render() Method

The render() method is the only required method in a class component. It examines this.props and this.state .

It returns one of the following types:

React elements – These are created via JSX(Not required).

For example, and <App /> are React elements that instruct React to render a DOM node, or another user-defined component, respectively.

Arrays and fragments - It is used to return multiple elements from render.

Portals – It is used to render children into a different DOM subtree.

String and numbers - These are rendered as text nodes in the DOM.

Booleans or null - It renders nothing. (Mostly exists to support return test && <Child /> pattern, where test is boolean.)

Note - The render() function should be pure, meaning that it does not modify component state, it returns the same result each time it's invoked, and it does not directly interact with the browser.

React Element

You can create a react element using React.createElement() method but there is a easy way to create element via JSX.

Using createElement() Method

React.createElement(“h1”, null, “Hello GeekyShows”);

Using JSX

<h1>Hello GeekyShows</h1>

React.createElement(type, props, children)

React.createElement(type, props, children) - It creates a React Element with the given arguments.

Syntax:- React.createElement(type, props, children)

- type: Type of the html element or component. (example : h1,h2,p,button..etc).
- props: The properties object.

  Example: {style :{ color:“blue”}} or className or event handlers etc.
- children: anything you need to pass between the dom elements.

Ex:-

React.createElement(‘h1’, null, ‘Hello GeekyShows’);

React Fragment

Fragment is used to group a list of children without adding extra nodes to the DOM.

Syntax:-

```
<React.Fragment>
</React.Fragment>
```

Ex:-

```
<React.Fragment>
    <h1>Hello</h1>
    <h2>GeekyShows</h2>
</React.Fragment>
```

Syntax:-

```
<>
</>
```

Ex:-

```
<>
    <h1>Hello</h1>
    <h2>GeekyShows</h2>
</>
```

Syntax:-

```
<React.Fragment key={id}>
</React.Fragment>
```

Ex:-

```
<React.Fragment key={item.id}>
   <h1>{item.title}</h1>
    <p>{item.description}</p>
</React.Fragment>
```

ReactDOM.render(element, DOMnode)

ReactDOM.render(element, DOMnode) - It takes a React Element and render it to a DOM node.

Syntax:- ReactDOM.render(element, DOMnode)

- The first argument is which component or element needs to render in the dom.
- The second argument is where to render in the dom.

Ex:-

ReactDOM.render(< App />, document.getElementById(“root”));

Webpack parses through the application starting at src/index.js, following any imported modules, until it has a complete dependency graph.

In order to convert the ES2015+ code that Webpack comes across into a version of JavaScript that will behave consistently across browsers, Webpack passes any JavaScript files it comes across through Babel.

Babel is a transpiler which parses newer and experimental JavaScript syntax, and transforms the code into a version of JavaScript which has better support across browsers.

Files src/index.js and public/index.html, these files can be modified as necessary, but the names and locations shouldn't be altered.

App.test.js is a simple unit test Create React App sets up to test if the App component renders without crashing.