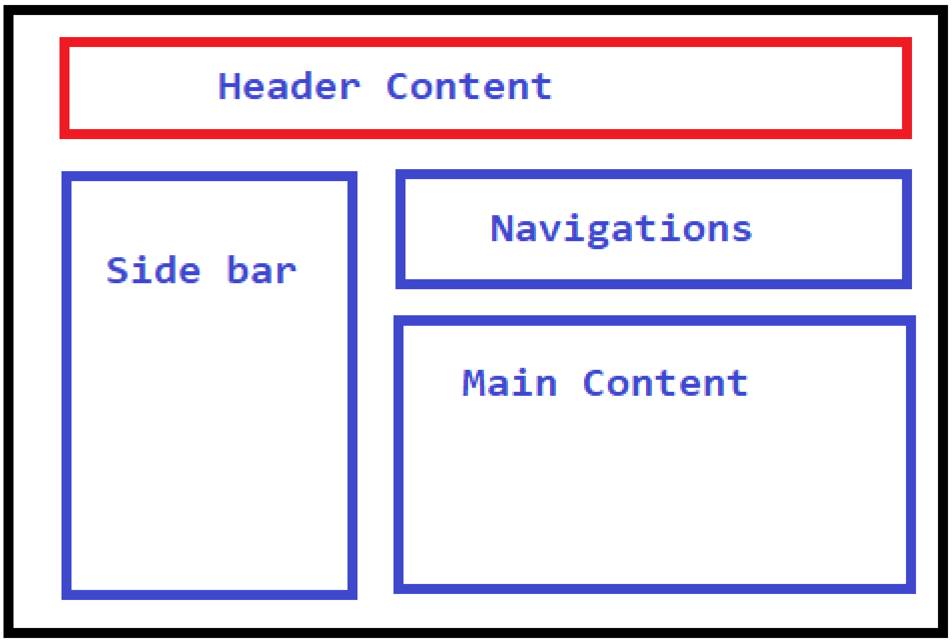
Prerequisites
HTML
Javascript (ES6 or above)
CSS

React and Angular:
Both are used to develop SPA, Angular is backed by Google and React is backed by Facebook, Both are good for small and medium market business, Both are powerful and flexible, while none of them are worse or better, than the other, Depending on customer app goals and particular system constraints developer choose react over angular and vice versa.
React is used only to build User interfaces it is composed of javascript library concentrates only on View part, whereas Angular is a fully fledged MVC.

What is React?
React is a Javascript library for building User Interfaces.
Javascript library is used to build Javascript driven app which is run on the browser not on the server, because it runs on the browser things happen instantly and we don’t have to wait for server response to get a new page or render something new.
User Interfaces are basically what the user sees on the web page, React is all about using components to build these User Interfaces.
Components: These are part of the webpage which user sees, a web page is made up of multiple components and every component can be independent from other component. If you think of a web page it can be split it into multiple sections like this



Here we can have header component, side bar component, navigations component, and main component
Why this is important because when we split up our website in to such components, we can build these components as contained pieces of code, we don't have to build our entire webpage as one big page, and we can build all these tiny things on their own so that it makes working in teams easier but even if you are working alone it makes it easy for us to keep these code manageable, so later if you want to change side bar content then we only need to go to side bar component and update it we don't have to find that code in our entire web page code.
You can also reuse the components means you create the component once and use them at multiple places because components are just like custom HTML elements which will replace the content the element is associated with at runtime when component is rendered on the browser.
Example: If you see some shopping cart web pages, the product component will have price components, rating components, discount components when you click on the product it opens a new component with bigger description but you could still see the price, ratings, discount

components with other components like reviews, comparisons and many more.

Another Example:
Facebook comments will have user profiles picture, name and comment components, all these are independent components because the profile picture component is rendered at many places when user posts something his/her profile picture must appear in the newsfeed, his/her profile must appear in the likes, comments and many more.

React is developed by Facebook and React has good community support for developers who uses React to build the web pages, Many of the Single page applications nowadays use Component based approach like GitHub, Facebook, Udemy, Angular official webpage, React official webpage, Gmail and etc.

The other alternatives for building single page applications are Angular and Vue.js.

You can create React from React tool below are

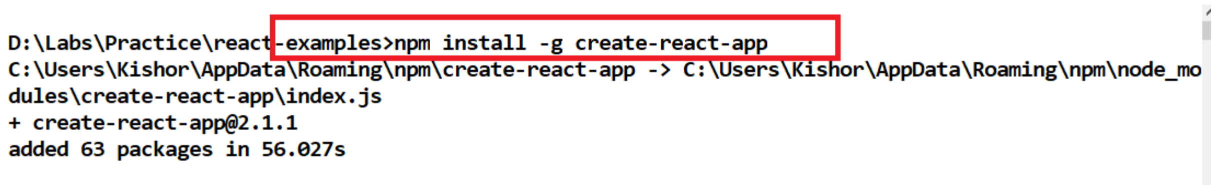
Software's required

Node.js

Visual Studio Code

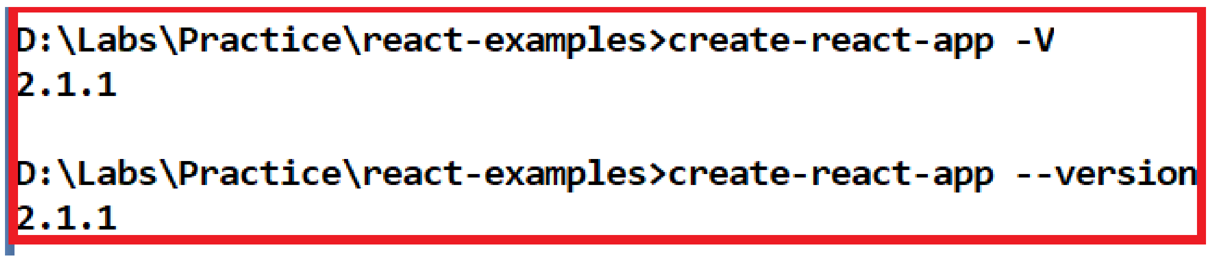
Browser

Installing React tool using npm command



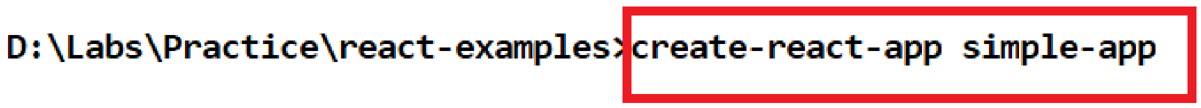
```
D:\Labs\Practice\react-examples>npm install -g create-react-app
C:\Users\Kishor\AppData\Roaming\npm\create-react-app -> C:\Users\Kishor\AppData\Roaming\npm\node_mo
dules\create-react-app\index.js
+ create-react-app@2.1.1
added 63 packages in 56.027s
```

Verify the React tool from below commands



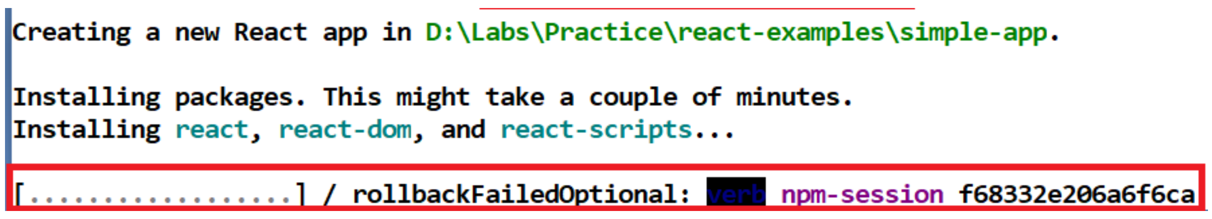
```
D:\Labs\Practice\react-examples>create-react-app -V
2.1.1

D:\Labs\Practice\react-examples>create-react-app --version
2.1.1
```

Create your first application using create-react-app <project-directory>



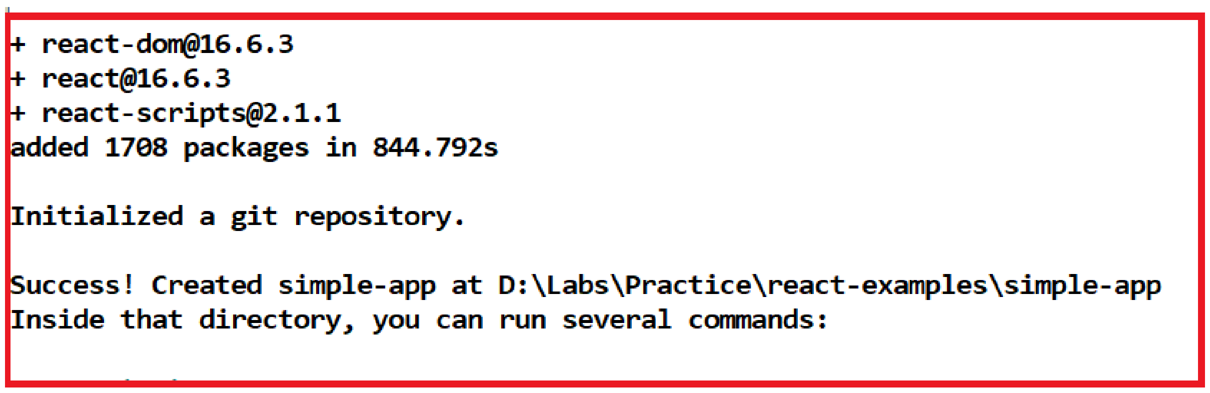
```
D:\Labs\Practice\react-examples>create-react-app simple-app
```

You will see below logs of react installing



```
Creating a new React app in D:\Labs\Practice\react-examples\simple-app.

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

[..................] / rollbackFailedOptional: verb npm-session f68332e206a6f6ca
```

Once it is installed you will see this success log

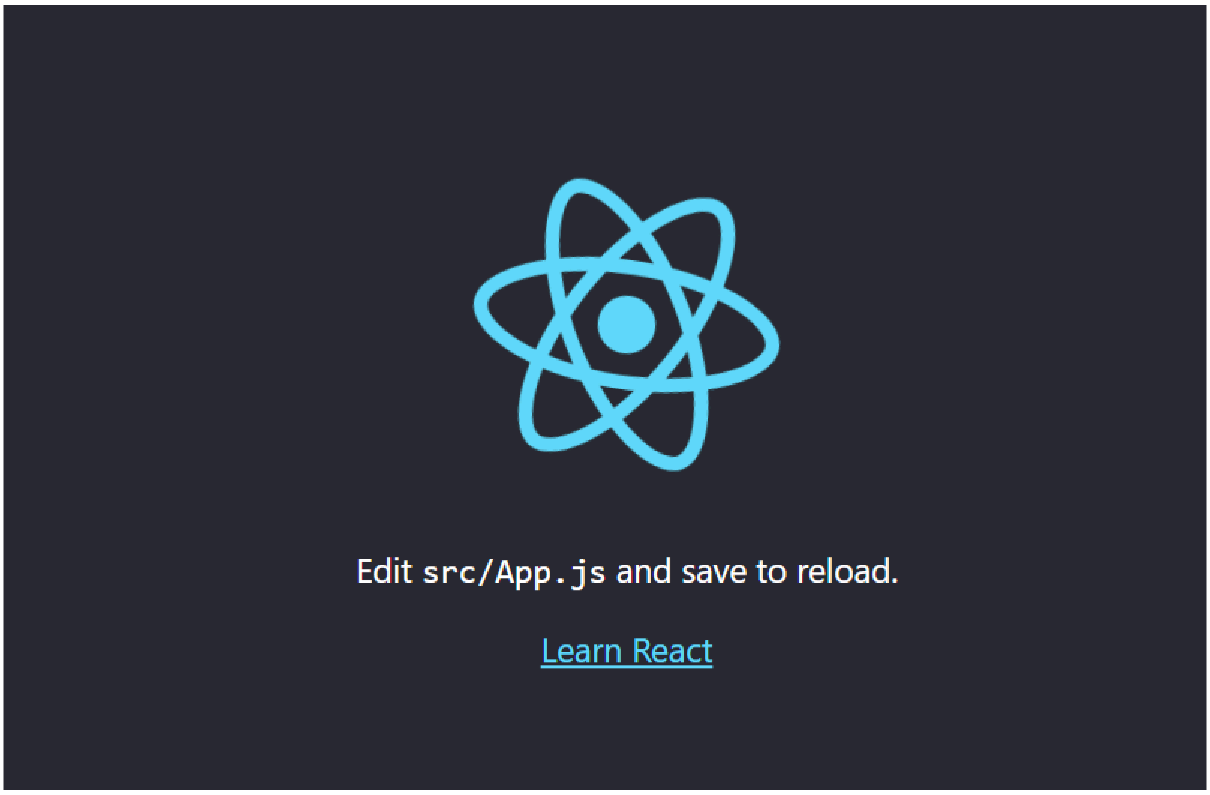


```
+ react-dom@16.6.3
+ react@16.6.3
+ react-scripts@2.1.1
added 1708 packages in 844.792s

Initialized a git repository.

Success! Created simple-app at D:\Labs\Practice\react-examples\simple-app
Inside that directory, you can run several commands:
```

Once installed take your command prompt to your project folder to start your application using *npm start*.

```
D:\Labs\Practice\react-examples\simple-app>npm start
_
```

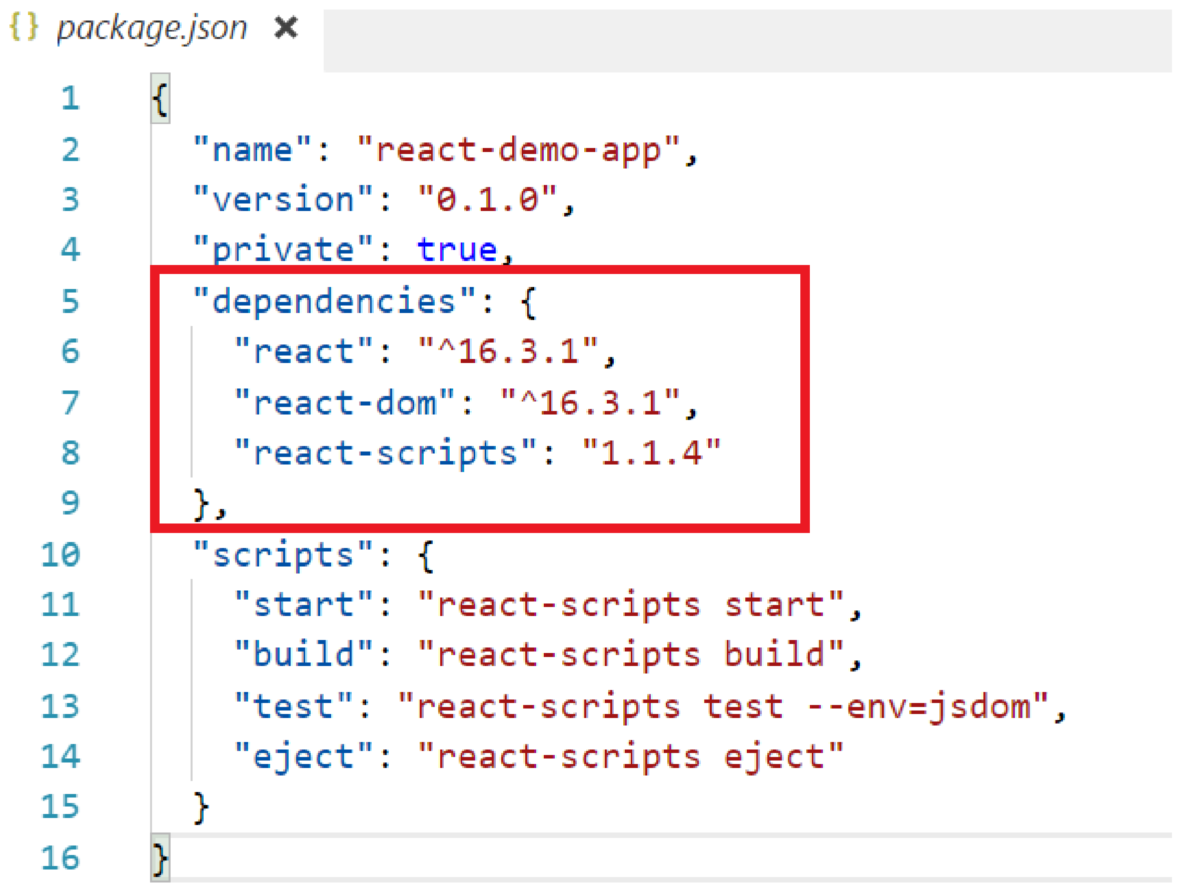
You can see in the browser below output or you can enter http://localhost:3000 to see the below output



Edit `src/App.js` and save to reload.

Learn React

You see lot of configuration files in the root folder, you can ignore the lock configuration files they are just locking the version of the dependency you are using.
package.json: It will have the general dependencies like react-dom, react-scripts, react-version which is shown as below



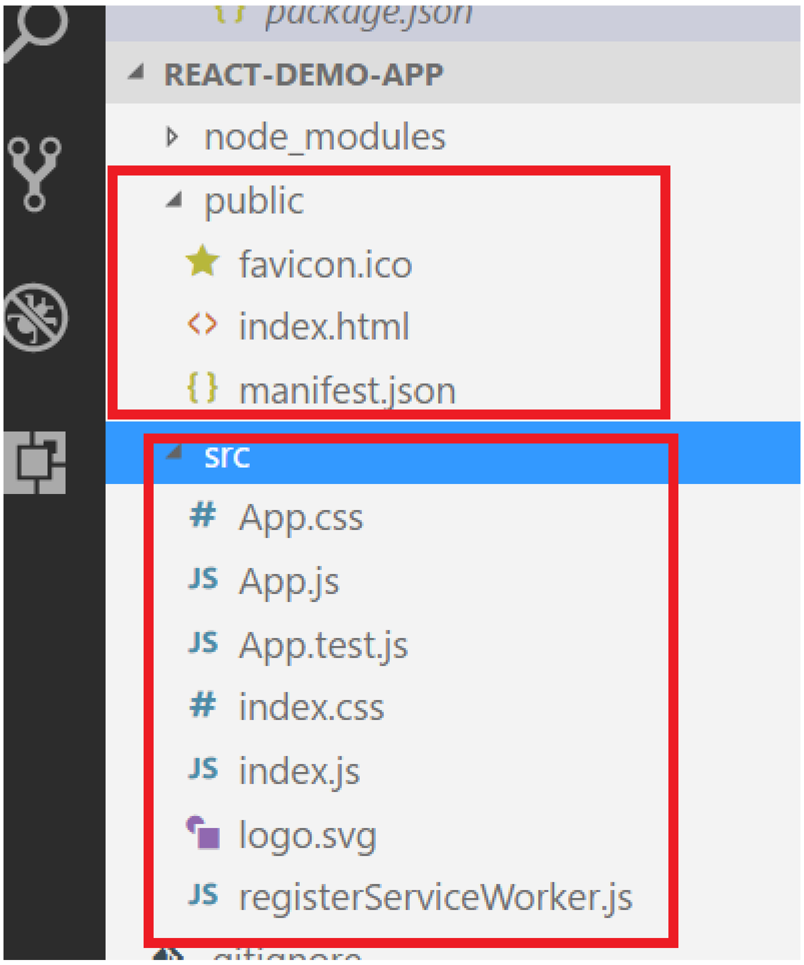
package.json

```json
{
  "name": "react-demo-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.3.1",
    "react-dom": "^16.3.1",
    "react-scripts": "1.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

You can see react version, react dom dependency which provides DOM specific methods that glues between components and DOM and
react-scripts which offer all the build workflow, development server, next generation javascript features support and all these things we are using in this project.
There are some scripts defined which you can run using npm command
You can run the start command which compiles your code and runs the application on development server i.e., npm start
The build command npm build will bundle the application for production which you can deploy on any server.
npm test starts the test runner
npm eject will remove the create-react-app tool, better not to use this unless you want to re-install

Instead of npm start you can also use yarn start which is, because npm start will internally call yarn start by looking into package.json, the same file is created for Angular applications also and when you hit npm start while developing angular application the npm start runs ng serve command, it means npm start is common command to run any application on the server and it reads package.json to know what script name to be run when you use npm followed by command name

Below image shows some important files & folders



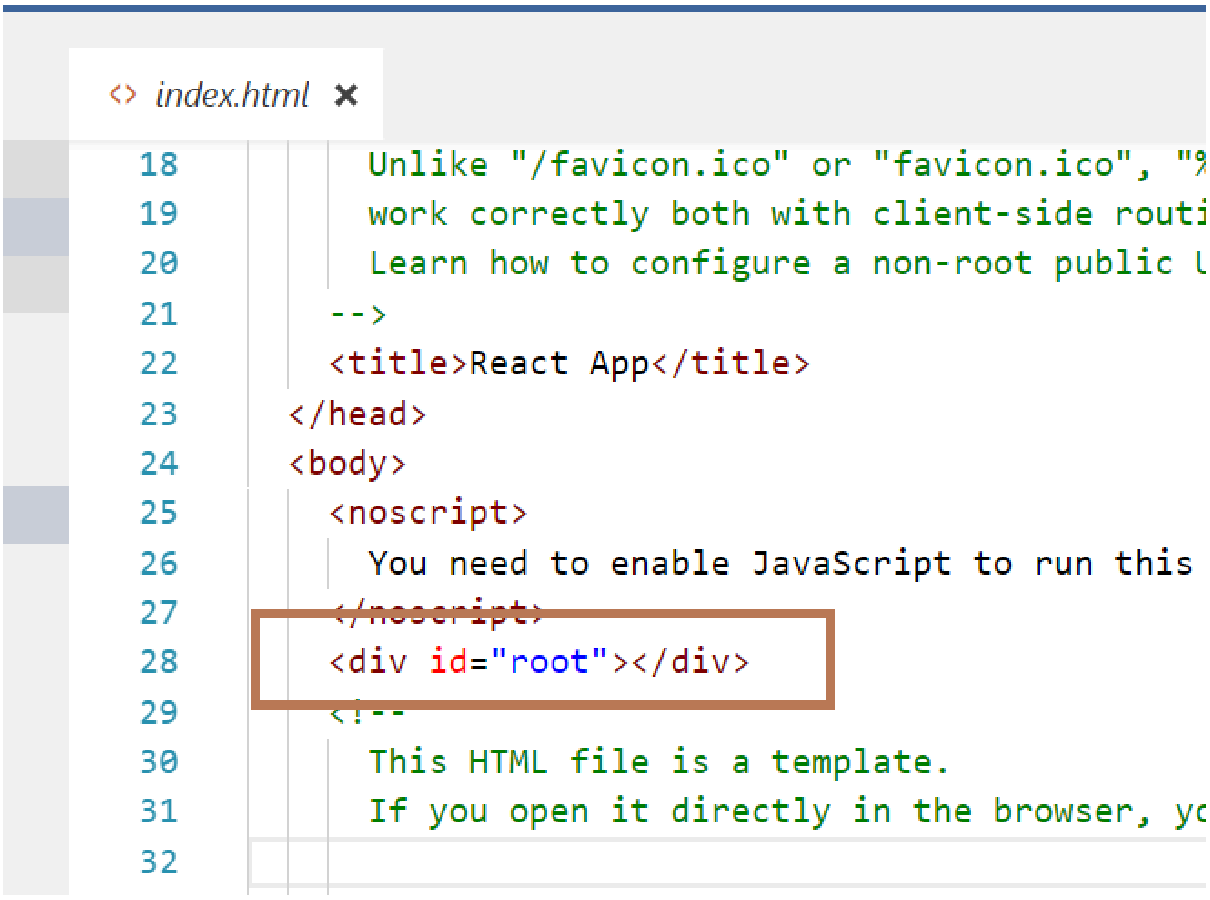
```
package.json
REACT-DEMO-APP
  node_modules
  public
    favicon.ico
    index.html
    manifest.json
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    registerServiceWorker.js
```

node-moudles: It holds all the dependencies and sub-dependencies of your project

public: This is a root folder basically served by the web server, it will have only one html file which you must not rename, it will be opened once you deploy your application, it will have all the components that has to be rendered, we will not create multiple html pages unless you are creating multiple page applications.

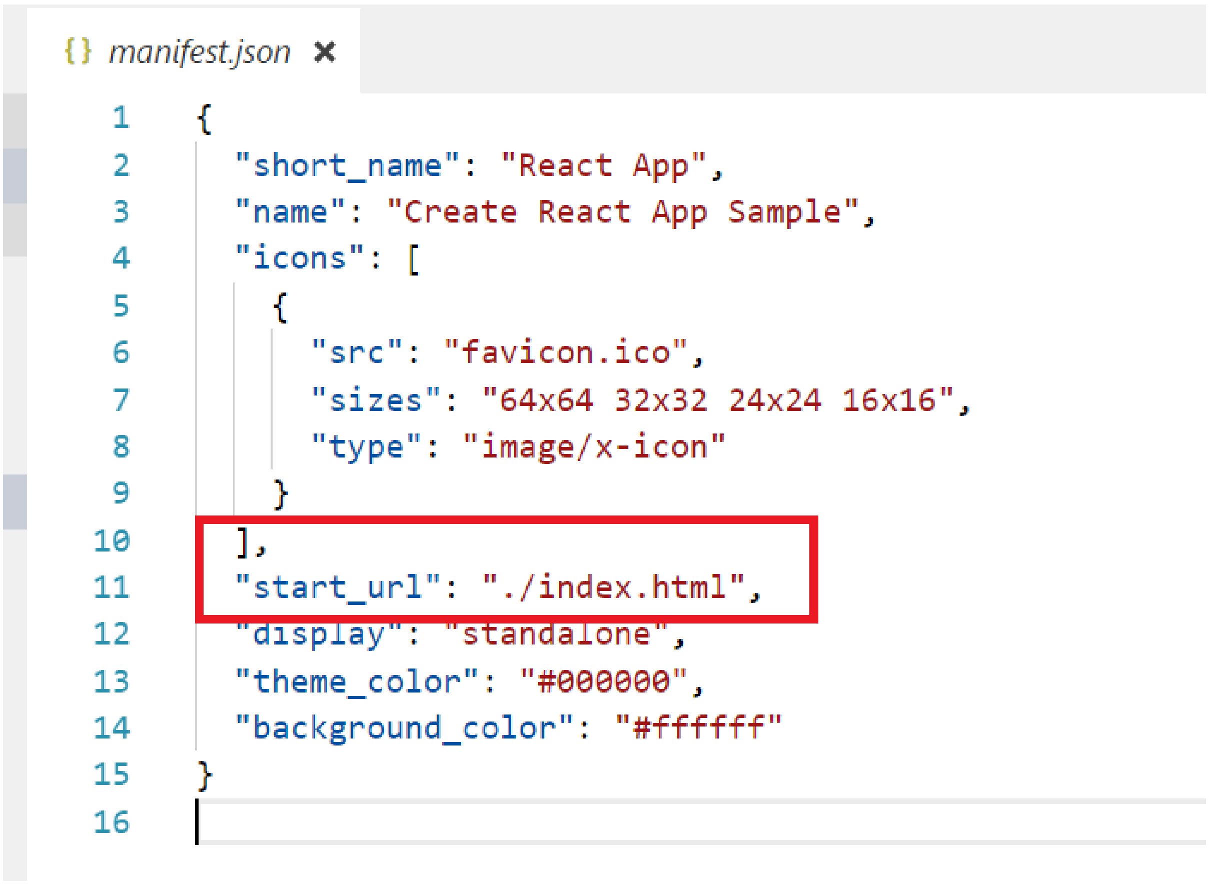
The index.html will have a div with id root on which we basically mound the react application later, whatever the components you develop you would do in the react application and mound that application to

<div id = ‘root’>



```
index.html
        Unlike "/favicon.ico" or "favicon.ico", "%
        work correctly both with client-side routi
        Learn how to configure a non-root public l
      -->
      <title>React App</title>
    </head>
    <body>
      <noscript>
        You need to enable JavaScript to run this
      </noscript>
      <div id="root"></div>
      <!--
        This HTML file is a template.
        If you open it directly in the browser, yo

```

manifest.json: It will tell the server what is the html file it has to launch when application is deployed

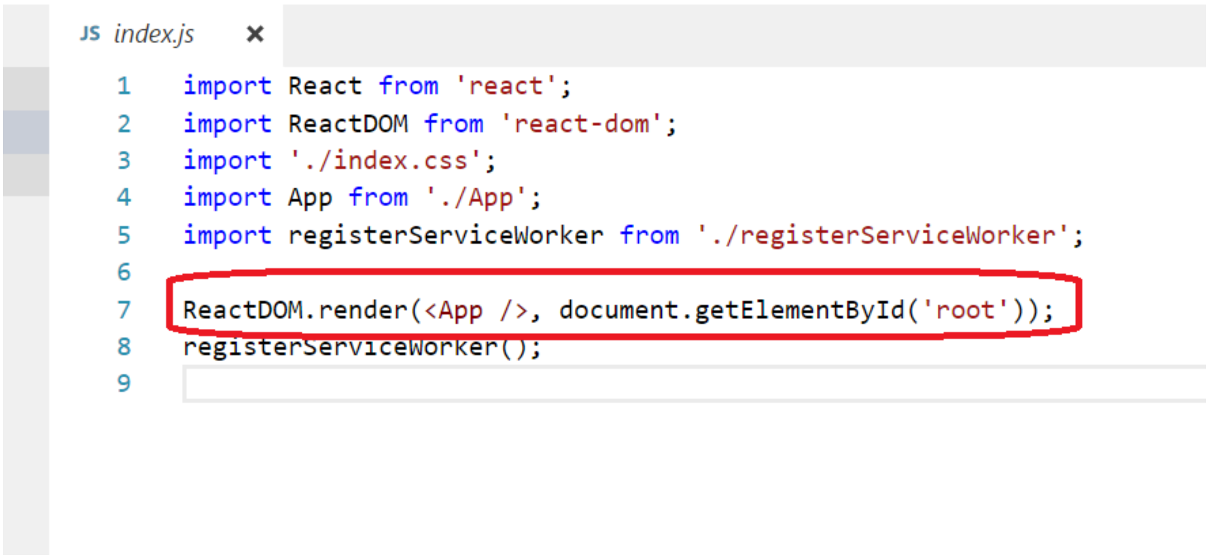


{} manifest.json ✕

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    }
  ],
  "start_url": "./index.html",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}

```

src folder will have scripts what we create and it is the folder which will have our React application code, one of the important file which is going to add our react application to the index.html <div> is index.js

index.js: It will have the code that adds the React App by accessing the root element <div> of our index.html.
Old code of React in index.js

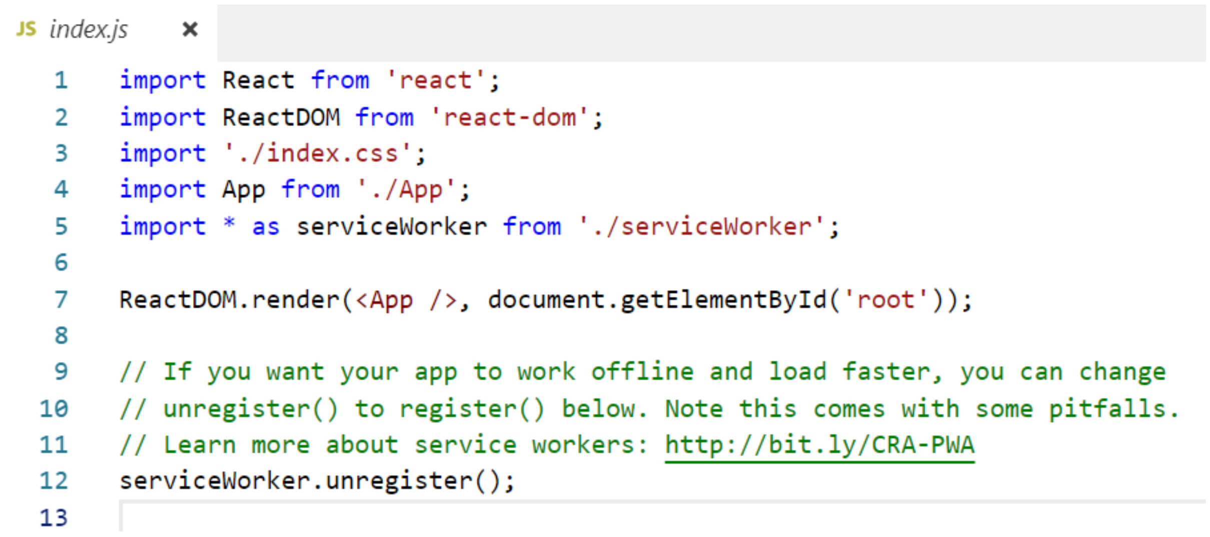


```
index.js

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(<App />, document.getElementById('root'));
registerServiceWorker();
```

Notice the render function taking <App /> which is the custom HTML element that is present in App.js which is also imported in the index.js, The content present in the App component is actually added to the root element.

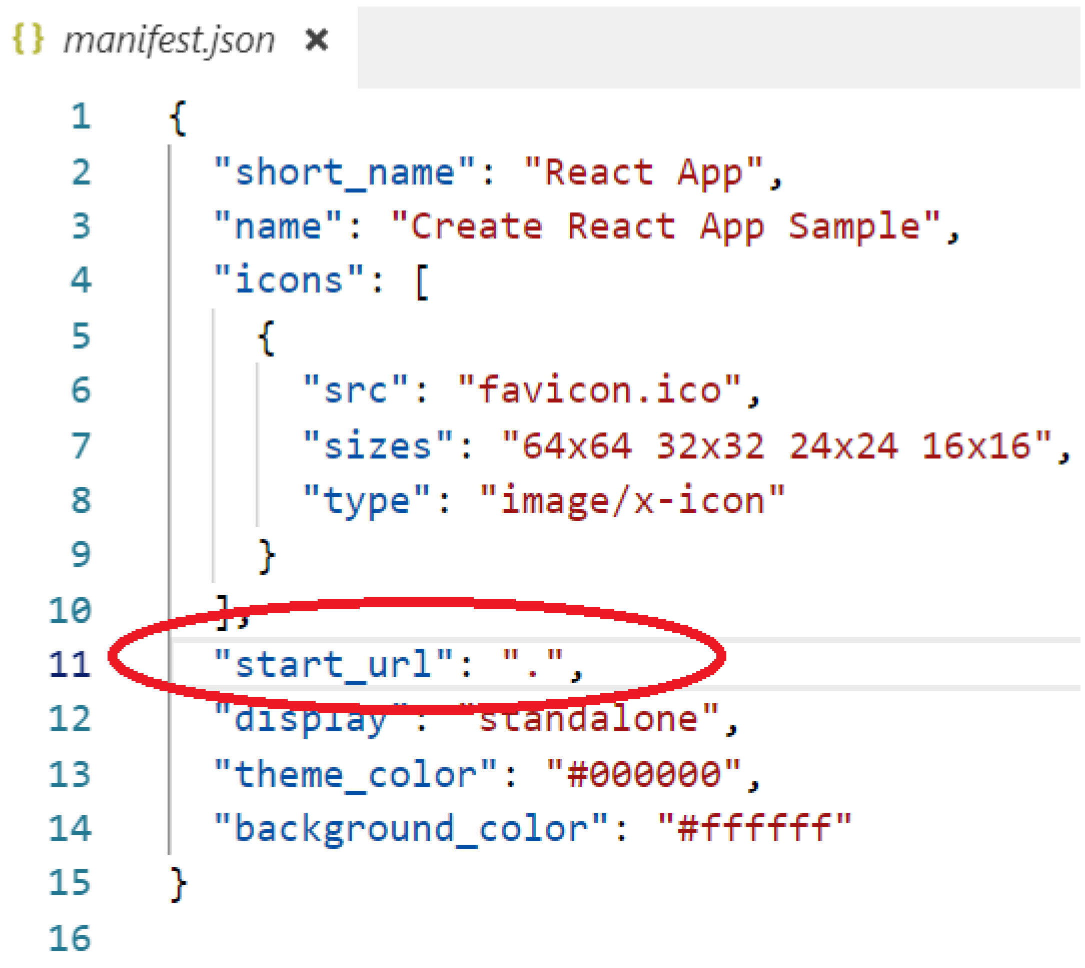
registerServiceWorker() will have a set up where the application to be launched its like a bootstrap, the code can be found in registerServiceWorker.js

New React Code the registerServicWorker.js is replaced by serviceWorker.js
This is how index.js looks now



```
index.js

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

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

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: http://bit.ly/CRA-PWA
serviceWorker.unregister();
```

manifest.json doesn't show index.html as home file it simply specifies . in start url



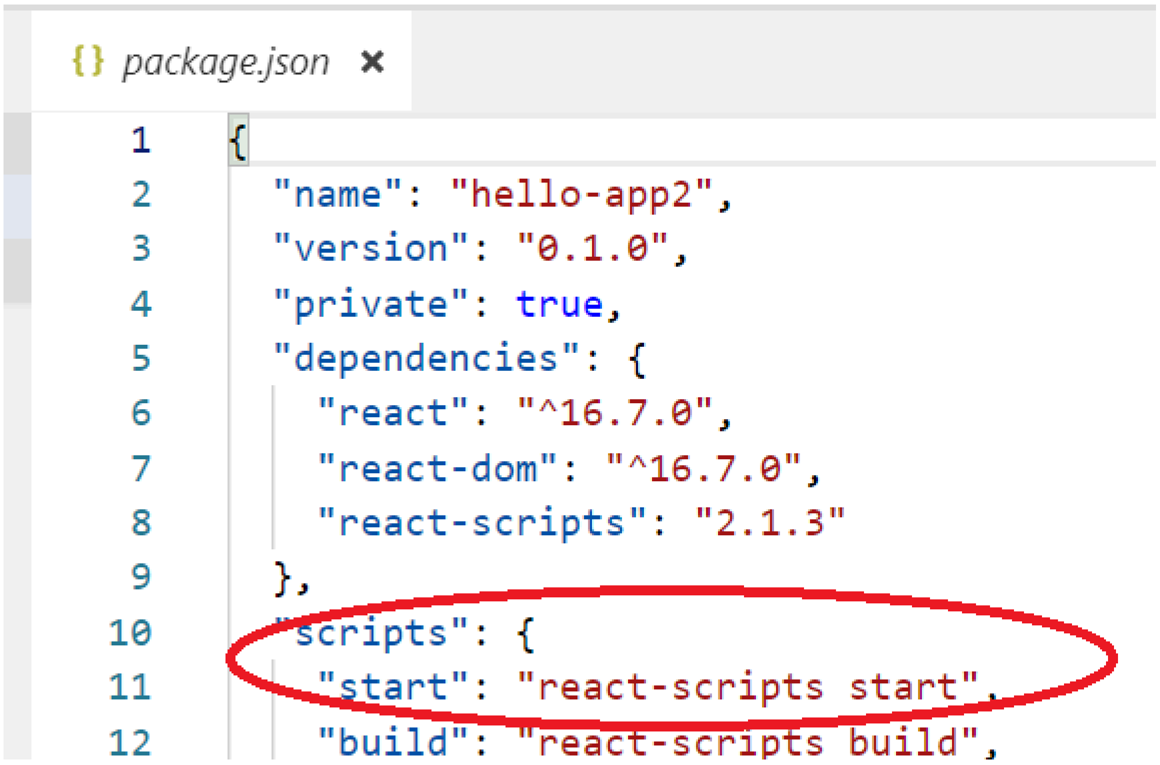
manifest.json

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}

```

In New code the react-script will have a code to call the code to open index.html which is abstract so that developers doesn't try modifying the home file,

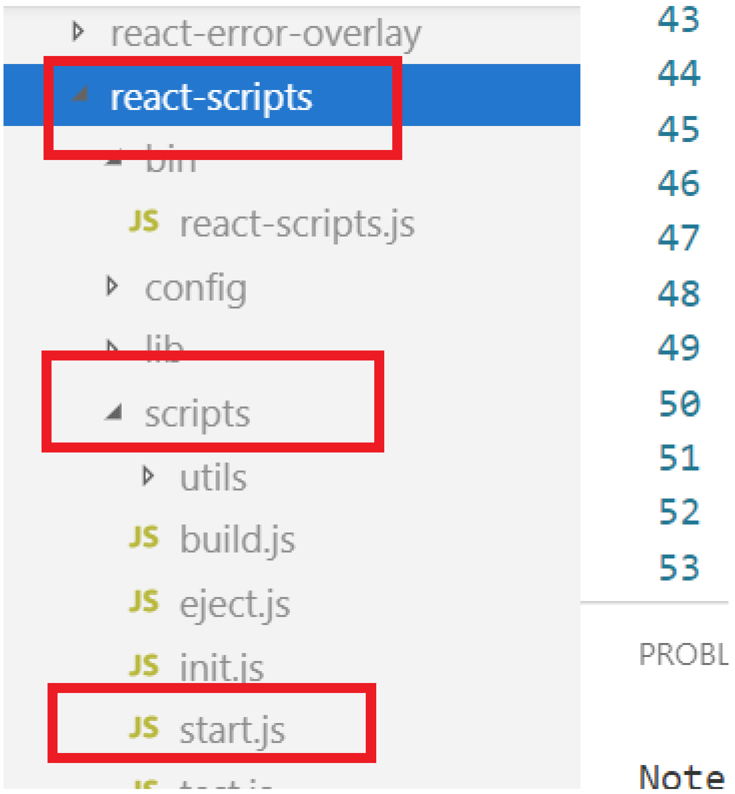
However this is how the flow goes to index.html

1. npm start -> calls -> react-scripts.js which you can observe in package.json,

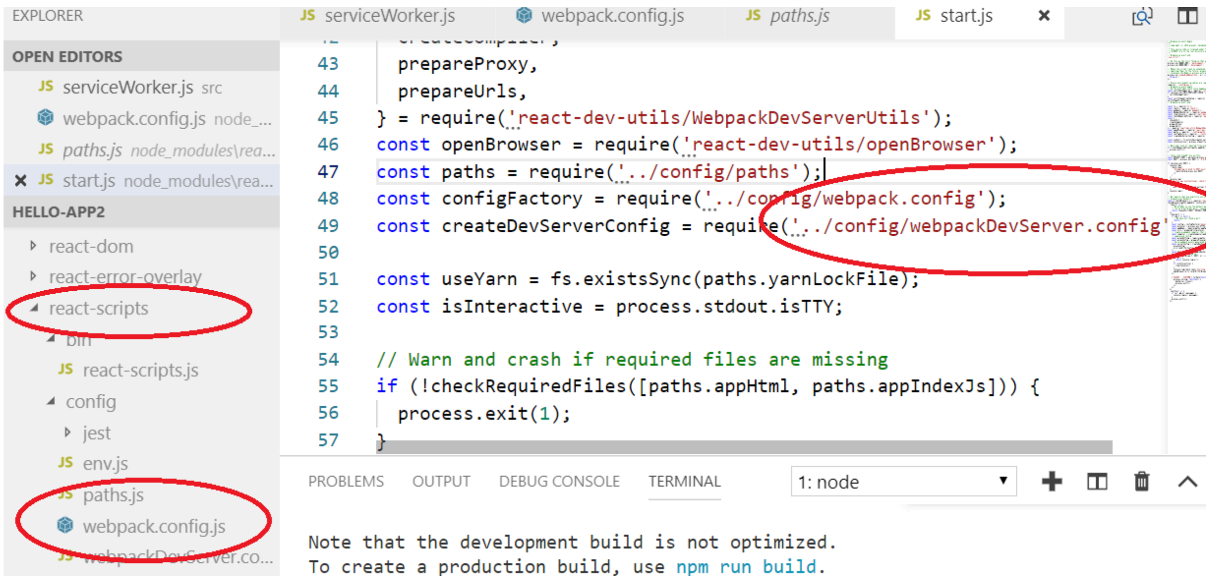


```
{
  "name": "hello-app2",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.7.0",
    "react-dom": "^16.7.0",
    "react-scripts": "2.1.3"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
```

2. react-scripts.js is present inside node_modules/bin has a code that searches scripts folder



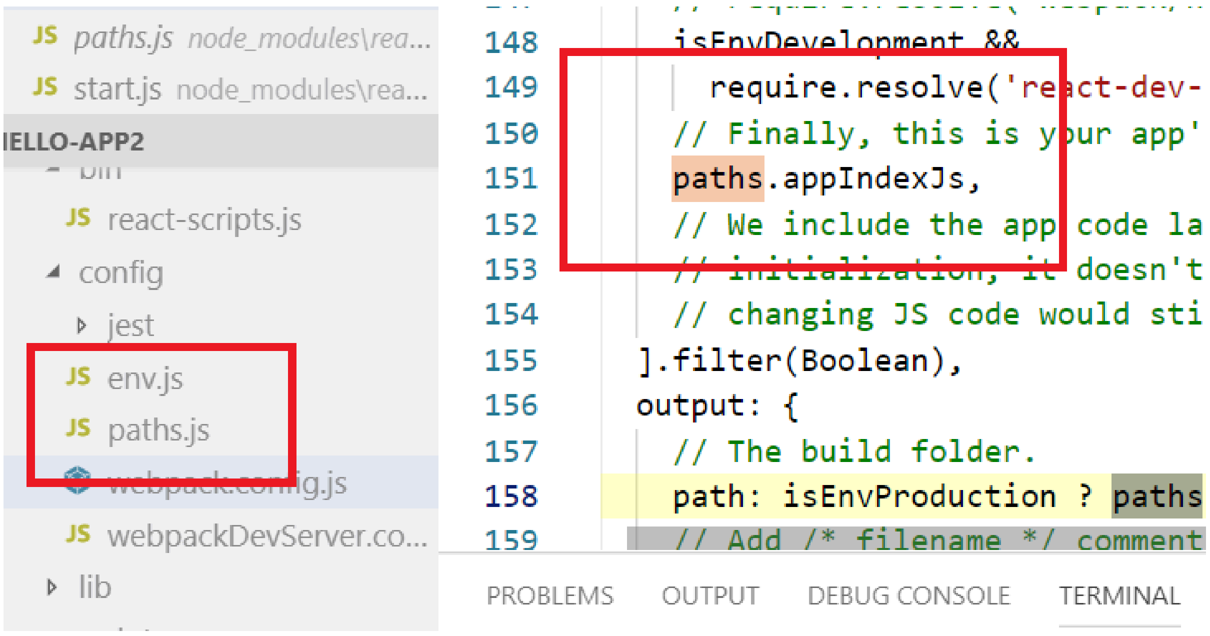
3. Since the commands is react-script start internally because of npm start you react executes the code in start.js which has a web pack config that actually present in the same folder with the name webpack.config.js



```
  prepareProxy,
  prepareUrls,
} = require('react-dev-utils/WebpackDevServerUtils');
const openBrowser = require('react-dev-utils/openBrowser');
const paths = require('../config/paths');
const configFactory = require('../config/webpack.config');
const createDevServerConfig = require('../config/webpackDevServer.config'

const useYarn = fs.existsSync(paths.yarnLockFile);
const isInteractive = process.stdout.isTTY;

// Warn and crash if required files are missing
if (!checkRequiredFiles([paths.appHtml, paths.appIndexJs])) {
  process.exit(1);
}
```

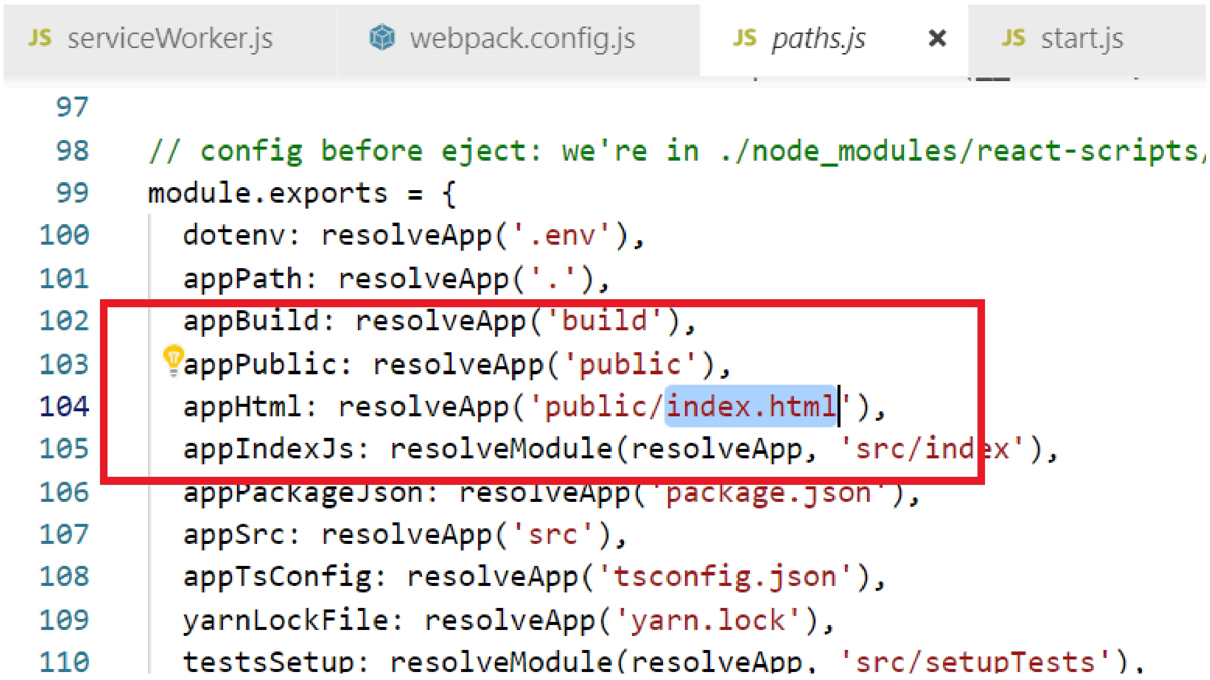
```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

4. The reference of webpack.conifig is supplied to a function in start.js that executes webpack.config.js that has a code to search paths.js which is in the same folder



```
    isEnvDevelopment &&
      require.resolve('react-dev-
    // Finally, this is your app'
    paths.appIndexJs,
    // We include the app code la
    // initialization, it doesn't
    // changing JS code would sti
  ].filter(Boolean),
  output: {
    // The build folder.
    path: isEnvProduction ? paths
    // Add /* filename */ comment
```

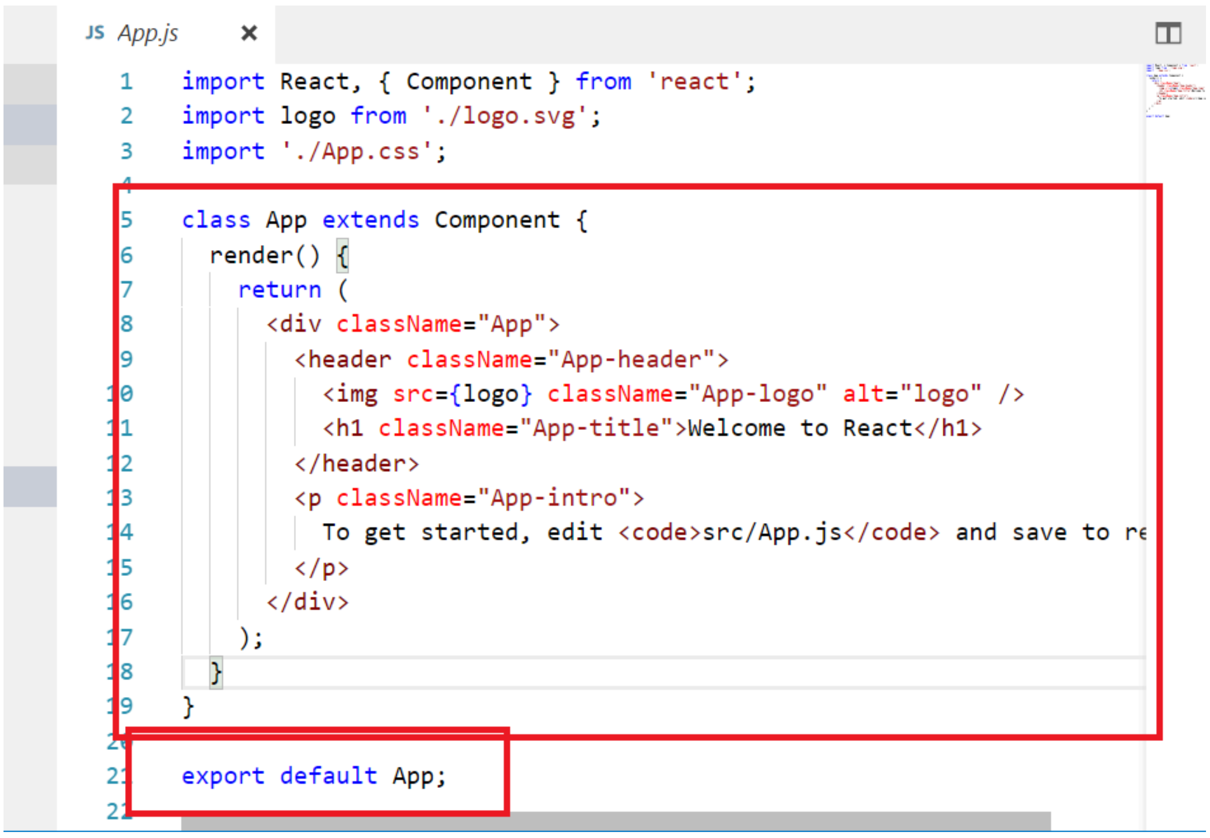
5. Finally the paths.js has the opening of index.html code



```
// config before eject: we're in ./node_modules/react-scripts,
module.exports = {
  dotenv: resolveApp('.env'),
  appPath: resolveApp('.'),
  appBuild: resolveApp('build'),
  appPublic: resolveApp('public'),
  appHtml: resolveApp('public/index.html'),
  appIndexJs: resolveModule(resolveApp, 'src/index'),
  appPackageJson: resolveApp('package.json'),
  appSrc: resolveApp('src'),
  appTsConfig: resolveApp('tsconfig.json'),
  yarnLockFile: resolveApp('yarn.lock'),
  testsSetup: resolveModule(resolveApp, 'src/setupTests'),
```

This paths.js is the file that ultimately served when you trigger npm start command ☺

The App.js looks as below


App.js

```
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" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to re
        </p>
      </div>
    );
  }
}

export default App;
```

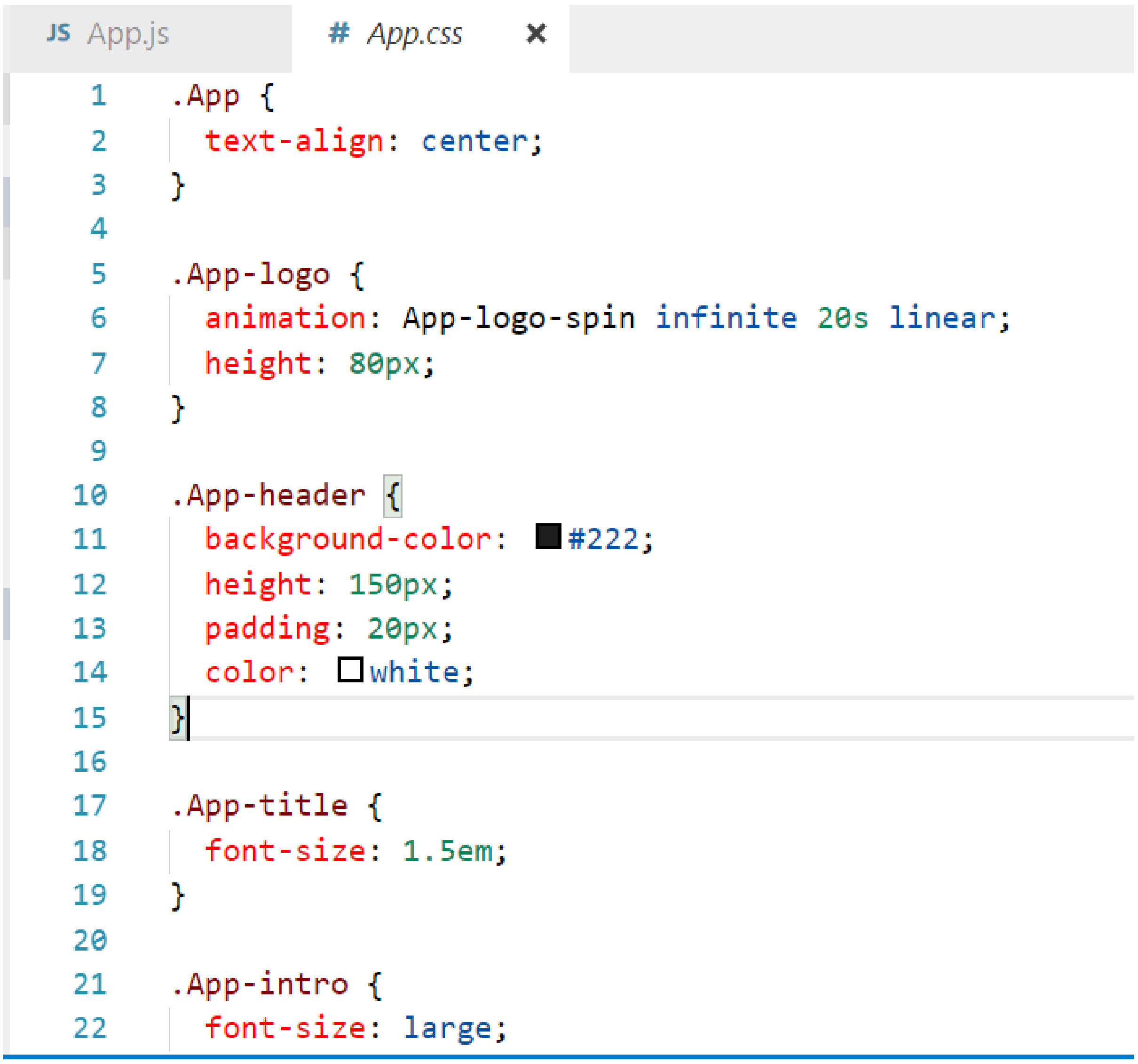
Once the class extends Component then the class will be a component whose name can be used as Custom HTML element and class must have a render() function which will have the content that has to rendered when you use the component.
The import statement has 2 types of syntax, if the class has default export then you need to use classnames without {} and if the class doesn't have default export but only export then it has to be imported using {}, like import React from 'react' and import {Component} from 'react', Since React is having default export and Component doesn't have the default export they both are imported differently.

Note: It is mandatory to import React even if you don't use inside your script because the compiler will convert the JSX to compatible Javascript i.e., if you *return(<div>hello</div>)* in JSX then the transpiler converts it into *React.createElement('div',null,'hello');*

The export keyword enables the App to be imported in other script files

App.js is also import App.css file which has the styling for the class names used in the render()

App.css



```
.App {
  text-align: center;
}

.App-logo {
  animation: App-logo-spin infinite 20s linear;
  height: 80px;
}

.App-header {
  background-color: ■#222;
  height: 150px;
  padding: 20px;
  color: □white;
}

.App-title {
  font-size: 1.5em;
}

.App-intro {
  font-size: large;
```

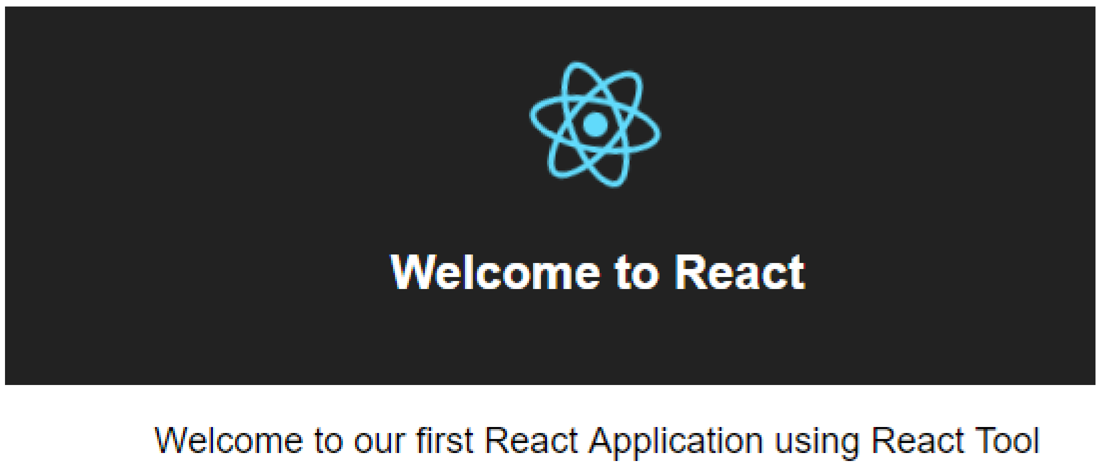
Change the code of App.js and save the code you can see browser automatically refreshes itself

App.js

```
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" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          Welcome to our first React Application using React Tool
        </p>
      </div>
    );
  }
}

export default App;
```

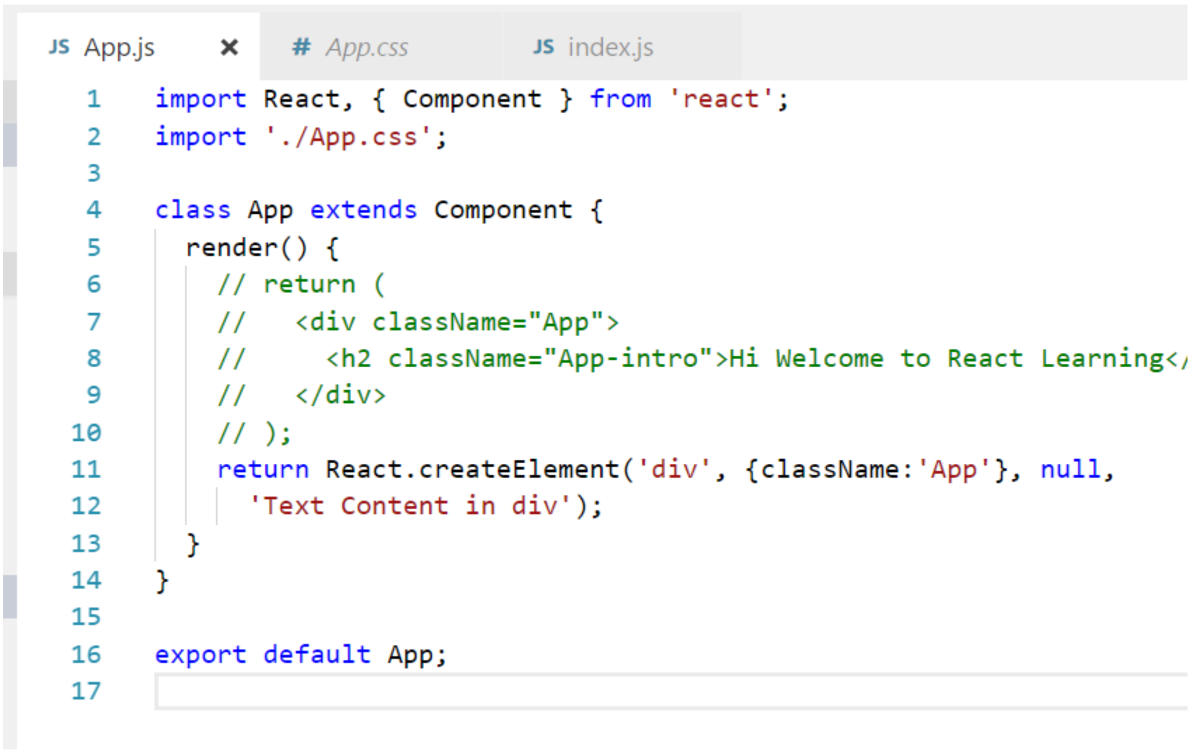
Output



Welcome to React

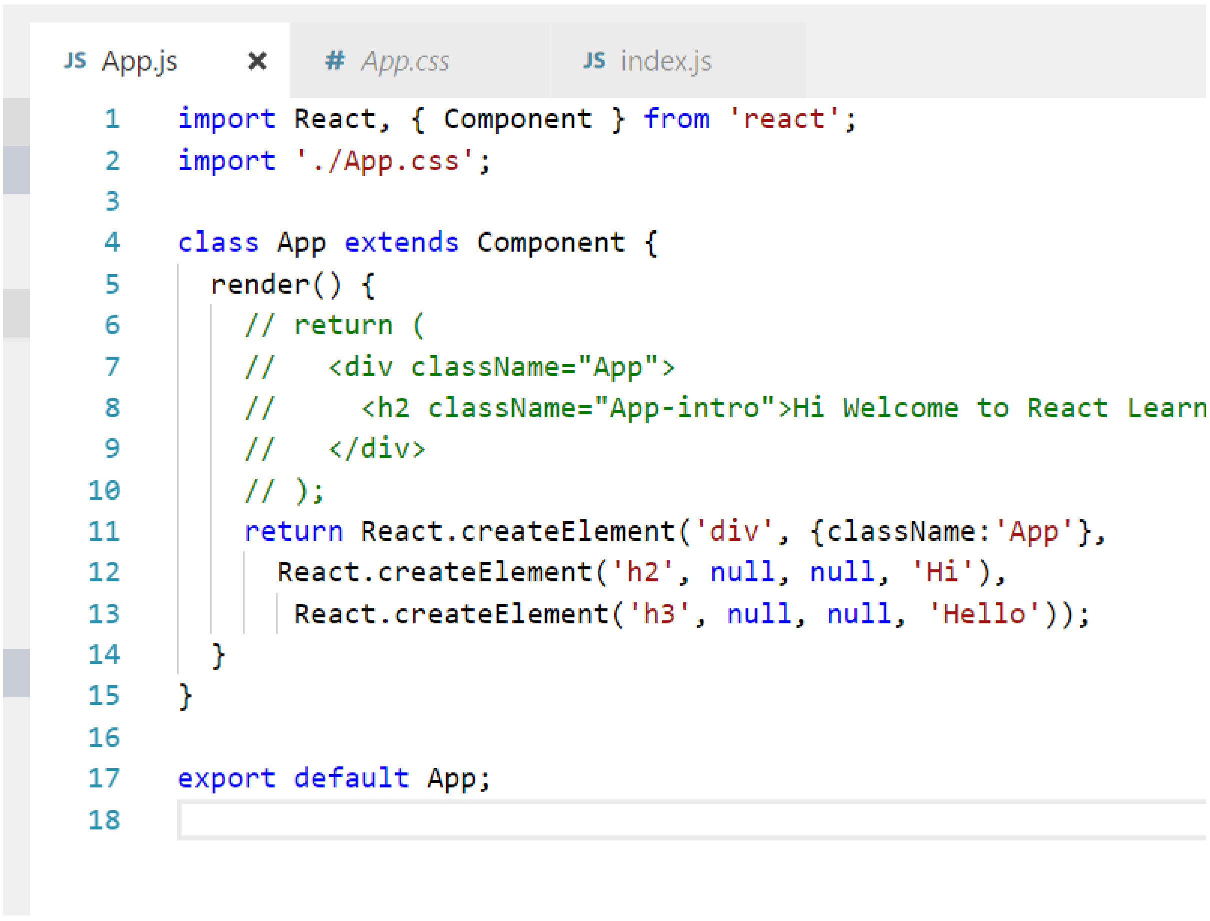
Welcome to our first React Application using React Tool

Another way of adding content in the render function is using JSX code using React.createElement() method which takes 3 parameters minimu i.e., React.createElement(elementName, JSObjects, childrens)The below code will render the content of HTML through without JSX i.e., Using Javascript function to create HTML elements which is quite cumbersome when you compare with JSX code which enables you to add HTML element



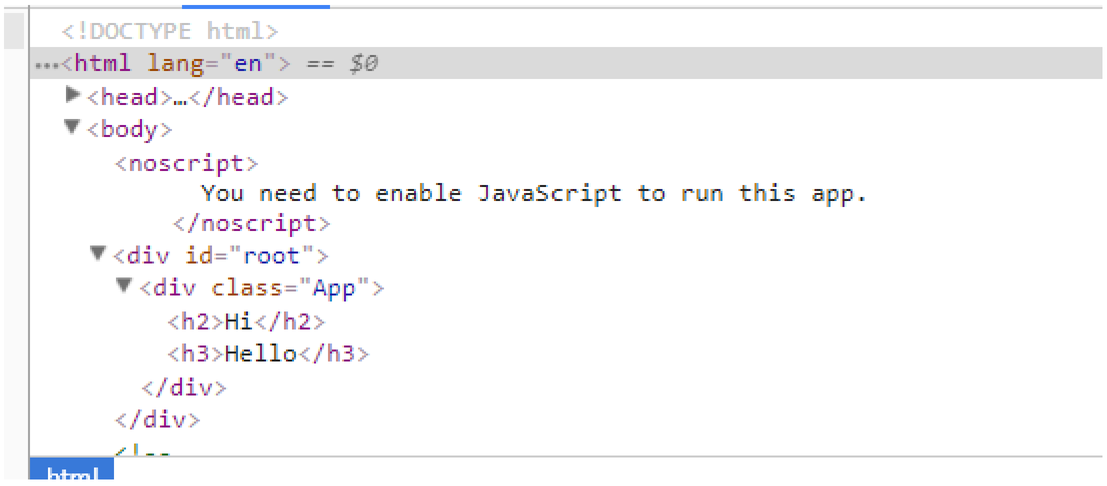
```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    // return (
    //   <div className="App">
    //     <h2 className="App-intro">Hi Welcome to React Learning</
    //   </div>
    // );
    return React.createElement('div', {className:'App'}, null,
      'Text Content in div');
  }
}

export default App;
```

The third parameter can be replaced with another React.createElement() function as below



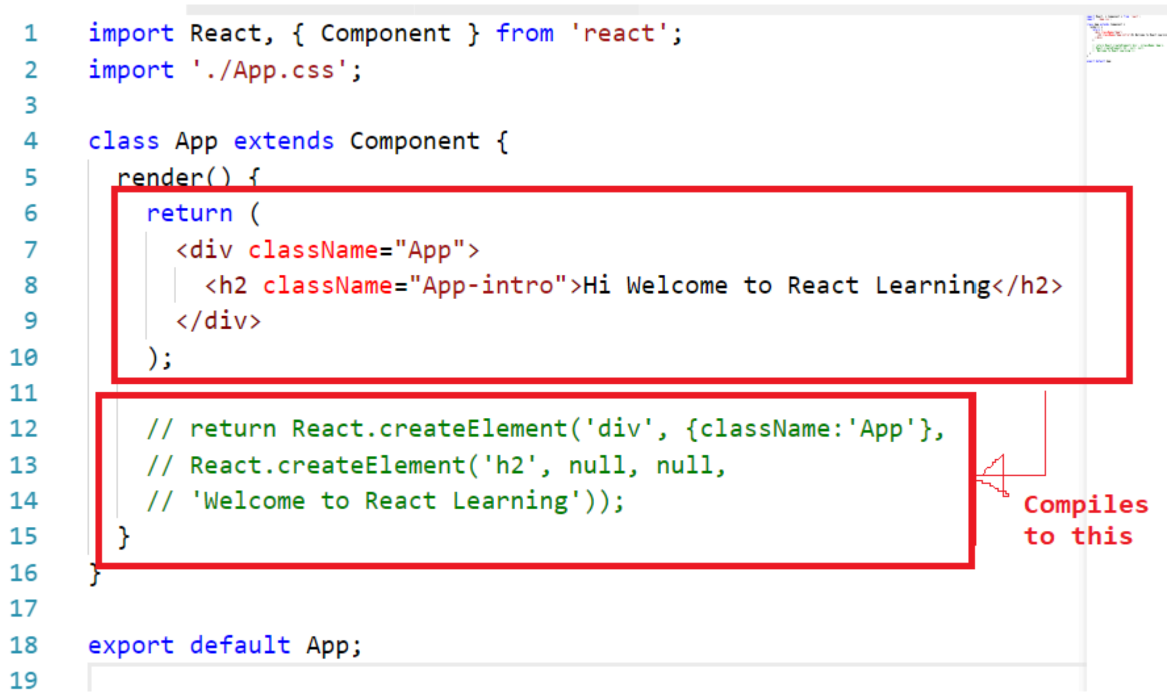
```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    // return (
    //   <div className="App">
    //     <h2 className="App-intro">Hi Welcome to React Learn
    //   </div>
    // );
    return React.createElement('div', {className:'App'},
      React.createElement('h2', null, null, 'Hi'),
       React.createElement('h3', null, null, 'Hello'));
  }
}

export default App;
```

The DOM looks as below



```
<!DOCTYPE html>
...<html lang="en"> == $0
  ▶<head>…</head>
  ▼<body>
      <noscript>
            You need to enable JavaScript to run this app.
          </noscript>
    ▼<div id="root">
      ▼<div class="App">
          <h2>Hi</h2>
          <h3>Hello</h3>
        </div>
      </div>
```

This still works but it is complex to write which is why we use JSX which is compiled to javascript code i.e.,



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

class App extends Component {
  render() {
    return (
      <div className="App">
        <h2 className="App-intro">Hi Welcome to React Learning</h2>
      </div>
    );

    // return React.createElement('div', {className:'App'},
    // React.createElement('h2', null, null,
    // 'Welcome to React Learning'));
  }
}

export default App;
```

Note:
While writing JSX you must have only one root element i.e., <div> in the App and also you must not use class attribute in the element instead you must use className

You can have <Person /> any number of times in the JSX since it is a component & is reusable.
You can also create a class that extends Component and have a render() function return HTML

How to use JS code inside JSX

You must wrap the JS code in curly braces {} as below
{Math.random()}
{date} if date = new Date().toString()

Create Test.js inside src folder

Test.js

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

export class Test extends Component {
    render() {
        const date = new Date().toString();
        return(
            <h4>You have visited the website at {date} </h4>
        );
    }
}
```

Since you have written export in the class definition you need to import Test within {}
i.e., import {Test} from ‘./Test’

App.js

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

class App extends Component {
  render() {
    return (
      <div className="App">
        <h2 className="App-intro">Hi Welcome to React Learning</h2>
        <p>I’m a person</p>
        <Test />
      </div>
    );
  }
}

export default App;
```

- The import ‘./App.css’ will import the styles from App.css which has the classes used like App, App-intro
- Since React as default export it is imported as import React from ‘react’
- Since Test doesn’t have default import it is wrapped in {} while importing

You can change your index.js with Test component

Create Test.js file in src folder and add below code

Test.js

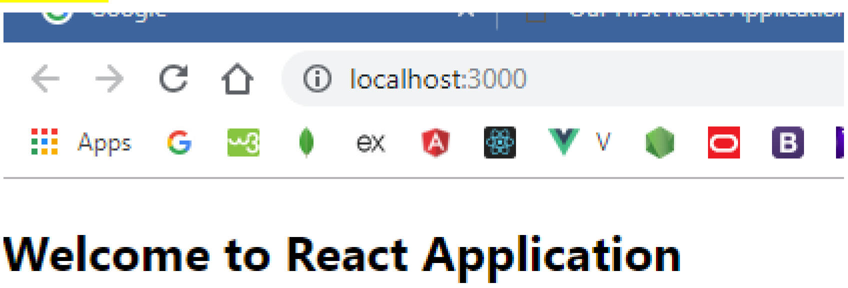
```
import React, {Component} from 'react';

class Test extends Component {
    render() {
       return(
           <div>
               <h2>Welcome to React Application</h2>
           </div>
       )
    }
}
export default Test;
```

Note: You will use import Test from ‘Test’ when you export default, if you write export before class then you will use import {Test} from ‘Test’

Add <Test /> in the render method of index.js as below

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from  './App';
import Test from './Test'
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<Test />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: http://bit.ly/CRA-PWA
serviceWorker.unregister();
```

Output



Welcome to React Application

Adding CSS to the Test component as below

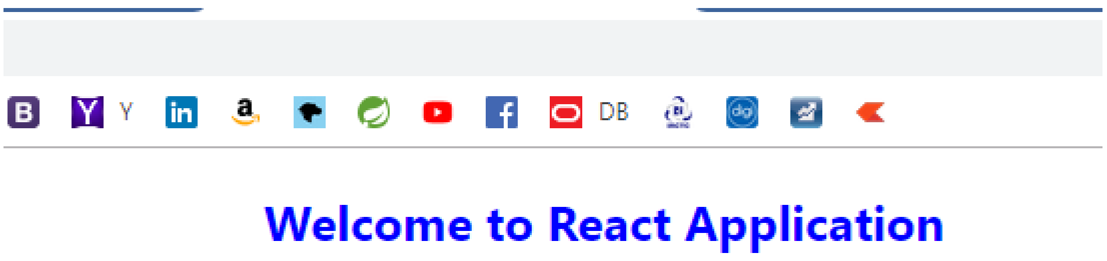
Create Test.css inside src folder

```
.blueStyle {
    color:blue;
    text-align: center;
}
```

Add this blueStyle in the Test Component using className attribute which is a JSX attribute

```
import React, {Component} from 'react';
import './Test.css';

class Test extends Component {
    render() {
       return(
           <div>
               <h2 className='blueStyle'>Welcome to React Application</h2>
           </div>
       )
    }
}
export default Test;
```

Output:



Welcome to React Application

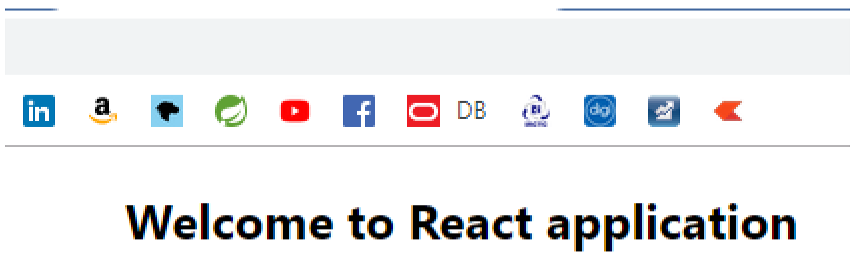
Now you can add multiple components inside App.js only, change your App.js code as below

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

class App extends Component {
  render() {
    return (
      <div className="App">
        <h2>Welcome to React application</h2>
      </div>
    );
  }
}

export default App;
```

index.js

```
…

import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));
….
```

Output:

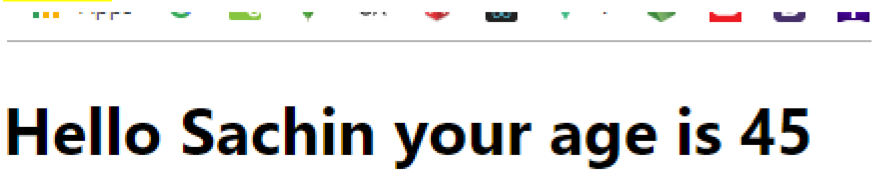

Welcome to React application

You can use {} as an expression language to access some variables inside render() function as below

App.js

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

class App extends Component {

  render() {
    let username = "Sachin";
    let age = 45;
    return (
      <div>
        <div>
          <h1>Hello {username} your age is {age}</h1>
        </div>
      </div>
    );
  }
}
export default App;
```

Output



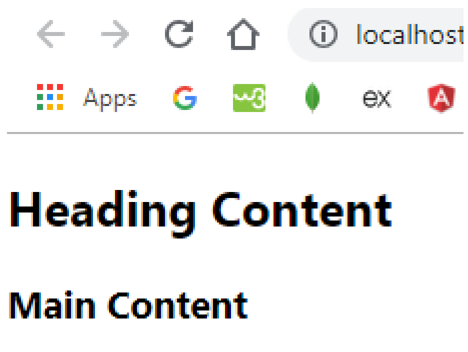
Hello Sachin your age is 45

Nesting multiple components

App.js

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

class Heading extends Component {
  render() {
    return (
      <div>
        <h2>Heading Content</h2>
      </div>
    )
  }
}
class Main extends Component {
  render() {
    return (
      <div>
        <h3>Main Content</h3>
      </div>
    )
  }
}

class App extends Component {
  render() {
    let username = "Sachin";
    let age = 45;
    return (
      <div>
        <Heading />
        <Main />
      </div>
    );
  }
}
export default App;
```

Output



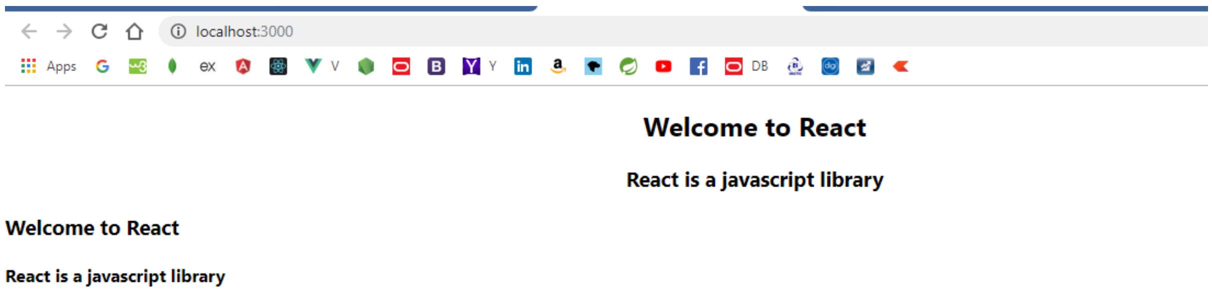
Heading Content

Main Content

App.js

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

class Heading extends Component {
  render() {
    return (
      <div className = 'App'>
        <h2>{this.props.titleData}</h2>
        <h3>{this.props.messageData}</h3>
      </div>
    )
  }
}
class Main extends Component {
  render() {
    return (
      <div>
        <h3>{this.props.title}</h3>
        <h4>{this.props.message}</h4>
      </div>
    )
  }
}

class App extends Component {
  render() {
    let title = "Welcome to React";
    let message = "React is a javascript library";
    return (
      <div>
        <Heading titleData = {title} messageData = {message}/>
        <Main title = {title} message = {message}/>
      </div>
    );
  }
}
export default App;
```

Output



Welcome to React

React is a javascript library

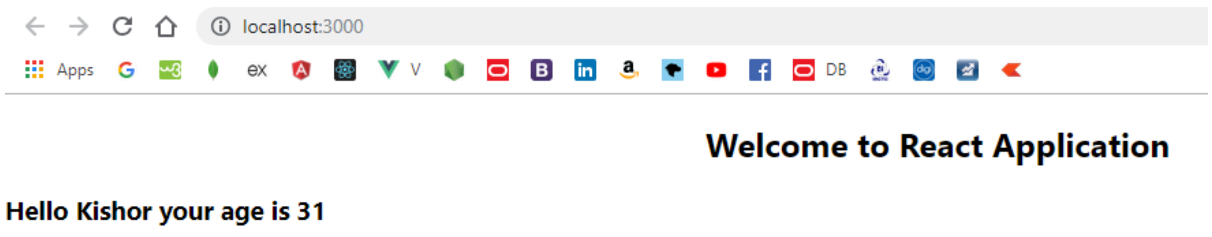
Welcome to React

React is a javascript library

Another simple example with Person component

```
class Person extends React.Component {
  render() {
    return (
      <h3>Hello {this.props.name} your age is {this.props.age}</h3>
    )
  }
}
class App extends Component {
  render() {
    let name = "Kishor";
    let age = 31;
    return (
      <div>
        <div className = 'App'><h2>Welcome to React Application</h2></div>
        <Person name = {name} age = {age}/>
      </div>
    )
  }
}
```

Outptut



Welcome to React Application

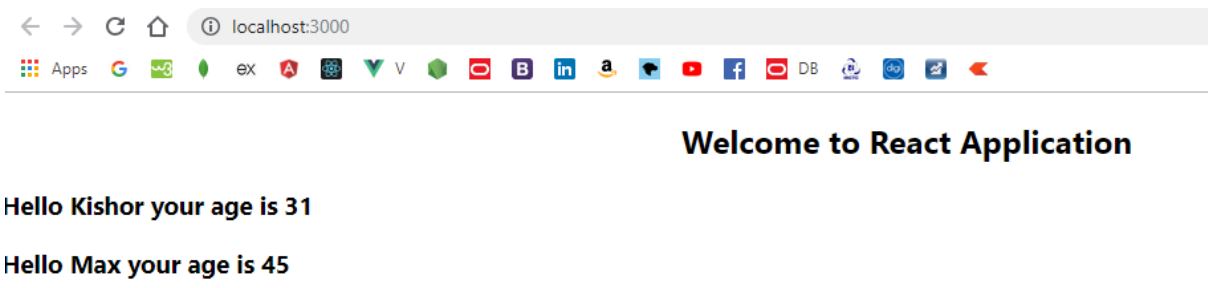
Hello Kishor your age is 31

Including <Person > multiple times

```
class App extends Component {
  render() {
    let name = "Kishor";
    let age = 31;
    return (
      <div>
```

```
        <div className = 'App'><h2>Welcome to React Application</h2></div>
        <Person name = {name} age = {age}/>
        <Person name = "Max" age = "45"/>
      </div>
    )
  }
}
```

Output


localhost:3000

Welcome to React Application

Hello Kishor your age is 31

Hello Max your age is 45

Exporting and Importing

If you have a JS file with multiple classes then you can have one class has default export that can be imported by any name in other javascript file

Say you have hello.js

export class A{}

export class B {}
class C{}
export default C;

Now you can import A, B by its name but C can be imported in any name
import Test from './hello';
import {A, B} from './hello';

Here Test means class C

Extracting Components

When you split components into smaller components and implement them independently, the components will not have any idea that it will be rendered inside which component so you can have props in more generic way.

example, assume you have comment component in social media which has to display user information, profile picture, comment and comment date then all these components can be implemented independently and render them in any component you want.

Since comment has to display user picture, information, comment and date, comment component will reuse UserDetails which will have picture and information component as below:

Assume you have a following comment in hard coded way where tommy and katty are users commented Hi, on a particular date

App.js

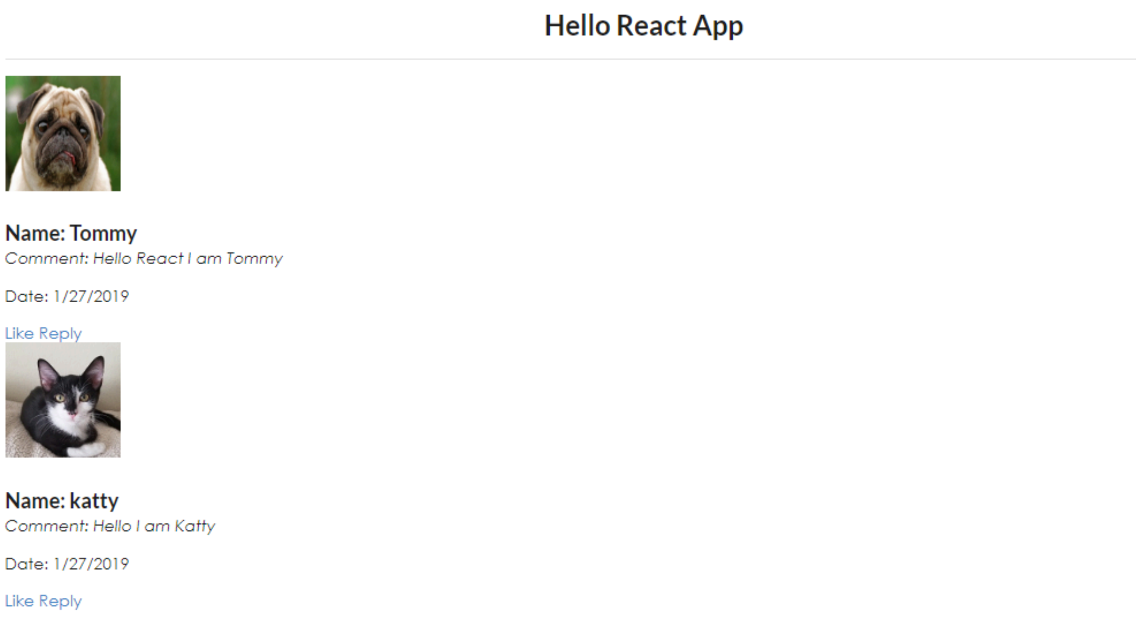
```
import React, { Component } from 'react';
import './App.css';
import Comment from './Comment'


class App extends Component {
  render() {
    let comment1 = {
      profile :
{author:"Tommy",imageURL:"https://www.petmd.com/sites/default/files/Acute-Dog-Diarrhea-47066074.jpg"},
      text : "Hello React I am Tommy",
      date : new Date().toLocaleDateString()
    }
    let comment2 = {
      profile : {author:"katty",imageURL:"https://pet-uploads.adoptapet.com/1/b/9/362017249.jpg"},
      text : "Hello I am Katty",
      date : new Date().toLocaleDateString()
    }
    return (
      <div class = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <Comment comment = {comment1}></Comment>
        <Comment comment = {comment2}></Comment>
      </div>
    );
  }
}

export default App;
```

Comment.js

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

```
export class UserImage extends Component {
    render() {
        return(
            <div>
                <img src = {this.props.imageURL}
                    width = {this.props.width}
                    height = {this.props.height}></img>
            </div>
        )
    }
}

export class UserProfile extends Component {
    render() {
        return(
            <div>
                <UserImage width = "100" height = "100"
                imageURL = {this.props.profile.imageURL}></UserImage>
                <h3>Name: {this.props.profile.author}</h3>
            </div>
        )
    }
}

class Comment extends Component {
    render() {
        return(
            <div>
                <UserProfile profile = {this.props.comment.profile}/>
                <p><i>Comment: {this.props.comment.text}</i></p>
                <p>Date: {this.props.comment.date}</p>
                <a href = "#">Like</a> <a href = "#">Reply</a>
            </div>
        )
    }
}
export default Comment;
```

Output



Hello React App

Name: Tommy

Comment: Hello React I am Tommy

Date: 1/27/2019

Like Reply

Name: katty

Comment: Hello I am Katty

Date: 1/27/2019

Like Reply

Looping the Components from another components

Assume you already have a component that renders Employee as below:

```
class Employee extends Component {
  render() {
    return(<div>
      <p>
        Hello, {this.props.employee.name} your id is {this.props.employee.id} and
        salary is {this.props.employee.salary}
      </p>
    </div>);
  }
}
```

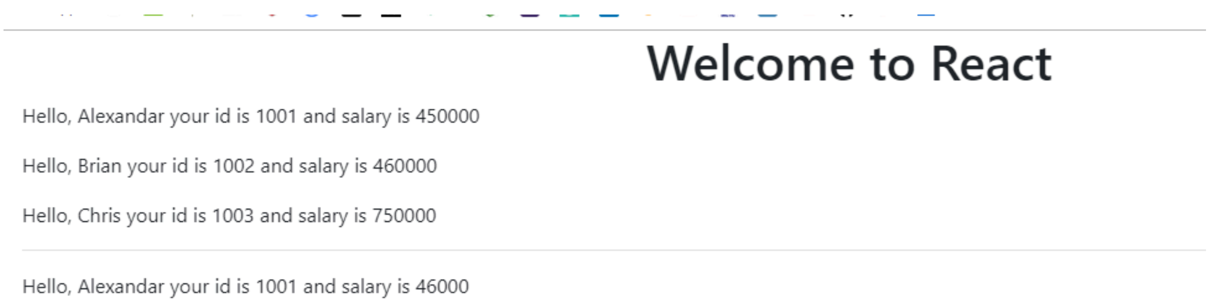
You want this to be re-used by another component that takes multiple employee objects to this components as below

```
class EmployeeList extends Component {
  render() {
    let employeeList = this.props.employees;
    return (<div>
      {employeeList.map((employee, i) => <Employee key = {i} employee = {employee}></Employee>)}
    </div>);
  }
}
```

Then your Root component can render this EmployeeList by passing the array of employee objects as below:

```
class App extends Component {
  render() {
    let employee = {id : 1001, name : "Alexandar", salary : 46000};
    let employees = [
      {id : 1001, name : "Alexandar", salary : 450000},
      {id : 1002, name : "Brian", salary : 460000},
      {id : 1003, name : "Chris", salary : 750000}
    ]
    return (
      <div className = "container-fluid">
        <h1 className = "text-center">Welcome to React</h1>
        <EmployeeList employees = {employees}></EmployeeList>
        <hr/>
        <Employee employee = {employee}></Employee>
      </div>
    );
  }
}
```

Now you can see below output



Welcome to React

Hello, Alexandar your id is 1001 and salary is 450000

Hello, Brian your id is 1002 and salary is 460000

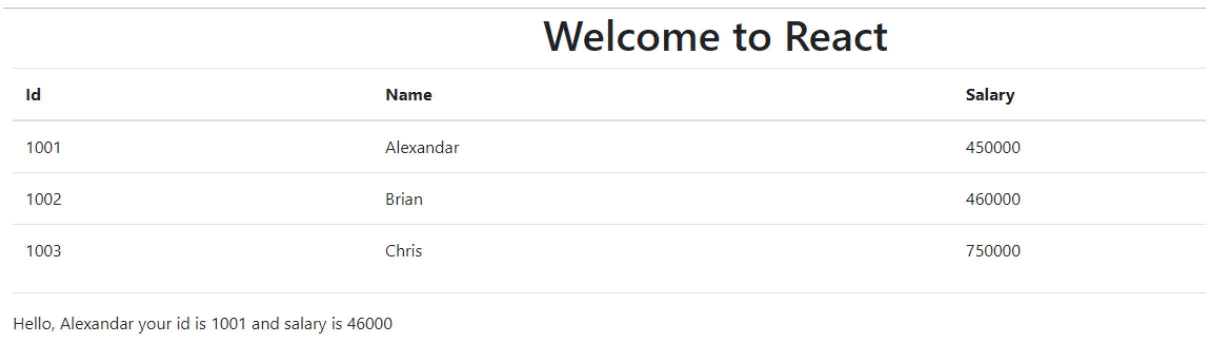
Hello, Chris your id is 1003 and salary is 750000

Hello, Alexandar your id is 1001 and salary is 46000

You can use table also as below:

```
class EmployeeList extends Component {
  render() {
    let employeeList = this.props.employees;
    return (<div>
      <table className = "table">
        <thead>
          <tr><th>Id</th><th>Name</th><th>Salary</th></tr>
        </thead>
        <tbody>
          {employeeList.map((employee, i) => <tr key = {i}>
            <td>{employee.id}</td><td>{employee.name}</td><td>{employee.salary}</td>
          </tr>)}
        </tbody>

      </table>
      {/* {employeeList.map((employee, i) => <Employee key = {i} employee = {employee}></Employee>)}
*/}
    </div>);
  }
}
```

Output:



Welcome to React

Id	Name	Salary
1001	Alexandar	450000
1002	Brian	460000
1003	Chris	750000

Hello, Alexandar your id is 1001 and salary is 46000

Note: Props are ready-only, whether you declare a class or a function; it must never modify its own props, because you need take input and return the result for the same input, Since the component changes at runtime over time the changes is done on State not on Props, State allow React components to change their output over time in response to user actions, network response

Adding Bootstrap or Semantic UI with React

Use the below command to install Bootstrap

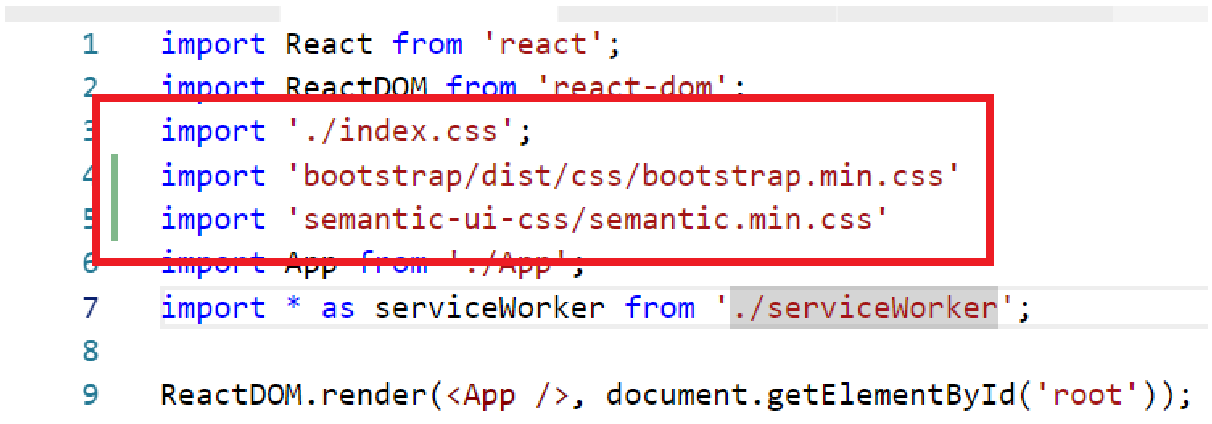
npm install --save bootstrap

Use the below command to install Semantic

npm install semantic-ui-react --save

npm install semantic-ui-css --save

Import the CSS either bootstrap or semantic any one you can import in the index.js as below



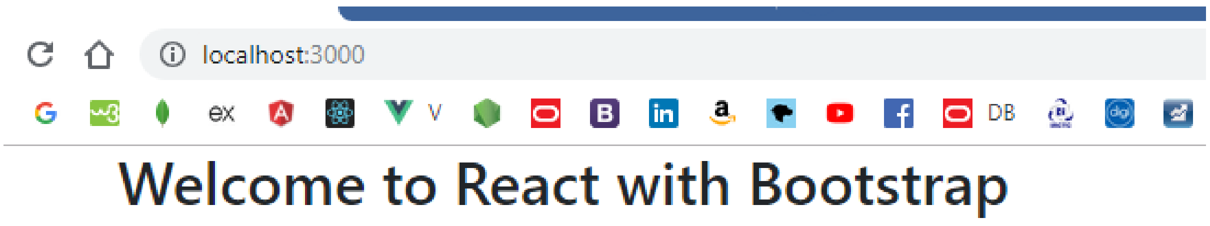
```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import 'bootstrap/dist/css/bootstrap.min.css'
import 'semantic-ui-css/semantic.min.css'
import App from './App';
import * as serviceWorker from './serviceWorker';

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

You can add other CSS imports after bootstrap so that they can take preference over bootstrap styles, add any one styles either bootstrap or semantic using both may not be a good idea.

Add className in your component as below

```
return (
      <div className = "container">
        <h2>Welcome to React with Bootstrap</h2>
      </div>
    );
```

Above code is written in the main component that is rendered in the root node, below is the output that shows content got css style of bootstrap.



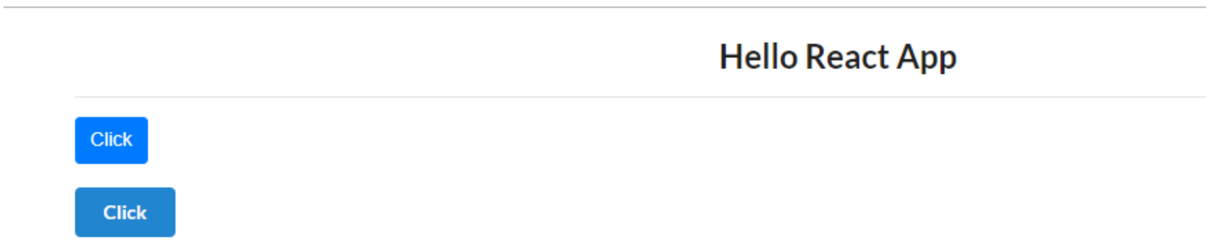
Welcome to React with Bootstrap

Adding bootstrap and semantic for a button

App.js

```
class App extends Component {
  render() {
    return (
      <div class = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <button className = "btn btn-primary">Click</button>
        <br />
        <br />
        <button className = "ui primary button">Click</button>
      </div>
    );
  }
}

export default App;
```

Output



Instead of adding classes for button semantic-ui-react gives you Button component that can also be used as below

You need to import the Button as below

```
import {Button} from 'semantic-ui-react'
```

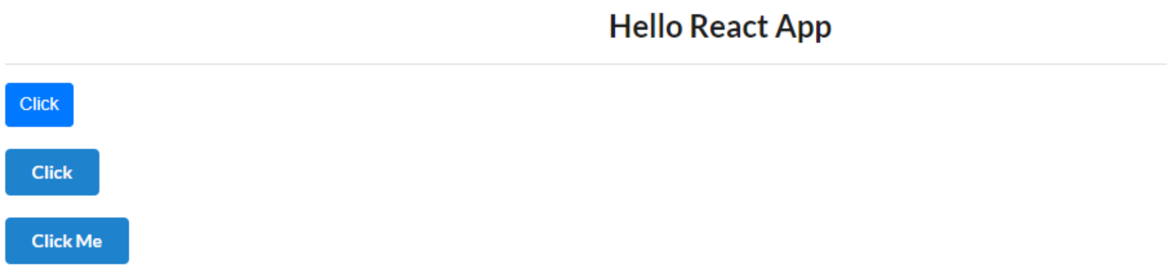
Add Button component in any other component as below

```
<Button primary>Click Me</Button>
```

App.js

```
import React, { Component } from 'react';
import {Button} from 'semantic-ui-react'
import './App.css';

class App extends Component {
  render() {
    return (
      <div class = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <button className = "btn btn-primary">Click</button>
        <br />
        <br />
        <button className = "ui primary button">Click</button>
        <br />
        <br />
        <Button primary>Click Me</Button>
      </div>
    );
  }
}

export default App;
```

Output



Commenting inside render()

function would be with *{/*….. */}* characters

State and Lifecycle

State is similar to props but its private and controlled by the component, every component will have Local States

A state can have multiple properties and values like

state = {date: new Date(), name:"Alex",...}

You need to initialize inside constructor with this.state = {..} and any updates you do must be done with this.setState({....}) method.

Note: state of any component is asynchronous which means if the state gets any updates from the server then its automatically updates the view

Below example has a Clock component that updates the date itself using state

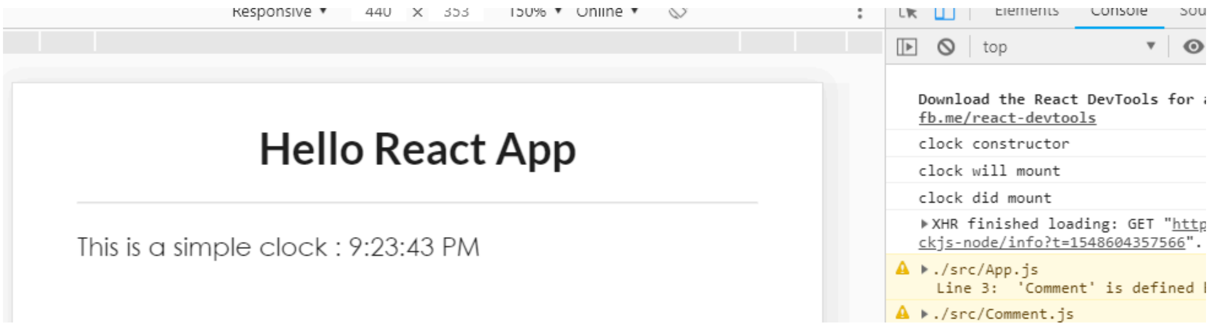
Clock.js

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

class Clock extends Component {
    constructor(props) {
        console.log('clock constructor')
        super(props);
        this.state = {date:new Date()}

    }
    updateTime() {
        this.setState({
            date:new Date()
        })
    }
    componentWillMount() {
        console.log('clock will mount')
    }
    componentDidMount() {
        console.log('clock did mount')
        setInterval(() => this.updateTime(), 1000);
    }

    render() {
        return(
            <div>
                This is a simple clock : {this.state.date.toLocaleTimeString()}
            </div>
        )
    }
}

export default Clock;
```

App.js

```
import React, { Component } from 'react';
import './App.css';
import Comment from './Comment'
import Clock from './Clock'

class App extends Component {
  render() {

    return (
      <div className = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <Clock />
      </div>
    );
  }
}

export default App;
```

Output



Hello React App

This is a simple clock : 9:23:43 PM

```
Download the React DevTools for a
fb.me/react-devtools
clock constructor
clock will mount
clock did mount
▸XHR finished loading: GET "http
ckjs-node/info?t=1548604357566".
⚠ ▸./src/App.js
   Line 3:  'Comment' is defined 
⚠ ▸./src/Comment.js
```

The above code will also show the way VDOM works where only the Clock node will refresh instead of the whole DOM node.

Component Life cycle

Component has below lifecycle methods

componentDidMount()
componentWillMount()
componentWillUnmount()
componentWillUpdate()

when you want to fetch all the posts and comments for the particular news feed component then componentWillMount() can have calling the fetch posts and comments that updates the state of the components

state will only update the properties that is changed by leaving all other properties intact.

i.e.,

constructor(props) { super(props);

this.user = “Alex”;
this.age = 35;
}

change() {
  this.setState({this.age = 36; });
}

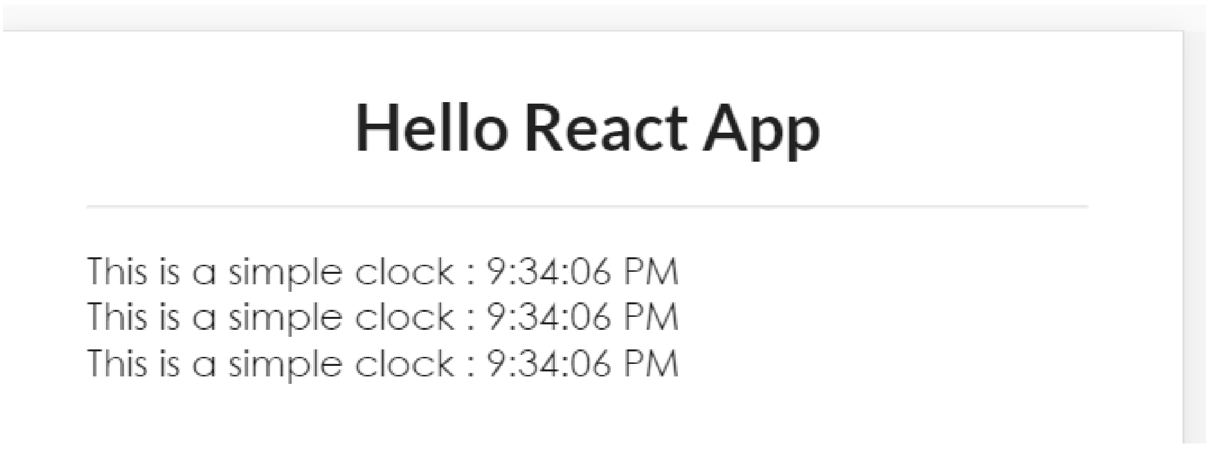
When the change() is called it updates only age by leaving user value same i.e., ‘Alex’;

Components are isolated

Every component will have their own state and multiple components of same name will maintain separate states, to see that you can render <Clock /> multiple times and see each Clock sets up its own timer and updates independently.

```
class App extends Component {
  render() {

    return (
      <div className = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <Clock />
        <Clock />
        <Clock />
      </div>
    );
  }
}
```

Output



Hello React App

This is a simple clock : 9:34:06 PM
This is a simple clock : 9:34:06 PM
This is a simple clock : 9:34:06 PM

Controlled Components (Forms), Props, States, Events

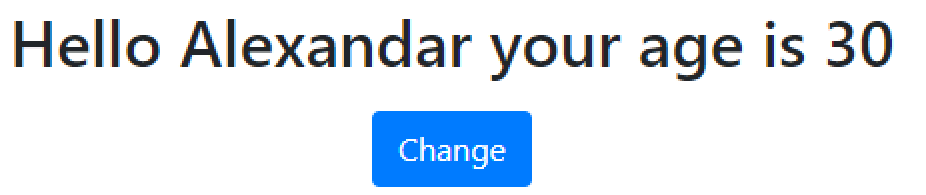
Handling Events

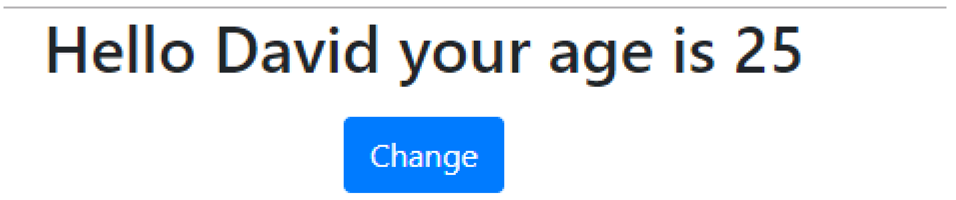
Changing the name from Alexandar to David by clicking

App.js

```
import React, { Component } from 'react';
import {FormControl} from './FormControl';

import logo from './logo.svg';
import './App.css';


class App extends Component {
  constructor(props) {
    super(props);
    this.handleClick = this.handleClick.bind(this)
  }
  state = {
    name : "Alexandar", age : 30
  }
  handleClick(event) {
    this.setState({
      name : "David", age : 25
    });
  }
  render() {

    return (
      <div className = 'App'>
        <h2>Hello {this.state.name} your age is {this.state.age}</h2>
        <p></p>
        <button onClick = {this.handleClick} className = 'btn btn-primary'>Change</button>
      </div>
    )
  }
}

export default App;
```

Output


Hello Alexandar your age is 30


Hello David your age is 25

Instead of binding the event to use the this keyword in a call back function you can assign arrow function to the function name so that binding is not required

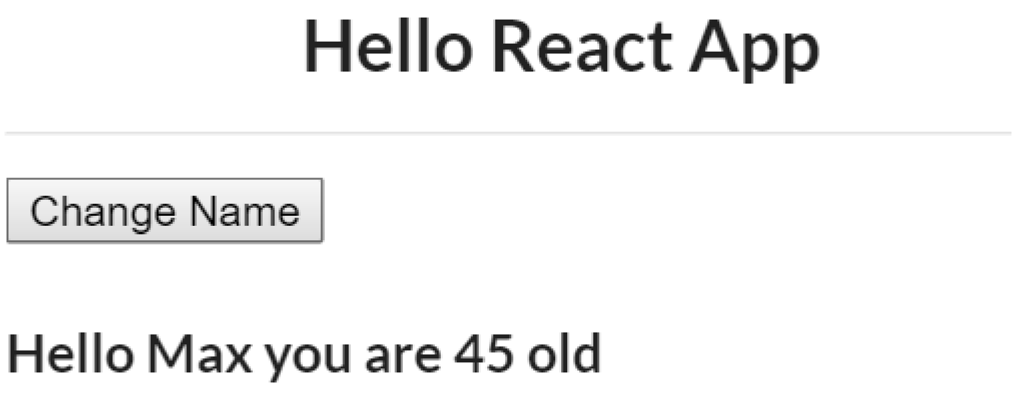
Button.js

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

export class Button extends Component{
    constructor(props) {
        super(props);
        this.state = {
            name:'Alex', age : 35
        }
    }
    handleClick = ()  => {
        this.setState({
            name:'Max', age : 45
        });
    }
    render() {
        return (
            <div>
                <button onClick = {this.handleClick}>Change Name</button>
                <h3>Hello {this.state.name} you are {this.state.age} old</h3>
            </div>

        )
    }
}
```

App.js

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

class App extends Component {
  render() {

    return (
      <div className = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <Button />
      </div>
    );
  }
}

export default App;
```

Output



Hello React App

Hello Max you are 45 old

Reusing Button names with props and state

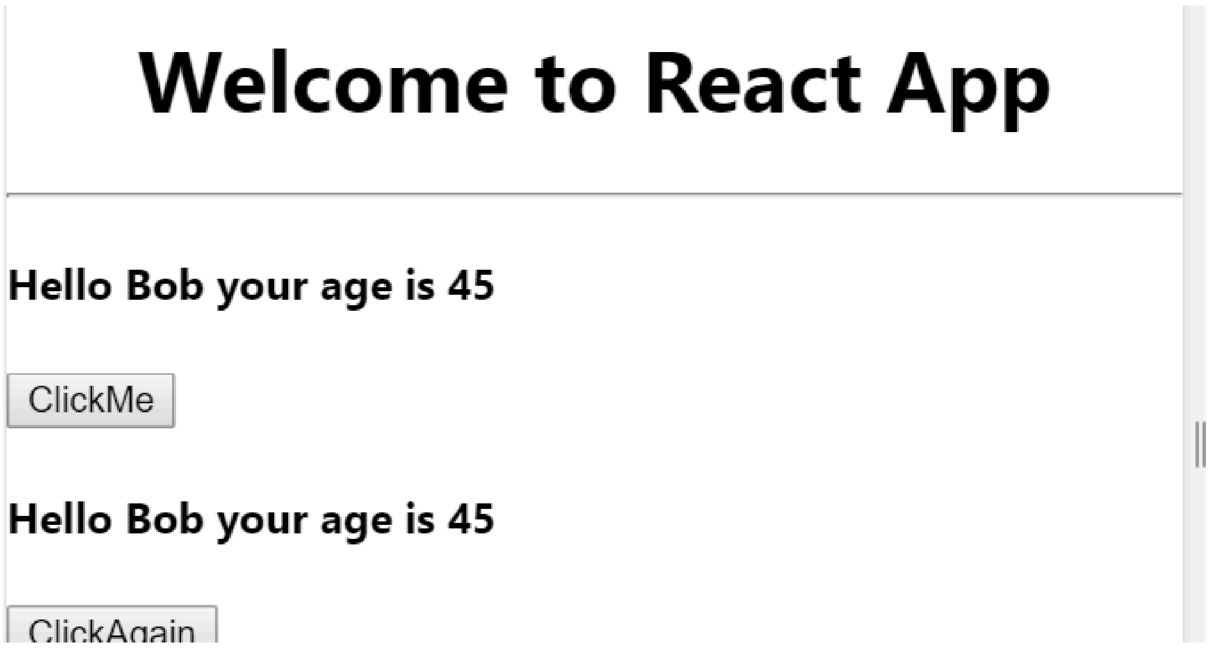
ButtonDemo.js

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

class ButtonDemo extends Component {
    constructor(props) {
        super(props);
        this.state= {
            name:'Alexandar', age:35
        }
        this.handleClick = this.handleClick.bind(this);
    }


    handleClick(event) {
       this.setState({
           name:'Bob', age:45
       })
    }
    render() {
        return <div>
            <h4>Hello {this.state.name} your age is {this.state.age}</h4>
            <button onClick = {this.handleClick}>{this.props.name}</button>
        </div>
    }
}
export default ButtonDemo;
```

App.js

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



class App extends Component {

  render() {

    return (
     <div>
       <div className = 'App'><h1>Welcome to React App</h1></div>
       <hr/>
       <ButtonDemo name = 'ClickMe'/>
       <ButtonDemo name = 'ClickAgain'/>
     </div>
    )
  }
}

export default App;
```

Output



Welcome to React App

Hello Bob your age is 45

Hello Bob your age is 45

Toggling the button Like & Unlike

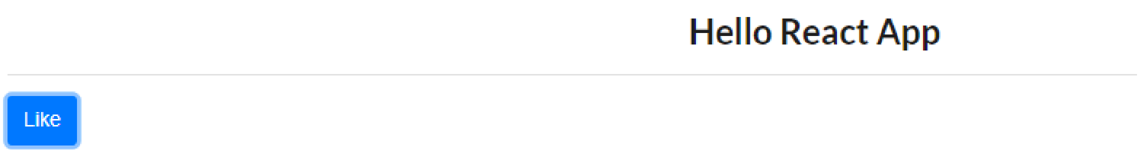
Button.js

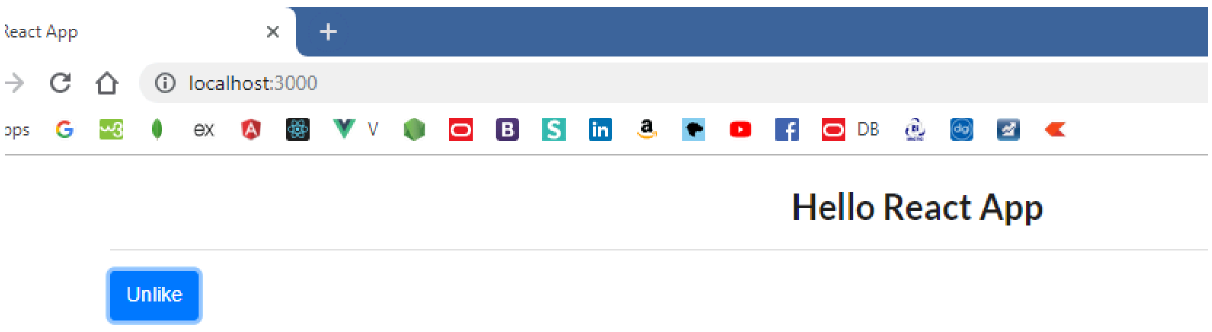
```
import React, {Component} from 'react';

export class Toggle extends Component {
   constructor(props) {
       super(props);
       this.state = {
           toggle:true
       }
       this.handleClick = this.handleClick.bind(this);
   }
   handleClick() {
    if(this.state.toggle) {
        this.setState({toggle:false})
    } else {
        this.setState({toggle:true})
    }
   }
    render() {
        return (
            <div>
                <button className = 'btn btn-primary'
                onClick = {this.handleClick}>{this.state.toggle?'Like':'Unlike'}</button>
            </div>
        )
    }
}
```

App.js

```
import React, { Component } from 'react';
import './App.css';
import {Button, ActionLink, Toggle} from './Button'

class App extends Component {
  render() {

    return (
      <div className = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />
        <Button />
        <ActionLink />
        <Toggle />
      </div>
    );
  }
}

export default App;
```

Output





You can create Like & Unlike buttons separately and have counters for each but you need to modify state by taking previous state as below:

this.setState((prevState) => ({
  likes : prevState + 1

}))

Creating a RegistrationForm and Adding to the main component

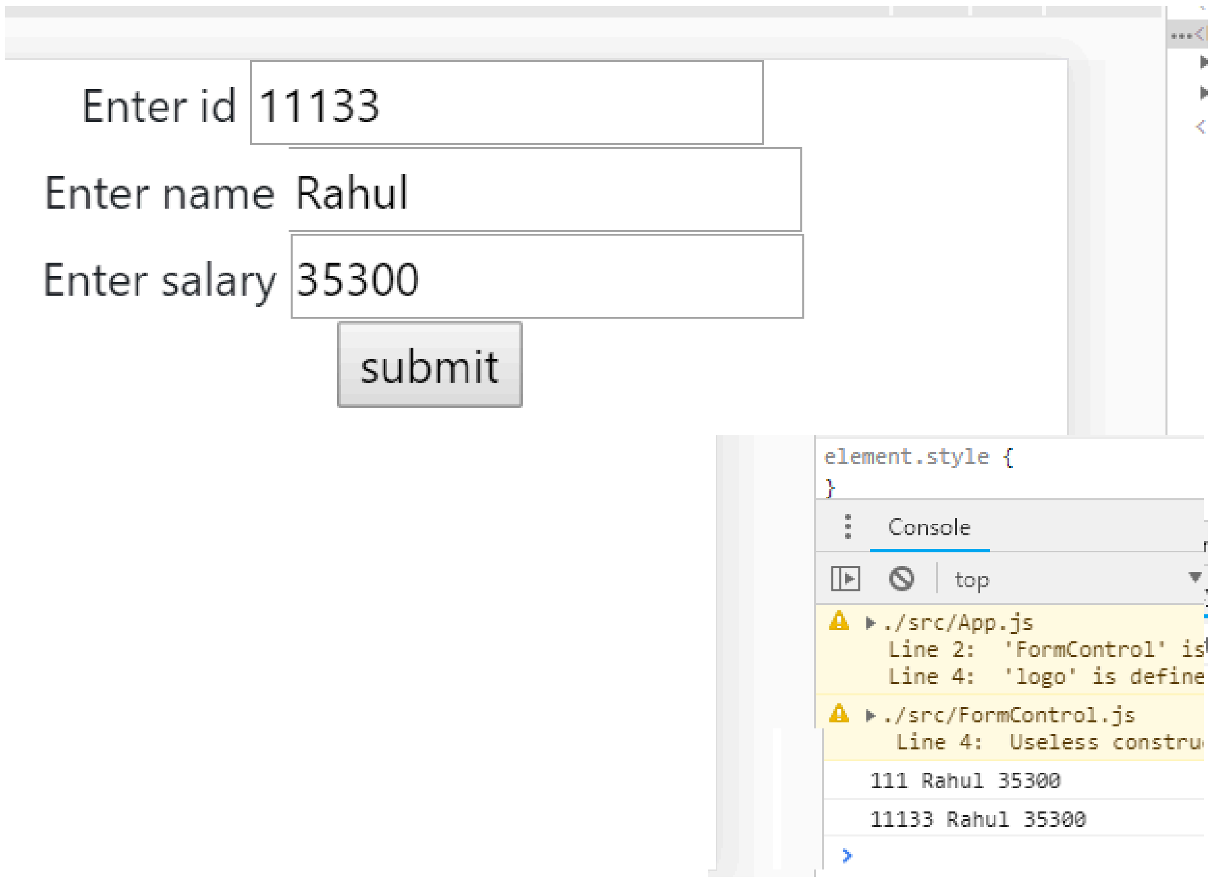
Registration Form class

```
class RegistrationForm extends Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.setState({
      _id:'',
      name:'',
      salary:''
    })
  }
  handleChange(event) {
    let tempName = event.target.name;
    let tempValue = event.target.value;
    this.setState({
      [tempName] : tempValue
    });
  }
  handleSubmit(event) {
    console.log(this.state._id, this.state.name, this.state.salary);
    event.preventDefault();
  }
  render() {
    return (
      <div>
        <form onSubmit = {this.handleSubmit}>
          Enter id <input type = 'text' name = '_id' onChange = 
{this.handleChange}></input> <br></br>
          Enter name <input type = 'text' name = 'name'  onChange = 
{this.handleChange}></input> <br></br>
          Enter salary <input type = 'text' name = 'salary'  onChange = 
{this.handleChange}></input> <br></br>
          <input type = "submit" value = "submit"></input>
        </form>
      </div>

    )
  }
}
```

App class

```
class App extends Component {
  constructor(props) {
    super(props);
    this.handleClick = this.handleClick.bind(this);
    this.handleChange = this.handleChange.bind(this);
  }
  state = {
    name : "Alexandar", age : 30
  }
  handleClick(event) {
    this.setState({

    })
  }
  handleChange(event) {
    let inputName = event.target.name;
    let inputValue = event.target.value;
    this.setState({
      [inputName] : inputValue
    });
  }
  render() {

    return (
      <div className = 'App'>
        <RegistrationForm/>
      </div>
    )
  }
}
```

Output



Enter id 11133

Enter name Rahul

Enter salary 35300

```
element.style {
}
```

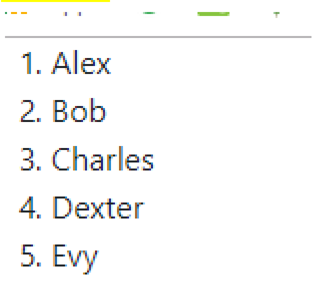
```
▸ ./src/App.js
   Line 2:  'FormControl' is
   Line 4:  'logo' is define
▸ ./src/FormControl.js
   Line 4:  Useless constru
111 Rahul 35300
11133 Rahul 35300
```

Lists and Keys

Iterating over array using map function

```
class ArrayItems extends Component {

  render() {
    let items = ["Alex", "Bob", "Charles", "Dexter", "Evy"]
    return (
     <ol>
      {items.map(i => <li>{i}</li>)}
     </ol>
    )
  }
}

class App extends Component {

  render() {

    return (
     <div>

       <ArrayItems />
     </div>
    )
  }
}
```

Output



```
1. Alex
2. Bob
3. Charles
4. Dexter
5. Evy
```

You will get a warning to use unique key while creating list of items this error is because React can perform an update to the particular item based on key thus React can handle minimal DOM change.

Keys help react identify which items have changed or added or removed, keys should be given to the elements inside the array to give the element a stable identity.

The best way to use the key is to pick up the value that uniquely identifies a list item, like id, if unique items are not present then you can use item index as a key.

Iterating over employee array using <table>

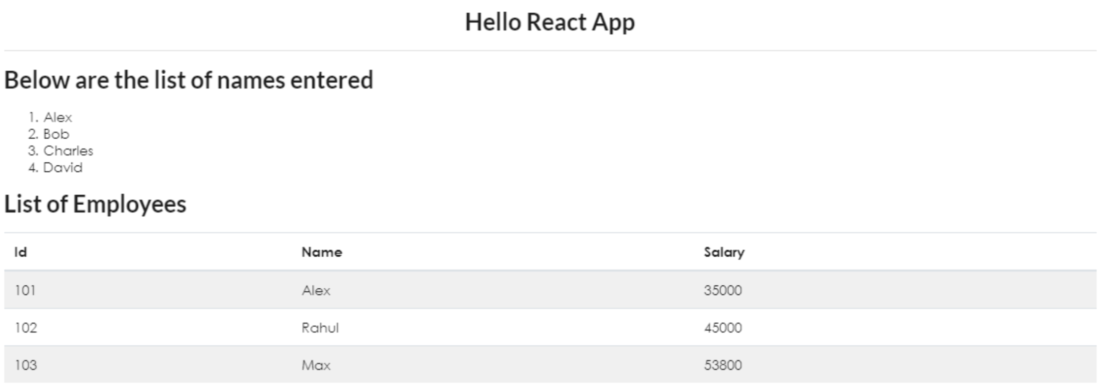
ArrayItems.js

```
export class EmployeeItems extends Component {
    render() {
        let employees = this.props.employees;
        return(
            <div>
                <h2>List of Employees</h2>
                <table className = 'table table-striped'>
                    <thead>
                        <tr>
                            <th>Id</th><th>Name</th><th>Salary</th>
                        </tr>
                    </thead>
                    <tbody>
                        {
                            employees.map(employee=><tr key = {employee.id}>
                                <td>{employee.id}</td>
                                <td>{employee.name}</td>
                                <td>{employee.salary}</td>
                            </tr>)
                        }
                    </tbody>
                </table>
            </div>
        )
    }
}
```

App.js

```
import React, { Component } from 'react';
import './App.css';
import {Button, ActionLink, Toggle} from './Button'
import {ArrayItem, EmployeeItems} from './ArrayItem'

class App extends Component {
  render() {
    let employees = [
      {id:101, name:"Alex", salary:35000},
      {id:102, name:"Rahul", salary:45000},
      {id:103, name:"Max", salary:53800}
    ]
    return (
      <div className = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />

        <ArrayItem />
        <EmployeeItems employees = {employees} />
      </div>
    );
  }
}

export default App;
```

Output



Hello React App

Below are the list of names entered

1. Alex
2. Bob
3. Charles
4. David

List of Employees

Id	Name	Salary
101	Alex	35000
102	Rahul	45000
103	Max	53800

Creating a form again with submit and change event handling

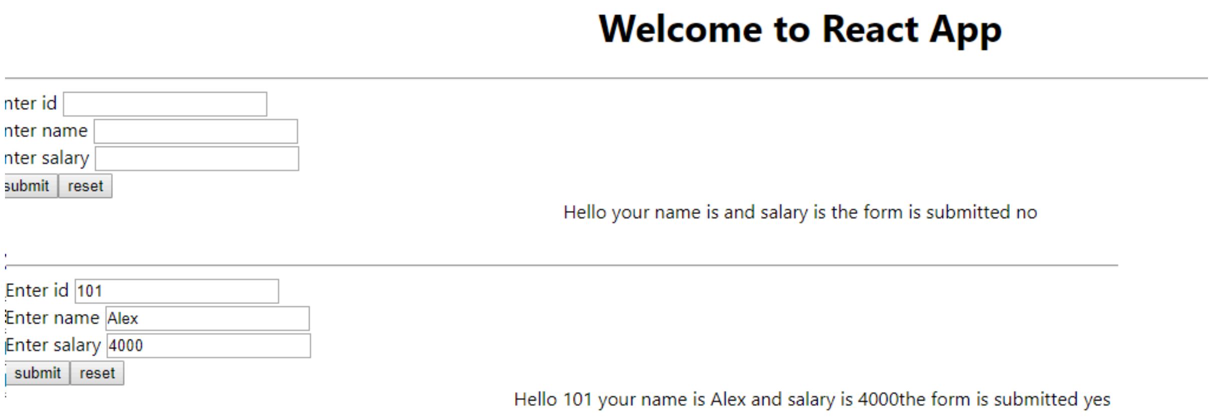
StoreForm.js

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


class StoreForm extends Component {
    constructor(props) {
        super(props);
        this.state = {
            _id: 0, name:'', salary:0, isSubmitted : 'no'
        }
        this.handleSubmit = this.handleSubmit.bind(this);
        this.handleChange = this.handleChange.bind(this);
        this.handleClick = this.handleClick.bind(this);
    }
    handleChange(event) {
        let tempName = event.target.name;
        let tempValue = event.target.value;
        this.setState({
            [tempName]  : tempValue
        });
    }
    handleSubmit(event) {
        this.setState({
            isSubmitted:'yes'
        })
        event.preventDefault();
    }
    handleClick(event) {
        this.setState({
            _id:'',name:'', salary:'',isSubmitted:'no'
        })
    }

    render() {
        return (
            <div>
            <form onSubmit = {this.handleSubmit}>
              Enter id <input type = 'text' name = '_id' onChange =
{this.handleChange}></input> <br></br>
              Enter name <input type = 'text' name = 'name'  onChange =
{this.handleChange}></input> <br></br>
              Enter salary <input type = 'text' name = 'salary'  onChange =
{this.handleChange}></input> <br></br>
              <input type = "submit" value = "submit"></input>
              <input type = "reset" value = "reset" onClick = {this.handleClick}></input>
            </form>
            <div className ='App'>
                Hello {this.state._id} your name is {this.state.name} and salary is
{this.state.salary}
                the form is submitted {this.state.isSubmitted}
            </div>
          </div>
        )
    }

}

export default StoreForm;
```

App.js

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



class App extends Component {

  render() {

    return (
     <div>
       <div className = 'App'><h1>Welcome to React App</h1></div>
       <hr/>
      <StoreForm />
       <ArrayItems title = "Collection of Employees"/>

     </div>
    )
  }
}

export default App;
```

Output

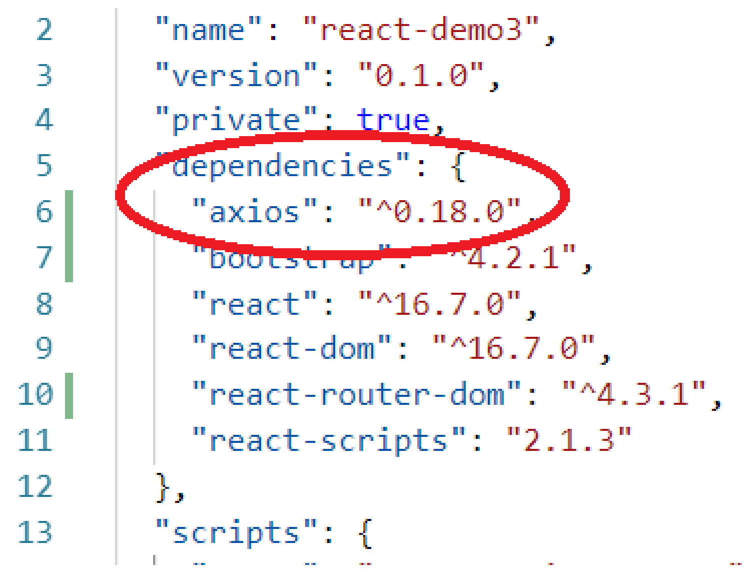


```
Welcome to React App

nter id
nter name
nter salary
submit  reset
Hello your name is and salary is the form is submitted no

Enter id 101
Enter name Alex
Enter salary 4000
submit  reset
Hello 101 your name is Alex and salary is 4000the form is submitted yes
```

Now we will interact with MongoDB through express js

In React we will use axios library to call the webservices which provides us all http methods, you must install axios by using npm command npm install axios --save

To verify the download check package.json that will have the below entries



```
  "name": "react-demo3",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "axios": "^0.18.0",
    "bootstrap": "^4.2.1",
    "react": "^16.7.0",
    "react-dom": "^16.7.0",
    "react-router-dom": "^4.3.1",
    "react-scripts": "2.1.3"
  },
  "scripts": {
```

You can import axios and call methods like put, get, post, delete and etc.

import axios from ‘axios’;

....

axios.post(url, data).then(response => {...});

axios.get(url).then(response => {..});

StoreForm.js

```
import React, {Component} from 'react';
import axios from 'axios'

class StoreForm extends Component {
    constructor(props) {
        super(props);
        this.state = {
            _id: 0, name:'', salary:0, isSubmitted : 'no', responseStatus : 0
        }
        this.handleSubmit = this.handleSubmit.bind(this);
        this.handleChange = this.handleChange.bind(this);
        this.handleClick = this.handleClick.bind(this);
    }
    handleChange(event) {
        let tempName = event.target.name;
        let tempValue = event.target.value;
        this.setState({
            [tempName]  : tempValue
        });
    }
    handleSubmit(event) {
        this.setState({
            isSubmitted:'yes'
        })

        let url = "http://localhost:3001/emp";
        let id = parseInt(this.state._id);
        let name = this.state.name;
        let salary = parseFloat(this.state.salary);
        let empJSON = {"_id":id, "name":name, "salary":salary};
        axios.post(url, empJSON).then(response => {
            if(response.ok != undefined) {

            this.setState({
                responseStatus : response.status
            })
            }
            else {
                this.setState({
                    responseStatus : 500
                })
            }
        })
        event.preventDefault();
    }
    handleClick(event) {
        this.setState({
            _id:'',name:'', salary:'',isSubmitted:'no'
        })
    }

    render() {
        return (
            <div>
            <form onSubmit = {this.handleSubmit}>
              Enter id <input type = 'text' name = '_id' onChange = {this.handleChange}></input>
<br></br>
              Enter name <input type = 'text' name = 'name'  onChange = {this.handleChange}></input>
<br></br>
              Enter salary <input type = 'text' name = 'salary'  onChange =
{this.handleChange}></input> <br></br>
              <input type = "submit" value = "submit"></input>
              <input type = "reset" value = "reset" onClick = {this.handleClick}></input>
            </form>
            <div className ='App'>
                Hello {this.state._id} your name is {this.state.name} and salary is
{this.state.salary}
                the form is submitted {this.state.isSubmitted} and
                returned response is {this.state.responseStatus}
            </div>
          </div>
        )
    }
```

```
}

export default StoreForm;
```

App.js

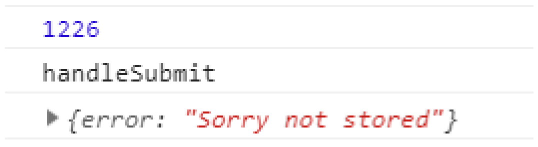
```
import React, { Component } from 'react';
import StoreForm from './StoreForm'
import ButtonDemo from './ButtonDemo';
import ArrayItems, {Employee} from './ArrayItems'
import logo from './logo.svg';
import './App.css';



class App extends Component {

  render() {

    return (
     <div>
       <div className = 'App'><h1>Welcome to React App</h1></div>
       <hr/>
      <StoreForm />
       <ArrayItems title = "Collection of Employees"/>

     </div>
    )
  }
}

export default App;
```

Express Module that has all the operations CRUD

express_demo.js

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const port = 3001;

var mongoClient = require("mongodb").MongoClient;
var url= "mongodb://localhost:27017";

// middleware to accept all the requests from client that has JSON format data
app.use(bodyParser.json());

// middleware to apply cross origin resource sharing filters
app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH,
DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    next();
});


// retrieve one record
app.get("/findEmp/:id", function(request, response){
    let empId = parseInt(request.params.id);
    mongoClient.connect(url, { useNewUrlParser: true }, function(err, client){
        let db = client.db("mydb");
        let docCursor = db.collection("Employee").find({_id:empId});
        var x = 0;
        docCursor.forEach(function(doc, err){
            x++;
            if( x > 0) {
                response.json(doc);
            }

        }, function(){
            if(x == 0) {
                response.send("Sorry no record found with an id "+empId)
                client.close();
            }

        });
    })
})

// authenticate
app.post("/login/:un/:pwd", function(request, response){
    let empId = parseInt(request.params.un);
    let pass = request.params.pwd;
    mongoClient.connect(url, {useNewUrlParser:true}, function(err, client){
        let db = client.db("mydb");
        let docCursor = db.collection("Employee")
        .find({$and:[{_id:empId}, {name:pass}]});
        let counter = 0;
        docCursor.forEach(function(doc, err){
            counter++;
            if(counter > 0) {
                response.json(doc);
            }

        }, function(){
            if(counter == 0) {
                response.json({"error":"No data found"});
                client.close();
            }
        });
    });
});

// retrieve all records
app.get("/getEmp",function(req,res){
    let emp = [];
    console.log('getting all emps')
    mongoClient.connect(url, {useNewUrlParser:true}, function(err, client){
        let db = client.db("mydb");
        let cursor = db.collection("Employee").find();
        cursor.forEach(function(doc, err){
            emp.push(doc)
        }, function(){
            client.close();
            res.json(emp);
        })
    })


})


// store data
app.post("/emp", function(req, response)  {

    let content = req.body;
    mdbClient.connect(dbUrl, {useNewUrlParser:true}, function(err, client){

        let db = client.db("mydb");
        db.collection("Employee").insertOne(content, function(err, res){
            if(err) {
                response.json({error:"Sorry not stored"})
            } else {
                response.json(res.ops[0]._id);
            }
            client.close();
        });
    });

});
```

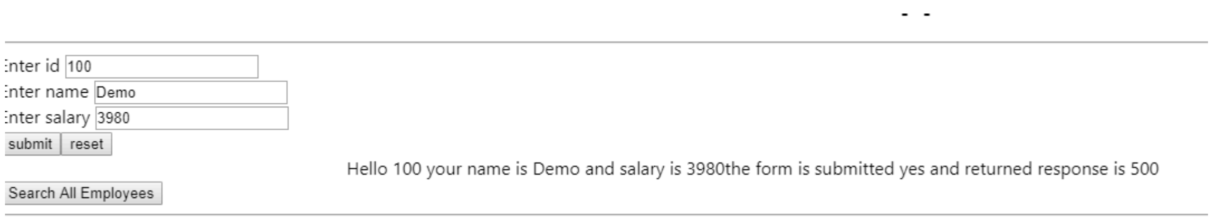
For store the console of browser would give the id inserted or error as below



```
1226
handleSubmit
▸ {error: "Sorry not stored"}
```

```
// update salary by id
app.put("/updateEmp/:id/:salary", function(request, response){
    let empId = parseInt(request.params.id)
    let salary2 = parseFloat(request.params.salary);

    let updateDoc = function(db, callback){
        db.collection("Employee")
        .updateOne({_id:empId}, {$set:{salary:salary2}}, function(err, result){
            callback();
            response.json(result);
        });
    }

    mongoClient.connect(url, {useNewUrlParser:true}, function(error, client){
        let db = client.db("mydb");
        updateDoc(db, function(){
            client.close();
        })
    })
})
// delete by id
app.delete("/deleteEmp/:id", function(request, response){
    let empId = parseInt(request.params.id);
    let deleteDoc = function(db, callback) {
        db.collection("Employee").deleteOne({_id:empId}, function(err, result){
            callback();
            response.json(result);
        })
    }
    mongoClient.connect(url, {useNewUrlParser:true}, function(err, client){
       let db = client.db("mydb");
       deleteDoc(db, function(){
           client.close();

       })
    });
});



app.listen(port,function(){
    console.log("Server is running on port number "+port)
});
```

Output



```
:nter id 100
:nter name Demo
:nter salary 3980
submit  reset
                                   Hello 100 your name is Demo and salary is 3980the form is submitted yes and returned response is 500
Search All Employees
```

FindAllEmps.js

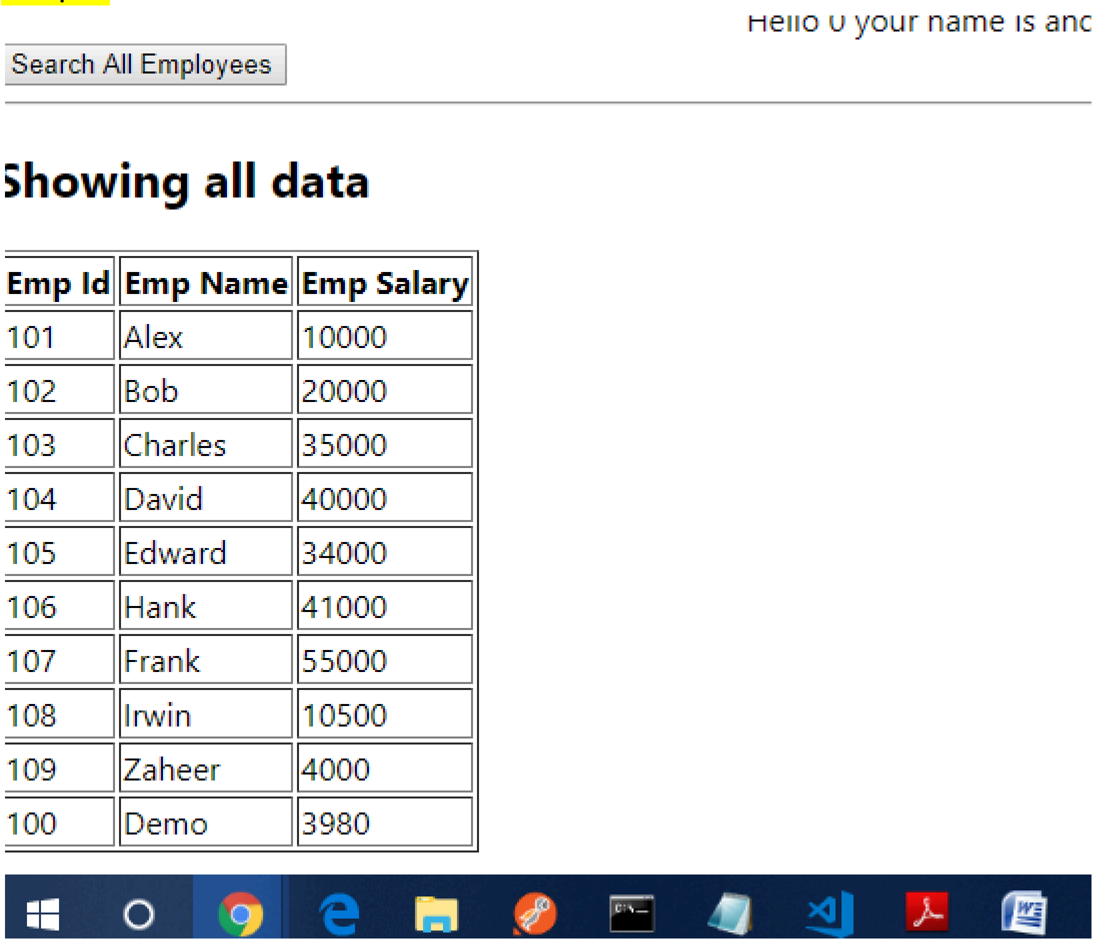
```
import React, {Component} from 'react';
import axios from 'axios'

class FindAllEmp extends Component {
    constructor(props) {
        super(props);
        this.state = {
            employees:[]
        }

        this.handleClick = this.handleClick.bind(this);
    }
    handleClick(event) {
        let url = "http://localhost:3001/getEmp";
        axios.get(url).then(response => {
            this.setState({
                employees: response.data
            })
        })
        event.preventDefault();
    }

    render() {
        return (
           <div>
               <button onClick = {this.handleClick}>Search All Employees</button>
               <hr/>
               <h2>Showing all data</h2>
               <div>
               <table border = '1'>
                <thead>
                    <tr>
                        <th>Emp Id</th><th>Emp Name</th><th>Emp Salary</th>
                    </tr>
                </thead>
                <tbody>
                    {
                        this.state.employees.map(employee => <tr key={employee._id}>
                            <td>{employee._id}</td><td>{employee.name}</td><td>{employee.salary}</td>
                        </tr>)
                    }
                </tbody>

               </table>

               </div>

           </div>
        );
    }
}

export default FindAllEmp;
```

App.js

```
import React, { Component } from 'react';
import StoreForm from './StoreForm'
import ButtonDemo from './ButtonDemo';
import ArrayItems, {Employee} from './ArrayItems'
import FindAllEmp from './FindAllEmp'
import logo from './logo.svg';
import './App.css';




class App extends Component {

  render() {

    return (
     <div>
       <div className = 'App'><h1>Welcome to React App</h1></div>
```

```
        <hr/>
       <StoreForm />
       <FindAllEmp />

      </div>
    )
  }
}

export default App;
```

Output



Hello u your name is anc

Search All Employees

Showing all data

Emp Id	Emp Name	Emp Salary
101	Alex	10000
102	Bob	20000
103	Charles	35000
104	David	40000
105	Edward	34000
106	Hank	41000
107	Frank	55000
108	Irwin	10500
109	Zaheer	4000
100	Demo	3980

UpdateForm.js

```
import React, {Component} from 'react';
import axios from 'axios'

class UpdateForm extends Component {
    constructor(props) {
        super(props);
        this.state = {
            _id: 0, salary:0, responseStatus : 0
        }
        this.handleSubmit = this.handleSubmit.bind(this);
        this.handleChange = this.handleChange.bind(this);
        this.handleClick = this.handleClick.bind(this);
    }
    handleChange(event) {
        let tempName = event.target.name;
        let tempValue = event.target.value;
        this.setState({
            [tempName]  : tempValue
        });
    }
    handleSubmit(event) {

        let id = parseInt(this.state._id);
        let salary = parseFloat(this.state.salary);
        let url = "http://localhost:3001/updateEmp/"+this.state._id+"/"+this.state.salary;

        axios.put(url).then(response => {
            console.log(response);
        });

        event.preventDefault();
    }
    handleClick(event) {
        this.setState({
            _id:'',  salary:''
        })
    }

    render() {
        return (
            <div>
            <form onSubmit = {this.handleSubmit}>
              Enter id <input type = 'text' name = '_id' onChange =
{this.handleChange}></input> <br></br>

              Enter salary <input type = 'text' name = 'salary'  onChange =
{this.handleChange}></input> <br></br>
              <input type = "submit" value = "Update"></input>
              <input type = "reset" value = "reset" onClick = {this.handleClick}></input>
            </form>

          </div>
        )
    }

}

export default UpdateForm;
```

App.js

```
import React, { Component } from 'react';
import StoreForm from './StoreForm'
import ButtonDemo from './ButtonDemo';
import ArrayItems, {Employee} from './ArrayItems'
import FindAllEmp from './FindAllEmp'
import logo from './logo.svg';
import './App.css';
import UpdateForm from './UpdateForm'



class App extends Component {

  render() {

    return (
     <div>
       <div className = 'App'><h1>Welcome to React App</h1></div>
       <hr/>
      <StoreForm />
      <hr></hr>
      <UpdateForm />
      <FindAllEmp />


     </div>
    )
  }
}

export default App;
```

React Router

npm install react-router-dom

change index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
//import 'bootstrap/dist/css/bootstrap.css'
import './index.css';
import App from './App';

import UpdateForm from './UpdateForm';
import StoreForm from './StoreForm'
import FindAllEmp from './FindAllEmp'

import {Route, BrowserRouter, Link} from 'react-router-dom'

import * as serviceWorker from './serviceWorker';




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



// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: http://bit.ly/CRA-PWA
serviceWorker.unregister();
```

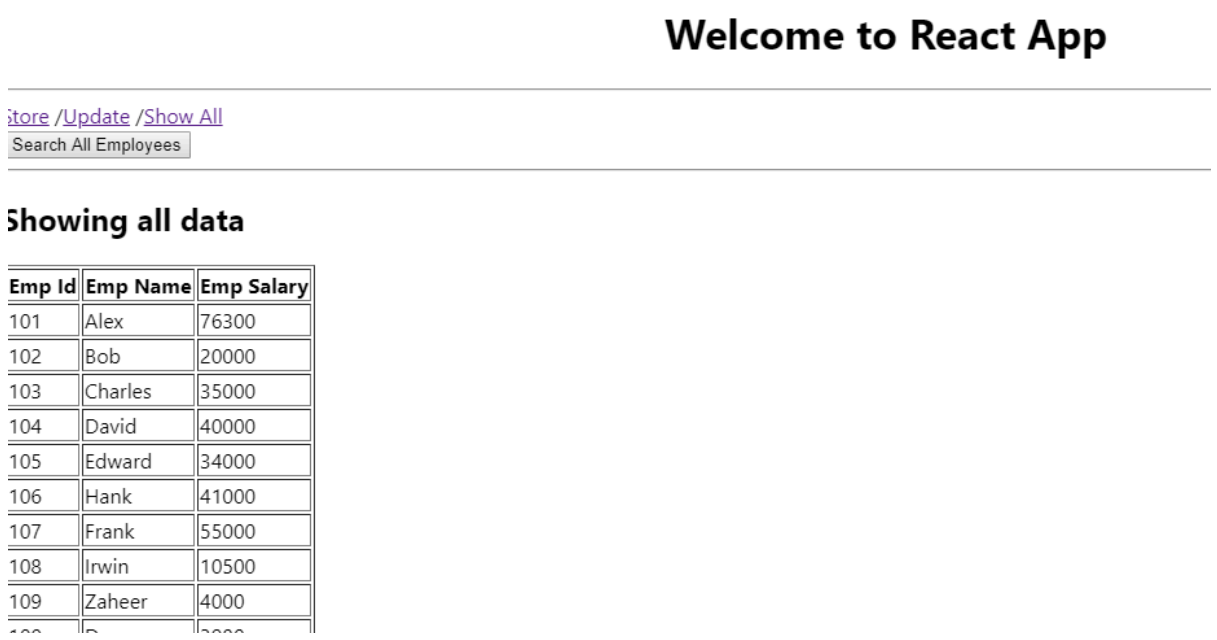
Create Links and Outlet

App.js

```
import React, { Component } from 'react';
import StoreForm from './StoreForm'
import ButtonDemo from './ButtonDemo';
import ArrayItems, {Employee} from './ArrayItems'
import FindAllEmp from './FindAllEmp'
import logo from './logo.svg';
import './App.css';
import UpdateForm from './UpdateForm'

import {Route, BrowserRouter, Link, Switch} from 'react-router-dom'


class Main extends Component {
  render() {
    return (
      <div>
        <Link to = "/store">Store</Link> /
        <Link to = "/update">Update</Link> /
        <Link to = "/findAll">Show All</Link>
      </div>
    )
  }
}

class AppRouter extends Component {
  render() {
    return (
      <div>
        <Switch>
          <Route path = "/store" component = {StoreForm}></Route>
          <Route path = "/update" component = {UpdateForm}></Route>
          <Route path = "/findAll" component = {FindAllEmp}></Route>
        </Switch>
      </div>
    )
  }
}



class App extends Component {

  render() {

    return (
     <div>
```

```
        <div className = 'App'><h1>Welcome to React App</h1></div>
        <hr/>
       <Main />
       <AppRouter />
      </div>
    )
  }
}

export default App;
```

Output



Welcome to React App

tore /Update /Show All

Search All Employees

howing all data

Emp Id	Emp Name	Emp Salary
101	Alex	76300
102	Bob	20000
103	Charles	35000
104	David	40000
105	Edward	34000
106	Hank	41000
107	Frank	55000
108	Irwin	10500
109	Zaheer	4000

Programatically you can navigate when router has path & component from a code

```
handleSubmit = (e) => {
        console.log('handleSubmit');
        e.preventDefault();
        this.props.history.push('/test');
    }
```

Another example on Router with App having <BrowserRouter>

App.js

```
import React, { Component } from 'react';
import './App.css';
import {Button, ActionLink, Toggle} from './Button'
import {ArrayItem, EmployeeItems} from './ArrayItem'
import { StoreForm, LoginForm, RetrieveUsers, SearchUserForm, Demo, UpdatePhoneNumberComponent } from
'./UserActions';
import {BrowserRouter, Switch, Route, Link} from 'react-router-dom'

class NavigationComponent extends Component {
  render() {
    return(
      <div>
        <Link to = "/register">Register</Link> |
        <Link to = "/login">Login</Link> |
        <Link to = "/update">Update</Link> |
        <Link to = "/search">Search User</Link> |
        <Link to = "/showAll">Show All</Link>

      </div>
    )
  }
}
class OutputComponent extends Component {
  render() {
    return(
      <div>
        <Switch>

          <Route path = "/register" component = {StoreForm} />
          <Route path = "/login" component = {LoginForm} />
          <Route path = "/update" component = {UpdatePhoneNumberComponent} />
          <Route path = "/search" component = {SearchUserForm} />
          <Route path = "/showAll" component = {RetrieveUsers} />
          <Route path = "/test" component = {Demo} />
        </Switch>
      </div>

    )
  }
}

class App extends Component {
  render() {

    return (
      <BrowserRouter>
        <div className = 'container'>
        <h2 className = 'text-center'>Hello React App</h2>
        <hr />

        <NavigationComponent />
        <OutputComponent />

      </div>
      </BrowserRouter>
    );
  }
}

export default App;
```

UserActions.js

```
import React, {Component} from 'react';
import axios from 'axios';

export class Demo extends Component {
    render() {
        return (<div>
            Home Component Test Demo!
        </div>)
    }
}
export class StoreForm extends Component {
    constructor(props) {
        super(props);
        this.state = {
            _id : null, name:"", password:"", gender: "", phoneNumber : null
        }
    }
    handleSubmit = (e) => {
        console.log('handleSubmit');
        e.preventDefault();
        let storeURL = "http://localhost:9999/user";
        let userData = {_id:parseInt(this.state._id), name:this.state.name,
            password:this.state.password,
            gender:this.state.gender,
            phoneNumber:parseInt(this.state.phoneNumber)
        };
        axios.post(storeURL, userData).then(res => {
            console.log(res.data);
        })
        this.props.history.push('/test');
    }
    handleChange = (e) => {
        this.setState({
            [e.target.name]: e.target.value
        })
    }
    render() {
        return(
            <div>
                <h2>User Registration Form</h2>
               <form onSubmit = {this.handleSubmit}>
                    <div className = "form-group">
                        <label>Enter User Id</label>
                        <input type = 'number' onChange = {this.handleChange} name = '_id' className =
'form-control'></input>
                    </div>
                    <div className = "form-group">
                        <label>Enter Name</label>
                        <input type = 'text' onChange = {this.handleChange} name = 'name' className =
'form-control'></input>
                    </div>
                    <div className = "form-group">
                        <label>Enter Password</label>
                        <input type = 'password' onChange = {this.handleChange} name = 'password'
className = 'form-control'></input>
                    </div>
                    <div className = "form-group">
                        <label>Select Gender: </label>
                        <label className = 'radio-inline'>
                        <input type = 'radio'  onChange = {this.handleChange} name = 'gender' value =
"Male"></input>
                        Male
                        </label>
                        <label className = 'radio-inline'>
                        <input type = 'radio'  onChange = {this.handleChange} name = 'gender' value =
"Female"></input>
                        Female
                        </label>
                    </div>

                    <div className = "form-group">
                        <label>Enter Phone Number</label>
                        <input type = 'text' onChange = {this.handleChange} name = 'phoneNumber'
className = 'form-control'></input>
                    </div>
                    <div className = 'form-group'>
                        <input type = 'submit' value = 'submit' className = 'btn btn-primary'></input>
                        <input type = 'reset' value = 'reset' className = 'btn btn-primary'></input>
                    </div>
               </form>
               <hr />

            </div>
        )
    }
}


export class UpdatePhoneNumberComponent extends Component {
    constructor(props) {
        super(props);
        this.state = {
            _id:null, phoneNumber : null, message : ""
        }

    }
    handleChange = (event) => {
        this.setState({
            [event.target.name] : event.target.value
        })
    }
    handleClick = (event) => {
        let id = parseInt(this.state._id);
        let ph = parseInt(this.state.phoneNumber);
        let updateURL = "http://localhost:9999/user/"+id+"/"+ph;
        axios.put(updateURL).then(res => {
            if(res.data.nModified == 1) {
                this.setState({message:"Updated Successfully"})
            } else {
                this.setState({message:"Not Updated"})
            }
            this.setState({_id:'', phoneNumber:''})
        })
        event.preventDefault()
    }
    render() {
        return(
            <div>
                <h2>Update PhoneNumber</h2>
                <form>
                    <div className ='form-group'>
                        <label>Enter Id</label>
                        <input value = {this.state._id} type = 'number' name = '_id' className =
'form-control' onChange = {this.handleChange} />
                    </div>
                    <div className = "form-group">
                        <label>Enter Phone Number</label>
                        <input value = {this.state.phoneNumber} type = 'text' onChange =
{this.handleChange} name = 'phoneNumber' className = 'form-control'></input>
                    </div>
                    <div className = 'form-group'>
                        <button className = 'btn btn-primary'
                        onClick = {this.handleClick}>Update</button>
                    </div>
                </form>
                <p>{this.state.message}</p>
            </div>
        )
    }
}

export class LoginForm extends Component {
    constructor(props) {
        super(props);
        this.state = {_id:null, password:""}
    }
    handleSubmit = (e) => {

        e.preventDefault();
    }
    handleChange = (e) => {
       this.setState({ [e.target.name] : e.target.value})
    }
    render() {
        return (
            <div>
                <h2>User Login Form</h2>
                <form onSubmit = {this.handleSubmit}>
                    <div className = 'form-group'>
                        <label>Enter UserId</label>
                        <input type = 'number' name = '_id' onChange = {this.handleChange} className =
'form-control'/>
                    </div>

                    <div className = 'form-group'>
                        <label>Enter Password</label>
                        <input type = 'password' name = 'password' onChange = {this.handleChange}
className = 'form-control'/>
                    </div>

                    <div className = 'form-group'>
                        <input type = 'submit' value = 'submit' className = 'btn btn-primary'></input>
                        <input type = 'reset' value = 'reset' className = 'btn btn-primary'></input>
                    </div>
                </form>
                <p>UserId {this.state._id}</p>
                <p>Password {this.state.password}</p>
            </div>
        )
    }
}
export class RetrieveUsers extends Component {
    constructor(props) {
        super(props);
        this.state = {users:[]}
    }
    handleClick = (e) => {
        e.preventDefault();
        let findAllUsersURL = "http://localhost:9999/findAllUsers";
        axios.get(findAllUsersURL).then(responseData => {
            // this.setState(users:responseData.data)
            console.log(responseData)
            this.setState({
                users:responseData.data
            });
        });
    }
    render() {
        return(
            <div>
```

```
                <h2>User Informations</h2>
                <a href = "#" onClick = {this.handleClick}>Click to view</a>
                <table className = 'table table-striped'>
                    <thead>
                        <tr>
                            <th>Id</th>
                            <th>Name</th>
                            <th>Gender</th>
                            <th>Phone Number</th>
                        </tr>
                    </thead>
                    <tbody>
                        {this.state.users.map(user => <tr key = {user._id}>
                            <td>{user._id}</td>
                            <td>{user.name}</td>
                            <td>{user.gender}</td>
                            <td>{user.phoneNumber}</td>
                        </tr>)}
                    </tbody>
                </table>

            </div>

        )
    }
}
export class SearchUserForm extends Component {
    render() {
        return (
            <div>
                <h2>Search User by id</h2>
            </div>

        )
    }
}
```

MongoDB

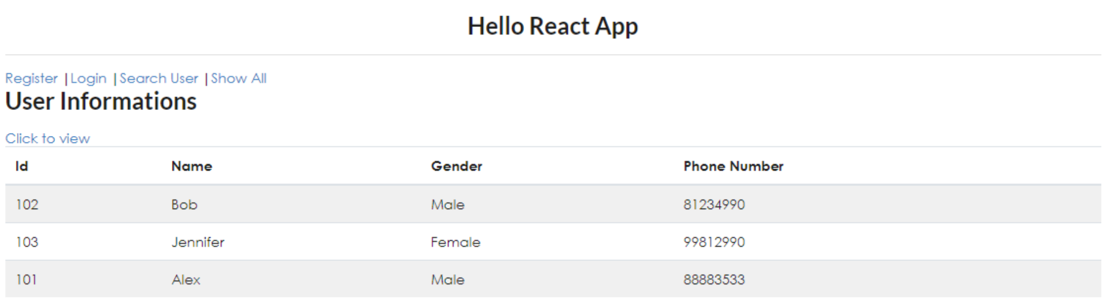
Express JS

Express Router

users-actions-webservices.js

```
let express = require("express");
let bodyParser = require("body-parser");
let mdbClient = require("mongodb").MongoClient;
let dbUrl = "mongodb://localhost:27017"
let app = express();
let port = 9999;

app.use(bodyParser.json());

// middleware to apply cross origin resource sharing filters
app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    next();
});


app.get("/findAllUsers", function(request, response){
  let users = [];
  mdbClient.connect(dbUrl, {useNewUrlParser:true}, function(err, client){
      let db = client.db("testdb");
      let cursor = db.collection("Users").find();
      cursor.forEach(function(doc, err){
          users.push(doc);
      }, function(){
          client.close();
          response.json(users);
      });
  });
});

app.post("/user", function(request, response){
    let content = request.body;

    mdbClient.connect(dbUrl, {useNewUrlParser:true}, function(err, client){
        let db = client.db("testdb");
        db.collection("Users").insertOne(content, function(err, res){
            if(err) {
                response.json({error:"Sorry not stored"})
            } else {
                response.json(res.ops[0]._id);
            }
            client.close();
        });
    });

});

app.put("/user/:id/:phone", function(request, response){
    let userId = parseInt(request.params.id);
    let phone = parseInt(request.params.phone);

    mdbClient.connect(dbUrl, {useNewUrlParser:true}, function(err, client){
        let db = client.db("testdb");
        db.collection("Users").updateOne({"_id":userId}, {$set:{"phoneNumber":phone}},
            function(err, res){
                response.json(res.result);
                console.log(res.result);
                client.close();
        });


    });
});

app.listen(port, () => {console.log(`Express running at ${port}`)})
```

Output

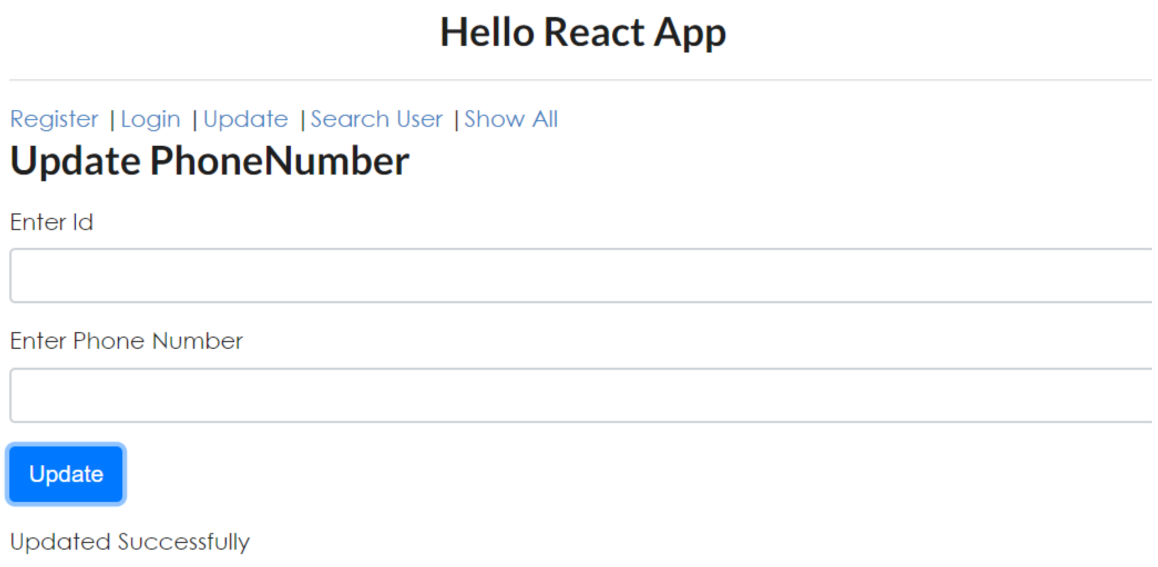

Hello React App

Register |Login |Search User |Show All

User Informations

Click to view

Id	Name	Gender	Phone Number
102	Bob	Male	81234990
103	Jennifer	Female	99812990
101	Alex	Male	88883533


Hello React App

Register |Login |Update |Search User |Show All

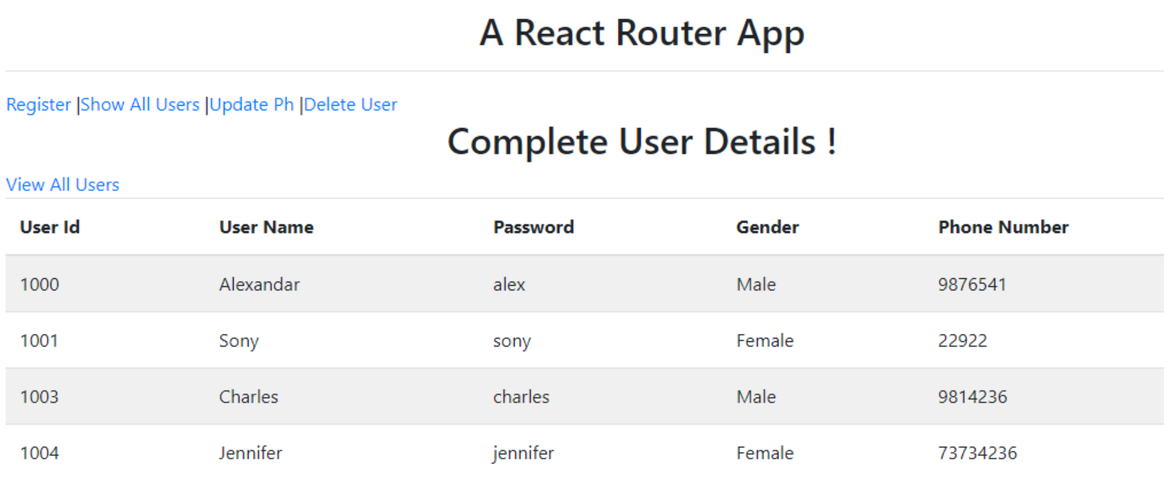
Update PhoneNumber

Enter Id

Enter Phone Number

Updated Successfully

Another Output: Below is a component that has content at the center instead towards the left


A React Router App

Register |Show All Users |Update Ph |Delete User

Complete User Details !

View All Users

User Id	User Name	Password	Gender	Phone Number
1000	Alexandar	alex	Male	9876541
1001	Sony	sony	Female	22922
1003	Charles	charles	Male	9814236
1004	Jennifer	jennifer	Female	73734236

Redux

A nice blog

https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835

https://appdividend.com/2018/06/14/how-to-connect-react-and-redux-with-example/