ReactJS Introduction

- React is a front-end JavaScript library developed by Facebook used for building user interfaces.
- React allows us to create reusable UI components.
- In a Model View Controller (MVC) architecture, React is the 'view' which is responsible for how the app looks and feel.
- React allows us to design and develop complex UIs from very small and isolated pieces of code referred to as "components".
- It is an open-source, component-based front-end library.
- Popular websites like Netflix, Airbnb, Yahoo!Mail, KhanAcademy, Dropbox and many more use React to build their UI.
- React apps have file extensions .js / .jsx / .ts / .tsx.
- ReactJS was developed by Jordan walke and was released in 2013.

Virtual Dom Vs Real Dom

Virtual Dom

- It updates faster.
- DOM manipulation is very easy.
- No memory wastage.
- It is only a virtual representation of the DOM.

Real Dom

- It updates slowly.
- DOM manipulation is very expensive.
- Too much memory wastage.
- It represents the UI of our application.

Features of ReactJS

Below there are some interesting features of ReactJS :-

- Easy creation of dynamic web applications.
- Reusable Components.
- Unidirectional Data Flow.
- Virtual DOM.
- Good Performance.
- Small Learning Curve.
- JSX.
- Dedicated tools for easy debugging.
- Server-Side rendering.

Advantages of ReactJS

There are some interesting benefits of using ReactJS includes :-

- Virtual DOM.
- Free & Open-source library.
- SEO Friendly.
- Reusable Components.
- Uses JSX.
- Fast Performance.

ReactJS Vs AngularJS

ReactJS

- ReactJS was developed by Facebook.
- React is just a library and works only the view layer of MVC.
- ReactJS uses virtual DOM.
- Unidirectional(one-way) data flow.
- Uses JavaScript for app development.
- Simple learning curve.
- React was officially released in 2013.

AngularJS

- Angular was developed by google.
- Angular is a fully fledged MVC framework.
- Angular uses real DOM.
- Bidirectional(two-way) data flow.
- Uses typescript for app development.
- Complex learning curve.
- Angular was officially released in 2010.

ReactJS Vs React Native

ReactJS

- It is used for developing web applications.
- React is a JavaScript library used for building user interfaces of web apps.
- ReactJS uses virtual DOM to refresh a particular section of a page.
- Uses HTML for app development.
- React.js use virtual DOM for rendering.
- React.js uses React-router for navigation.
- A developer can easily incorporate animations & gestures with the help of CSS in ReactJS.

React Native

- It is used for developing mobile applications.
- React Native is a JavaScript framework especially used for mobile app development.
- React Native uses Native API to render browser code on mobile.
- React Native not uses HTML in app development.
- React Native uses native APIs for rendering the components on mobile.
- React Native uses third-party library like react-navigation for navigating screens.
- React Native uses APIs, i.e., Animated & Responder, to create all animations & gestures, respectively.

ReactJS Prerequisites

The required things to learn ReactJS are :-

- Basic knowledge of HTML, CSS, and JavaScript.
- Basic understanding of ES6 features.
- Basic understanding of JSX, Babel & webpack.
- Basic understanding of how to use npm.
- Basic understanding of Git & CLI.

So, for creating React-App, we need to have node install on our local machine

https://nodejs.org/en

download the node installer (LTS) and install nodeJS

Companies Uses ReactJS

ReactJS is used by top companies for their applications that includes :-

- Facebook
- Instagram
- Twitter
- Netflix
- WhatsApp
- Dropbox
- Reddit

ReactJS Working

- The Document Object Model (DOM) is a programming interface for HTML and XML (Extensible markup language) documents. It defines the logical structure of documents and the way a document is accessed and manipulated. When a webpage is loaded, the browser creates DOM of the page. It represents the documents as nodes or objects.
- DOM creation and update are a slower process. To speed up this process, Facebook developed ReactJS.
- ReactJS uses the concept of virtual DOM which makes the loading faster. A virtual DOM is just a lightweight JavaScript object which is just the copy of a real DOM.

Virtual DOM works in the following 3 steps :-

a) In ReactJS, whenever any underlying part of the web page changes, the entire UI is re-rendered in Virtual DOM representation.
b) Then the difference between the previous DOM (real DOM) and the new DOM (virtual DOM) is evaluated using the diffing algorithm.
c) After the final calculations, the real DOM will only be updated with those elements which have actually changed.

ReactJS Environmental Setup

There are two procedure by which we can configure ReactJS in our local machine :-

a) Using ReactJS from CDN links
b) Using NPM Packages

1) Using ReactJS from CDN links

Go to the official site of ReactJS to get the CDN links,
https://reactjs.org/docs/cdn-links.html

1.1) ReactJS Development Version

```
<script crossorigin
src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js"></script>
```

1.2) ReactJS Production Version

```
<script crossorigin
src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-
dom.production.min.js"></script>
```

1.3) Babel Link

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

1.4) Overall Code (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
```

```
<body>

  <div id="app"></div>

  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  <script crossorigin
src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js"></script>

  <script type="text/babel">
     class App extends React.Component{
       render(){
         return(
           <div>
             <h1>Hello React</h1>
           </div>
         )
       }
     }

     ReactDOM.render(<App/>,document.getElementById('app'))
  </script>
</body>
</html>
```

2) Using NPM Packages (Installing ReactJS Via "npx")

- Create a project folder for ReactJS and open that folder in terminal / CMD.
- When we create ReactJS project this way we can automatically get "webpack & babel" configuration along with our project.
- To install npx, you need to use 'npm install -g npx' command.

Commands to Install ReactJS

For executing ReactJS project directly
"npx create-react-app appname / yarn create-react-app appname"

For changing app directory
"cd appname"

For running ReactJS project
"npm start / yarn start"

Components In ReactJS

- A component is a small reusable chunk of code. Components are the core building block of React apps.
- Components lets you split the UI into independent, reusable pieces & think about each piece in isolation.
- When creating a React component, the component's name must start with an upper-case letter.
- React treats components starting with lowercase letters as DOM tags. For example, represents an HTML div tag, but <App/> represents component requires to be in scope.

Types of Components in ReactJS

There are two types of components in ReactJS :-

a) Functional Components or stateless Components.
b) Class Components or statefull Components.

a) Functional Components

- A functional component represents the simplest form of a ReactJS component.
- We can create a functional component in React by writing a JavaScript function.
- There is no render method used in functional components.
- These can be typically defined using arrow functions but can also be created with the regular function keyword.

Syntax

```
import React from "react";
function App() {
  return (
    <h1>Hello World</h1>
  );
}
export default App;
```

or,

```
import React from "react";

const App = () => {
  return (
    <h1>Hello World again n again</h1>
  );
}

export default App;
```

b) Class Components

- We can use JavaScript ES6 classes to create class based components in React which extends React.Component class.
- It must have render() method returning html.
- These components are depreciated and not used nowadays.

Syntax

```
import React, { Component } from "react";

class App extends Component {
  render() {
    return (
      <h1>Hello World</h1>
    );
  }
}

export default App;
```

Functional Components Vs Class Components

Functional Components

- Functional components are easier to write and faster in performance than that of class components.
- We cannot manage states with functional components.
- There is no render method used in functional components.
- We cannot use lifecycle methods with functional components.
- Functional Components are also called "Stateless Components".

b) Class Components

- Class Components are also called "Statefull Components".
- Class based components are complex and slower in performance than that of functional components.
- We cannot manage states with class components.
- It must have render() method returning App Layout / UI.
- We can use lifecycle methods with class-based components.

Benefits of Using Functional Components

We have following benefits of using functional components in our React app :-

- Less Code.
- Simplicity.
- Simpler to test.
- No 'this' binding.
- Stateless.
- Easier to extract small components.

Understanding JSX

- JSX is acronym for JavaScript eXtension / JavaScript Syntax in XML.
- JSX is basically a syntactic sugar for the React.createElement method, where JSX code is finally transpiled into pure JavaScript function calls with React.createElement.
- JSX allows us to write HTML in React also JSX makes it easier to write and add HTML in React.
- JSX converts HTML tags into react elements.

Benefits of Using JSX

- JSX is faster than standard JavaScript, because it optimize the source code while compiling it.
- JSX code is much more readable than the plain JavaScript.
- JSX is easy to learn & write in comparison to JavaScript.
- It is type-safe, and most of the errors can be found at compilation time.

JSX Syntax Vs JavaScript Syntax

JSX Code
Hello ReactJS

JavaScript Output ---->
React.createElement("div", null, "Hello ReactJS");

Comments, Line Breaks and Spacing in ReactJS

Comments in ReactJS

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.

Example – outside JSX

```
import React from "react";

const App = () => {

// Single Line Comment

  /*
    Multiline Comments
  */

  return(
    <div>
      <h1>Hello World</h1>
    </div>
  )
}

export default App;
```

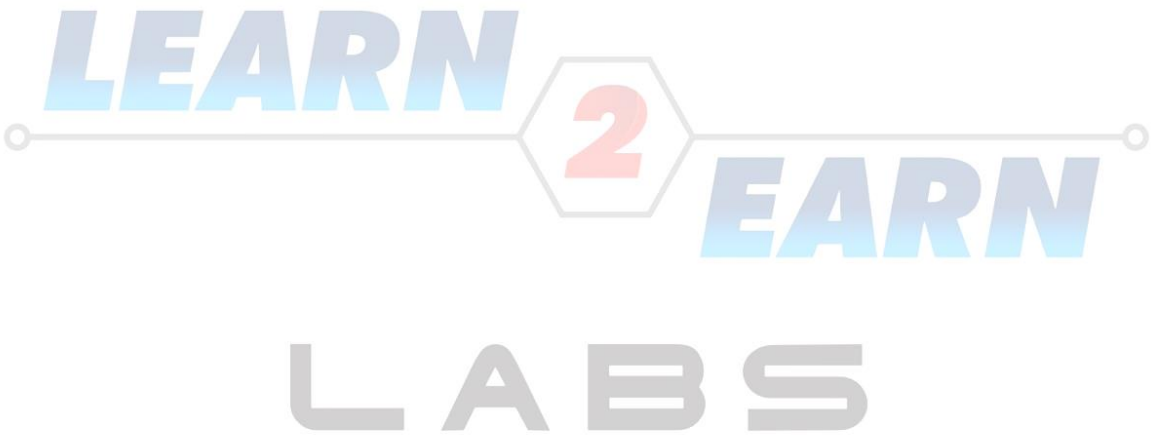
Example – inside JSX

```
import React from "react";

const App = () => {

 return (
            <div>
              <h1>Hello World</h1>
              {/* Comments in ReactJS */}
            </div>
 );
}

export default App;
```



Line Breaks in ReactJS

- When creating components in React Native there are situations where newlines are required inside the render function.
-
 creates a new line in ReactJS.

Example

```
import React from 'react'

const App = () => {
 return (
   <div>
        <p>Hello <br /> ReactJS App</p>
 </div>
 )
}

export default App;
```

Spacing in ReactJS

For spacing we can use html entities like " " / " " / " ".

Example

```
import React from 'react'

const App = () => {
  return (
    <div>
      <p>Hello  ReactJS App</p>
    </div>
  )
}

export default App;
```

Fragments

- A React Fragments let us group a list of children without adding extra nodes to the DOM.
- React Fragment helps in returning multiple elements. The other alternative is to use a html element like div to wrap them.
- React Fragments were introduced in React 16.2.0.

Simple Example

```
import React from 'react'

const App = () => {
  return (
    <React.Fragment>
                <h1>Hello Heading</h1>
                <p>Hello Para</p>
</React.Fragment>

  )
}

export default App;
```

Shorthand Example

```
import React from 'react'

const App = () => {
  return (
    <>
                <h1>Hello Heading</h1>
                <p>Hello Para12</p>
</>
  )
}

export default App;
```

Example : Keyed Fragments

```
import React from 'react'
const App = () => {
  var item={
    term:”chair”,
    description:"this is the unique chair"
  };
  return (
        <React.Fragment key={item.id}>
    <h1>{item.term}</h1>
    <p>{item.description}</p>
</React.Fragment>
  )
}
export default App;
```