1 – Create React App

npx create-react-app my-app

cd my-app

npm start

2. Create first file and render Some text through react

Delete all file from src and create index.js file and write following code

import React from 'react';

import ReactDom from 'react-dom';

const message = <h1>Sample Message</h1>;

ReactDom.render(message, document.getElementById('root'));

3. Create first React component

Create a new file App.js in src folder and in that file write

import React from 'react';

class App extends React.Component {

render(){

return <h1>Sample Message</h1>

}

}

export default App;

and import App component in index.js and replace variable with App component

import App from ‘./App’;

ReactDom.render(<App />, document.getElementById(‘root’);

4. Add CSS file

create a new file index.css in src folder and in index.js file import this css file

import ‘./index.css’;

5. Add Bootstrap

npm install –save react-bootstrap

In index.js file import bootstrap

**import** 'bootstrap/dist/css/bootstrap.css';

6. Adding Styles and classes

class App extends React.Component {

styles = {

fontSize : 12,

fontWeight:normal

}

render(){

return <h1 style={this.styles} className=”score-display” > Your Score is {this.state.score}</h1>;

}

}

or

class App extends React.Component {

render(){

return <h1 style={ { fontSize:12, fontweight:’normal’} className=”score-display” > Your Score is {this.state.score}</h1>;

}

}

Note:

* attributes should be in camelCase
* values should be in quotes
* numeric value can be used without quotes
* Adding px is optional but if added this should be placed in quotes like ‘10px’
* attributes should be separated with comas
* inline styling in jsx should be wrapped in double curly braces { { } }

7. Adding State

class App extends React.Component {

state = { score : 70 }

render(){

return <h1 className={this.getClass()}>Your Score is {this.state.score}</h1>;

}

}

8. Adding class dynamically

class App extends React.Component {

state = { score : 70 }

getScoreClass(){

let classes ="blue-heading result-";

classes += this.state.score > 30 ? 'pass' : 'fail';

return classes;

}

render(){

return <h1 className={ this.getScoreClass() }>Your Score is { this.state.score }</h1>;

}

}

9. Conditional Rendering

class App extends React.Component {

state = { score : 70 }

getScoreClass(){

let classes ="blue-heading result-";

classes += this.state.score > 30 ? 'pass' : 'fail';

return classes;

}

render(){

return (

<h1 className={ this.getScoreClass() }>

{ this.state.score > 30 && <span> Great </span> }

or {this.state.score > 30 ? <span>Great! </span> : <span>Sorry! </span>}

Your Score is { this.state.score }

</h1>

);

}

}

OR

class App extends React.Component {

state = { score : 70 }

getScoreClass(){

let classes ="blue-heading result-";

classes += this.state.score > 30 ? 'pass' : 'fail';

return classes;

}

renderScoreText(){

if (this.state.score > 70 ) return <span>Great! </span>;

return <span>Sorry </span>;

}

render(){

return (

<h1 className={ this.getScoreClass() }>

{ this.renderScoreText() }

Your Score is { this.state.score }

</h1>

);

}

}

Note :

* If conditions can not be used inside JSX
* for conditional rendering in JSX either we use logical && operator or ternary operator
* or we can use if conditions in a function to conditional rendering and use that function inside JSX

10. Render Lists

state = {

score : 70,

topScorer : [“Sahil”, “Vikram”, “Deepak”],

yourScore : [

{subject : "Physics", score: 80},

{subject : "Maths", score: 60},

{subject : "Chemistry", score: 70}

]

<ul>

{this.state.topScorer.map((item) => <li key={item}>{item}</li>)}

</ul>

<table>

{ this.state.yourScore.map((item) => {

return (

<tr>

<td>{item.subject}</td>

<td> : </td>

<td>{item.score}</td>

</tr>

);

})

}

</table>

Note :

* List items should always have unique key

11. Add event handler

class App extends React.Component {

constructor(){

super();

this.handleScoreDisplay = this.handleScoreDisplay.bind(this);

}

handleScoreDisplay(){

this.state.showScore == false ? this.setState({showScore:true}) : this.setState({showScore:false})

}

render(){

return <button onClick={this.handleScoreDisplay} > Show Details Scores </button>;

}

}

or

class App extends React.Component {

handleScoreDisplay = () => {

this.state.showScore == false ? this.setState({showScore:true}) : this.setState({showScore:false})

}

render(){

return <button onClick={this.handleScoreDisplay} > Show Details Scores </button>;

}

}

Note –

* We don’t use onClick={eventhandler()}, we use onClick={eventhndler}
* We need to bind this with event handler in constructor method or if we are using arrow function for event handler we don’t need to bind this

12. Passing Event Arguments

We can not pass arguments in event like onClick={eventHandler(arg)}

we have to write inline arrow function to pass arguments like

<button onClick={ () => this.handleScoreDisplay( { id : 1 } ) } />

or <button onClick={ () => this.handleScoreDisplay( arg ) }

13. Change State

we don’t update state directly like

this.state.score = 40;

we need to use setState method to update state

this.setState({score : 40})

1. Composing Components / using subcomponents
2. Passing data as props
3. children props
4. Difference in Props and state
5. Raising events to parent
6. updating state

Q&A

* Whats npx and difference in npm and npx

***npx*** is a npm package runner. It helps to execute packages without installing explicitly**. create-react-app** is an npm package that is expected to be run only once in a project's lifecycle. Hence, it is preferred to use npx to install and run it in a single step.