Introduction to React

What is ReactJS?

- "React is a **JavaScript library** for building **User Interfaces**" – **reactjs.org**
- React is a **client-side framework** for building **single-page client applications** using **HTML and JavaScript + JSX (Java Script Xml).**

React = Html + JS + JSX

- Created by **Jordan Walke** in **May 29, 2013.**
- It is maintained by **Facebook**.
- React can be integrated with any **server-side technologies** such as **Java, NodeJS, Asp.Net, PHP, Python**
- React is very popular with **MERN** Stack development.
- React is the mostly-used as a client side framework compare with other like **Angular, Vue, etc**.
- Current version of React is **18.2.0 (June 14, 2022)**

Organizations using ReactJS:

Features:

1. **Open-source**: source code of reactjs is available online for free of cost **(commercially too).**

2. **Declarative:**
   React makes it painless to create interactive UIs.
   Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

   Declarative views make your code more predictable and easier to **debug**.

3. **Component-Based:**
   Build encapsulated components that manage their own state, then compose them to make complex UIs.

   Since component logic is written in **JavaScript** instead of **templates**, you can easily pass rich data through your app and keep state out of the DOM.

4. **Flexible:**
   Learn Once, Write Anywhere.
   We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

5. **Supports Native Environment**:
   React can also render on the server using Node and power mobile apps using React Native.

6. **Huge Community**.

Working with Hello World Application

1. Download and install the Nodejs (If not done)

2. Install **create-react-app.**
   Open CMD and run the following command.
   >npm install -g create-react-app

   Note
   create-react-app package includes the global command for Create React App.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\asdas>npm install -g create-react-app
C:\Users\asdas\AppData\Roaming\npm\create-react-app -> C:\Users\asdas\AppData\Roaming\npm\node_modules\create-react-app\in
dex.js
+ create-react-app@4.0.1
added 6 packages from 4 contributors, removed 37 packages and updated 12 packages in 31.655s

C:\Users\asdas>create-react-app --version
4.0.1
```

3. Create workspace as **ReactJS** folder
   C:\Users\username>d:
   D:\>mkdir ReactJS
   D:\>cd ReactJS

4. Create **hello-app**
   D:\ReactJS>npx create-react-app hello-app

   Note: It takes several minutes to create **hello-app** project.

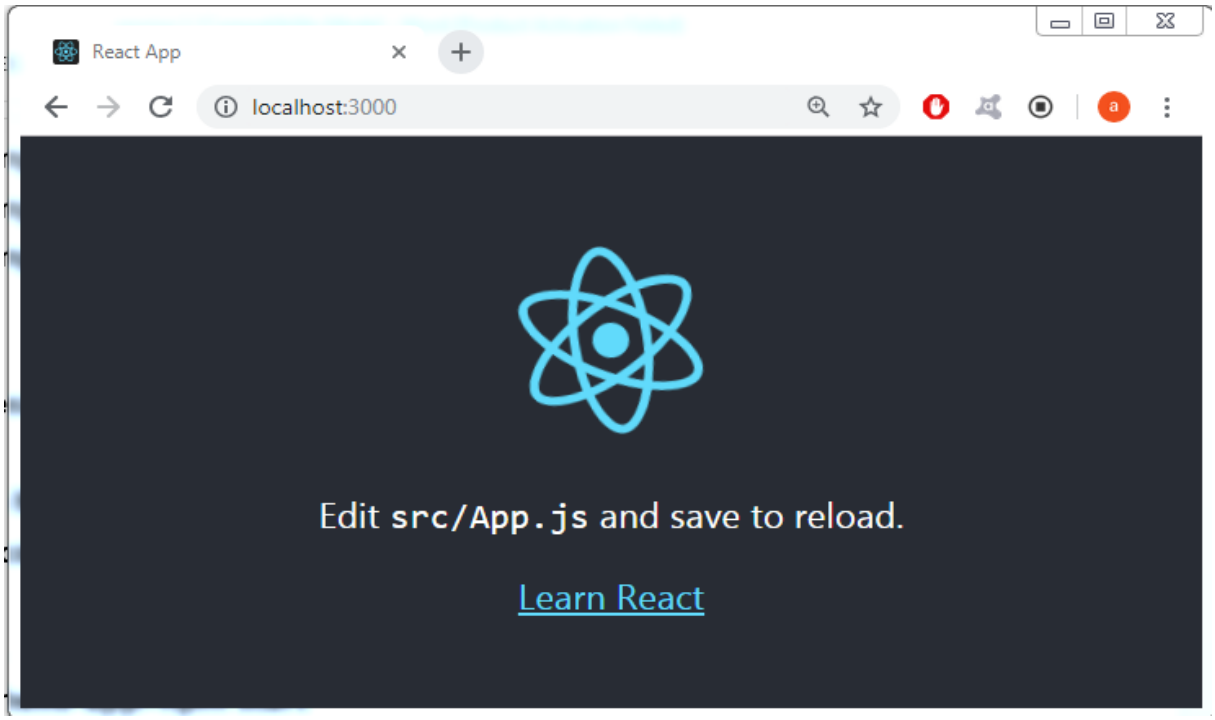
   npm is a CLI tool to install node packages.
   npx is a CLI tool to execute node packages and it comes with npm version 5.2+

5. Move into the **hello-app**
   D:\ReactJS>cd hello-app

6. Start server.
   D:\ReactJS\hello-app>npm start

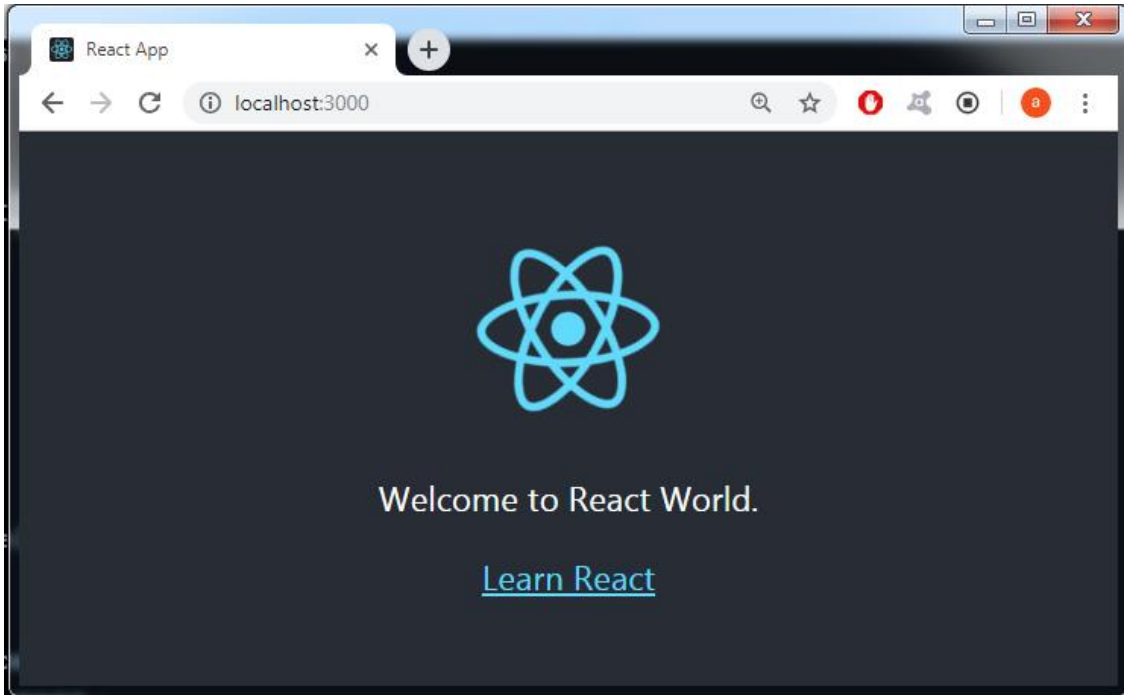
```
Windows PowerShell
D:\ReactJS>cd hello-app

D:\ReactJS\hello-app>npm start

> hello-app@0.1.0 start D:\ReactJS\hello-app
> react-scripts start

i ℹ wds ℹ: Project is running at http://0.0.0.0:3000/
i ℹ wds ℹ: webpack output is served from
i ℹ wds ℹ: Content not from webpack is served from D:\ReactJS\hello-app\public
i ℹ wds ℹ: 404s will fallback to /
Starting the development server...
Compiled successfully!

You can now view hello-app in the browser.

  http://localhost:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

7. Open the browser and access the following url.
   http://localhost:3000/



8. Edit **src/App.js file** and just save it.

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <p>
            Welcome to React World.
          </p>
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React
          </a>
        </header>
      </div>
    );
  }
}
export default App;
```



Working with Elements

- **Elements** are the smallest building blocks of React apps.
- An element specifies what should be there in our UI.
- An Element is a plain object describing what we want to render in terms of the DOM nodes.
- An Element can be created 2 ways.
    1. React using JSX
    2. React without JSX

Creating first React Element using JSX:

JSX:

- **JSX** stands for **J**ava **S**cript **X**ml.
- It allows to write Html in React
- JSX is a React extension that allows us to write JavaScript that looks like **HTML**.
- JSX would be compiled into Plain JavaScript using **Babel**.
  (Babel is JavaScript compiler that converts ES2015+ code into backwards compatible version of JavaScript in current and older browsers)
- JSX is not a requirement for using React.

index.js

```
import React from 'react'
import ReactDOM from 'react-dom'

//<h1> element
const element = <h1>Welcome to React World!</h1>;

//Render <h1>element at root container
ReactDOM.render(element, document.getElementById('root'))
```

ReactDOM.render(element, container[, callback]) controls the contents of the container node you pass in. Any existing DOM elements inside are replaced when first called.

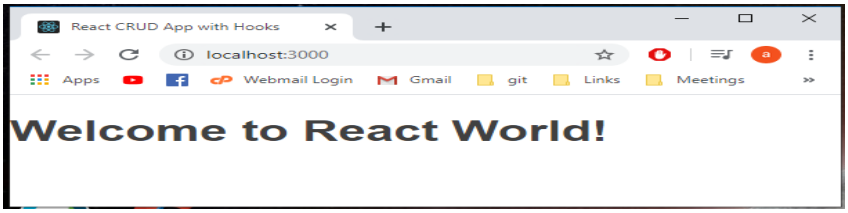
We have created an Object of type **h1** and assigned it to a variable called as **element**.

This element should be rendered into the Browser DOM, and for that we need a container.

In **index.html**, there is a ** ** element with id as root and we will use that div as our container.

```
<!DOCTYPE html>
<html lang="en">
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
  </body>
</html>
```

>npm start



To add css styles to this element:

Create a class with classname as **customClass** in **index.css**.

hello-app\src\index.css

```
body {
    margin: 0;
    padding: 0;
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
      "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
      sans-serif;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }

  code {
    font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
      monospace;
  }

  .customClass{
      color: brown;
      background-color: silver;
  }
```

Import index.css and apply **customClass** to the element using **className** attribute in index.js

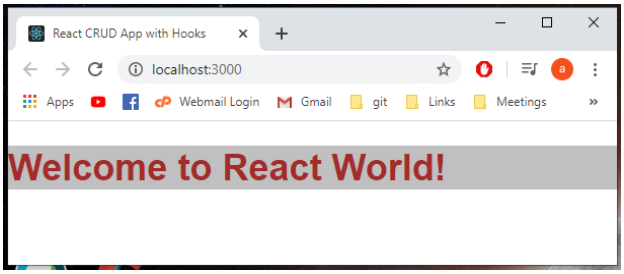
hello-app\src\index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

//<h1> element
const element = <h1 className="customClass">Welcome to React World!</h1>;

//Render <h1>element at root container
ReactDOM.render(element, document.getElementById('root'))
```

Note:

An Element contains **type and properties**,

here, **h1** is the **type** and **className** as **property**

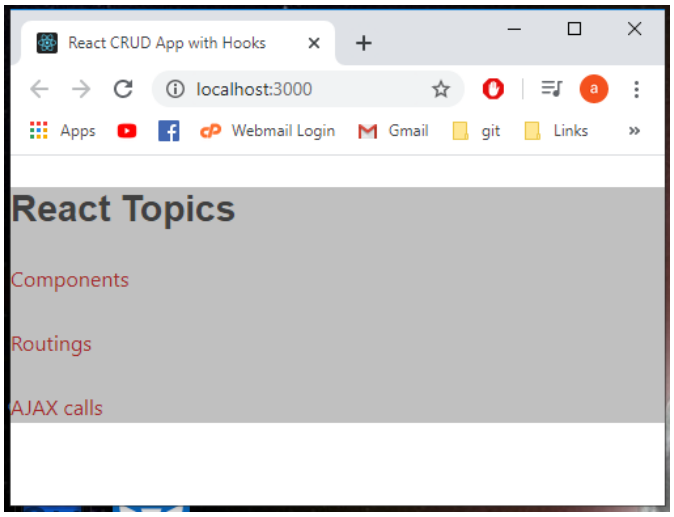


Creating a group or nested Elements:

hello-app\src\index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'


const element = (
    <div className="customClass">
        <h1>React Topics</h1>
        <p>Components</p>
        <p>Routings</p>
        <p>AJAX calls</p>
    </div>
);

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

Output:



React without JSX:

Using React without JSX is especially convenient when you don't want to set up compilation in our build environment.

Eg: Creating a Single Element:

hello-app\src\index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

const element = React.createElement("h1",{className:"customClass"},"Welcome to React World!");

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

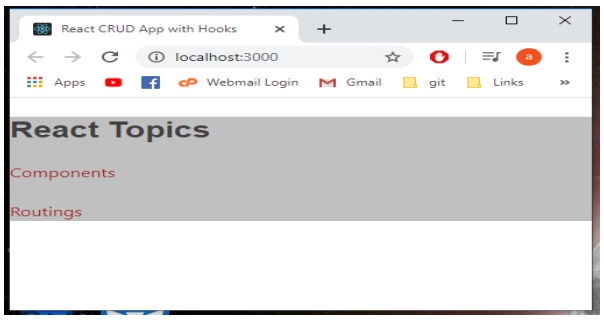
React.createElement(type, prop, children);

- This method takes few parameters.
- First one is the type of Element we want to create like h1 or h2 or div.
- Second parameter specifies the Properties we want to apply to this element.
- Third one represents the Child Element or Elements that has be placed inside the Parent one.

Eg: Creating a group or nested elements:

hello-app\src\index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

const element = React.createElement(
    "div", { className: "customClass" },
    React.createElement("h1", null, 'React Topics'),
    React.createElement("p", null, 'Components'),
    React.createElement("p", null, 'Routings'),
);

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



Working with Components

- Components are the building blocks of React app.
- Components are used to split the UI into independent and reusable pieces.

  Component = Template (HTML) + User Interactivity (Events) + Appearance (CSS)

- Each component represents certain section of the web page.
  For example,
  "login form" is represented as a "Login Component".
  "register form" is represented as a "Register Component".
  "header section" is represented as "Header Component".
  "footer section" is represented as "Footer Component".

- Conceptually a component is a JavaScript class or function that accepts inputs which are properties (props) and returns a React element that describes how a section of the User Interface should appear.

- A component can be created 2 ways:
  1. Function Component
  2. Class Component.

Function Components:

- These are also called as **Stateless components**.
- In React, function components are a way to write components that only contain a render() method.
- It must **import React from 'react'**;

Eg:
hello-app\src\index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

function DisplayEmployeeInfo (employee) {
    return <div>
        <p>Id : {employee.id}</p>
        <p>Name : {employee.name}</p>
        <p>Salary : {employee.salary}</p>
    </div>;
}

const element = <DisplayEmployeeInfo id="E101" name="Jones" salary="5500" />;

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

Note:
ComponentName always start with Capital letter only.
React treats components starting with lowercase letters as DOM tags.

Creating functional component using lambda expression:

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

let DisplayEmployeeInfo = (employee) => {
    return <div>
        <p>Id : {employee.id}</p>
        <p>Name : {employee.name}</p>
        <p>Salary : {employee.salary}</p>
    </div>;
}

const element = <DisplayEmployeeInfo id="E101" name="Jones" salary="5500" />;

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

Composing Components:

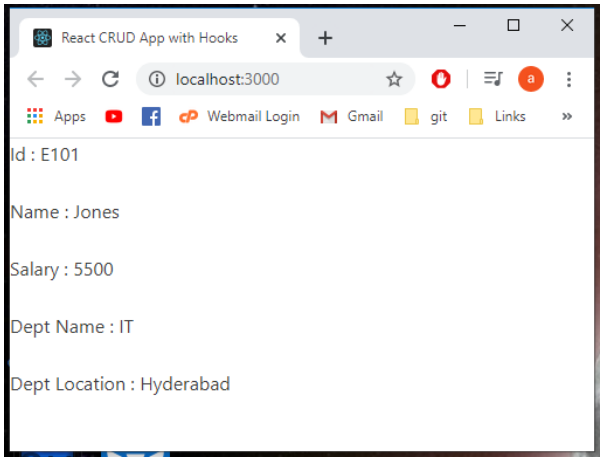
- Components can refer to other components in their output.

Example:

Along with the Employee Details, we would like to Display Employee Department information as well.
One way is to write the Code to display department information in the Employee Component.
But it is not a good programing practice to keep everything in one component.
To promote Code reusability, we will split them into different components.
Department Component which will display only Department Information and this Component can be used by any other component.

hello-app\src\index.js

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

let Department=(deptInfo)=>{
    return <div>
        <p>Dept Name : {deptInfo.dept}</p>
        <p>Dept Location : {deptInfo.location}</p>
    </div>;
  }

let DisplayEmployeeInfo = (employee) => {
    return <div>
        <p>Id : {employee.id}</p>
        <p>Name : {employee.name}</p>
        <p>Salary : {employee.salary}</p>
        <Department dept={employee.dept} location={employee.location}/>
    </div>;
}


const element = <DisplayEmployeeInfo id="E101" name="Jones" salary="5500" dept="IT" locati
on="Hyderabad" />;

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


React CRUD App with Hooks
localhost:3000
Apps
Webmail Login
Gmail
git
Links
Id : E101
Name : Jones
Salary : 5500
Dept Name : IT
Dept Location : Hyderabad