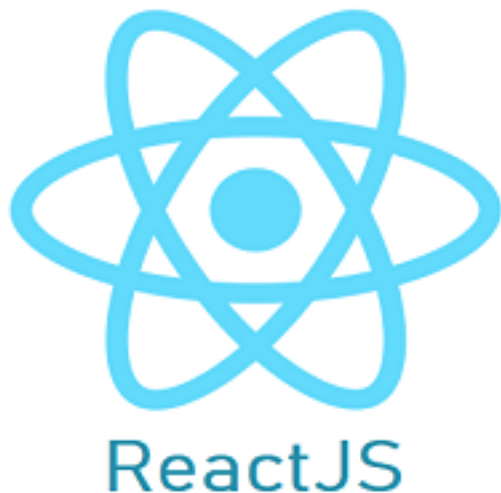


ReactJS Tutorial



ReactJS tutorial provides basic and advanced concepts of ReactJS. Currently, ReactJS is one of the most popular JavaScript front-end libraries which has a strong foundation and a large community.

ReactJS is a **declarative, efficient, and flexible JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

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. It was created by **Jordan Walke**, who was a software engineer at **Facebook**. It was initially developed and maintained by Facebook and was later used in its products like **WhatsApp & Instagram**. Facebook developed ReactJS in **2011** in its newsfeed section, but it was released to the public in the month of **May 2013**.

Today, most of the websites are built using MVC (model view controller) architecture. In MVC architecture, React is the 'V' which

stands for view, whereas the architecture is provided by the Redux or Flux.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

To create React app, we write React components that correspond to various elements. We organize these components inside higher level components which define the application structure. For example, we take a form that consists of many elements like input fields, labels, or buttons. We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself. The form components would specify the structure of the form along with elements inside of it.

React Environment Setup

In this section, we will learn how to set up an environment for the successful development of ReactJS application.

Pre-requisite for ReactJS

- NodeJS and NPM
- React and React DOM
- Webpack
- Babel

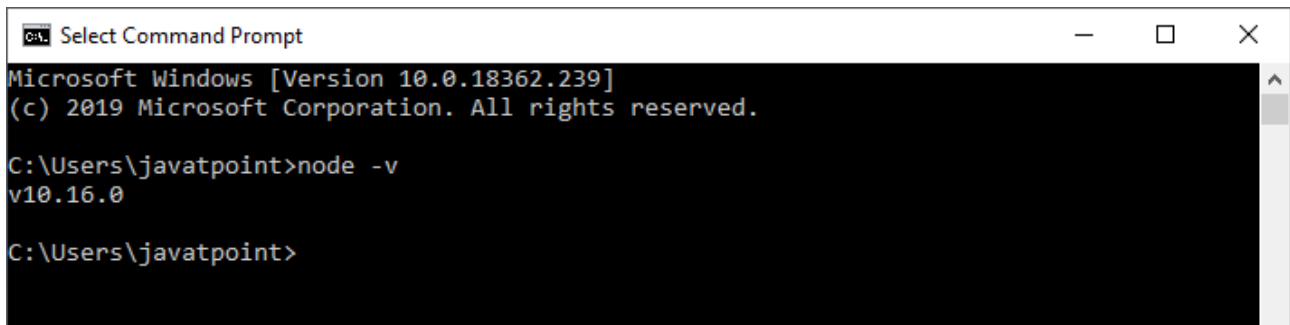
Ways to install ReactJS

There are two ways to set up an environment for successful ReactJS application. They are given below.

- Using the npm command
- Using the create-react-app command

Run the following command to check the Node version in the command prompt.

\$ node -v



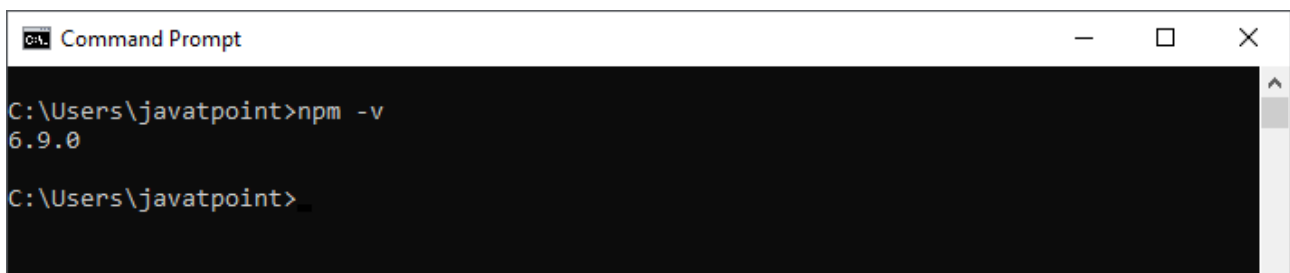
```
Microsoft Windows [Version 10.0.18362.239]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\javatpoint>node -v
v10.16.0

C:\Users\javatpoint>
```

Run the following command to check the NPM version in the command prompt.

\$ npm -v



```
C:\Users\javatpoint>npm -v
6.9.0

C:\Users\javatpoint>
```

Installation

Here, we are going to learn how we can install React using **CRA** tool. For this, we need to follow the steps as given below.

Install React

We can install React using npm package manager by using the following command. There is no need to worry about the complexity of React installation. The create-react-app npm package manager will manage everything, which needed for React project.

```
C:\Users\javatpoint> npm install -g create-react-app
```

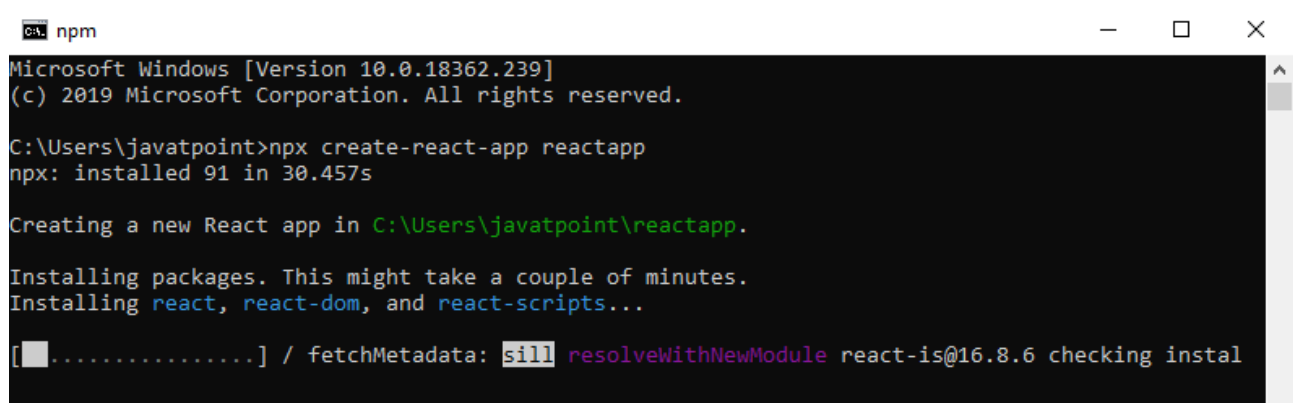
Create a new React project

Once the React installation is successful, we can create a new React project using create-react-app command. Here, I choose "reactproject" name for my project.

```
C:\Users\javatpoint> create-react-app reactproject
```

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

```
C:\Users\javatpoint> npx create-react-app reactproject
```



```
npm
Microsoft Windows [Version 10.0.18362.239]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\javatpoint>npx create-react-app reactapp
npx: installed 91 in 30.457s

Creating a new React app in C:\Users\javatpoint\reactapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

[█.....] / fetchMetadata: sill resolveWithNewModule react-is@16.8.6 checking instal
```

The above command will take some time to install the React and create a new project with the name "reactproject." Now, we can see the terminal as like below.

```
Command Prompt
added 1385 packages from 675 contributors and audited 902050 packages in 305.218s
found 0 vulnerabilities

Success! Created reactapp at C:\Users\javatpoint\reactapp
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 reactapp
  npm start

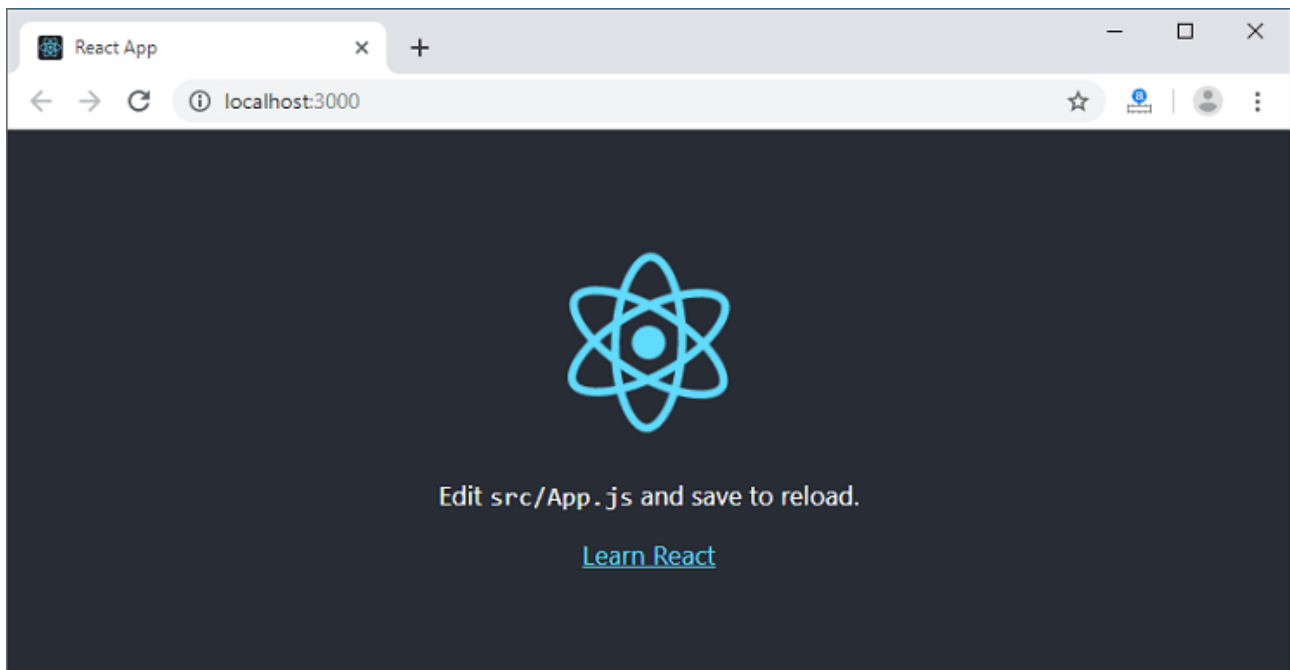
Happy hacking!
C:\Users\javatpoint>
```

The above screen tells that the React project is created successfully on our system. Now, we need to start the server so that we can access the application on the browser. Type the following command in the terminal window.

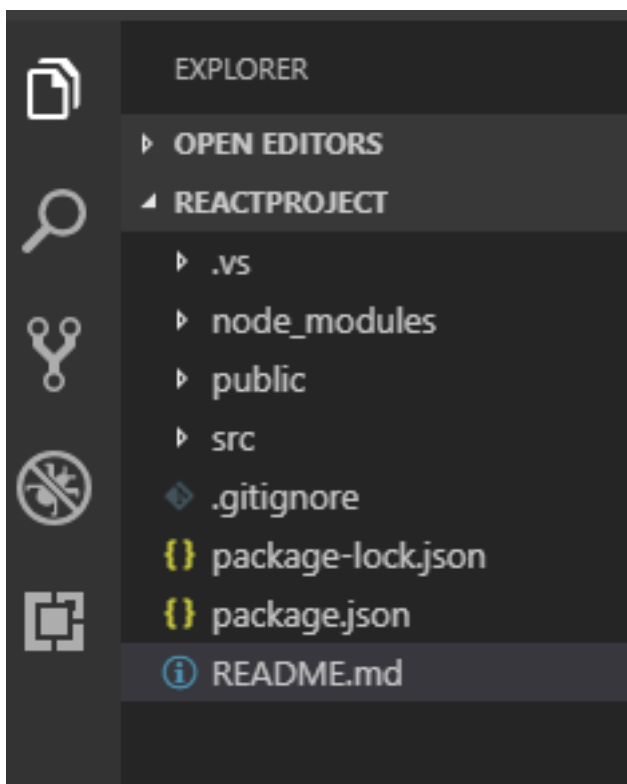
```
$ cd Desktop
```

```
$ npm start
```

NPM is a package manager which starts the server and access the application at default server <http://localhost:3000>. Now, we will get the following screen.



Next, open the project on Code editor. Here, I am using Visual Studio Code. Our project's default structure looks like as below image.



In React application, there are several files and folders in the root directory. Some of them are as follows:

node_modules: It contains the React library and any other third party libraries needed.

public: It holds the public assets of the application. It contains the index.html where React will mount the application by default on the `<div id="root"></div>` element.

src: It contains the App.css, App.js, App.test.js, index.css, index.js, and serviceWorker.js files. Here, the App.js file always responsible for displaying the output screen in React.

package-lock.json: It is generated automatically for any operations where npm package modifies either the node_modules tree or package.json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package.

package.json: It holds various metadata required for the project. It gives information to npm, which allows to identify the project as well as handle the project's dependencies.

README.md: It provides the documentation to read about React topics.

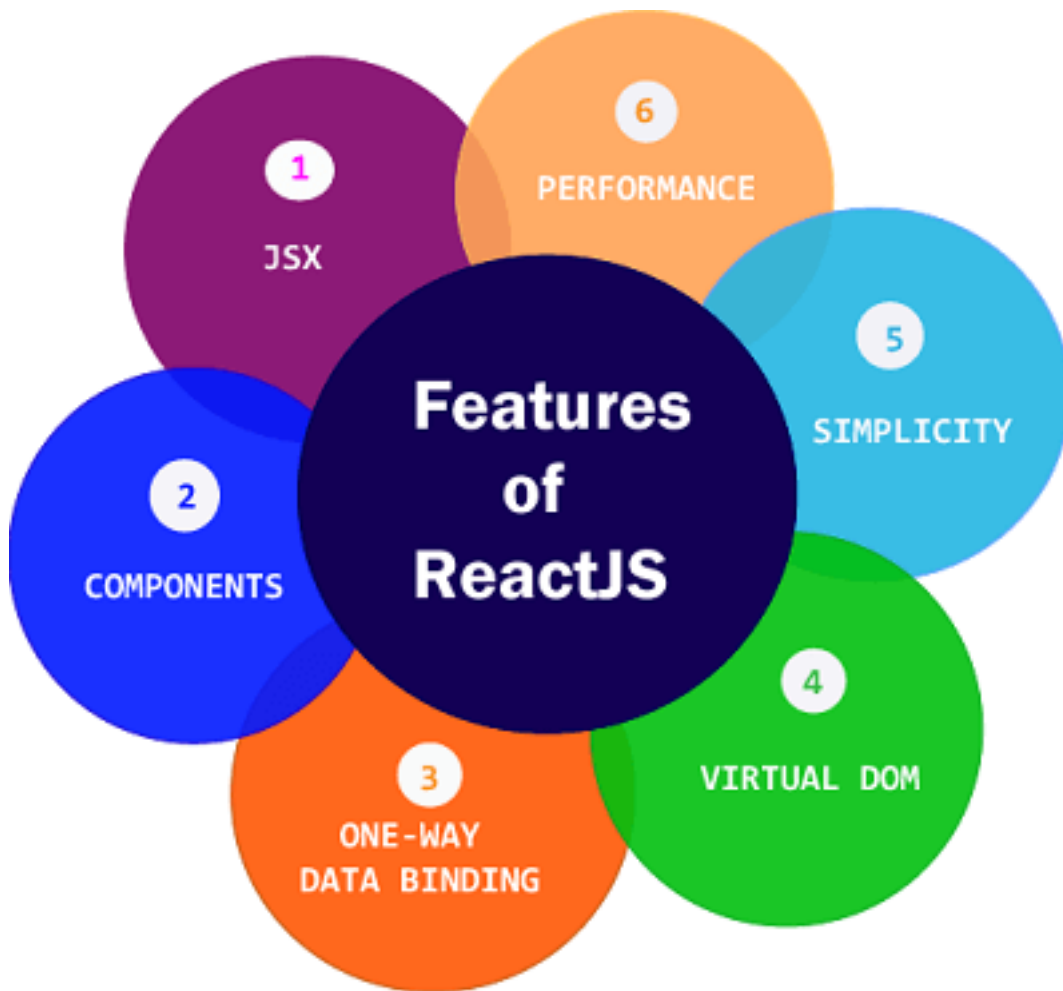
React Environment Setup

Now, open the **src >> App.js** file and make changes which you want to display on the screen. After making desired changes, **save** the file. As soon as we save the file, Webpack recompiles the code, and the page will refresh automatically, and changes are reflected on the browser screen. Now, we can create as many components as we want, import the newly created component inside the **App.js** file and that file will be included in our main **index.html** file after compiling by Webpack.

Next, if we want to make the project for the production mode, type the following command. This command will generate the production build, which is best optimized.

```
$ npm build
```

React Features



Currently, ReactJS gaining quick popularity as the best JavaScript framework among web developers. It is playing an essential role in the front-end ecosystem. The important features of ReactJS are as following.

JSX

Components

One-way Data Binding

Virtual DOM

Simplicity

Performance

JSX

JSX stands for JavaScript XML. It is a JavaScript syntax extension. Its an XML or HTML like syntax used by ReactJS. This syntax is processed into JavaScript calls of React Framework. It extends the ES6 so that HTML like text can co-exist with JavaScript react

code. It is not necessary to use JSX, but it is recommended to use in ReactJS.

Components

ReactJS is all about components. ReactJS application is made up of multiple components, and each component has its own logic and controls. These components can be reusable which help you to maintain the code when working on larger scale projects.

One-way Data Binding

ReactJS is designed in such a manner that follows unidirectional data flow or one-way data binding. The benefits of one-way data binding give you better control throughout the application. If the data flow is in another direction, then it requires additional features. It is because components are supposed to be immutable and the data within them cannot be changed. Flux is a pattern that helps to keep your data unidirectional. This makes the application more flexible that leads to increase efficiency.

Virtual DOM

A virtual DOM object is a representation of the original DOM object. It works like a one-way data binding. Whenever any modifications happen in the web application, the entire UI is re-rendered in virtual DOM representation. Then it checks the difference between the previous DOM representation and new DOM. Once it has done, the real DOM will update only the things that have actually changed. This makes the application faster, and there is no wastage of memory.

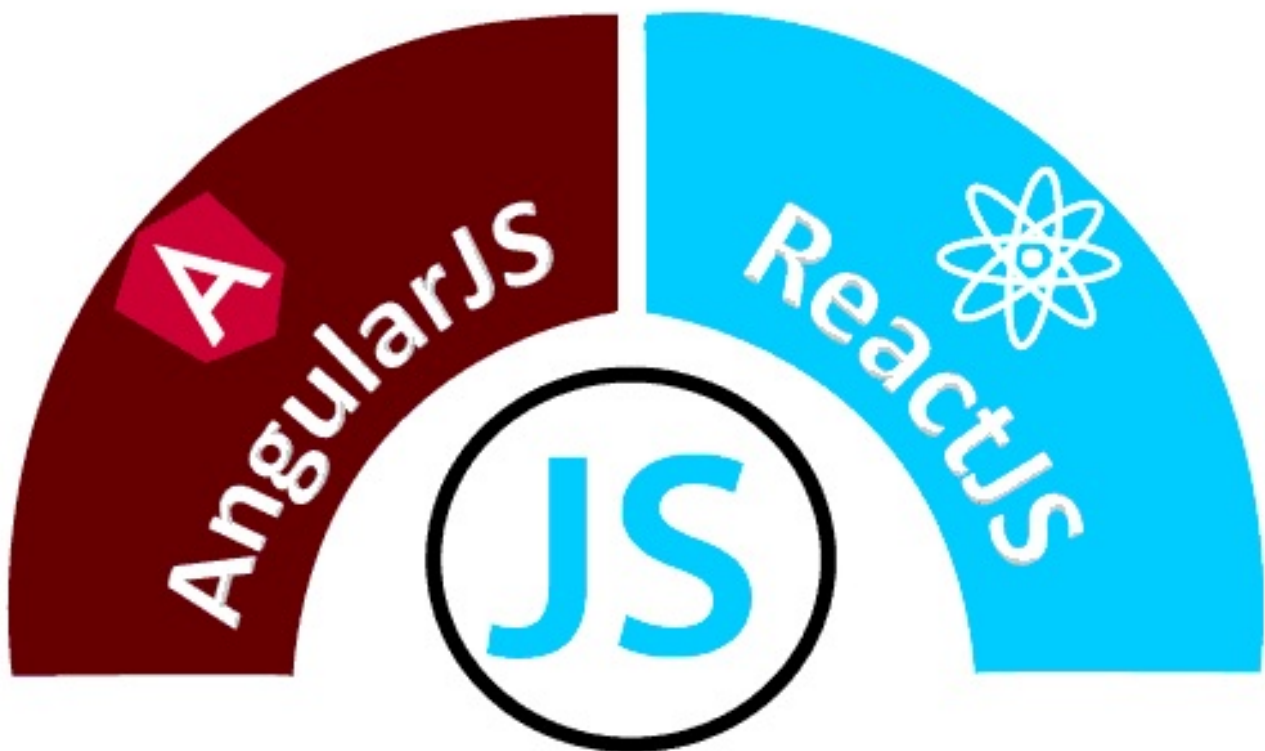
Simplicity

ReactJS uses JSX file which makes the application simple and to code as well as understand. We know that ReactJS is a component-based approach which makes the code reusable as your need. This makes it simple to use and learn.

Performance

ReactJS is known to be a great performer. This feature makes it much better than other frameworks out there today. The reason behind this is that it manages a virtual DOM. The DOM is a cross-platform and programming API which deals with HTML, XML or XHTML. The DOM exists entirely in memory. Due to this, when we create a component, we did not write directly to the DOM. Instead, we are writing virtual components that will turn into the DOM leading to smoother and faster performance.

AngularJS Vs. ReactJS

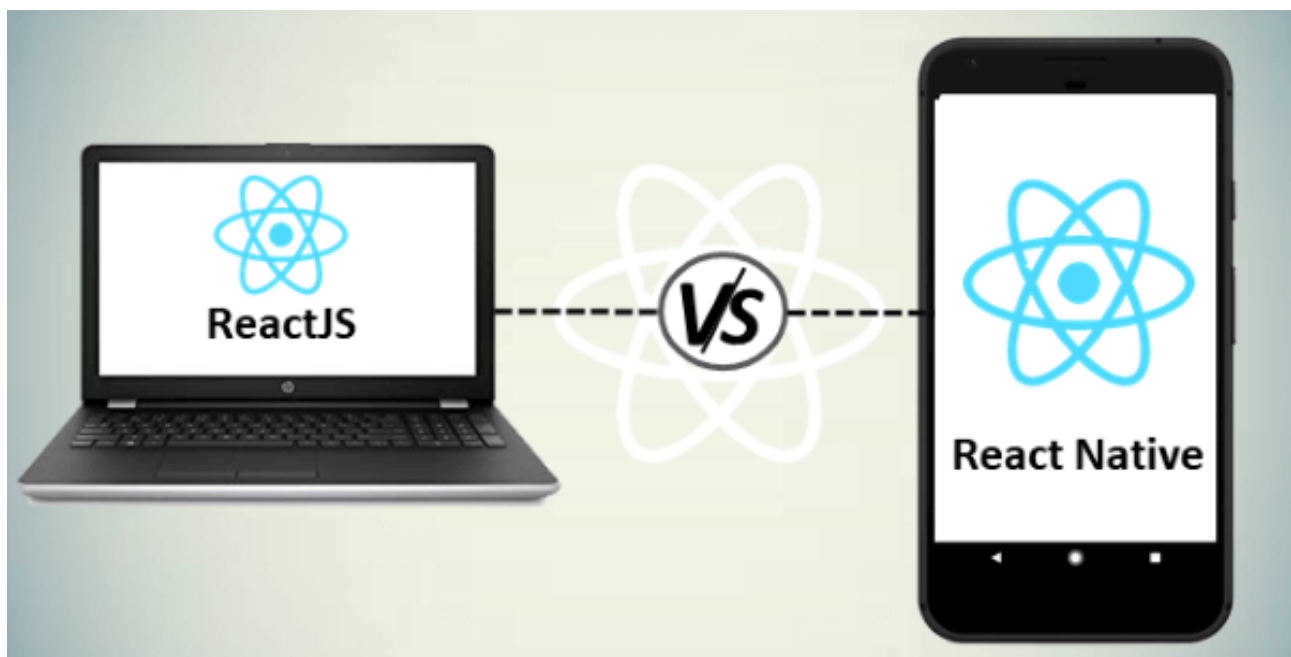


	AngularJS	ReactJS
Author	Google	Facebook Community

Developer	Misko Hevery	Jordan Walke
Initial Release	October 2010	March 2013
Latest Version	Angular 1.7.8 on 11 March 2019.	React 16.8.6 on 27 March 2019
Language	JavaScript, HTML	JSX
Type	Open Source MVC Framework	Open Source JS Framework
Rendering	Client-Side	Server-Side
Packaging	Weak	Strong
Data-Binding	Bi-directional	Uni-directional
DOM	Regular DOM	Virtual DOM
Testing	Unit and Integration Testing	Unit Testing
App Architecture	MVC	Flux

Dependencies	It manages dependencies automatically.	It requires additional tools to manage dependencies.
Routing	It requires a template or controller to its router configuration, which has to be managed manually.	It doesn't handle routing but has a lot of modules for routing, eg., react-router.
Performance	Slow	Fast, due to virtual DOM.
Best For	It is best for single page applications that update a single view at a time.	It is best for single page applications that update multiple views at a time.

ReactJS Vs React Native



S N	ReactJS	React Native
1	The ReactJS initial release was in 2013.	The React Native initial release was in 2015.

2	It is used for developing web applications.	It is used for developing mobile applications.
3	It can be executed on all platforms.	It is not platform independent. It takes more effort to be executed on all platforms.
4	It uses a JavaScript library and CSS for animations.	It comes with built-in animation libraries.
5	It uses React-router for navigating web pages.	It has built-in Navigator library for navigating mobile applications.
6	It uses HTML tags.	It does not use HTML tags.
7	It can use code components, which saves a lot of valuable time.	It can reuse React Native UI components & modules which allow hybrid apps to render natively.
8	It provides high security.	It provides low security in comparison to ReactJS.
9	In this, the Virtual DOM renders the browser code.	In this, Native uses its API to render code for mobile applications.

Why use JSX?

It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.

Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both. We will learn components in a further section.

It is type-safe, and most of the errors can be found at compilation time.

It makes easier to create templates.

Nested Elements in JSX

To use more than one element, you need to wrap it with one container element. Here, we use **div** as a container element which has **three** nested elements inside it.

App.JSX

```
import React, { Component } from 'react';
class App extends Component{
  render(){
    return(
      <div>
        <h1>JavaTpoint</h1>
        <h2>Training Institutes</h2>
        <p>This website contains the best CS tutorials.</p>
      </div>
    );
  }
}
export default App;
```

Output:



JSX Attributes

JSX use attributes with the HTML elements same as regular HTML. JSX uses **camelcase** naming convention for attributes rather than standard naming convention of HTML such as a class in HTML becomes **className** in JSX because the class is the reserved keyword in JavaScript. We can also use our own custom attributes in JSX. For custom attributes, we need to use **data- prefix**. In the below example, we have used a custom attribute **data-demoAttribute** as an attribute for the **<p>** tag.

Example

```

import React, { Component } from 'react';
class App extends Component{
  render(){
    return(
      <div>
        <h1>JavaTpoint</h1>
        <h2>Training Institutes</h2>
        <p data-demoAttribute = "demo">This website contains the best CS tutorials.</p>
      </div>
    );
  }
}
export default App;

```

In JSX, we can specify attribute values in two ways:

As String Literals: We can specify the values of attributes in double quotes:

```

var element = <h2 className = "firstAttribute">Hello JavaTpoint</h2>;

```

Example

```

import React, { Component } from 'react';
class App extends Component{
  render(){
    return(
      <div>
        <h1 className = "hello">JavaTpoint</h1>
        <p data-demoAttribute = "demo">This website contains the best CS tutorials.</p>
      </div>
    );
  }
}
export default App;

```

Output:

JavaTpoint

This website contains the best CS tutorials.

2. As Expressions: We can specify the values of attributes as expressions using curly braces {}: - Interpolation

```
var element = <h2 className = {varName}>Hello JavaTpoint</h2>;
```

Example

```
import React, { Component } from 'react';
class App extends Component{
  render(){
    return(
      <div>
        <h1 className = "hello" >{25+20}</h1>
      </div>
    );
  }
}
export default App;
```

JSX Comments

JSX allows us to use comments that begin with /* and ends with */ and wrapping them in curly braces {} just like in the case of JSX expressions. Below example shows how to use comments in JSX.

Example

```
import React, { Component } from 'react';
class App extends Component{
  render(){
    return(
      <div>
```



```

    <h1 className = "hello" >Hello JavaTpoint</h1>
    {/* This is a comment in JSX */}
  </div>
);
}
}
export default App;

```

JSX Styling

React always recommends to use **inline** styles. To set inline styles, you need to use **camelCase** syntax. React automatically allows appending **px** after the number value on specific elements. The following example shows how to use styling in the element.

Example

```

import React, { Component } from 'react';
class App extends Component{
  render(){
    var myStyle = {
      fontSize: 80,
      fontFamily: 'Courier',
      color: '#003300'
    }
    return (
      <div>
        <h1 style = {myStyle}>www.javatpoint.com</h1>
      </div>
    );
  }
}
export default App;

```

Example

```

import React, { Component } from 'react';
class App extends Component{
  render(){

```

```
var i = 5;  
return (  
  <div>  
    <h1>{i == 1 ? 'True!' : 'False!'}</h1>  
  </div>  
)  
}  
}  
export default App;
```

Output:

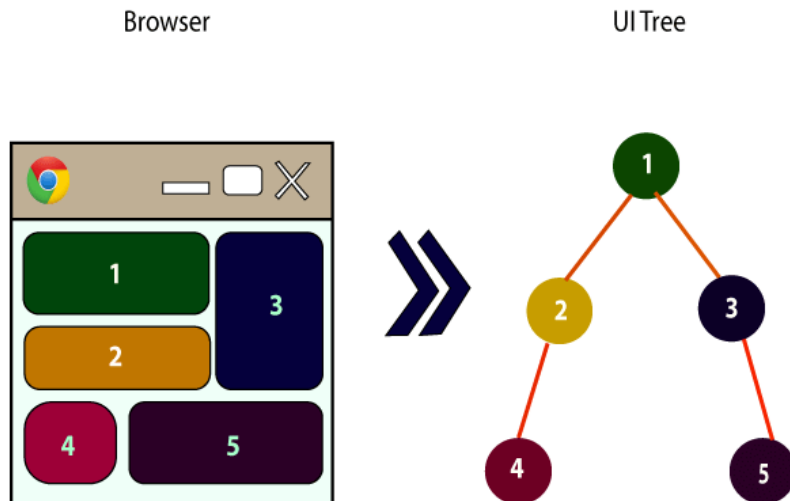
False!

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

Functional Components
Class Components

Functional Components

In React, function components are a way to write components that only contain a render method and don't have their own state. They are simply JavaScript functions that may or may not receive data as parameters. We can create a function that takes props(properties) as input and returns what should be rendered. A valid functional component can be shown in the below example.

```
function WelcomeMessage(props) {  
  return <h1>Welcome to the , {props.name}</h1>;  
}
```

The functional component is also known as a stateless component because they do not hold or manage state. It can be explained in the below example.

Example

```
import React, { Component } from 'react';
class App extends React.Component {
  render() {
    return (
      <div>
        <First/>
        <Second/>
      </div>
    );
  }
}
class First extends React.Component {
  render() {
    return (
      <div>
        <h1>JavaTpoint</h1>
      </div>
    );
  }
}
class Second extends React.Component {
  render() {
    return (
      <div>
        <h2>www.javatpoint.com</h2>
        <p>This websites contains the great CS tutorial.</p>
      </div>
    );
  }
}
export default App;
```

Output:

