React JS report:(in short )

we are studying documentation of React Js <https://react.dev/>

Udemy course: React JS beginners and advance level

# React Introduction:

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library responsible only for the view layer of the application

## Using the create-react-app command

****Install NodeJS and NPM****

NodeJS and NPM are the platforms need to develop any ReactJS application. You can install NodeJS and NPM package manager by the link given below.

<https://www.javatpoint.com/install-nodejs-on-linux-ubuntu-centos>

****Install React****

You can install React using npm package manager by using the below command. There is no need to worry about the complexity of React installation. The create-react-app npm package will take care of it.

1. javatpoint@root:~/>npm install -g create-react-app

****Create a new React project****

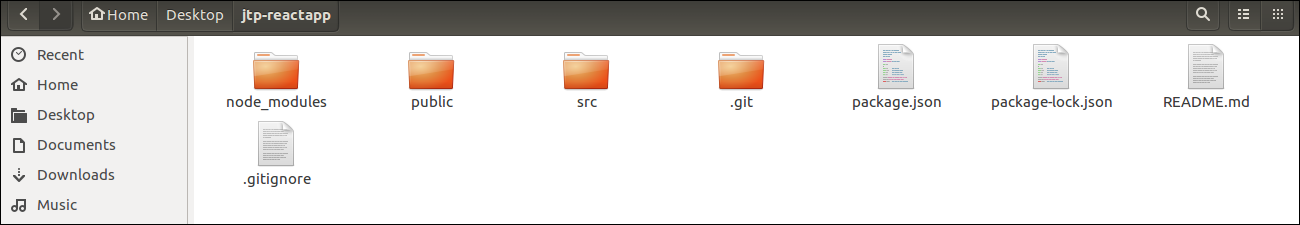
After the installation of React, you can create a new react project using create-react-app command. Here, I choose ****jtp-reactapp**** name for my project.

1. javatpoint@root:~/>create-react-app jtp-reactapp

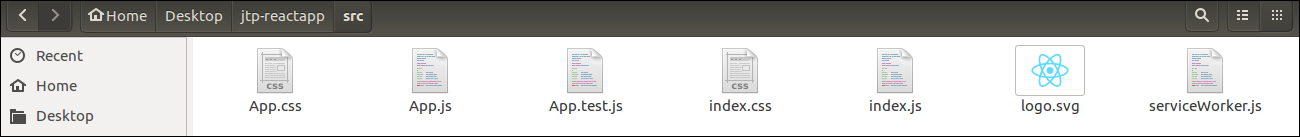
#### NOTE: You can combine the above two steps in a single command using npx. The npx is a package runner tool that comes with npm 5.2 and above version.

1. javatpoint@root:~/>npx create-react-app jtp-reactapp

The above command will install the react and create a new project with the name jtp-reactapp. This app contains the following sub-folders and files by default which can be shown in the below image.



Now, to get started, open the ****src**** folder and make changes in your desired file. By default, the src folder contain the following files shown in below image.



For example, I will open ****App.js**** and make changes in its code which are shown below.

**App.js**

1. **import** React from 'react';
2. **import** logo from './logo.svg';
3. **import** './App.css';
5. function App() {
6. **return** (
7. <div className="App">
8. <header className="App-header">
9. <img src={logo} className="App-logo" alt="logo" />
10. <p>
11. Welcome To JavaTpoint.
13. <p>To get started, edit src/App.js and save to reload.</p>
14. </p>
15. <a
16. className="App-link"
17. href="https://reactjs.org"
18. target="\_blank"
19. rel="noopener noreferrer"
20. >
21. Learn React
22. </a>
23. </header>
24. </div>
25. );
26. }
28. export **default** App;

#### **NOTE:** You can also choose your own favorite code editor for editing your project. But in my case, I choose Eclipse. Using the below link, you can download Eclipse for Ubuntu and install.

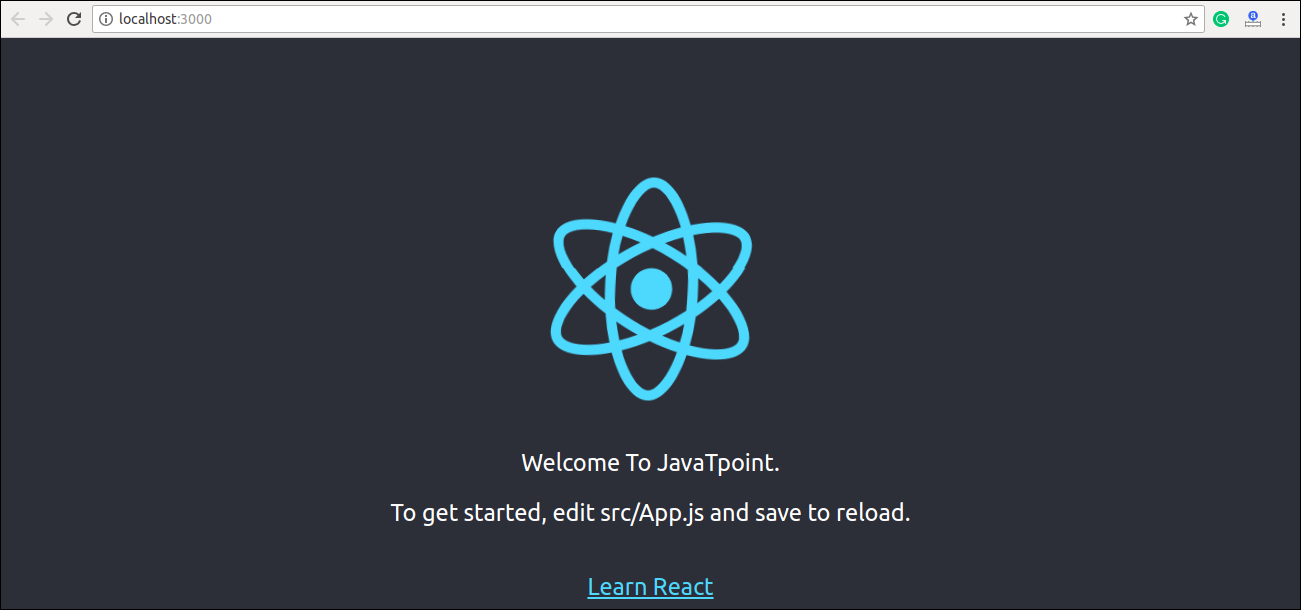
**[click Here to download Eclipse for Ubuntu and install](https://www.javatpoint.com/how-to-install-eclipse-in-ubuntu)**

****Running the Server****

After completing the installation process, you can start the server by running the following command.

1. javatpoint@root:~/Desktop>cd jtp-reactapp
2. javatpoint@root:~/Desktop/jtp-reactapp>npm start

It will show the port number which we need to open in the browser. After we open it, you will see the following output.



# Pros and Cons of ReactJS

### Advantage of ReactJS

****1. Easy to Learn and USe****

****2. Creating Dynamic Web Applications Becomes Easier****

****3. Reusable Components****

****4. Performance Enhancement****

****5. The Support of Handy Tools****

****6. Known to be SEO Friendly****

****7. The Benefit of Having JavaScript Library****

****8. Scope for Testing the Codes****

### Disadvantage of ReactJS

1. ****The high pace of development****
2. ****Poor Documentation****
3. ****View Part****
4. ****JSX as a barrier****

# React JSX

The render function specifies the HTML output of a React component. JSX(JavaScript Extension), is a React extension which allows writing JavaScript code that looks like HTML. In other words, JSX is an HTML-like syntax used by React that extends ECMAScript so that ****HTML-like**** syntax can co-exist with JavaScript/React code. The syntax is used by ****preprocessors**** (i.e., transpilers like babel) to transform HTML-like syntax into standard JavaScript objects that a JavaScript engine will parse.

# React Components

Earlier, the developers write more than thousands of lines of code for developing a single page application. These applications follow the traditional DOM structure, and making changes in them was a very challenging task. If any mistake found, it manually searches the entire application and update accordingly. The component-based approach was introduced to overcome an issue. In this approach, the entire application is divided into a small logical group of code, which is known as components.

A Component is considered as the core building blocks of a React application. It makes the task of building UIs much easier. Each component exists in the same space, but they work independently from one another and merge all in a parent component, which will be the final UI of your application.

Every React component have their own structure, methods as well as APIs. They can be reusable as per your need. For better understanding, consider the entire UI as a tree. Here, the root is the starting component, and each of the other pieces becomes branches, which are further divided into sub-branches.

In ReactJS, we have mainly two types of components. They are

1. Functional Components
2. Class Components

## Functional Components

1. function WelcomeMessage(props) {
2. **return** <h1>Welcome to the , {props.name}</h1>;
3. }

## Class Components

**class** MyComponent **extends** React.Component {

  render() {

**return** (

      <div>This is main component.</div>

    );

  }

}

# React State

The state is an updatable structure that is used to contain data or information about the component. The state in a component can change over time. The change in state over time can happen as a response to user action or system event. A component with the state is known as stateful components. It is the heart of the react component which determines the behavior of the component and how it will render. They are also responsible for making a component dynamic and interactive.

## Changing the State

We can change the component state by using the setState() method and passing a new state object as the argument. Now, create a new method toggleDisplayBio() in the above example and bind this keyword to the toggleDisplayBio() method otherwise we can't access this inside toggleDisplayBio() method.

**this**.toggleDisplayBio = **this**.toggleDisplayBio.bind(**this**);

# React Props

Props stand for "****Properties****." They are ****read-only**** components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

Props are ****immutable**** so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as ****this.props**** and can be used to render dynamic data in our render method.

When you need immutable data in the component, you have to add props to ****reactDom.render()**** method in the ****main.js**** file of your ReactJS project and used it inside the component in which you need. It can be explained in the below example.

### Example

****App.js****

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

**class** App **extends** React.Component {

   render() {

**return** (

          <div>

            <h1> Welcome to { **this**.props.name } </h1>

            <p> <h4> Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad. </h4> </p>

          </div>

      );

   }

}

export **default** App;

****Main.js****

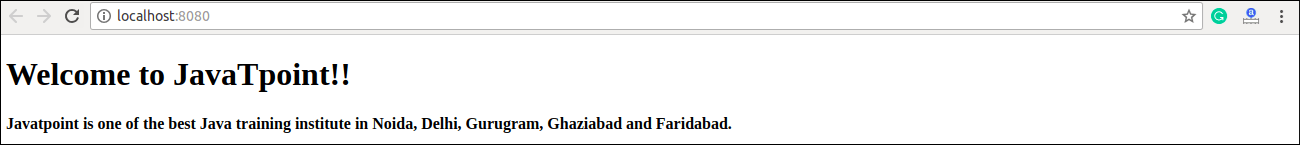
**import** React from 'react';

**import** ReactDOM from 'react-dom';

**import** App from './App.js';

ReactDOM.render(<App name = "JavaTpoint!!" />, document.getElementById('app'));

****Output****



## Default Props

It is not necessary to always add props in the reactDom.render() element. You can also set ****default**** props directly on the component constructor. It can be explained in the below example.

### Example

****App.js****

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

**class** App **extends** React.Component {

   render() {

**return** (

          <div>

              <h1>Default Props Example</h1>

            <h3>Welcome to {**this**.props.name}</h3>

            <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.</p>

          </div>

        );

    }

}

App.defaultProps = {

   name: "JavaTpoint"

}

export **default** App;

****Main.js****

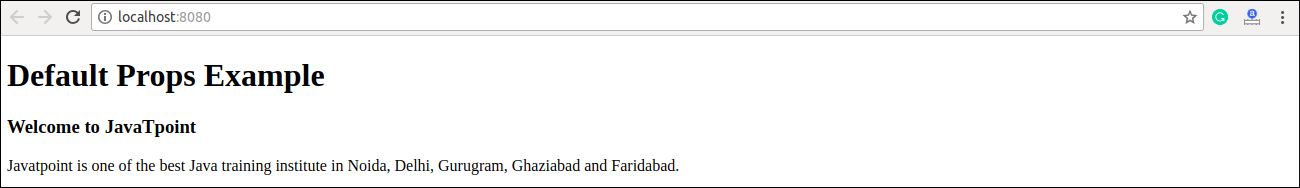
**import** React from 'react';

**import** ReactDOM from 'react-dom';

**import** App from './App.js';

ReactDOM.render(<App/>, document.getElementById('app'));

****Output****



## State and Props

It is possible to combine both state and props in your app. You can set the state in the parent component and pass it in the child component using props. It can be shown in the below example.

### Example

****App.js****

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

**class** App **extends** React.Component {

   constructor(props) {

**super**(props);

**this**.state = {

         name: "JavaTpoint",

      }

   }

   render() {

**return** (

         <div>

            <JTP jtpProp = {**this**.state.name}/>

         </div>

      );

   }

}

**class** JTP **extends** React.Component {

   render() {

**return** (

          <div>

              <h1>State & Props Example</h1>

              <h3>Welcome to {**this**.props.jtpProp}</h3>

              <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.</p>

         </div>

      );

   }

}

export **default** App;

****Main.js****

**import** React from 'react';

**import** ReactDOM from 'react-dom';

**import** App from './App.js';

ReactDOM.render(<App/>, document.getElementById('app'));

# React Forms

Forms are an integral part of any modern web application. It allows the users to interact with the application as well as gather information from the users. Forms can perform many tasks that depend on the nature of your business requirements and logic such as authentication of the user, adding user, searching, filtering, booking, ordering, etc. A form can contain text fields, buttons, checkbox, radio button, etc.

## Creating Form

React offers a stateful, reactive approach to build a form. The component rather than the DOM usually handles the React form. In React, the form is usually implemented by using controlled components.

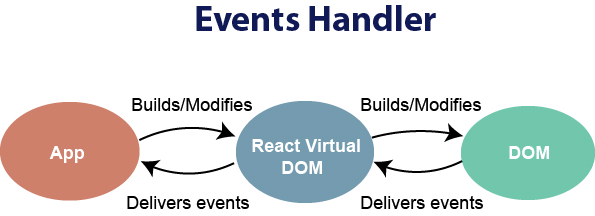
There are mainly two types of form input in React.

1. Uncontrolled component
2. Controlled component

# React Events

An event is an action that could be triggered as a result of the user action or system generated event. For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.

React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.



# React Conditional Rendering

# React Fragments

In React, whenever you want to render something on the screen, you need to use a render method inside the component. This render method can return ****single**** elements or ****multiple**** elements. The render method will only render a single root node inside it at a time. However, if you want to return multiple elements, the render method will require a '****div****' tag and put the entire content or elements inside it. This extra node to the DOM sometimes results in the wrong formatting of your HTML output and also not loved by the many developers.

// Rendering with div tag

**class** App **extends** React.Component {

     render() {

**return** (

         //Extraneous div element

         <div>

           <h2> Hello World! </h2>

           <p> Welcome to the JavaTpoint. </p>

         </div>

      );

     }

}

## Why we use Fragments?

The main reason to use Fragments tag is:

1. It makes the execution of code faster as compared to the div tag.
2. It takes less memory.

# React Router

Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application. When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route.

## Need of React Router

React Router plays an important role to display multiple views in a single page application. Without React Router, it is not possible to display multiple views in React applications. Most of the social media websites like Facebook, Instagram uses React Router for rendering multiple views.

## React Router Installation

React contains three different packages for routing. These are:

1. ****react-router:**** It provides the core routing components and functions for the React Router applications.
2. ****react-router-native:**** It is used for mobile applications.
3. ****react-router-dom:**** It is used for web applications design.

It is not possible to install react-router directly in your application. To use react routing, first, you need to install react-router-dom modules in your application. The below command is used to install react router dom.

1. $ npm install react-router-dom --save

## Components in React Router

There are two types of router components:

* ****<BrowserRouter>:**** It is used for handling the dynamic URL.
* ****<HashRouter>:**** It is used for handling the static request.

### Example

****Step-1:**** In our project, we will create two more components along with ****App.js****, which is already present.

****About.js****

**import** React from 'react'

**class** About **extends** React.Component {

  render() {

**return** <h1>About</h1>

  }

}

export **default** About

****Contact.js****

**import** React from 'react'

**class** Contact **extends** React.Component {

  render() {

**return** <h1>Contact</h1>

  }

}

export **default** Contact

****App.js****

**import** React from 'react'

**class** App **extends** React.Component {

  render() {

**return** (

      <div>

        <h1>Home</h1>

      </div>

    )

  }

}

export **default** App

****Step-2:**** For Routing, open the index.js file and import all the three component files in it. Here, you need to import line: ****import { Route, Link, BrowserRouter as Router } from 'react-router-dom'**** which helps us to implement the Routing. Now, our index.js file looks like below.

## What is Route?

It is used to define and render component based on the specified path. It will accept components and render to define what should be rendered.

## Adding Navigation using Link component

Sometimes, we want to need ****multiple**** links on a single page. When we click on any of that particular ****Link****, it should load that page which is associated with that path without ****reloading**** the web page. To do this, we need to import ****<Link>**** component in the ****index.js**** file.

### What is < Link> component?

This component is used to create links which allow to ****navigate**** on different ****URLs**** and render its content without reloading the webpage.

# React CSS

CSS in React is used to style the React App or Component. The ****style**** attribute is the most used attribute for styling in React applications, which adds dynamically-computed styles at render time. It accepts a JavaScript object in ****camelCased**** properties rather than a CSS string. There are many ways available to add styling to your React App or Component with CSS. Here, we are going to discuss mainly ****four**** ways to style React Components, which are given below:

1. Inline Styling
2. CSS Stylesheet
3. CSS Module
4. Styled Components

# React Bootstrap

Single-page applications gaining popularity over the last few years, so many front-end frameworks have introduced such as Angular, React, Vue.js, Ember, etc. As a result, jQuery is not a necessary requirement for building web apps. Today, React has the most used JavaScript framework for building web applications, and Bootstrap become the most popular CSS framework. So, it is necessary to learn various ways in which Bootstrap can be used in React apps, which is the main aim of this section

## Adding Bootstrap for React

We can add Bootstrap to the React app in several ways. The ****three**** most common ways are given below:

1. Using the Bootstrap CDN
2. Bootstrap as Dependency
3. React Bootstrap Package

## Using the Bootstrap CDN

It is the easiest way of adding Bootstrap to the React app. There is no need to install or download Bootstrap. We can simply put an ****<link>**** into the ****<head>**** section of the ****index.html**** file of the React app as shown in the following snippet

## Bootstrap as Dependency

If we are using a build tool or a module bundler such as Webpack, then importing Bootstrap as dependency is the preferred option for adding Bootstrap to the React application. We can install Bootstrap as a dependency for the React app. To install the Bootstrap, run the following commands in the terminal window.

1. $ npm install bootstrap --save

## React Bootstrap Package

The React Bootstrap package is the most popular way to add bootstrap in the React application. There are many Bootstrap packages built by the community, which aim to rebuild Bootstrap components as React components. The ****two**** most popular Bootstrap packages are:

1. ****react-bootstrap:**** It is a complete re-implementation of the Bootstrap components as React components. It does not need any dependencies like bootstrap.js or jQuery. If the React setup and React-Bootstrap installed, we have everything which we need.
2. ****reactstrap:**** It is a library which contains React Bootstrap 4 components that favor composition and control. It does not depend on jQuery or Bootstrap JavaScript. However, react-popper is needed for advanced positioning of content such as Tooltips, Popovers, and auto-flipping Dropdowns.

# React Map

## In React, the map() method used for:

1. Traversing the list element.