**Index**

1. Chapter 1 🡪Basics of react
2. Chapter 2 🡪Setting of dev environment and creating Hello-world app
3. Chapter 3 🡪Understanding folder structure and flow
4. Chapter 4 🡪 Understanding components and Component types
5. Chapter 5 🡪 JSX (JavaScript XML)
6. Chapter 6 🡪 props (properties) in react component
7. Chapter 7 🡪 state
8. Chapter 8 🡪 Event handling and Event binding
9. Chapter 9 🡪 Methods as props
10. Chapter 10 🡪Rendering (Conditional Rendering, List Rendering)
11. Chapter 11 🡪 Styling React Component
12. Chapter 12 ---------------------------------------------
13. Chapter 13 ---------------------------------------------
14. Chapter 14 ---------------------------------------------
15. Chapter 15 ---------------------------------------------
16. Chapter 16 ---------------------------------------------
17. Chapter 17 ---------------------------------------------
18. Chapter 18 ---------------------------------------------
19. Chapter 19 ---------------------------------------------
20. Chapter 20 ---------------------------------------------

**Snippet:**

1. **ES7 React/Redux:**

To create functional component: rfce

To create Class component: rce

To create constructor in Class component: rcosnt

**Chapter -1**

**Basics of react**

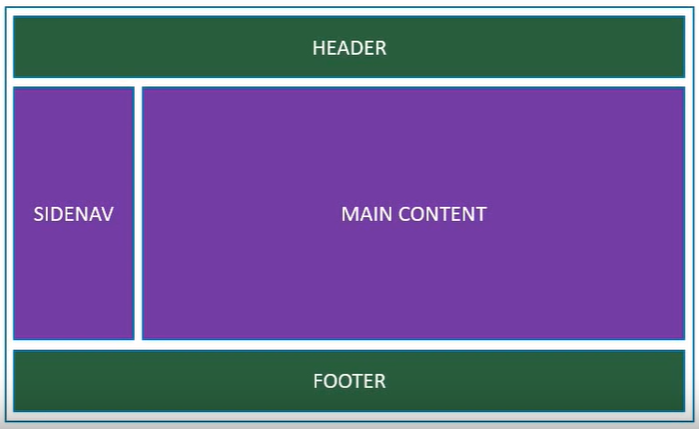
React is created and maintained by Facebook

What is react?

* React is an open source open source java script library for building user interface (It is not a framework).
* We can user react to build user interface only (we can’t handle http request or routing etc.)
* React has rich eco system so we can get other libraries for handling of http request or routing.

Architecture of react applications

React has component bases architecture.



**React is declarative in nature**

* Tell react what you want and react will build for you.
* React will handle efficiently updating and rendering of the component.
* DOM updates are handle gracefully in react.
* React can seamlessly be integrated into any of your application (portion of your page or a complete page or even an entire application.
* Later on we can learn **React Native** for native mobile application development.

**Chapter-2**

**Setting up the development environment**

1. Download **nodeJS** from <https://nodejs.org/en/> and install it into your system.
2. Download and install **VSCode** from <https://code.visualstudio.com/download>.

[Creating first react app](https://github.com/facebook/create-react-app)

1. Create a folder in your system with any name (say **React**) and open this folder in VSCode.
2. Open integrated terminal in VSCode and type a command

**npx create-react-app hello-world**

1. Navigate inside the newly created “hello-world” folder.

**cd .\hello-world**

1. Now start the server.

**npm start**

This will open browser at port 3000

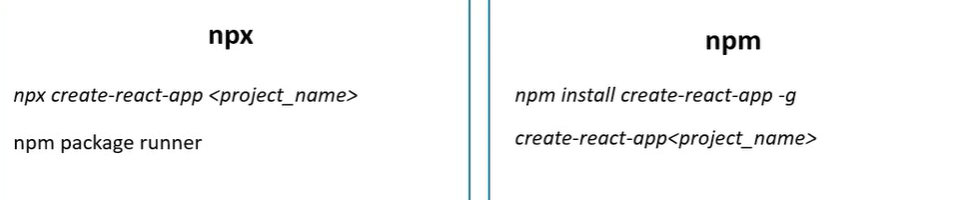
**2nd approach**

* npm install create-react-app –g
* create-react-app <project-name>

Note: if nothings works then use these steps

1. create-react-app my-app
2. cd my-app
3. npm install react-scripts@2.1.8
4. npm start

Difference between npx and npm methods



**Chapter-3**

**Understanding folder structure and flow of react application**

**Flow**

index.html -> index.js -> App.js

Public folder src folder

Hello-world > public > index.js

<!DOCTYPE html>

<html lang="en">

  <head> </head>

  <body>

    <div id="root"></div>

  </body>

</html>

Hello-world > src > 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'));

serviceWorker.unregister();

Hello-world > src > App.js

import React from "react";

import logo from "./logo.svg";

import "./App.css";

function App() {

  return (

    <div className="App">

    </div>

  );

}

export default App;

**Chapter-4**

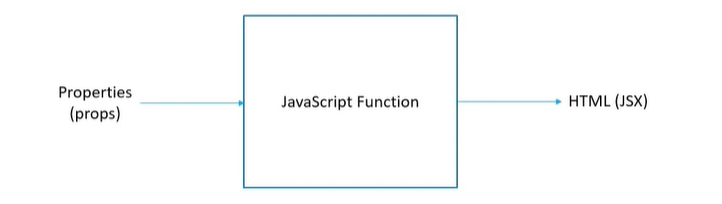
**Understanding components**

* Components are basically JavaScript files with **js** or **jsx** extension. E.g. for app component App.js
* Naming convention for components: Pascal naming conversion E.g. “HelloWorld”

**Component type**

* Stateless functional component
* Stateful class component

**Stateless functional component**



Steps to create function component

1. Create a folder by the name of “components” in src folder
2. Create a JavaScript file within component folder (say Greet.js)

Hello-world > src > components > Greet.js

import React from 'react'

function Greet(){

    return <h1>Hello world</h1>

}

export default Greet

OR

import React from 'react'

const Greet =()=><h1>Hello World</h1>

export default Greet

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Greet from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Greet/>

</div>

  );

}

export default App;

**Stateful class component**



Steps to create function component

1. Create a folder by the name of “components” in src folder
2. Create a JavaScript file within component folder (say Welcome.js)

Hello-world > src > components > Welcome.js

import React, { Component } from "react";

class Welcome extends Component {

  render() {

    return (

      <div>

        <h1>Hello World</h1>

      </div>

    );

  }

}

export default Welcome;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Welcome from "./components/Welcome"

import "./App.css";

function App() {

  return (

    <div className="App">

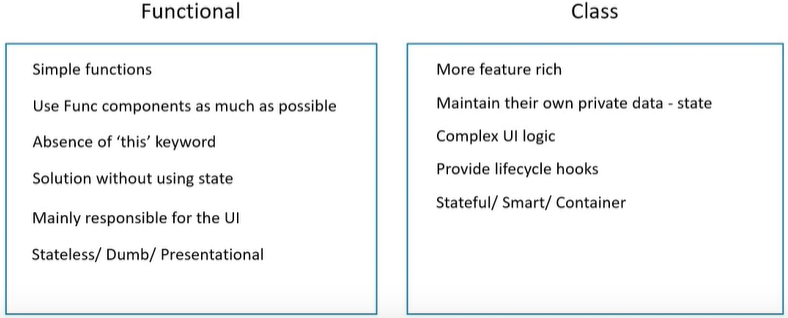
<Welcome/></div>

);

}

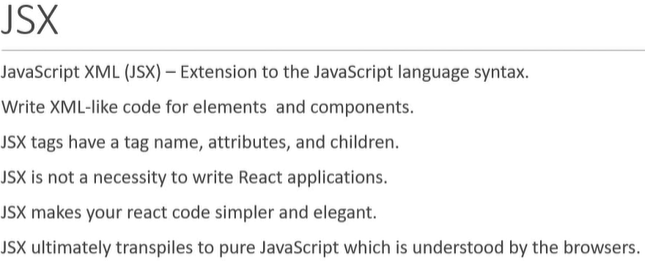
export default App;

**Comparison between functional and class component**



**Chapter-5**

JSX (JavaScript XML)



Comparison for react component with and without JSX

E.g. create a component to display “hello world” with and without using jsx.

**Using JSX**

Hello-world > src > components > Hello.js

import React from 'react'

const Greet =()=><h1>Hello World</h1>

export default Greet

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Hello/></div>

  );

}

export default App;

Hello-world > src > components > Hello.js

import React from "react";

const Hello = () => {

  return React.createElement(

    "div",

    null,

    React.createElement(

      "h1",

      { id: "mtId", className: "myClass" },

      "hello world"

    )

  );

};

export default Hello;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Hello/>

</div>

  );

}

export default App;

**Chapter-6**

**Props (properties) in react component**

* Props are the optional input that a component can accept.

(It is a JavaScript Object)

* It also allows the component to be dynamic (by).

**6.1 Using props with class component**

Hello-world > src > components > Greet.js

import React from "react";

const Greet = props => {

  return (

    <h1>

      Greet {props.name} a.k.a {props.heroName}

    </h1>

  );

};

export default Greet;

**Or we can destructure props and then use**

import React from "react";

const Greet = props => {

const { name, heroName } = props;

  return (

    <h1>

      Greet {name} a.k.a {heroName}

    </h1>

  );

};

export default Greet;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Greet name="rohit" heroName="hitman" />

<Greet name="virat" heroName="Mr. Consistan" />

<Greet name="dhoni" heroName="Finisher" />

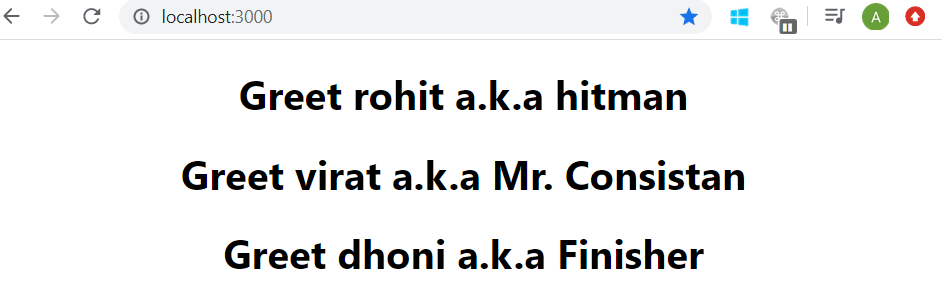
</div>

  );

}

export default App;

**Output:-**



**Note:**

Sometime we may not have an idea what content may have passed in.

In such situation we can pass the content in <p> </p> in opening and closing tag of the component and retrieving it using reserved “**children**” property of props object.

**6.2 Using props.children**

Hello-world > src > components > Greet.js

import React from "react";

const Greet = props => {

  return (

    <div>

      <h1>

        Greet {props.name} a.k.a {props.heroName}

      </h1>

      {props.children}

    </div>

  );

};

export default Greet;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Greet name="rohit" heroName="hitman">

<p> This is details of rohit</p>

</Greet>

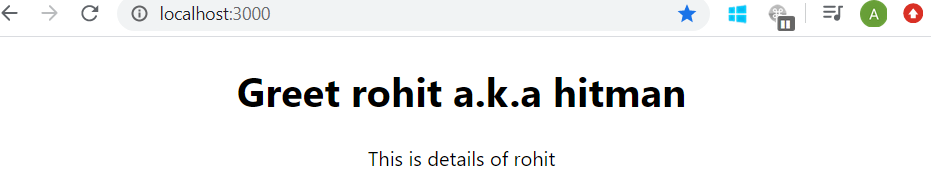
</div>

  );

}

export default App;

**Output:-**



**6.3 Using props with class component**

Hello-world > src > components > Welcome.js

import React, { Component } from "react";

class Welcome extends Component {

  render() {

    return (

      <h1>

        Welcome {this.props.name} a.k.a {this.props.heroName}

      </h1>

    );

  }

}

export default Welcome;

**or** we can destructure props and then use

import React, { Component } from "react";

class Welcome extends Component {

  render() {

    const { name, heroName } = this.props;

    return (

      <h1>

        Welcome {name} a.k.a {heroName}

      </h1>

    );

  }

}

export default Welcome;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

      <Welcome name="rohit" heroName="hitman" />

<Welcome name="virat" heroName="Mr. Consistan" />

      <Welcome name="dhoni" heroName="Finisher" />

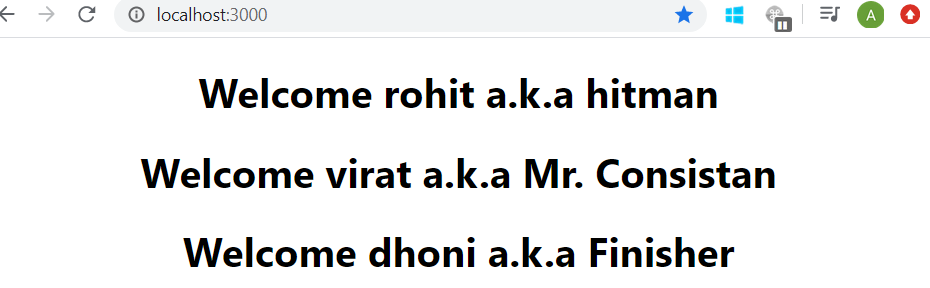
</div>

  );

}

export default App;

**Output:-**



**6.4 Props are immutable**

Props are immutable i.e. its value can’t be changed.

E.g. let’s try to change the value of props.name

Hello-world > src > components > Greet.js

import React from "react";

const Greet = props => {

 props.name = "Abhishek";

  return (

    <h1>

      Greet {props.name} a.k.a {props.heroName}

    </h1>

  );

};

export default Greet;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Greet name="rohit" heroName="hitman" />

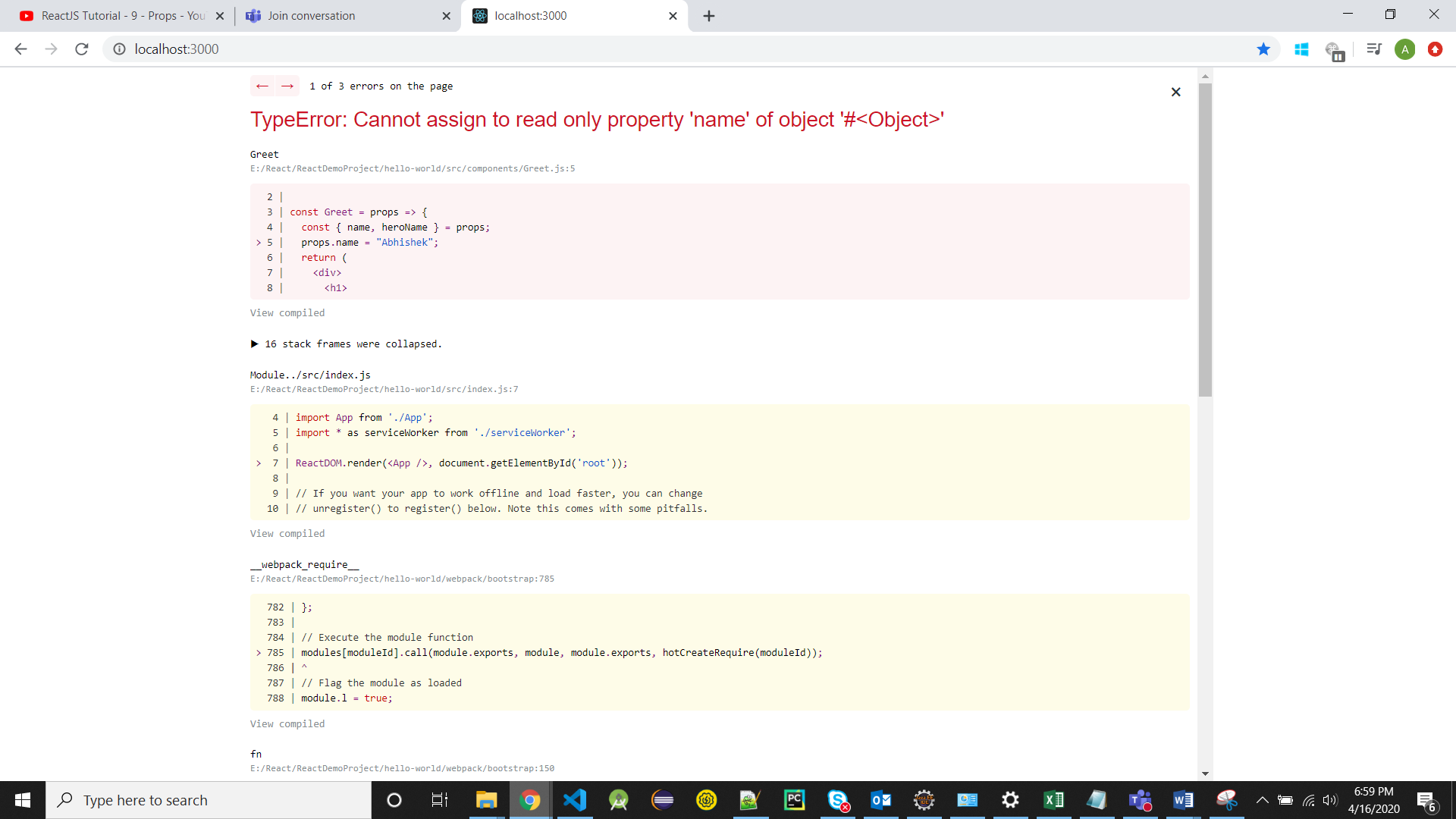
</div>

  );

}

export default App;

Output:-



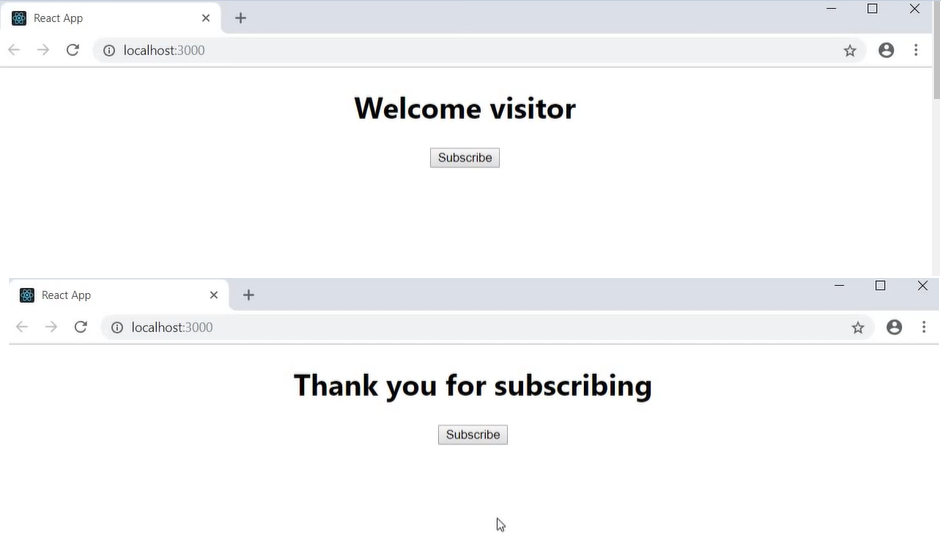
**Chapter-7**

**State**

State is also like props where its value can be changed

|  |  |
| --- | --- |
| **props** | **state** |
| Props get passed to the component  E.g. props are similar to function parameter | state is managed within the component  E.g. state is similar to variable declared in the function body |
| props are immutable | state can be changed |
| Using props in component:  In functional Component: props  In Class Component: this.props | Using state in component:  In functional Component: useState Hook  In Class Component: this.state |

**E.g.** Create an app where text displaying welcome visitor changes to thank you for subscribing after clicking subscribe button.



Hello-world > src > components > Message.js

import React, { Component } from "react";

class Message extends Component {

  constructor() {

    super();

    this.state = {

      message: "Welcome visitor"

    };

  }

  changeMessage() {

    this.setState({

      message: "thank you for subscribing"

    });

  }

  render() {

    return (

      <div>

        <h1>{this.state.message}</h1>

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

      </div>

    );

  }

}

export default Message;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Hello/>

</div>

  );

}

export default App;

**setState**

* To update/modify the state always use setState method and never modify state directly.
* setState method can accept 2 parameters:

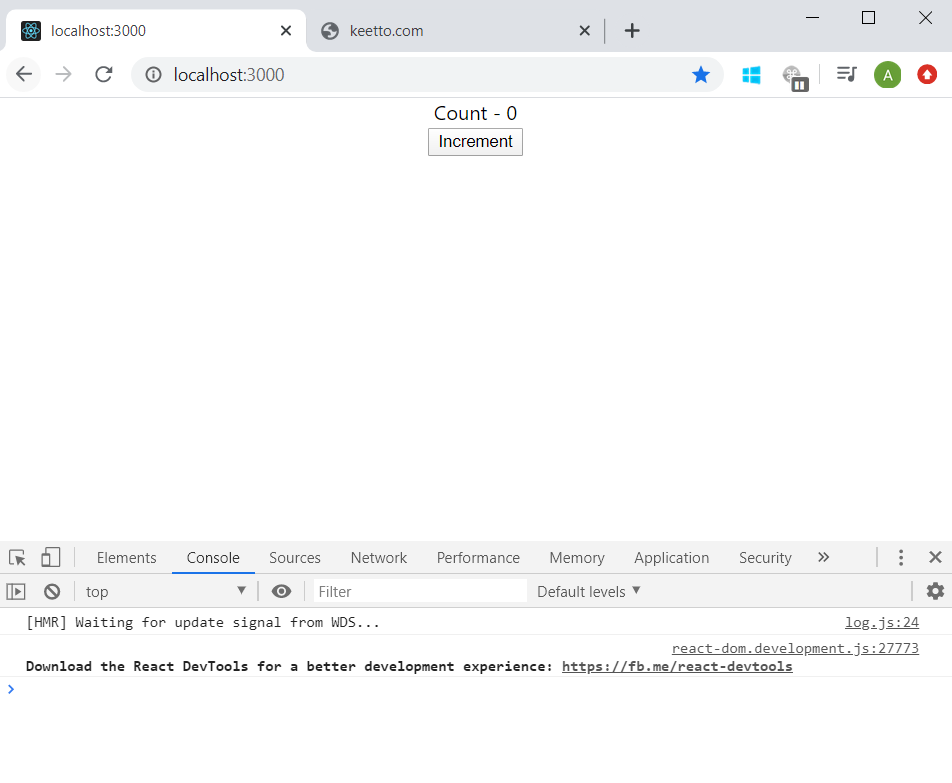
1st) State object **or** function:

We use function as an argument when we need to update the state based on previous state otherwise we use state object as an argument.

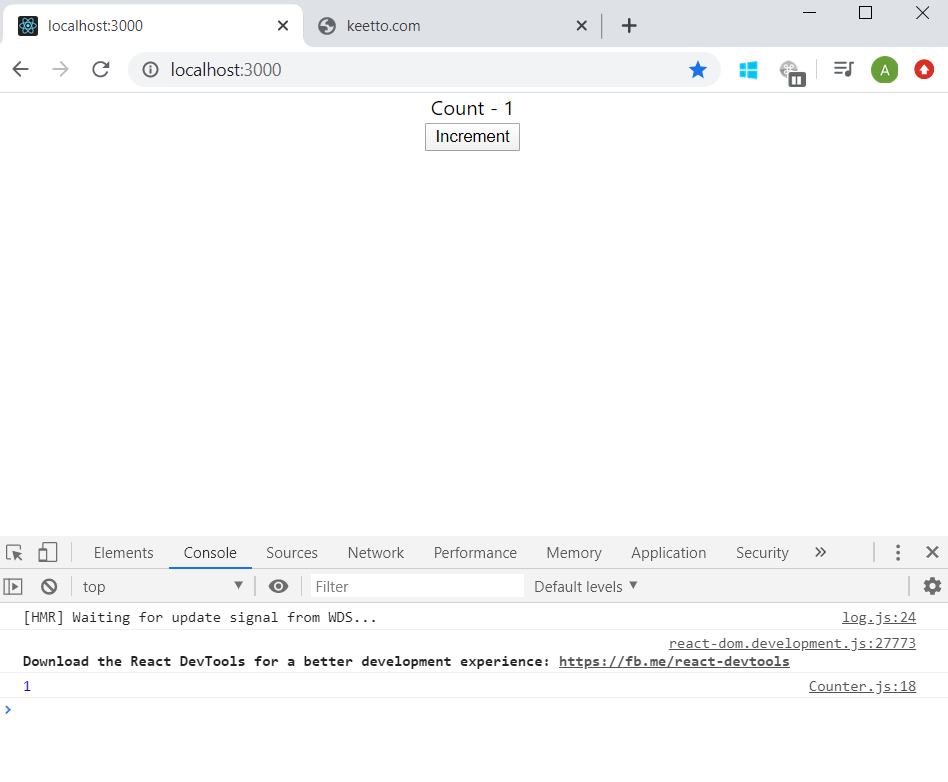
2nd) Callback function: - it’s an optional parameter. This function gets executed once the state is set

**(Whenever we want to executed some code once state is set then place that code within callback function which is passed as 2nd parameter to setState method)**

E.g. for callback function) Create an app where value of count can increase with click of “increment” button and also incremented value should get logged in console.



After clicking Increment



Hello-world > src > components >Counter.js

import React, { Component } from "react";

class Counter extends Component {

  constructor(props) {

    super(props);

    this.state = {

      count: 0

    };

  }

  increment() {

    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()}>Increment</button>

      </div>

    );

  }

}

export default Counter;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Counter/>

</div>

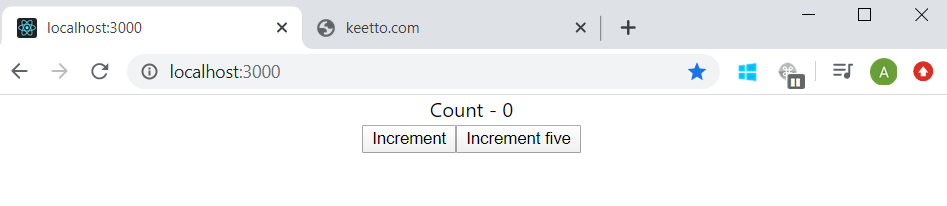
  );

}

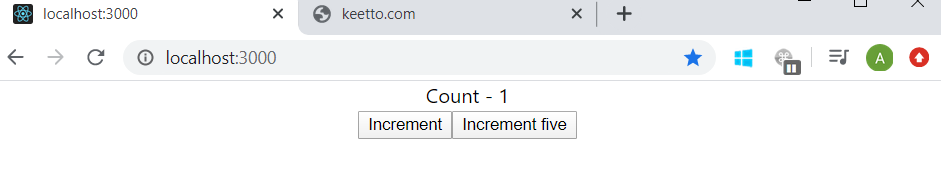
export default App;

Whenever we required to update the state based on previous state then we need to pass function as an argument to setState method instead of passing in an object

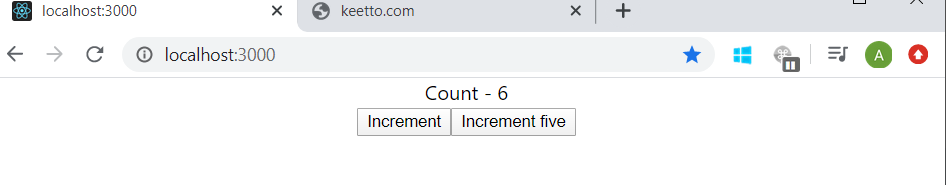
E.g. create an app where count can increase by 1 and by 5 on click to “increment” and “increment five” button resp.



Increment is clicked



Then Increment five is clicked



Hello-world > src > components > Counter.js

import React, { Component } from "react";

class Counter extends Component {

  constructor(props) {

    super(props);

    this.state = {

      count: 0

    };

  }

  increment() {

    this.setState(prevState => ({

      count: prevState.count + 1

    }));

  }

  incrementFive() {

    this.increment();

    this.increment();

    this.increment();

    this.increment();

    this.increment();

  }

  render() {

    return (

      <div>

        <div>Count - {this.state.count}</div>

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

        <button onClick={() => this.incrementFive()}>Increment five</button>

      </div>

    );

  }

}

export default Counter;

Hello-world > src > App.js

import React from "react"

import logo from "./logo.svg"

import Hello from "./components/Greet"

import "./App.css";

function App() {

  return (

    <div className="App">

<Counter/>

</div>

  );

}

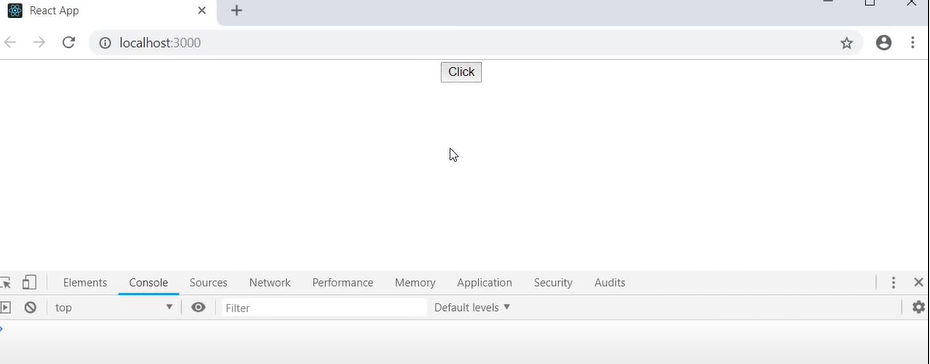
export default App;

**Chapter-8**

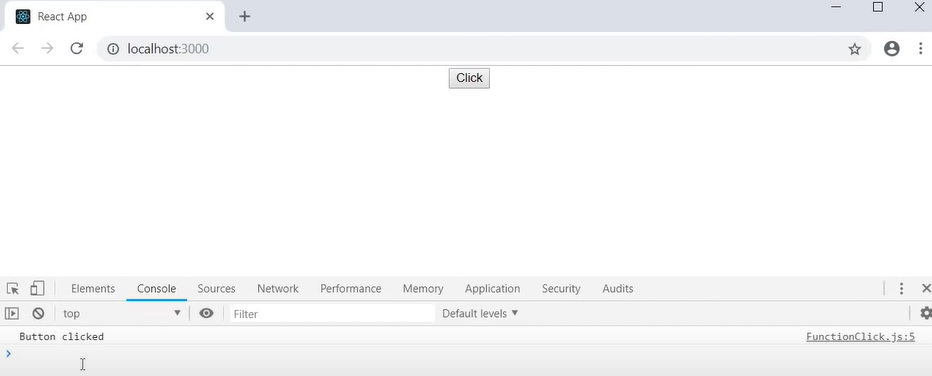
**Event handling:**

**Implementation in functional component:**

E.g. create an app where a button “click” is present and on clicking this button “button is clicked” should get logged to the console.



After clicking on “click”



Hello-world > src > components > FunctionClick.js

import React from "react";

function FunctionClick() {

  function clickHandler() {

    console.log("button clicked");

  }

  return (

    <div>

      <button onClick={clickHandler}>Click</button>

    </div>

  );

}

export default FunctionClick;

Hello-world > src > App.js

import React from "react"

import FunctionClick from "./components/FunctionClick "

import "./App.css";

function App() {

  return (

    <div className="App">

<FunctionClick/>

</div>

  );

}

export default App;

Note: event handlers should be a function "clickHandler" not a function call"clickHandler()"

**Implementation in Class component:**

E.g. create an app where a button “click” is present and on clicking this button “button is clicked” should get logged to the console.

Hello-world > src > components > ClassClick.js

import React, { Component } from "react";

class ClassClick extends Component {

  clickHandler() {

    console.log("button clicked");

  }

  render() {

    return (

      <div>

        <button onClick={this.clickHandler}>click</button>

      </div>

    );

  }

}

export default ClassClick;

Hello-world > src > App.js

import React from "react"

import ClassClick from "./components/ClassClick"

import "./App.css";

function App() {

  return (

    <div className="App">

<ClassClick/>

</div>

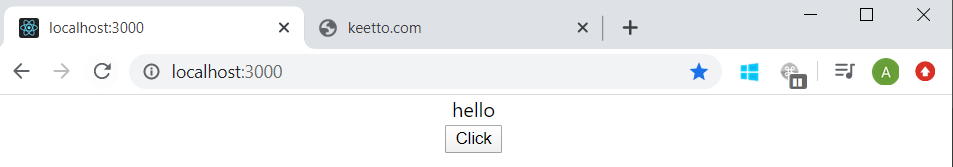
  );

}

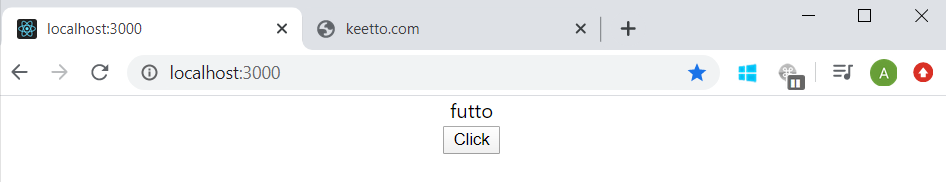
export default App;

**Event binding (Binding Event handlers)**

E.g. create an app where text displaying “hello” and have “click” button bellow it and on clicking “click” button text message changes to “futto”



On clicking “click”



**Theory:-**

We need to bind event handler just because of how “this” keyword work in JavaScript.

**“this” keyword is undefined in event handler so we need to bind it to use it there are 4 methods to bind event handler**

<button onClick={this.clickHandler}>Click</button> 🡪 it’s won’t work, so to make it work we have 4 below approaches.

* **Binding in the render method**
* **Using arrow function in render method**
* **Binding event handler in constructor**
* **Using arrow function as class property**

Note: use approach 3 or 4

1) **Binding in the render method** 🡪 although this method works but if we are using this every update in state will let component to rerender once again and intern generating new event handler every time, so this this is not suggested

<button onClick={this.clickHandler.bind(this)}>Click</button>

Hello-world > src > components > EventBind.js

import React, { Component } from "react";

class EventBind extends Component {

  constructor(props) {

    super(props);

    this.state = {

      message: "hello"

    };

  }

  clickHandler() {

    this.setState({

      message: "futto"

    });

  }

  render() {

    return (

      <div>

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

        <button onClick={this.clickHandler.bind(this)}>Click</button>

      </div>

    );

  }

}

export default EventBind;

2) **Using arrow function in render method🡪** this method also has performance implication in some scenarios

**<**button onClick={() => this.clickHandler()}>Click</button>

Hello-world > src > components > EventBind.js

import React, { Component } from "react";

class EventBind extends Component {

  constructor(props) {

    super(props);

    this.state = {

      message: "hello"

    };

  }

  clickHandler() {

    this.setState({

      message: "futto"

    });

  }

  render() {

    return (

      <div>

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

        <button onClick={() => this.clickHandler()}>Click</button>

      </div>

    );

  }

}

export default EventBind;

3) **Binding event handler in constructor** -> this approch is given in react docs

<button onClick={this.clickHandler}>Click</button>

Hello-world > src > components > EventBind.js

import React, { Component } from "react";

class EventBind extends Component {

  constructor(props) {

    super(props);

    this.state = {

      message: "hello"

    };

    this.clickHandler = this.clickHandler.bind(this);

  }

  clickHandler() {

    this.setState({

      message: "futto"

    });

  }

  render() {

    return (

      <div>

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

        <button onClick={this.clickHandler}>Click</button>

      </div>

    );

  }

}

export default EventBind;

4) **Using arrow function as class property**

<button onClick={this.clickHandler}>Click</button>

Hello-world > src > components > EventBind.js

import React, { Component } from "react";

class EventBind extends Component {

  constructor(props) {

    super(props);

    this.state = {

      message: "hello"

    };

  }

  clickHandler = () => {

    this.setState({

      message: "futto"

    });

    console.log(this);

  };

  render() {

    return (

      <div>

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

        <button onClick={this.clickHandler}>Click</button>

      </div>

    );

  }

}

export default EventBind;

Hello-world > src > App.js

import React from "react"

import EventBind from "./components/EventBind"

import "./App.css";

function App() {

  return (

    <div className="App">

<EventBind/>

</div>

  );

}

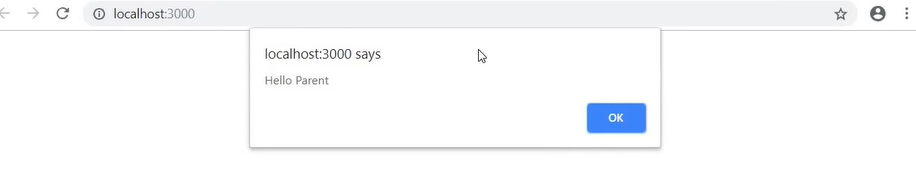
export default App;

**Chapter-9**

**Methods as props**



On clicking “Greet Parent” button



Hello-world > src > components > ParentComponent.js

import React, { Component } from "react";

import ChildComponent from "./ChildComponent";

class ParentComponent extends Component {

  constructor(props) {

    super(props);

    this.state = {

      parentName: "parent"

    };

    this.greetParent = this.greetParent.bind(this);

  }

  greetParent(childName) {

    alert(`hello ${this.state.parentName}`);

  }

  render() {

    return (

      <div>

        <ChildComponent greethandler={this.greetParent} />

      </div>

    );

  }

}

export default ParentComponent;

Hello-world > src > components > ChildComponent.js

import React from "react";

function ChildComponent(props) {

  return (

    <div>

      <button onClick={props.greethandler}>Greet Parent</button>

    </div>

  );

}

export default ChildComponent;

Hello-world > src > App.js

import React from "react"

import ParentComponent from "./components/ParentComponent"

import "./App.css";

function App() {

  return (

    <div className="App">

<ParentComponent/>

</div>

  );

}

export default App;

**passing the parameter from childComponent when calling the methods of parentComponent:** for this arrow function is approach is really useful

Hello-world > src > components > ChildComponent.js

import React from "react";

function ChildComponent(props) {

  return (

    <div>

    <button onClick={() => props.greethandler("child")}>Greet Parent</button>

    </div>

  );

}

export default ChildComponent;

Hello-world > src > components > ParentComponent.js

import React, { Component } from "react";

import ChildComponent from "./ChildComponent";

class ParentComponent extends Component {

  constructor(props) {

    super(props);

    this.state = {

      parentName: "parent"

    };

    this.greetParent = this.greetParent.bind(this);

  }

  greetParent(childName) {

    alert(`hello ${this.state.parentName} from ${childName}`);

  }

  render() {

    return (

      <div>

        <ChildComponent greethandler={this.greetParent} />

      </div>

    );

  }

}

export default ParentComponent;

Hello-world > src > App.js

import React from "react"

import ParentComponent from "./components/ParentComponent"

import "./App.css";

function App() {

  return (

    <div className="App">

<ParentComponent/>

</div>

  );

}

export default App;

Chapter-10

**Rendering**

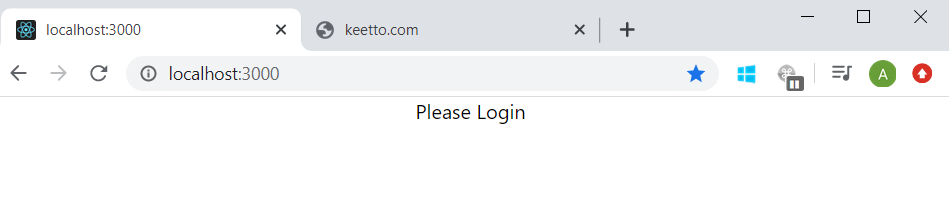
**10.1) Conditional Rendering**

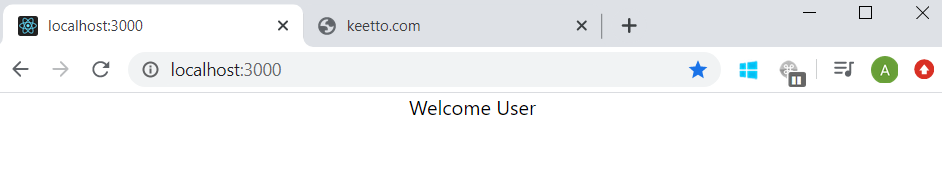
When we are building react application we often need to show or hide html based on certain condition

There are 4 approaches for conditional rendering

* **If/else**
* **element variable**
* **Ternary conditional operator**
* **short circuit operator**

**E.g.** create an app to display “Please Login” when user is not logged in but display “Welcome User” when user is logged in.





**Using “if/else” approach:** here we check the condition with if/else outside the return statement and then return the required html.

Hello-world > src > components >UserGreeting.js

import React, { Component } from "react";

class UserGreeting extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isLoggedIn: true //true or false

    };

  }

  render() {

    if (this.state.isLoggedIn) {

      return (

        <div>

          <div>Welcome user</div>

        </div>

      );

    } else {

      return (

        <div>

          <div>Please Login</div>

        </div>

      );

    }

  }

}

export default UserGreeting;

**Element variable:** here also we use if/else only but here we create a variable and assign this variable to required html based on if/else condition and then return this variable itself.

Hello-world > src > components >UserGreeting.js

import React, { Component } from "react";

class UserGreeting extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isLoggedIn: true //true or false

    };

  }

  render() {

    let message;

    if (this.state.isLoggedIn) {

      message = <div>Welcome User</div>;

    } else {

      message = <div>Please Login</div>;

    }

    return message;

  }

}

export default UserGreeting;

**Ternary conditional operator:** this is most preferred approach as we can use this inside return statement.

Hello-world > src > components >UserGreeting.js

import React, { Component } from "react";

class UserGreeting extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isLoggedIn: true //true or false

    };

  }

  render() {

    return this.state.isLoggedIn ? (

        <div>Welcome User</div>

      ) : (

        <div>Please Login</div>

      );

  }

}

export default UserGreeting;

**Short circuit operator**: it is the special case of ternary used when if u want something to display or absolutely nothing to display

Hello-world > src > components >UserGreeting.js

import React, { Component } from "react";

class UserGreeting extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isLoggedIn: true //true or false

    };

  }

  render() {

    return this.state.isLoggedIn && <div>Welcome User</div>;

  }

}

export default UserGreeting;

Hello-world > src > App.js

import React from "react"

import UserGreeting from "./components/UserGreeting"

import "./App.css";

function App() {

  return (

    <div className="App">

<Usergreeting/>

</div>

  );

}

export default App;

**10.2) List Rendering:**

When we built react application one the most common scenarios is to display list of items E.g. list of students, list of courses etc. In such scenarios out intension is to repeat some html for each item of the list.

We will be using map method of JavaScript to implement this.

**Understanding map method**

var array1= [1, 3, 5, 7];

//pass function to map

const map1 = array1.map(x=>x\*2);

console.log(map1);

//expected output: Array [2, 6, 10, 14]

**Without using map method**

Hello-world > src > components >NameList.js

import React from "react";

function NameList() {

  const names = ["rohit", "virat", "dhoni"];

  return (

    <div>

      <h2>{names[0]}</h2>

      <h2>{names[1]}</h2>

      <h2>{names[2]}</h2>

    </div>

  );

}

export default NameList;

Using map method

Hello-world > src > components >ListRendusingMap.js

import React from "react";

function ListRendusingMap() {

  const names = ["rohit", "virat", "dhoni"];

  return (

    <div>

     names.map(name=><h2>{name}</h2>)

    </div>

  );

}

export default ListRendusingMap;

**Or**

Hello-world > src > components >ListRendusingMap.js

import React from "react";

function ListRendusingMap() {

  const names = ["rohit", "virat", "dhoni"];

  const nameList = names.map(name => <h2>name</h2>);

  return <div>{nameList}</div>;

}

export default ListRendusingMap;

Hello-world > src > components >PersonData.js

import React from "react";

import PersonDisplay from "./PersonDisplay";

function PersonData() {

  const persons = [

    { id: 1, name: "rohit", age: 30, skill: "hitting" },

    { id: 2, name: "virat", age: 32, skill: "batting" },

    { id: 3, name: "dhoni", age: 40, skill: "wicket keeper" }

  ];

  const personList = persons.map(person => (

    <h2>

      My name is {person.name}. I am {person.age}. i'm known for {person.skill}

    </h2>

  ));

  return <div> {personList}</div>;

}

export default PersonData;

Hello-world > src > App.js

import React from "react"

import PersonData from "./components/PersonData"

import "./App.css";

function App() {

  return (

    <div className="App">

<PersonDate/>

</div>

  );

}

export default App;

**OR** we can refactor the jsx into separate component (this is the recommended way)

Hello-world > src > components >PersonData.js

import React from "react";

import PersonDisplay from "./PersonDisplay";

function PersonData() {

  const persons = [

    { id: 1, name: "rohit", age: 30, skill: "hitting" },

    { id: 2, name: "virat", age: 32, skill: "batting" },

    { id: 3, name: "dhoni", age: 40, skill: "wicket keeper" }

  ];

  const personList = persons.map(person => (

    <PersonDisplay person={person}></PersonDisplay>

  ));

  return <div> {personList}</div>;

}

export default PersonData;

Hello-world > src > components >PersonDisplay.js

import React from "react";

function PersonDisplay({ person }) {

  return (

    <div>

      <h2>

        My name is {person.name}. I am {person.age}. i'm known for {person.skill}

      </h2>

    </div>

  );

}

export default PersonDisplay;

Hello-world > src > App.js

import React from "react"

import PersonData from "./components/PersonData"

import "./App.css";

function App() {

  return (

    <div className="App">

<PersonDate/>

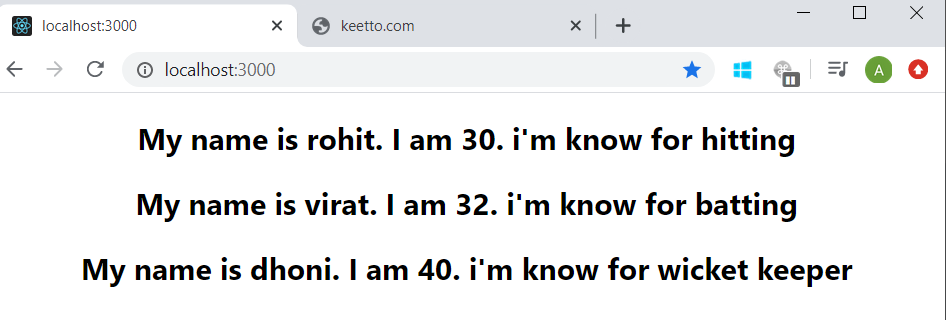
</div>

  );

}

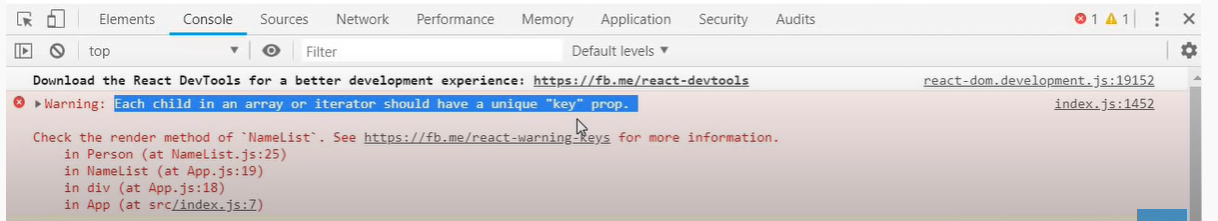
export default App;

**0utput: -**



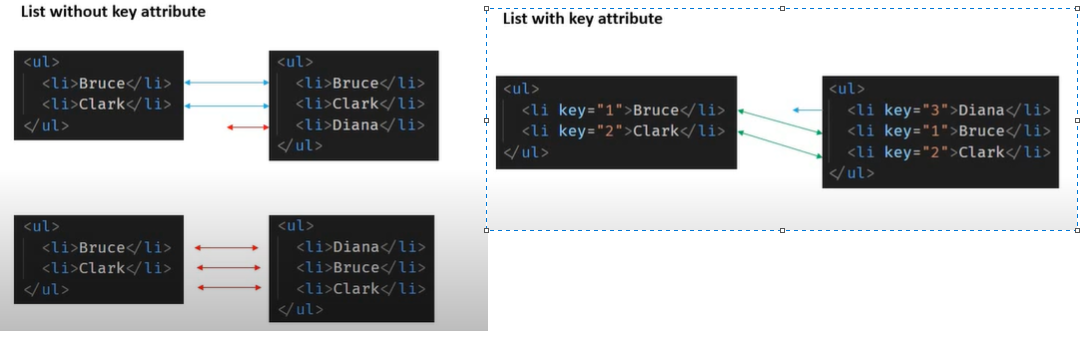
Note:-

While rendering the list if we have not assigned any unique key with each list item then we’ll receive below warning in console.



**Lists and keys in react**

* A Key is a special string attribute we need to include when creating list of elements.
* Keys give the elements a stable identity. It helps react to identity which items have changed (added or removed).



Hello-world > src > components >PersonData.js

import React from "react";

import PersonDisplay from "./PersonDisplay";

function PersonData() {

  const persons = [

    { id: 1, name: "rohit", age: 30, skill: "hitting" },

    { id: 2, name: "virat", age: 32, skill: "batting" },

    { id: 3, name: "dhoni", age: 40, skill: "wicket keeper" }

  ];

  const personList = persons.map(person => (

    <PersonDisplay key={person.id} person={person}></PersonDisplay> ));

  return <div> {personList}</div>;

}

export default PersonData;

Hello-world > src > components >PersonDisplay.js

import React from "react";

function PersonDisplay({ person }) {

  return (

    <div>

      <h2>

        My name is {person.name}. I am {person.age}. i'm known for {person.skill}

      </h2>

    </div>

  );

}

export default PersonDisplay;

Hello-world > src > App.js

import React from "react"

import PersonData from "./components/PersonData"

import "./App.css";

function App() {

  return (

    <div className="App">

<PersonDate/>

</div>

  );

}

export default App;

**Note: -** key is a reserved keyword for prop, we cannot use it to pass values.

NB: - sometime we may have just simple list of items where there is no ids. In such scenarios id all items are different from one another then we can use those items themselves as keys otherwise we can **use index of item as key**.

import React from "react";

function IndexAsKey() {

  const names = ["rohit", "virat", "dhoni", "pandya"];

  const nameList = names.map((name, index) => (

    <h2 key={index}>

      {name}

    </h2>

  ));

  return (

    <div>

      <h2>{nameList}</h2>

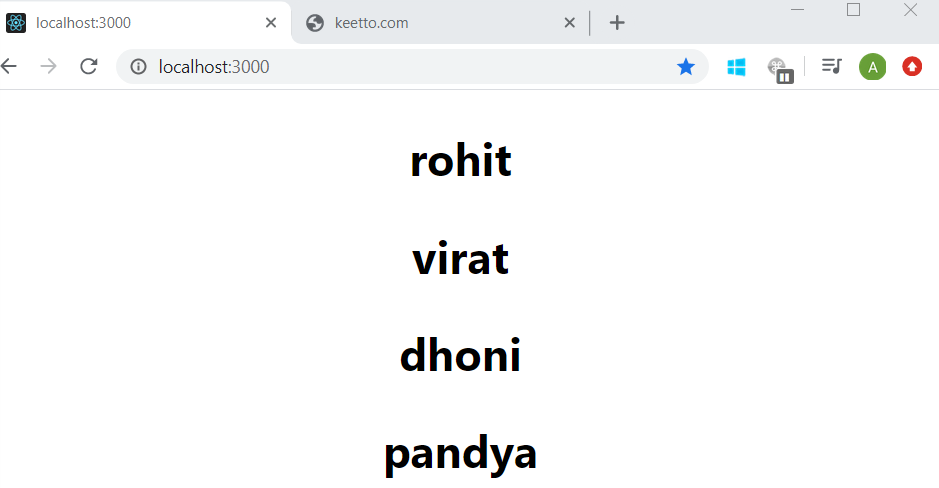
    </div>

  );

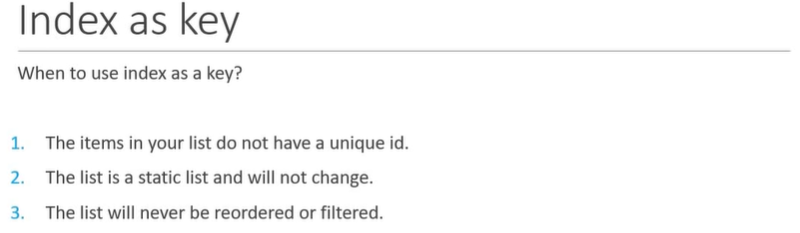
}

export default IndexAsKey;

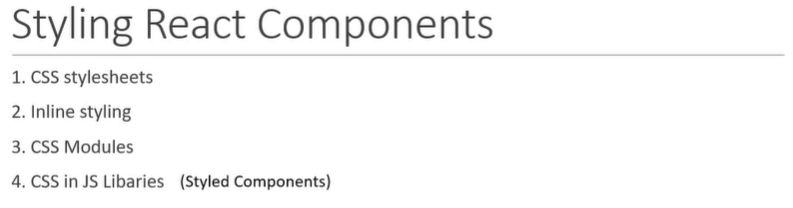
output: -



Note: -



Chapter: 11



1. **CSS stylesheet**

Hello-world > src > CSS >Mystyles.css

.primary {

  color: orange;

}

.increaseFont {

  font-size: 72px;

}

Hello-world > src > CSS >StyleSheet.js

import React from "react";

import "./Mystyles.css";

function StyleSheet() {

  return (

    <div>

      <h1 className="primary increaseFont">Stylesheets</h1>

    </div>

  );

}

export default StyleSheet;

Hello-world > src > App.js

import React from "react"

import Stylesheet from "./CSS/StyleSheet";

import "./App.css";

function App() {

  return (

    <div className="App">

<Stylesheet />

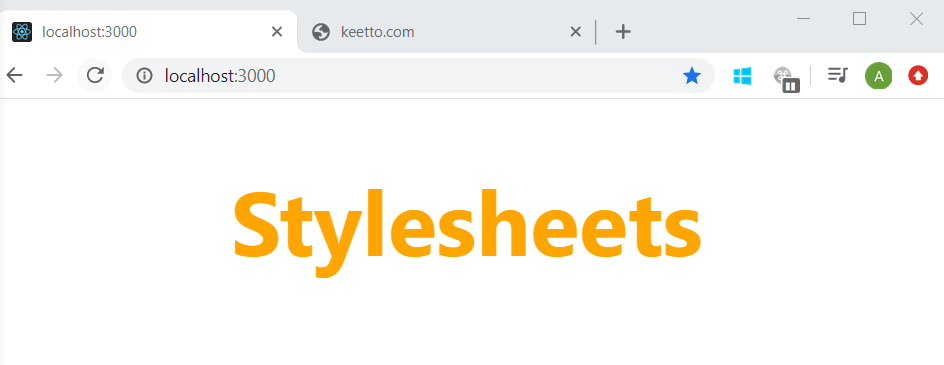
</div>

  );

}

export default App;

output: -



1. **Inline styling**

Hello-world > src > CSS >InlineStyling.js

import React from "react";

const heading = {

  fontSize: "72px",

  color: "blue"

};

function InlineStyling() {

  return (

    <div>

      <h1 style={heading}>Inline styling demo</h1>

    </div>

  );

}

export default InlineStyling;

Hello-world > src > App.js

import React from "react"

import InlineStyling from "./CSS/InlineStyling";

import "./App.css";

function App() {

  return (

    <div className="App">

<InlineStyling />

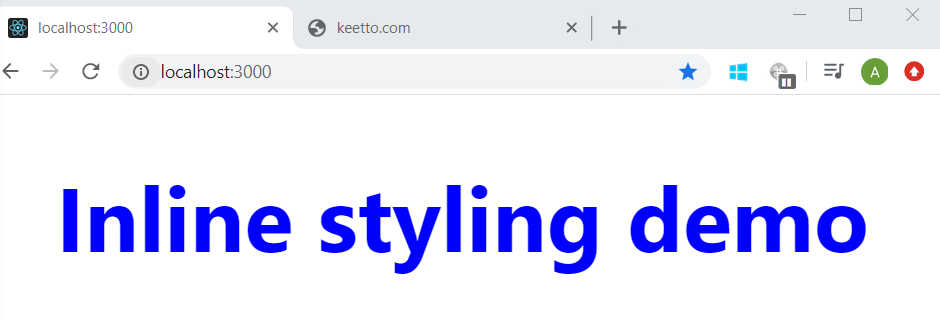
</div>

  );

}

export default App;

Output: -



1. **CSS Module**

* For using CSS Module there is file naming convention i.e. file name should end with “.module.css” e.g. If appstyle.css id regular CSS stylesheet then “appStyle.module.css” is CSS module stylesheet
* When we use regular CSS stylesheet then if we import that sheet in any component then we can use its classes the any of the children component of that component. Which may lead to unintentionally using the classes in children component. E.g. if we are importing regular CSS stylesheet in App component then we can use it’s classes in every other component.
* We have import css and module.css files in different way (see example below)
* So we can use CSS module, the main advantage of using CSS module is that **classes are locally scoped**.

Hello-world > src > CSS >appstyle.css

.showRed {

  color: red;

}

Hello-world > src > CSS >appstyle.module.css

.showGreen {

  color: green;

}

Hello-world > src > App.js

import React from "react"

import "./App.css";

import "./CSS/appStyle.css";

import style from "./CSS/appStyle.module.css";

function App() {

  return (

    <div className="App">

<h1 className="showRed">Message from regular styleSheet</h1>

      <h1 className={style.showGreen}>Message from regular styleSheet</h1>

</div>

  );

}

export default App;

output

