ReACTJS-Notes

**React:**

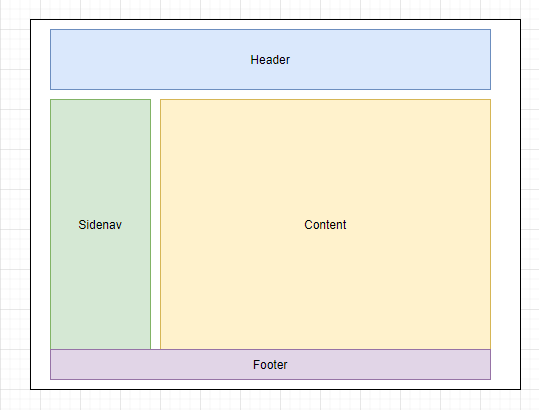
What is React?

* Open source library for building user interfaces
* Not a framework
* Focus on UI
* Rich ecosystem: means provide other library which is used for routing, service call etc.

Why learn React?

* Created and maintained by Facebook
* More than 100k stars on github
* Huge community so if you face any problem during react development you can get answer easily online.
* It is demand skillset.

React has component-based architecture, means all website can be broken into small chunks called components. E.g.



It is also reusable code

React is declarative, tell react what you want and react will build the actual UI. Means you just need to give instruction and react will be build your UI.

React will handle efficiently updating and rendering of the components.

DOM updates are handles gracefully in react.

Seamlessly integrate react into any of your applications.

**Prerequisites:**

1. HTML, CSS and JavaScript fundamentals
2. ES6
3. JavaScript- this keyword, filter map and reduce.

**Software Requirement:**

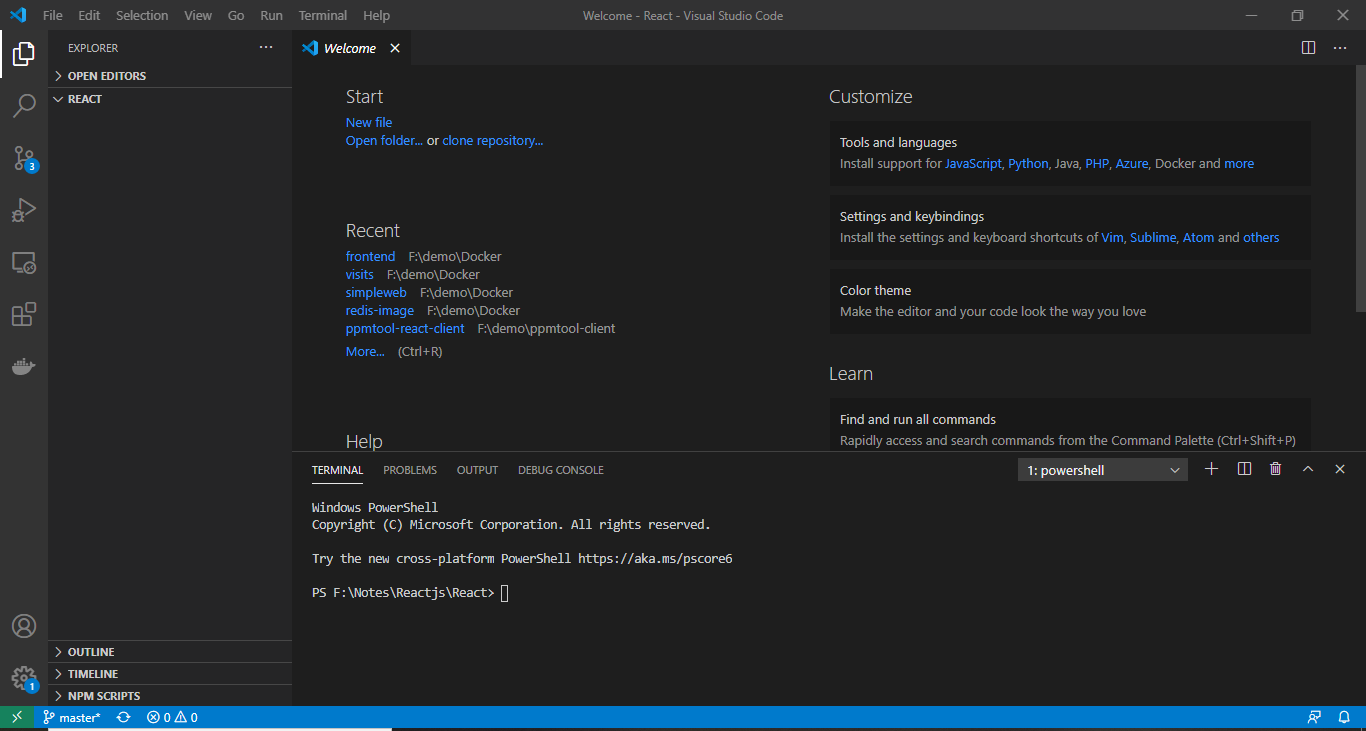
1. Node: <https://nodejs.org/en/>
2. Editor of your choice.: <https://code.visualstudio.com/>

You can also install some plugins in VS Code which help you improve your development experience.

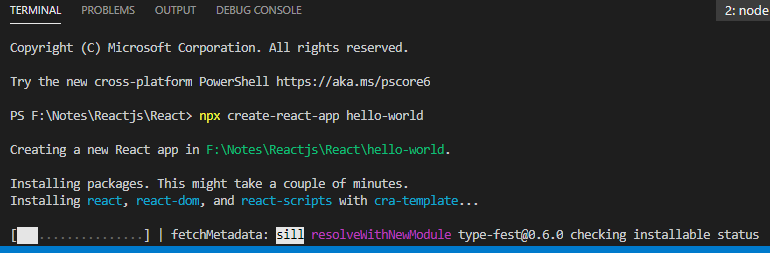
1. Auto import
2. Prettier code formatter
3. Import beautify

Create a hello-world react app.

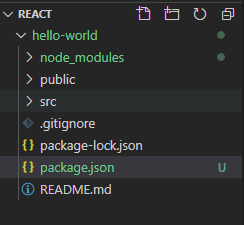
1. Create a folder in your system
2. Open that folder in vs code.



1. Open terminal from terminal menu.
2. We are going to use create-react-app utilities to create react app. We can either use **npx create-react-app <Project-name>** (which is npm package runner) or directly installed create-react-app in your system using **npm install -g create-react-app** and then **create-react-app <Project-name>**.



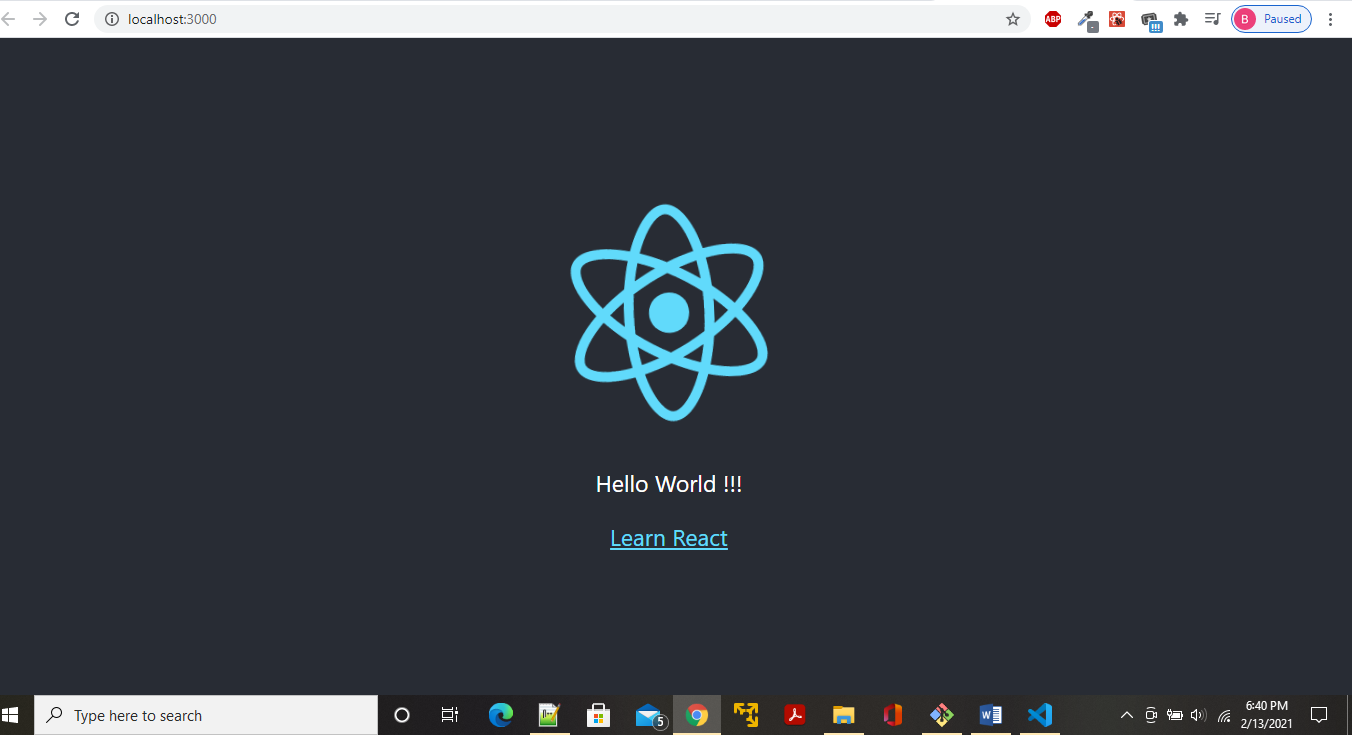
Once it completed you can see below folder structure.



Run cd hello-world to change directory and Open App.js file and remove below line

 Edit <code>src/App.js</code> and save to reload.

Now run npm start command to start server. Once it started you can see below page.



Now let see what are the folders file created under react hello-world app.

In root level we can see three folders node\_modules, public and src and some files.

1. Package.json

Package.json file contains application name, dependencies which is used to run react app and scripts to build and run react application.

1. Based on what you used npm or yarn you can see package-lock.json or yarn-lock.json. it ensures consistence installation of dependency and you don’t to bother about it.
2. First folder is node\_modules which contain all the dependency which is used in react application.
3. Now in public folder there are some favicon.icon, png files and other files which you don’t have to bother about. Only file we need to look is index.html which is the only html file we gone have in entire project. In title you can change title of your application main thing in this index.html is that div tag which has id=”root” which is used by react to render out page.

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

1. Next folder is src folder. Starting point of react application is index.js file where we define the that which element on index.html file needs to be render. App.js is component class where we write out html code. It also has App.css and App.test.js file which is css file and test file for app.js file. We have index.css which is used by whole project. Rest of the file we don’t need to bother.

**Components:**

If we go back to example we have seen above. It has 5 components, header, footer, sidenav, content and root component which hold all these components.

Component code generally we right in .js and .jsx file.

There are two types of components

1. Stateless function component: it is generally javascript function

Function welcome(prop){

Return <h1> welcome </h1>

}

1. Stateful class component: contains class extending Component and render method which returning html code. You can take example of app.js file.

**Functional components:**

It is just a javascript function which may take properties and return html(jsx).

1. Create folder components.
2. Create Greet.js file inside it
3. Writes below codes

import React from 'react';

// function Greet(){

//     return <h1>Hello World!!!</h1>;

// }

const Greet = () => <h1>Hello World!!!</h1>;

export default Greet;

You can either use function or arrow operator to write your codes.

1. In above code we have imported React, then using arrow operator written our react app code. Till here it is component which is not visible to other components to make it visible we need to export that component that’s why we have written that export line.
2. Now to use this component inside App.js we need to import this component inside App.js

import './App.css';

import Greet from './Components/Greet';

function App() {

  return (

    <div className="App">

      <Greet/>

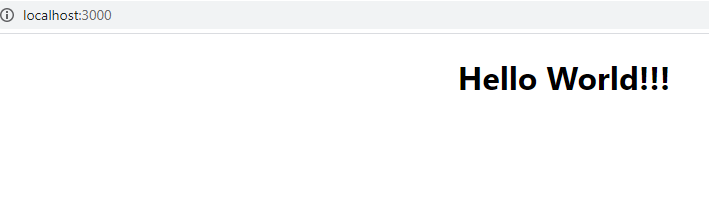
    </div>

  );

}

export default App;

now you will able to see changes in browser as shown in below screenshot.



**Note:** Here we are exporting Greet as default by which we can import it with any name

**e.**g. import MyComponent from ‘./components/Greet’;

<MyComponent />

But if we use below code we need to import with that name.

export const Greet = () => <h1>Hello World!!!</h1>;

if you change this in current code after changing mcomponent thing you will get error in project saying you haven’t used default export.

**Class Component:**

Class component is ES6 class which is also take prop as i/p and return HTML(JSX). And it also maintain state which is private to that class.

1. Create a js file MyGreet.js
2. Write below code in it

import React, { Component} from 'react';

class MyGreet extends Component {

    render (){

        return (

            <div>

            Hello from Class components!!!

            </div>

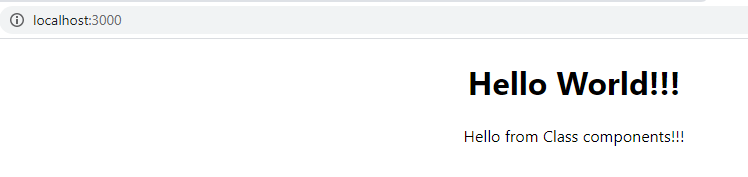
        );

    }

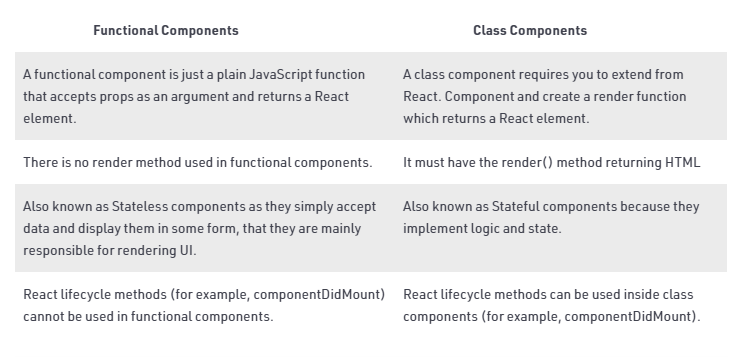
}

export default MyGreet;

1. Import this in App.js file like we did in case of functional component and we can able to see below page.



Difference between functional component and class component.



In functional component new feature called hooks is added by which we can use state inside functional component. It also have back version support means this new change don’t break previous code.

**JSX:**

JavaScript XML(JSX): extension to the JavaScript language syntax. Using this we can write xml like code for elements and components. JSX tags have a tag name, attributes and children. JSX is not a necessity to write react applications without using this we can write the code. But JSX makes your react code simpler and elegant. It ultimately transpiles to pure JavaScript which is understood by the bowsers.

Now let take example,

1. Create a Hello.js file inside components folder.
2. And write below code and import this class and tag in App.js file.

import React from 'react';

const Hello = () =>{

// JSX example

    // return (

    //     <div>

    //     <h2>Hello World!!!</h2>

    //     </div>

    // )

    return React.createElement('div', {id: 'Hello',className:'dummy'},

     React.createElement( 'h2',null, 'Hello World!!!'));

}

export default Hello;

In above example we first code represent JSX example. Where as second code represent without JSX. In second example we have used React.createElement which will take first parameter as tag name, second is properties which is added to the tag and third value of that tag. So, achieve the JSX like thing we need to use ro React.createElement method. We didn’t use class in properties because it is reserved instead of it to add css class we need to use className.

JSX difference:

Class🡪 className

For 🡪 htmlFor

onclick🡪 onClick

tabindex 🡪tabIndex etc.

**Properties:**

We can reuse components in react app. Suppose we want to use greet component multiple time in app.js file. We just need to repeat tags that many times. But if you do so you can see ‘Hello World!!!’ that many times. What if we want to greet some person using his/her name. For this we need to use properties concept in react app. We can pass properties to component using attributes.

e.g. for reusing component.

<Greet/>

      <Greet/>

      <Greet/>



e.g. passing properties.

 <Greet name="shaelraj"/>

      <Greet name="shaelemor"/>

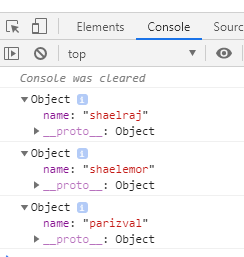
      <Greet name="parizval"/>

const Greet = (props) => {

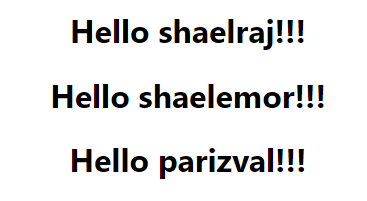
    console.log(props);

    return <h1>Hello World!!!</h1>;

}



In above example we passed name attribute to greet component and using props we are just logging that to console. So, when we want to use these name properties in component we need to use props.name inside greet component. But if we use props.name directly it will not be evaluated by react. So, to evaluate it we need to put props.name inside curly braces.



const Greet = (props) => {

    console.log(props);

    return <h1>Hello {props.name}!!!</h1>;

}

Take another example, suppose we only want to so some elements to first greet tag. What we can do we can use props.children concept. For this we need to change greet component. We need to use below code. If we try to return multiple things from return it will give error because in JSX can only return one html tag.

const Greet = (props) => {

  console.log(props);

  return (

    <div>

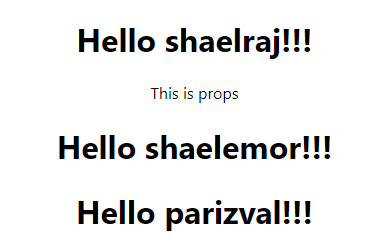
      <h1>Hello {props.name}!!!</h1>

      {props.children}

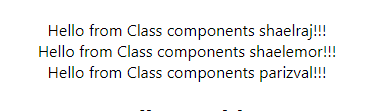
    </div>

  );

};



Now take example how we can do same thing in class component. In class component also we can pass props as attribute but in class component we can use this.props to access those properties as shown in below example.



render (){

        return (

            <div>

            Hello from Class components {this.props.name}!!!

            </div>

        );

    }

We can’t modify props object it is read only.

**State:**

|  |  |
| --- | --- |
| **Props** | **State** |
| Props get passed to component | State managed within components |
| Passed as functional parameter | Variables declared in the function body. |
| Props are immutable | State can be changed |
| We can access props in functional component using props parameter where as in class component using this.props. | Using useState in functional component where as in class component using this.state. |

e.g.

1) Create a class component called message.js

2) write below code on it.

import React, { Component } from "react";

class Message extends Component {

  constructor() {

    super();

    this.state = {

      message: 'Welcome vistor!!!'

    };

  }

  changeMessage(){

      this.setState({

        message: 'Thanks for subscribing!!!'

      });

  }

  render() {

    return (

      <div>

        <h1> {this.state.message} </h1>

        <button onClick={()=> this.changeMessage()}>Subscribe</button>

      </div>

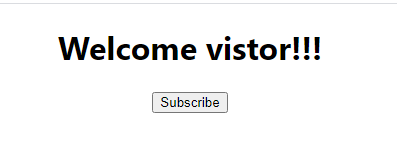
    );

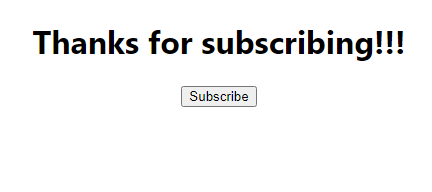
  }

}

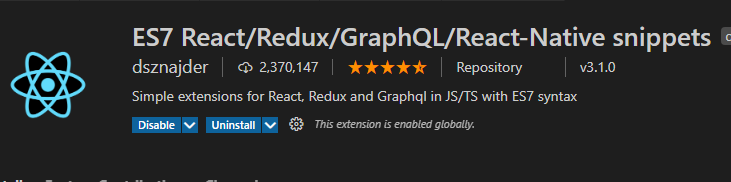
export default Message;

1. Import message class inside the app.js and then you can see below page. When you click on subscribe method message will change automatically.





**setState Method:**



Install above plugin in vs code.

1. Create a counter.js file type **rce** it will automatically generate your code.
2. Create constructor using snippet **rconst**

import React, { Component } from "react";

class Counter extends Component {

  constructor(props) {

    super(props);

    this.state = {

      count: 0,

    };

  }

  increment(){

    //   this.state.count = this.state.count +1

      this.setState({

          count: this.state.count +1

      })

console.log(this.state.count)

  }

  render() {

    return (

      <div>

        <div>count : {this.state.count}</div>

        <button onClick={() => this.increment()}>Subscribe</button>

      </div>

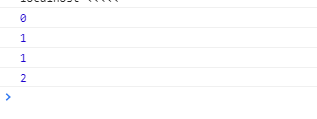
    );

  }

}

export default Counter;

1. If you uncomment commented code and comment this.setState method. If you open your browser you can see that when you click on button the value of state is getting incremented but it doesn’t render the component. But when you use the.setState method it is going to render the component. Also, you can see that in console in case of setState method value is not changing directly it is because setState method is asynchronous. But in application we might want to call some functionality only when state is changed for this we can pass second argument in setState method as fallback function.

increment(){

    //   this.state.count = this.state.count +1

    this.setState({

        count: this.state.count +1

    },  ()=> console.log(this.state.count))

    console.log(this.state.count)

  }

Now take a scenario that we have incrementFive method which will call this.increment method five times.

incrementFive(){

      this.increment()

      this.increment()

      this.increment()

      this.increment()

      this.increment()

  }

  render() {

    return (

      <div>

        <div>count : {this.state.count}</div>

        <button onClick={() => this.increment()}>Subscribe</button>

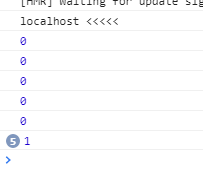
        <button onClick={() => this.incrementFive()}>Subscribe Five</button>

      </div>

    );

  }

When you click on subscribe Five button you can see five time 0 is logged and five time fallback method is called with value 1. It is because react may group multiple setState call to single call.

To make it work we need to change pass function as argument in setstate method where we can pass prevState not current state.

increment(){

    //   this.state.count = this.state.count +1

    // this.setState({

    //     count: this.state.count +1

    // },  ()=> console.log(this.state.count))

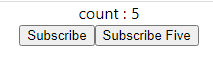
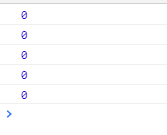
    this.setState(prevState=>({

        count: prevState.count +1

    }))

    console.log(this.state.count)

  }

1. Always use setState method and never try to modify state directly.
2. Code which needs to be executed after state change should be passed as call back function to setState method second argument.
3. When you have to update state based on prevState always pass function to setState method instead of object.