React is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta and a community of individual developers and companies.

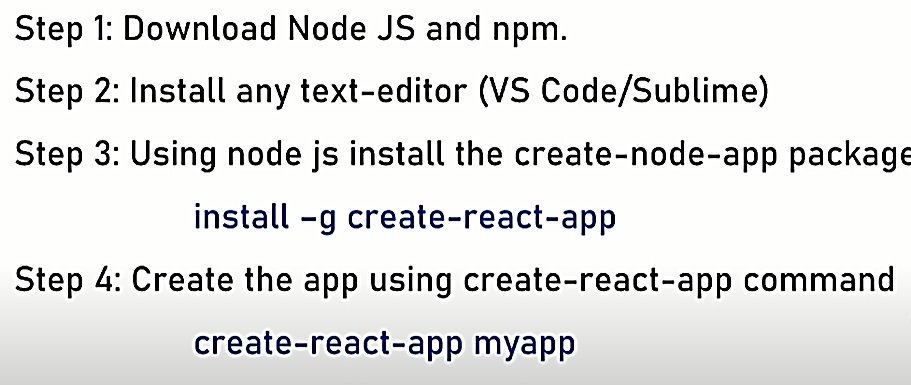
React is a JavaScript library created by Facebook

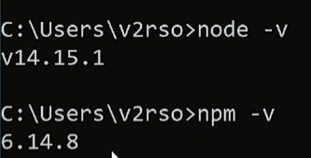
React is a User Interface (UI) library

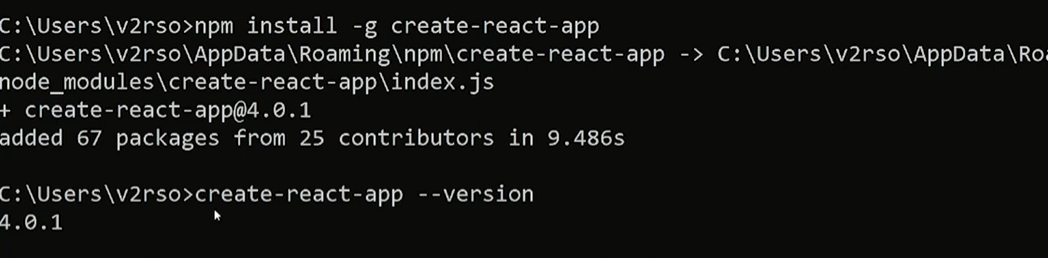
React is a tool for building UI components

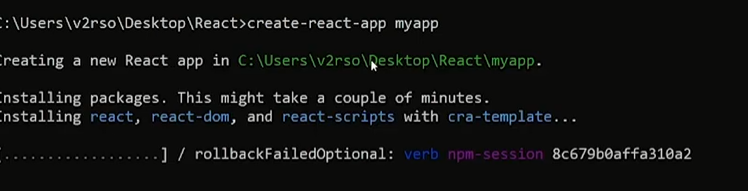
**Why we use ReactJS?**

The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps.









How to run:

Cd project

Npm start

React is a **JavaScript** library created by **Facebook**

React is a **User Interface** (UI) library

React is a tool for building **UI components**

Adding React to an HTML Page

This quickstart tutorial will add React to a page like this:

Example

<!DOCTYPE html>  
<html lang="en">  
<title>Test React</title>  
  
**<!-- Load React API -->**<script src= "https://unpkg.com/react@16/umd/react.production.min.js"></script>  
**<!-- Load React DOM-->**<script src= "https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>  
**<!-- Load Babel Compiler -->**<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>  
  
<body>  
  
<script type="text/babel">  
**//  JSX Babel code goes here**</script>  
  
</body>  
</html>

What is Babel?

Babel is a JavaScript compiler that can translate markup or programming languages into JavaScript.

With Babel, you can use the newest features of JavaScript (ES6 - ECMAScript 2015).

Babel is available for different conversions. React uses Babel to convert JSX into JavaScript.

Please note that <script type="text/babel"> is needed for using Babel.

What is JSX?

JSX stands for **J**ava**S**cript **X**ML.

JSX is an XML/HTML like extension to JavaScript.

Example

const element = <h1>Hello World!</h1>

As you can see above, JSX is not JavaScript nor HTML.

JSX is a XML syntax extension to JavaScript that also comes with the full power of ES6 (ECMAScript 2015).

Just like HTML, JSX tags can have a tag names, attributes, and children. If an attribute is wrapped in curly braces, the value is a JavaScript expression.

Note that JSX does not use quotes around the HTML text string.

React DOM Render

The method ReactDom.render() is used to render (display) HTML elements:

Example

<div id="id01">Hello World!</div>  
  
<script type="text/babel">  
ReactDOM.render(  
    <h1>Hello React!</h1>,  
    document.getElementById('id01'));  
</script>

JSX Expressions

Expressions can be used in JSX by wrapping them in curly **{}** braces.

Example

<div id="id01">Hello World!</div>  
  
<script type="text/babel">  
const **name** = 'John Doe';  
ReactDOM.render(  
    <h1>Hello **{name}**!</h1>,  
    document.getElementById('id01'));  
</script>

React Elements

React applications are usually built around a single **HTML element**.

React developers often call this the **root node** (root element):

<div id="root"></div>

**React elements** look like this:

const element = <h1>Hello React!</h1>

Elements are **rendered** (displayed) with the ReactDOM.render() method:

ReactDOM.render(element, document.getElementById('root'));

React elements are **immutable**. They cannot be changed.

The only way to change a React element is to render a new element every time:

Example

function tick() {  
    const element = (<h1>{new Date().toLocaleTimeString()}</h1>);  
    ReactDOM.render(element, document.getElementById('root'));  
}  
setInterval(tick, 1000);

React Components

React components are JavaScript functions.

This example creates a React **component** named "Welcome":

Example

function Welcome() {  
    return <h1>Hello React!</h1>;  
}  
ReactDOM.render(<Welcome />, document.getElementById('root'));

React can also use ES6 classes to create components.

This example creates a React component named Welcome with a render **method**:

Example

class Welcome extends React.Component {  
    render() { return(<h1>Hello React!</h1>); }  
}  
ReactDOM.render(<Welcome />, document.getElementById('root'));

React Component Properties

This example creates a React **component** named "Welcome" with property arguments:

Example

function Welcome(props) {  
    return <h1>Hello {props.name}!</h1>;  
}  
ReactDOM.render(<Welcome name="John Doe"/>, document.getElementById('root'));

React can also use ES6 classes to create components.

This example also creates a React component named "Welcome" with property arguments:

Example

class Welcome extends React.Component {  
    render() { return(<h1>Hello {this.props.name}</h1>); }  
}  
ReactDOM.render(<Welcome name="John Doe"/>, document.getElementById('root'));

JSX Compiler

The examples on this page compiles JSX in the browser.

For production code, the compilation should be done separately.

Create React Application

Facebook has created a **Create React Application** with everything you need to build a React app.

It is a a development server that uses Webpack to compile React, JSX, and ES6, auto-prefix CSS files.

The Create React App uses ESLint to test and warn about mistakes in the code.

To create a Create React App run the following code on your terminal:

Example

npx create-react-app react-tutorial

Make sure you have Node.js 5.2 or higher. Otherwise you must install npx:

Example

npm i npx

Start one folder up from where you want your application to stay:

Example

C:\Users\myUser>npx create-react-app react-tutorial

Success Result:

npx: installed 63 in 10.359s  
Creating a new React app in C:\Users\myUser\react-tutorial.  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts...  
+ react-dom@16.5.2  
+ react@16.5.2  
+ react-scripts@2.0.4  
added 1732 packages from 664 contributors and audited 31900 packages in 355.501s  
found 0 vulnerabilities+ react@16.5.2  
  
Success! Created react-tutorial at C:\Users\myUser\react-tutorial  
Inside that directory, you can run several commands:  
  
npm start  
Starts the development server.  
  
npm run build  
Bundles the app into static files for production.  
  
npm test  
Starts the test runner.  
  
npm run eject  
Removes this tool and copies build dependencies, configuration files  
and scripts into the app directory. If you do this, you can't go back!  
  
We suggest that you begin by typing:  
  
cd react-tutorial  
npm start

**Components add in React App:**

import React, { Component } from 'react'

/\*

render() {

return (

<div>ControlledComponent</div>

)

}

}

export default ControlledComponent

\*/

class ControlledComponent extends Component {

constructor(props) {

super(props);

this.state = {value: ''};

this.handleChange = this.handleChange.bind(this);

this.handleSubmit = this.handleSubmit.bind(this);

}

handleChange(event) {

this.setState({value: event.target.value});

}

handleSubmit(event) {

alert('You have submitted the input successfully: ' + this.state.value);

//event.preventDefault();

}

render() {

return (

<form onSubmit={this.handleSubmit}>

<h1>Controlled Form Example</h1>

<label>

Name:

<input type="text" value={this.state.value} onChange={this.handleChange} />

</label>

<input type="submit" value="Submit" />

</form>

);

}

}

export default ControlledComponent;

import React, { Component } from "react";

class Person extends Component

{

render()

{

return(

<div>

<div> Name: Adam Adam </div>

<div> Gender: Male </div>

</div>

);

}

}

export default Person;