-side apps, a team of FB developers realized that the DOM is slow.

What is DOM ?

→ TheDocument Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

- So to make it faster, **React implements a virtual DOM** that is basically a DOM tree representation in JavaScript but it is very efficient in memory.
- So when we are updating something in components that update takes place in virtual DOM and when the state changes the react will compare virtual DOM with the real DOM and it will only take the updated part to the real DOM and it will update it in real DOM.
- This is the reason why react is faster.

Latest version of React JS ?

→ React 18 (released in 2022).

Why named React ?

→ **React is called React because it reacts to changes in data.** This is in contrast to other JS Libraries, which often re-render the entire UI when data changes. React's ability to **only re-render the parts of the UI that have changed** makes it much more efficient.

Date: 23/05/2023

Overall about react

- React was originally created by Facebook In 2011 to improve the performance of their news feed. Facebook's news feed is constantly changing , so they needed a library that could react to changes in data quickly and efficiently.
- React is a **declarative library**, which means that you tell React what you want the UI to look like, and React figures out how to render it. This makes React much easier to learn and use than other JS libraries which often require you to write a lot of code to get the desired result.
- React is a **component-based library**, which means that you can break down your UI into smaller, reusable components. This makes React much more flexible and scalable than other JS libraries.

Things came to know while working on react.

- React uses **(JSX) JavaScript Syntax Extension** to insert dynamic JavaScript with HTML.(Combination of JS and HTML)
- In react we can create components in two ways :
    1) **Class Based Components.**
    2) **Function Based Components.**

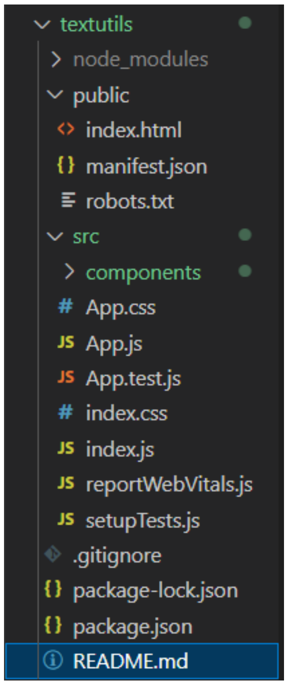
Date: 24/05/2023

Created React App named textutils.

CLI Tools ?

- CLI stands for **Command Line Interface** tools**,** which are programs that can be run from the command line. They are often used for tasks such as compiling code, running scripts, and managing files.
- **create-react-app** is a tool that helps you create a new React application. It provides a boilerplate project with all the necessary files and dependencies.

- Benefits of CLI tools : **Save time**, **Improves quality** of code, make your **code more portable** by generating code that can be run in a variety of environments.
- When we run the command **create-react-app app_name,** it creates a new React project with the following folder structure:



```
textutils
  node_modules
  public
    index.html
    manifest.json
    robots.txt
  src
    components
    App.css
    App.js
    App.test.js
    index.css
    index.js
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

- The **public** folder contains the static files that are served by the web server. This includes the HTML, CSS, and JavaScript files that make up the user interface.
- **manifest.json** file is a JSON file that contains information about React application. It is used by the browser to display application in the app launcher, and to control how your application behaves when it is installed.
- The **src** folder contains the source code for the application. This includes the React components, as well as any other code that is needed to run the application.(component folder is created for the components files)
- The **App.js** file is the main entry point for the application. It contains the code that renders the user interface. The **App.css** file contains the styles for the user interface. The **App.test.js** file contains the unit tests for the application.
- The **index.css** file contains the global styles for the application. The **index.js** file contains the code that bootstraps the application. The **logo.svg** file is the logo for the application.
- The **package.json** file is a manifest file that contains information about React project. It includes the following information:

  - The name of the project.
  - The version of the project.
  - The dependencies of the project.
  - The scripts that are used to build and test the project.
- The **package.json** file is used by npm to install and manage your project's dependencies. It is also used by the React development tools to build and test your project.

What are props and state of components ?

- Components **prop** means **the details which we are sending** or we want to send into any components.
- Components **state** means what **details are present in that component**. For example heading, this-that content inside the page.
- React Components are put inside the **src folder.**
- Props are sended to the components and there will be a state of components.
- The major code of react application is written through **components**.

While writing code……!

- Inside the src folder App.js is a component and index.js is our entry point.
- Index.js is rendering the App component

Code

D:\RKIT Internship\React Tutorials\textutils\public\index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <!-- This line specifies the URL of the favicon, which is a small icon that appears in the browser's address bar. -->
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta  name="viewport" content="width=device-width, initial-scale=1" />
  <meta  name="theme-color" content="#000000" />
  <meta name="description" content="Web site created using create-react-app"  />

  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

  <!--  manifest.json provides metadata used when your web app is installed on a  user's mobile
device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/  -->

  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

  <!-- This line specifies the URL of the manifest file,

    which contains information about the document,

    such as the permissions that the document needs.

    Notice the use of %PUBLIC_URL% in the tags above.

    It will be replaced with the URL of the `public` folder during the build.

    Only files inside the `public` folder can be referenced from the HTML.

    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will

    work correctly both with client-side routing and a non-root public URL.

    Learn how to configure a non-root public URL by running `npm run build`.

  -->

  <title>React App</title>

 </head>

 <body>

  <noscript>You need to enable JavaScript to run this app.</noscript>

  <!-- This line displays a message if the browser does not have JavaScript enabled. -->

  <div id="root"></div>

  <!-- This line creates a div element with the id of "root".

      This div element will be used to hold the content of the React application.

      This div is populated with the use of javascript. -->

  <!-- This HTML file is a template.

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

    You can add web fonts, meta tags, or analytics to this file.

    The build step will place the bundled scripts into the <body> tag.

    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->
 </body>
</html>
<!-- All rendering is done here in this file . Other major code is written in components. -->
```

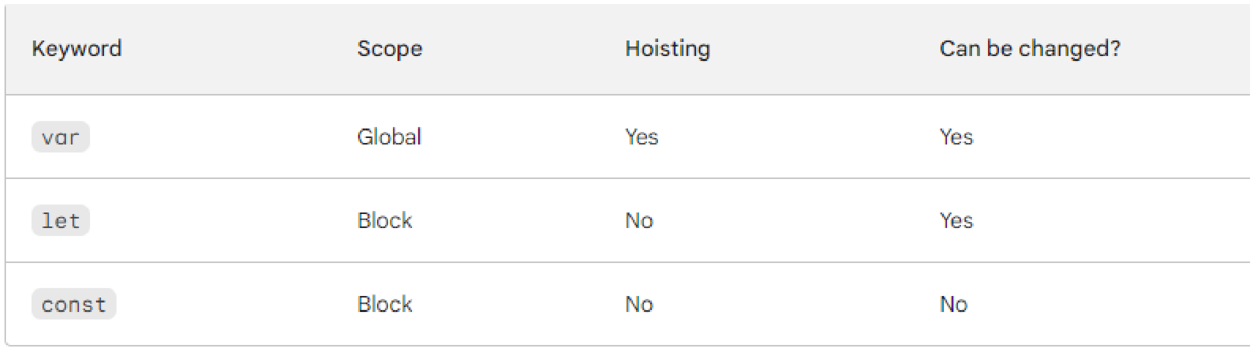
D:\RKIT Internship\React Tutorials\textutils\src\index.js

```
// It is the starting point for a React application.
// It imports the React and ReactDOM libraries,
// as well as the App component and the reportWebVitals function.
// It then creates a root element and renders the App component into it.
// Finally,  it calls the reportWebVitals function to start measuring the performance of the application.
import React from 'react';
import ReactDOM from 'react-dom/client'; //ReactDOM is a JS library for rendering React components to the DOM.
import './index.css';
import App from './App'; //(imports App Component) The App component is the main component for the application.
import reportWebVitals from './reportWebVitals'; // The reportWebVitals function is used to start measuring the performance of the application.



// creates a root element and assigns it to the root variable.
// The root element is the top-level element in the application.
const root = ReactDOM.createRoot(document.getElementById('root'));

//This line renders the App component into the root element.
//The React.StrictMode component is used to enable strict mode in the application.
//Strict mode helps to prevent errors and improve the performance of the application.
root.render(
  <React.StrictMode>
  <App />
 </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
// calls the reportWebVitals function. The reportWebVitals function starts measuring the
// performance of the application.
// The results of the performance measurement can be used to
// improve the performance of the application.
```

- Whatever Changes made in **App.js Component** will come to this `document.getElementById('root')` with the help of **index.js file.**

While Refreshing the Concepts of JavaScript.

3 Ways to create variable : **let , var , const**



Keyword	Scope	Hoisting	Can be changed?
`var`	Global	Yes	Yes
`let`	Block	No	Yes
`const`	Block	No	No

let : it can be updated but can't be redeclared into the scope.

var : it can be updated and redeclared into the scope.

const : it cannot be updated and redeclared into the scope.

Hoisting :

- Hoisting is a concept in JavaScript that refers to the process of moving variable declarations to the top of their scope. This means that even though a variable may be declared in the middle of a block of code, it will be available as soon as the block starts.
- Hoisting is important because it allows you to use variables before they are declared. This can be useful for things like declaring variables that are used in multiple places in a block of code.
- However, hoisting can also lead to errors if you are not careful. For example, if you try to use a variable that has not been declared, you will get an error.
- To avoid errors, it is important to be aware of hoisting and to declare your variables early in your code.

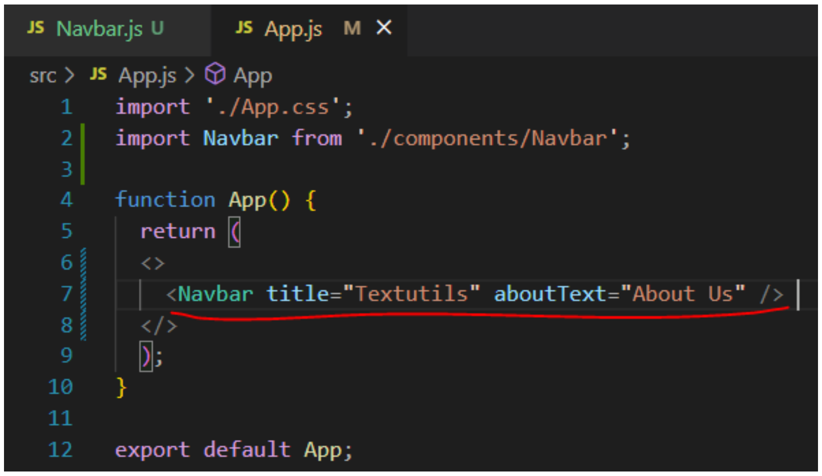
Date: 25/05/2023

- While importing modules from other files . We write the module name inside { } which we are not exporting by default.
- First folder created for react app was : **D:\RKIT Internship\React Tutorials\textutils\src\components**
- Inside components folder : New File created (new component : **D:\RKIT Internship\React Tutorials\textutils\src\components\Navbar.js**

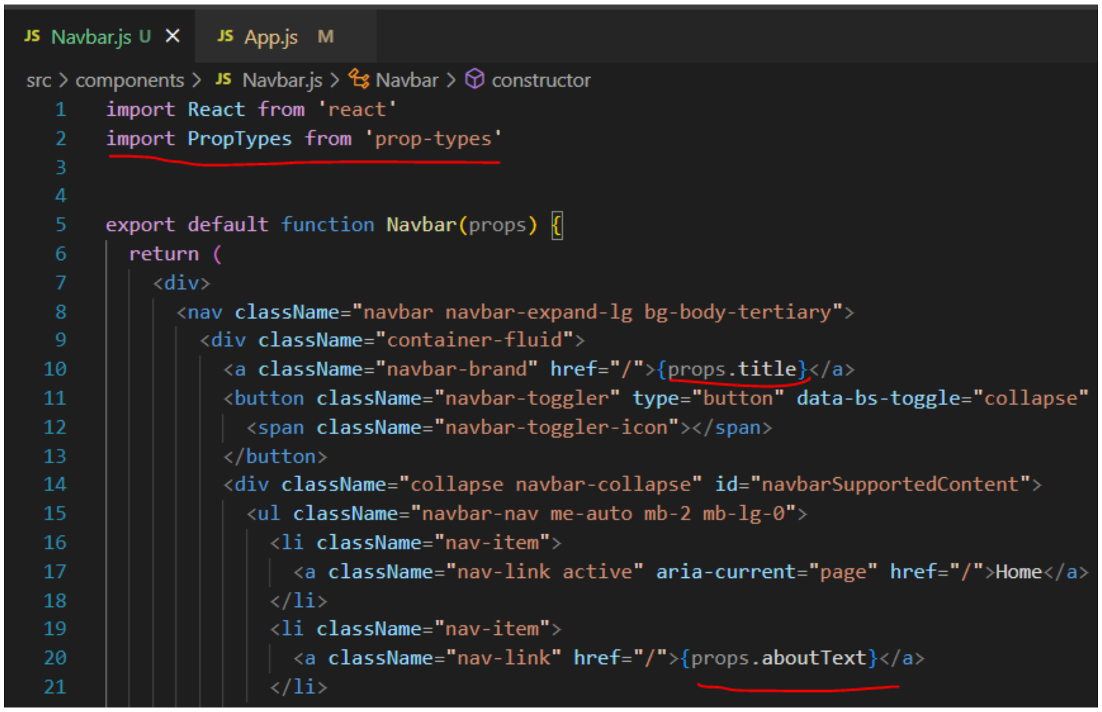
Props and PropTypes implementation in React.

- Props stands for **properties**. They are used to pass data from one component to another.
- Props are **read-only**, which means that they cannot be changed from the child component. If you need to change the value of a prop, you need to do so in the parent component.
- **PropTypes :** Proptypes are a way to validate the props that are passed to a React component. They are used to ensure that the props are of the correct type and that they have the correct values.
- Proptypes are defined using the **PropTypes** object. The PropTypes object has a number of methods that can be used to validate props.

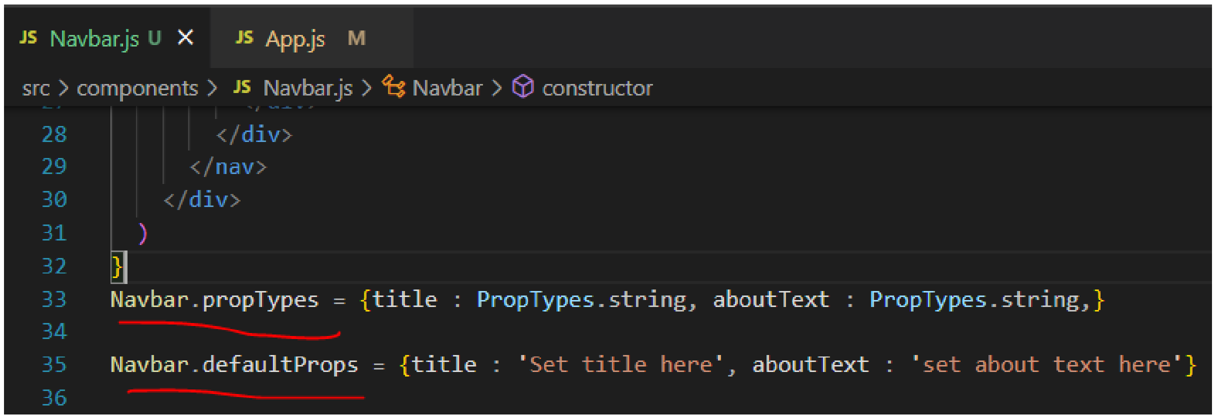
- Default props are a way to provide default values for props that are not passed to a React component. They are defined using the **defaultProps** property on the component definition.



```
import './App.css';
import Navbar from './components/Navbar';

function App() {
  return (
    <>
      <Navbar title="Textutils" aboutText="About Us" />
    </>
  );
}

export default App;
```



```
import React from 'react'
import PropTypes from 'prop-types'


export default function Navbar(props) {
  return (
    <div>
      <nav className="navbar navbar-expand-lg bg-body-tertiary">
        <div className="container-fluid">
          <a className="navbar-brand" href="/">{props.title}</a>
          <button className="navbar-toggler" type="button" data-bs-toggle="collapse"
            <span className="navbar-toggler-icon"></span>
          </button>
          <div className="collapse navbar-collapse" id="navbarSupportedContent">
            <ul className="navbar-nav me-auto mb-2 mb-lg-0">
              <li className="nav-item">
                <a className="nav-link active" aria-current="page" href="/">Home</a>
              </li>
              <li className="nav-item">
                <a className="nav-link" href="/">{props.aboutText}</a>
              </li>
```


JS Navbar.js U × JS App.js M

src > components > JS Navbar.js > Navbar > constructor

```
            </div>
          </nav>
        </div>
      )
    }
    Navbar.propTypes = {title : PropTypes.string, aboutText : PropTypes.string,}

    Navbar.defaultProps = {title : 'Set title here', aboutText : 'set about text here'}

```

Date: 26/05/2023

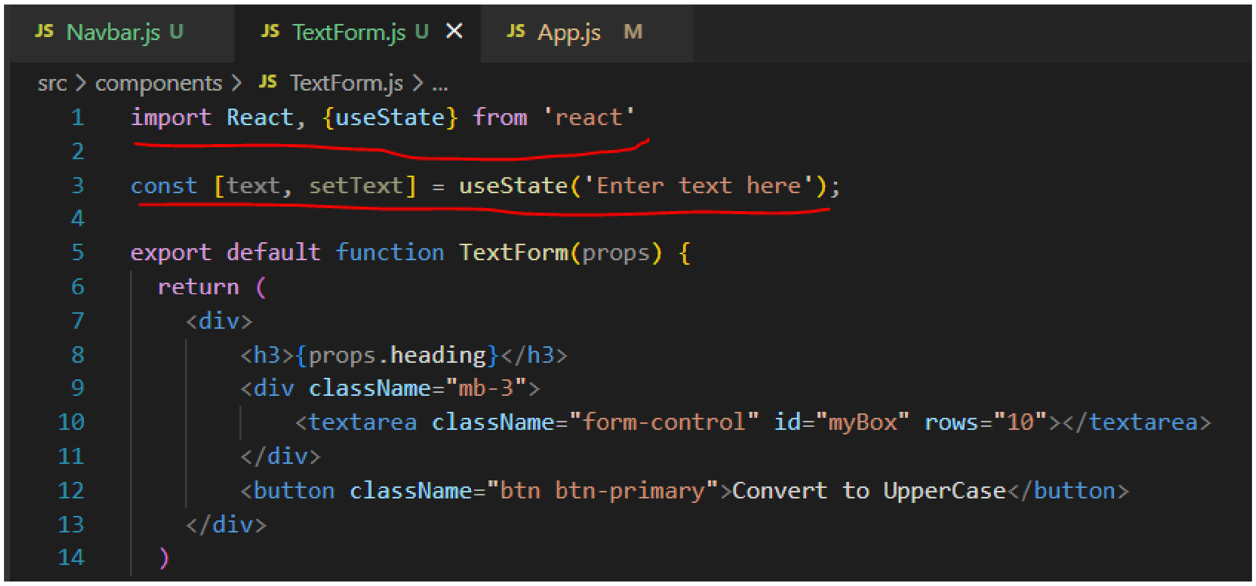
Understanding State and Handling Events in React.

NewFile Created (Component) : **D:\RKIT Internship\React Tutorials\textutils\src\components\TextForm.js**

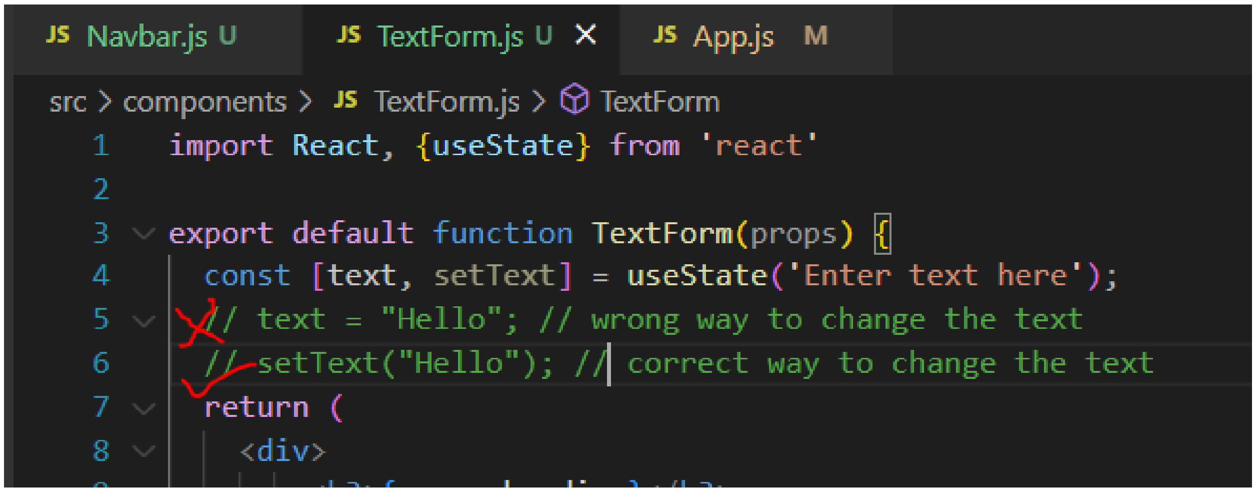
- State in React is a **way to store data that can be changed over time**. It is used to **keep track of the current state of a component**. State is always local to a component, and it is not shared with other components.
- State is defined using the **state** property on the component definition. The **state** property is an object that contains the current state of the component.
- Import useState from react. **useState is a hook.**
- **What is hook ? ->** They let us use state and other React features without writing a class.
- The **useState** hook is a React hook that allows us to manage state in your components.

```
const [text, setText] = useState('Enter text here');
```

- The **useState** hook to create a state variable called **text** and a function called **setText**. The **text** variable is initialized with the value **'Enter text here'.** The **setText** function takes a new value for the **text** variable as an argument and updates the state of the **text** variable.
- Directly we can't change the value assigned to text , we need to used setText function to change the value of text.



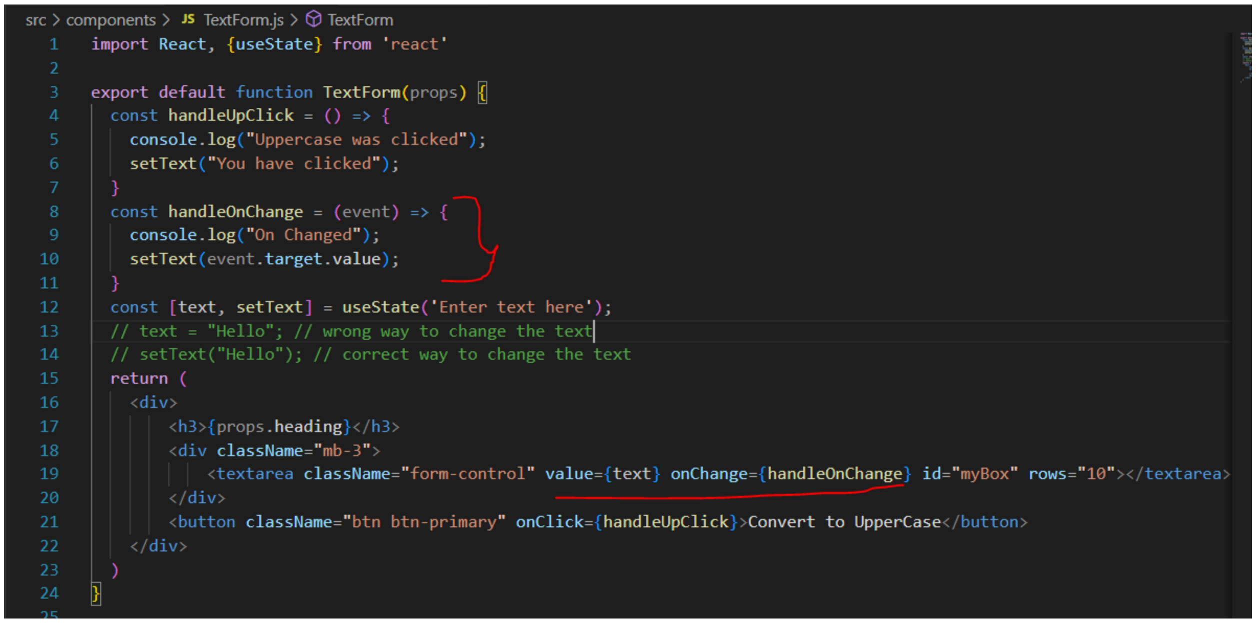
```
import React, {useState} from 'react'

const [text, setText] = useState('Enter text here');

export default function TextForm(props) {
  return (
    <div>
        <h3>{props.heading}</h3>
        <div className="mb-3">
            <textarea className="form-control" id="myBox" rows="10"></textarea>
        </div>
        <button className="btn btn-primary">Convert to UpperCase</button>
    </div>
  )
```



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

export default function TextForm(props) {
    const [text, setText] = useState('Enter text here');
    // text = "Hello"; // wrong way to change the text
    // setText("Hello"); // correct way to change the text
    return (
      <div>
```

How to handle events in react ?

1. Create a function that will handle the event. This function will be called when the event is triggered.
2. Pass the function to the event handler attribute of the DOM element that you want to handle the event.
3. The event handler attribute takes a function as its value. The function will be called when the event is triggered.



```
src > components > JS TextForm.js > TextForm
import React, {useState} from 'react'

export default function TextForm(props) {
  const handleUpClick = () => {
    console.log("Uppercase was clicked");
    setText("You have clicked");
  }
  const handleOnChange = (event) => {
    console.log("On Changed");
    setText(event.target.value);
  }
  const [text, setText] = useState('Enter text here');
  // text = "Hello"; // wrong way to change the text
  // setText("Hello"); // correct way to change the text
  return (
    <div>
        <h3>{props.heading}</h3>
        <div className="mb-3">
          <textarea className="form-control" value={text} onChange={handleOnChange} id="myBox" rows="10"></textarea>
        </div>
        <button className="btn btn-primary" onClick={handleUpClick}>Convert to UpperCase</button>
    </div>
  )
}
```

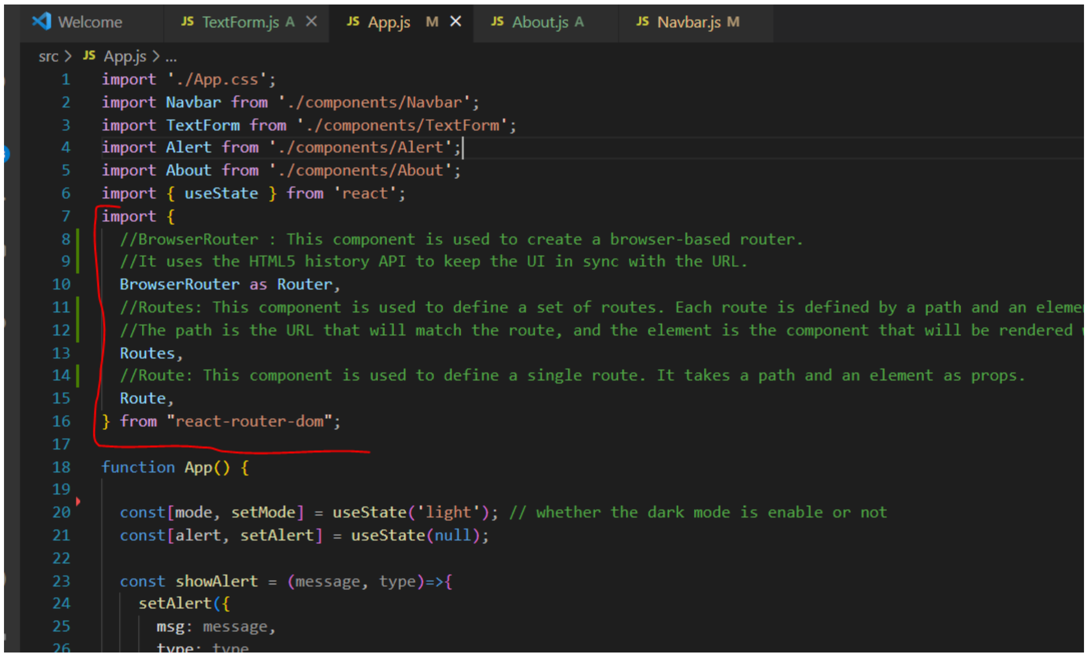
- The **handleOnChange** function is a regular JavaScript function. It takes one argument, which is the event object. The **event** object contains information about the event that was triggered. In this case, the event object will contain information about the text area that was changed.
- The **setText(event.target.value)**; statement sets the text state variable to the new value of the text area. The **event.target.value** property contains the new value of the text area.
- The **onChange={handleOnChange}** attribute is used to bind the handleOnChange function to the onChange event of the text area. This means that the handleOnChange function will be called whenever the value of the text area is changed.

React Router Setup + Usage

- **Routing** in React is the process of navigating between different pages in a React application. React Router is a popular library for handling routing in React applications.
- To use React Router, first we need to install it. We can do this with the following command: **npm install react-router**

- Once we have installed React Router, we can start using it to define routes in our application. Routes are defined using the **Route** component.
- The **Route** component takes a path and a component as props. The path is the URL that will be used to match the route. The component is the component that will be rendered when the route is matched.

D:\RKIT Internship\React Tutorials\TextUtils-React-App\textutils\src\App.js



```
import './App.css';
import Navbar from './components/Navbar';
import TextForm from './components/TextForm';
import Alert from './components/Alert';
import About from './components/About';
import { useState } from 'react';
import {
  //BrowserRouter : This component is used to create a browser-based router.
  //It uses the HTML5 history API to keep the UI in sync with the URL.
  BrowserRouter as Router,
  //Routes: This component is used to define a set of routes. Each route is defined by a path and an eleme
  //The path is the URL that will match the route, and the element is the component that will be rendered
  Routes,
  //Route: This component is used to define a single route. It takes a path and an element as props.
  Route,
} from "react-router-dom";

function App() {

  const[mode, setMode] = useState('light'); // whether the dark mode is enable or not
  const[alert, setAlert] = useState(null);

  const showAlert = (message, type)=>{
    setAlert({
      msg: message,
      type: type
```

- **BrowserRouter:** This component is used to create a browser-based router. It uses the HTML5 history API to keep the UI in sync with the URL.
- **Routes:** This component is used to define a set of routes. Each route is defined by a **path** and an **element**. The **path** is the URL that will match the route, and the **element** is the component that will be rendered when the route matches.
- **Route**: This component is used to define a single route. It takes a path and an **element** as props.
- **Link**: This component is used to create links to other routes. It takes a to prop, which is the path of the route that the link should navigate to.

🎉Here TextUtils React App Completes….! (Function based Components are Used here)