Unit-2: React JS

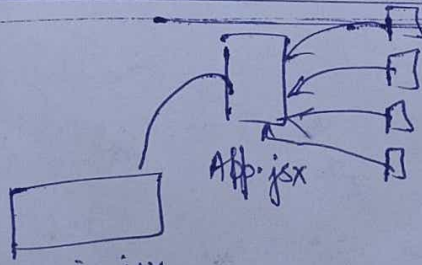
Introduction, Templating using JSX, classes using JSX, Components, State & Props, Lifecycle of components, Rendering Lists & Portals, Error Handling, Routers, Redux, & Redux Saga, Immutable.js, Service Side Rendering, Unit Testing, Webpack.

Getting Started with React App. / How to create App.

cmd
```
node -v          // to check the version
npm create vite@4.1.0 / latest
npm (create-react-app)
(Y/N)
Project name : react-app
Select a framework: >> React
Select a variant: JavaScript
Scaffolding... (Y/N)
Done. Now run:
cd react-app
npm install / i
code i.
npm run dev
```
// to create a react app.

Main.jsx
```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
import './index.css'
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

VS Code Explorer
- REACT-APP
  - node_modules
  - public
  - src
    - assets
    - App.css
    - App.jsx
    - index.css
    - main.jsx → (List of components)
    - Message.jsx
  - .gitignore
  - index.html
  - package-lock.json
  - package.json



Go to file → New file → Message.jsx
→ Make a new component
message

Message.jsx
```
function Message(){
  return <h1>Helloworld</h1>
}
export default Message;
```

App.jsx
```
import Message from "./Message";
function App(){
  return(
    <div>
      <Message/>
    </div>
  );
}
export default App;
```

React JS : It is a Java Script library for building front end application or User Interface. → It allows us to create reusable UI components. (It is created by Facebook).

(2)

1. What are the features of React?

JSX: JSX is a syntax extension to JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code.

Components: Components are the building blocks of any React application, and a single app usually consists of multiple components. It splits the user interface into independent, reusable parts that can be processed separately.

Virtual DOM: React keeps a lightweight representation of the real DOM in the memory, and that is known as the virtual DOM. When the state of an object changes, virtual DOM changes only that object in the real DOM, rather than updating all the objects.

One-way data-binding: React's one-way data binding keeps everything modular and fast. A unidirectional data flow means that when designing a React app, you often nest child components within parent components.

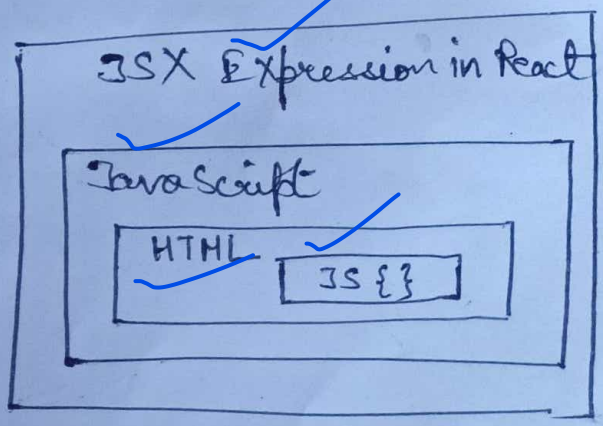
High performance: React updates only those components that have changed, rather than updating all the components at once. This results in much faster web applications.

Advantages : → Reusable Components, → Open Source → Efficient and fast → works in browser → Large community.

2. What is JSX?

JSX is a syntax extension of JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code.

```
render() {
  return (
    <div>
      <h1> This is a JSX code</h1>
    </div>
  );
}
```



Component → User.jsx

```
function User()
{ return (
  <div> {10+20} </div> ) } export default User;
```

Component 1 — Rahul

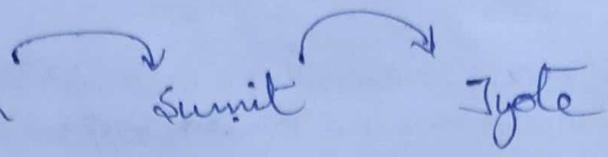
Component 2 — Sonam

Component 3 — Sumit

Component 4 — Jyoti

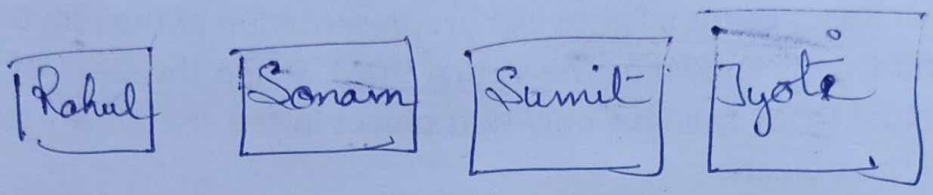
```
<div>
  Rahul
  Code2
  Code3
</div>
```

```
<div>
  Sonam
  Code2
  Code3
</div>
```



Customer

```
<div>
  Name
  Code2
  Code3
</div>
```

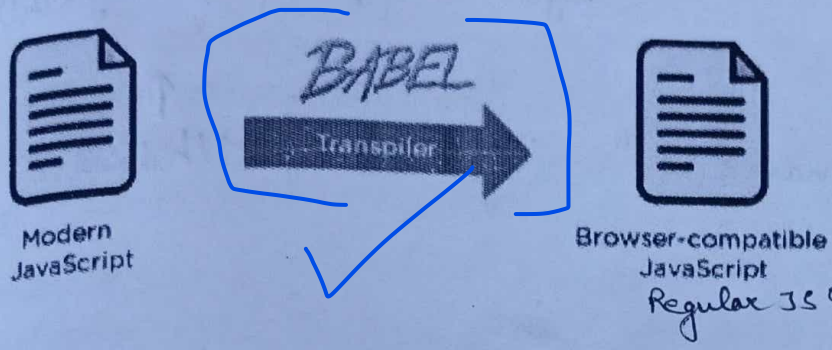



```
<Customer name
  ='Rahul'/>
```

```
<Customer
  Name='Sonam'/>
```

3. Can web browsers read JSX directly?

- Web browsers cannot read JSX directly. This is because they are built to only read regular JS objects and JSX is not a regular JavaScript object
- For a web browser to read a JSX file, the file needs to be transformed into a regular JavaScript object. For this, we use Babel

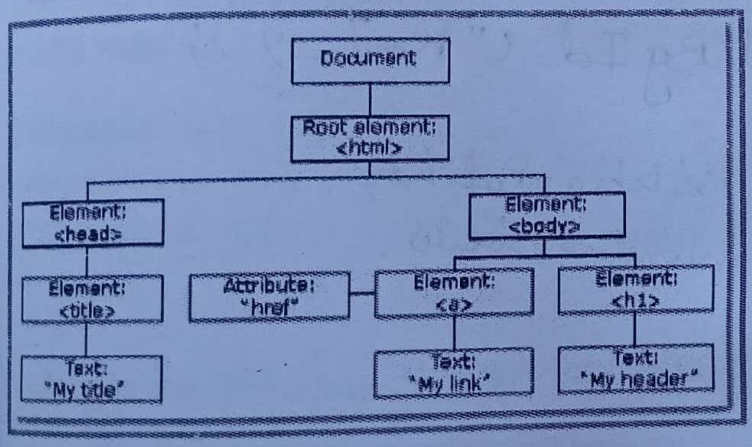


Regular JS Object.

OK!

4. What is the virtual DOM?

DOM stands for Document Object Model. The DOM represents an HTML document with a logical tree structure. Each branch of the tree ends in a node, and each node contains objects.



React keeps a lightweight representation of the real DOM in the memory, and that is known as the virtual DOM. When the state of an object changes, the virtual DOM changes only that object in the real DOM, rather than updating all the objects.

index.jsx

(4) **Template Literals** `` ` ` `` uses back tick

```
import React from "React";
import ReactDOM from "react-dom";
const fname = "Kritika";
const lname = "Balihar";

ReactDOM.render(
  <>
  <h1>my name is {fname + "  " + lname}</h1>
```

or `<h1> my name is {fname} {lname} </h1>`

Or In Literals

```
[<h1>{my name is {`${fname}  ${lname}`}</h1>
or <h1>{' my first name is ${fname} and my last name is ${lname}'}</h1>
<p> my lucky number is {5+5}</p>
</>
document.getElementById("Root"));
```

o/p: my name is Kritika Balihar.
my lucky number is 10.

5. Why use React instead of other frameworks, like Angular?

Easy creation of dynamic applications: React makes it easier to create dynamic web applications because it provides less coding and provides more functionality, whereas, with JavaScript applications, code tends to get complex very quickly.

Improved performance: React uses virtual DOM, which makes web applications perform faster. Virtual DOM compares its previous state and updates only those components in the real DOM, whose states have changed, rather than updating all the components — like conventional web applications.

Reusable components: Components are the building blocks of any React application, and a single app usually consists of multiple components. These components have their own logic and controls, and they can be reused through the application, which, in turn, dramatically reduces the development time of an application.

Unidirectional data flow: React follows a unidirectional data flow. This means that when designing a React app, we often nest child components within parent components. And since the data flows in a single direction, it becomes easier to debug errors and know where the problem occurs in an application at the moment.

Dedicated tools for easy debugging: Facebook has released a chrome extension that we can use to debug React applications. This makes the process of debugging React to web applications faster and easier.

6. What is the difference between the ES6 and ES5 standards?

This is one of the most frequently asked react interview questions.

These are the few instances where ES6 syntax has changed from ES5 syntax:

- **Components and Function**

```
// ES5
var MyComponent = React.createClass({
  render: function() {
    return(
      <h3>Hello Simplilearn</h3>
    );
  }
});

// ES6
class MyComponent extends React.Component {
  render() {
    return(
      <h3>Hello Simplilearn</h3>
    );
  }
}
```

- **exports vs export**

```
sqoop export --connect
jdbc:mysql://localhost/retail_db -username
root --password cloudera --table dept --
export-dir /user/cloudera/departments
```

- **require vs import**

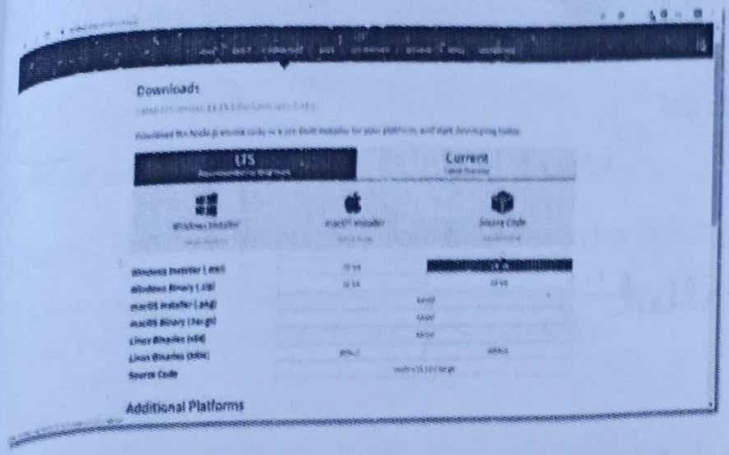
```
// ES5
var React = require('react');

// ES6
import React from 'react';
```

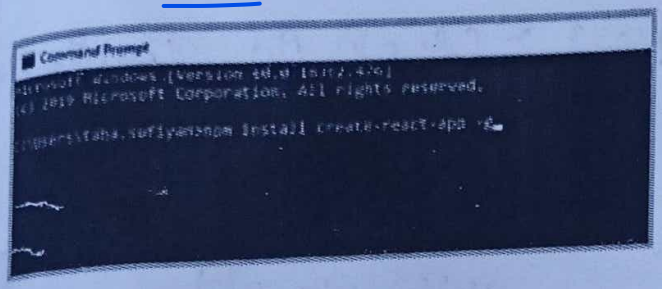
7. How do you create a React app?

These are the steps for creating a React app:

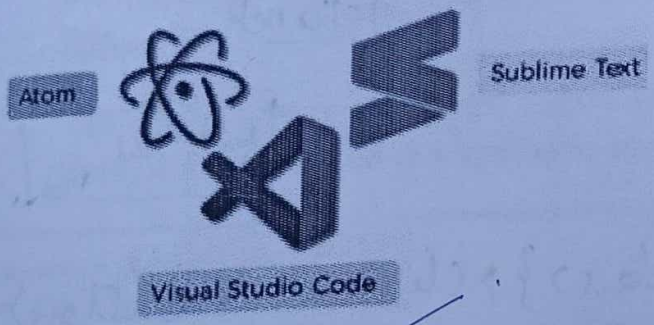
- Install NodeJS on the computer because we need npm to install the React library. Npm is the node package manager that contains many JavaScript libraries, including React.



- Install the **create-react-app** package using the command prompt or terminal.



- Install a text editor of your choice, like VS Code or Sublime Text.



8. What is an event in React?

An event is an action that a user or system may trigger, such as pressing a key, a mouse click, etc.

- React events are named using camelCase, rather than lowercase in HTML.
- With JSX, you pass a function as the event handler, rather than a string in HTML.

```
<Button onPress={lightItUp} />
```

(8)

App.jsx

```
import logo from './logo.svg';
import './App.css';
function App() {
  function apple()
   { alert ("function called");
   }
  return (
  <div className = "App">
  <h1> Hello World ! </h1>
  <button onclick = {apple} > Click Me, </button>
  </div>
  ); }
  export default App;
```

o/p :-

Hello World!

[Click Me]

[localhost:3000says
functn called
OK]

or

```
<button onclick={()=>apple()}>Click Me </button>
```

arrow

```
function app( ){
  let data = " Gtbit clg";
  function apple ()
    {
      data = "Peter";
       alter alert (data);
    }
  return (
  <div className = "App">
   <h1> {data} </h1>
  <button onclick = {apple} > Click Me </button>
  </div> ); } export default App;
```

o/p :-

Gtbit clg

[Click Me]

Peter

9. How do you create an event in React?

A React event can be created by doing the following:

```
class Simple extends React.Component {
    work() {
        alert("Good Work!");
    }
    render() {
        return (
            <button onClick={this.work}>Do some work!</button>
        );
    }
}
```

10. What are synthetic events in React?

- Synthetic events combine the response of different browser's native events into one API, ensuring that the events are consistent across different browsers.
- The application is consistent regardless of the browser it is running in. Here, **preventDefault** is a synthetic event.

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('You just clicked a Link.');
  }
  return (
    <a href="#" onClick={handleClick}>
        Click_Me
    </a>
  );
}
```

11. Explain how lists work in React

- We create lists in React as we do in regular JavaScript. Lists display data in an ordered format
- The traversal of lists is done using the map() function

```
const names = ['Kohli', 'Saif', 'Arun', 'Aamir', 'Arif'];

const listOfNames = () => {
  const listItems = names.map((name) =>
      <li key={name}>
      {name}
      </li>
  );
  return (
      <ul>{listItems}</ul>
  );
}
```

12. Why is there a need for using keys in Lists?

Keys are very important in lists for the following reasons:

- A key is a unique identifier and it is used to identify which items have changed, been updated or deleted from the lists
- It also helps to determine which components need to be re-rendered instead of re-rendering all the components every time. Therefore, it increases performance, as only the updated components are re-rendered

13. What are forms in React?

React employs forms to enable users to interact with web applications.

- Using forms, users can interact with the application and enter the required information whenever needed. Form contain certain elements, such as text fields, buttons, checkboxes, radio buttons, etc
- Forms are used for many different tasks such as user authentication, searching, filtering, indexing, etc

14. How do you create forms in React?

We create forms in React by doing the following:

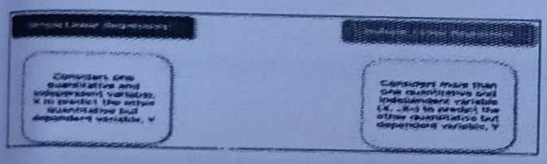
```
class NameForm extends React.Component {
 this.state = {value: ''};

 handleChange(event) {
  this.setState({value: event.target.value});
 }

 handleSubmit(event) {
  alert('A name was entered: ' + this.state.value);
  event.preventDefault();
 }

 render() {
  return (
   <form onSubmit={this.handleSubmit.bind(this)}>
    <label>
     Name:
     <input type="text" value={this.state.value}
onChange={this.handleChange.bind(this)} />
    </label>
    <input type="submit" value="Submit" />
   </form>
  );
 }
}
```

The above code will yield an input field with the label **Name** and a submit button. It will also alert the user when the submit button is pressed.



15. How do you write comments in React?

There are basically two ways in which we can write comments:

- Single-line comments

```
In [8]: #Returns sum of two values
        def sum(a, b):
            return a + b

        x = sum(4, 7)
        print(x)

        11
```

- Multi-line comments

```
Return (
  { /*
    Multi
    line
    comment
  */ }
  <div>
    <p>Hello</p>
  </div>
);
```

16. What is an arrow function and how is it used in React?

- An arrow function is a short way of writing a function to React.
- It is unnecessary to bind **'this'** inside the constructor when using an arrow function. This prevents bugs caused by the use of **'this'** in React callbacks.

Without Arrow function

```
render() {
  return(
    <MyInput onChange={this.handleChange.bind(this) } />
  );
}
```

With Arrow function

```
render() {
  return(
    <MyInput onChange={ (e) => this.handleOnChange(e) } />
  );
}
```

17. How is React different from React Native?

	React	React Native

(13)

Components: are like JS Functions. They accept arbitrary i/p (called "props") & return React elements describing what should appear on the screen.

Props → properties

→ Always start component names with a capital letter.

→ React treats components starting with lowercase letters as DOM tags. (For example, represents an HTML div tag but <App/> represents a component requires App to be in scope)

Release	2013	2015
Platform	Web	Mobile – Android, iOS
HTML	Yes	No
CSS	Yes	No

Ok!

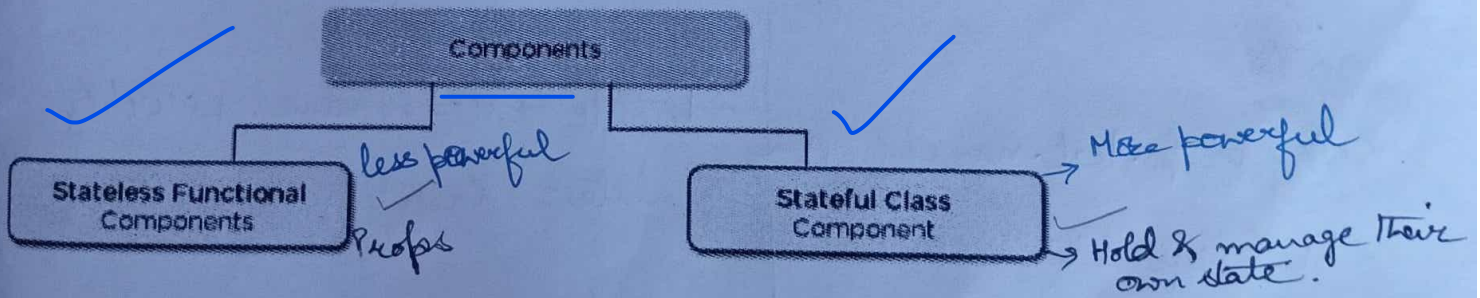
18. How is React different from Angular?

	Angular	React
Author	Google	Facebook
Architecture	Complete MVC	View layer of MVC
DOM	Real DOM	Virtual DOM
Data-Binding	Bi-directional	Uni-directional
Rendering	Client-Side	Server-Side
Performance	Comparatively slow	Faster due to Virtual DOM

19. What are the components in React?

Components are the building blocks of any React application, and a single app usually consists of multiple components. A component is essentially a piece of the user interface. It splits the user interface into independent, reusable parts that can be processed separately.

There are two types of components in React:



- **Functional Components:** These types of components have no state of their own and only contain render methods, and therefore are also called

(14) Functional component :

```
import React from 'react'
function Profile()
 {
  return (
  <div>
   <h1> Profile Cmp </h1>
  </div>
  )
 export default Profile
```

```
import Profile from './Profile'
function App() {
return (
 <div className = "App" >
  <Profile />
 </div>
 )
}
export default App;
```

→ o/p : Profile Cmp

To pass the data

```
function Profile(props)
{
return(
 <div>
  <h1> {props.text} </h1>
 </div>
```

```
<Profile text = "Hello Props" />
```

⇒ o/p : Hello Props

To pass the object

```
<Profile text = {{name: 'Peter'}}/>
```

```
<h1> {props.text.name} </h1>
```

⇒ o/p : Peter

To Pass obj & data both

```
<h1> {props.text.name} </h1>
<h1> {props.data} </h1>
```

```
<Profile text = {{name: 'peter'}}
  data = "Profile data" />
```

⇒ o/p : Peter
Profile Data

stateless components. They may derive data from other components as props (properties).

```
function Greeting(props) {
 return <h1>Welcome to {props.name}</h1>;
}
```

- **Class Components:** These types of components can hold and manage their own state and have a separate render method to return JSX on the screen. They are also called Stateful components as they can have a state.

```
class Greeting extends React.Component {
 render() {
  return <h1>Welcome to {this.props.name}</h1>;
 }
}
```

20. What is the use of render() in React?

- It is required for each component to have a render() function. This function returns the HTML, which is to be displayed in the component.
- If you need to render more than one element, all of the elements must be inside one parent tag like , <form>.

```
import React from 'react'

class App extends React.Component {
  render (){
    return (
      <h1>Hello Simplilearn</h1>
    )
  }
}
export default App
```

21. What is a state in React?

(16) Class Component | Profile.jsx

```
import React from 'react'

class Profile extends React.Component
{
  render()
  {
    return(
      <div>
        <h1> Hello props </h1>
      </div>
    )
  }
} export default profile;
```

o/p: Hello props

To Pass data & obj both

```
import React from 'react'
import Profile from './Profile'

function App(){
return(
<div class Name = "App">
  <Profile/>
</div>
);
}
export default App;
```

```
<Profile text = {{name:
'Peter'}} data = "Profile
Data"/>
```

```
<h1> {this.props.data} </h1>
<h1> {this.props.text.name} </h1>
```

o/p: Profile Data
Peter.

State → Profile.jsx
Component

```
import React from 'react'

class Profile extends React.Component
{Render()
  {return(
    <div>
    <h1> Hello State </h1>
    </div>
    );}}
  export default profile
```

```
import React from 'react'
import Profile from './Profile'
function App() {
return(
<div>
  <Profile/>
</div>
);
}
export default App;
```

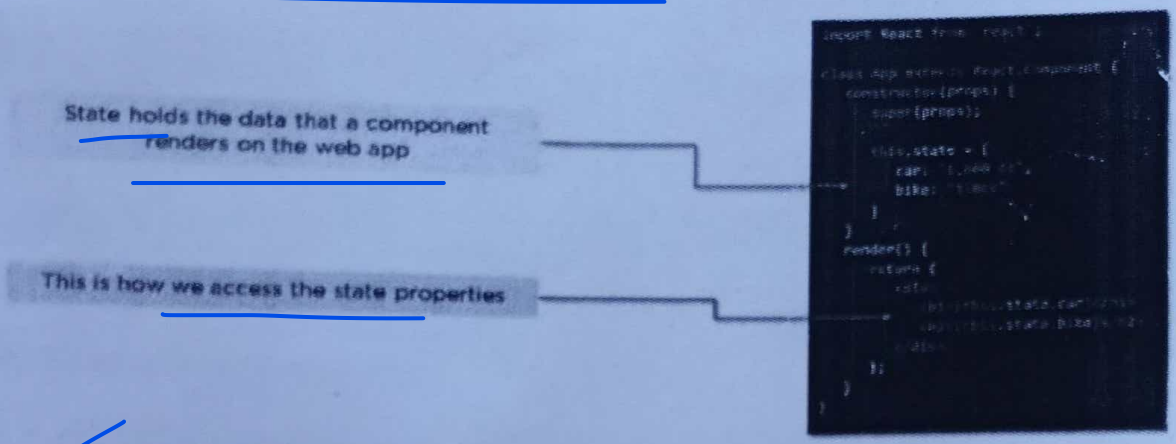
o/p: Hello State

State

(it is private)

- The state is a built-in React object that is used to contain data or information about the component. The state in a component can change over time, and whenever it changes, the component re-renders.
- The change in state can happen as a response to user action or system-generated events. It determines the behavior of the component and how it will render.

22. How do you implement state in React?



23. How do you update the state of a component?

We can update the state of a component by using the built-in **'setState()'** method:

State Implementation

Profile.jsx

```
class Profile extends React.Component
{
  constructor()
  {
    super();
    this.state = {
      name : 'peter';
      email : 'peter@test.com'
    }
  }
  render()
  {
    return(
      <div>
        <h1> Hello {this.state.name} </h1>
        <h1> Email : {this.state.email} </h1>
      </div>
    );
  }
```

o/p: Hello peter
Email : peter@test.com

To Update the state

```
updateState()
{ this.setState({
  name : 'Bruce'
});
```

```
<button onClick={()=>
this.updateState()}}> Update
</button>
```

o/p: Hello peter
Email : peter@test.com
Hello Bruce

```
class App extends React.Component {
  constructor() {
    super();
    this.state = {
      message: "Welcome to Simplilearn"
    };
    this.buttonPress = this.buttonPress.bind(this);
  }
  buttonPress() {
    this.setState({
      message:"The best place to learn"
    });
  }
  render() {
    return (
      <div>
        <h1>{this.state.msg}</h1>
        <button onClick = {this.buttonPress}>Click Me!</button>
      </div>
    );
  }
}
```

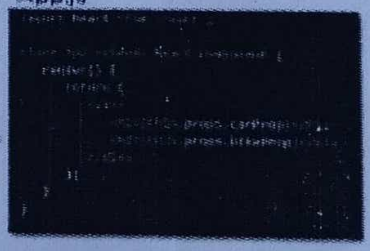
24. What are props in React?

- Props are short for Properties. It is a React built-in object that stores the value of attributes of a tag and works similarly to HTML attributes.
- Props provide a way to pass data from one component to another component. Props are passed to the component in the same way as arguments are passed in a function.

25. How do you pass props between components?

App.js

This is how we access the properties passed to a component



This is how we pass the properties to a component

Main.js

26. What are the differences between state and props?

	State	Props
Use	Holds information about the components	Allows to pass data from one component to other components as an argument
Mutability	Is mutable	Are immutable
Read-Only	Can be changed	Are read-only
Child components	Child components cannot access	Child component can access
Stateless components	Cannot have state	Can have props

27. What is a higher-order component in React?

A higher-order component acts as a container for other components. This helps to keep components simple and enables re-usability. They are generally used when multiple components have to use a common logic.

28. How can you embed two or more components into one?

We can embed two or more components into one using this method:

```
class App extends React.Component {
  render (){
    return (
      <div>
      <h1>Hello</h1>
      <Simple/>
      </div>
    )
  }
}

class Simple extends React.Component {
  render (){
    return (
      <h1>Simplilearn</h1>
    )
  }
}

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

29. What are the differences between class and functional components?

	Class Components	Functional Components
State	Can hold or manage state	Cannot hold or manage state
Simplicity	Complex as compared to the stateless component	Simple and easy to understand
Lifecycle methods	Can work with all lifecycle methods	Does not work with any lifecycle method
Reusability	Can be reused	Cannot be reused

- **Class components example:**

Component Life Cycle

① **Mounting Component**: These methods are called in the following order when an instance of a component is being created & inserted into the DOM:

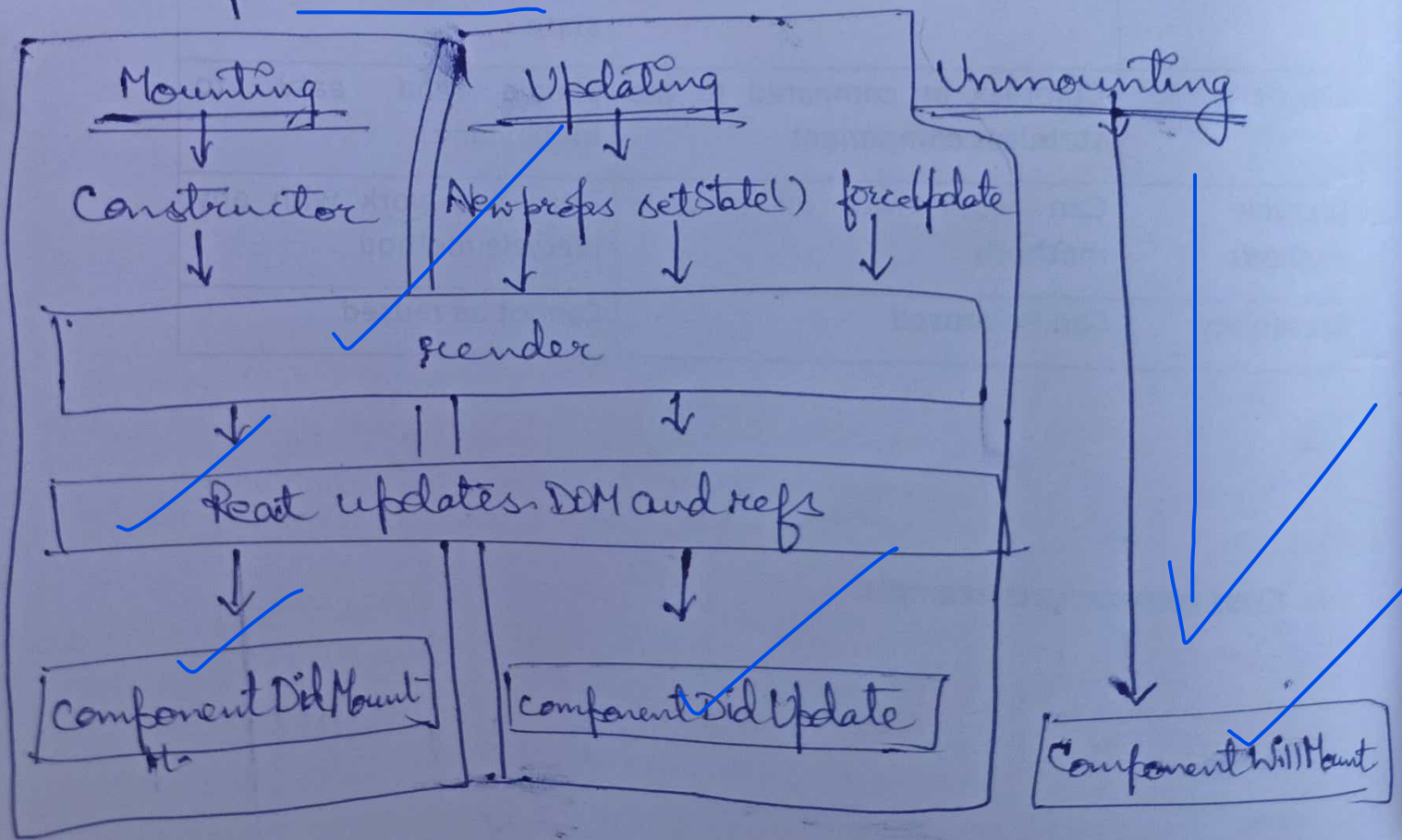
- constructor()
- static getDerivedStateFromProps()
- render()
- componentDidMount()

② **Updating Component**: An update can be caused by changes to props or state. These methods are called in the following order when a component is being re-rendered:

- static getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

③ **Unmounting Component**:

- componentWillUnmount()



```
class StatefulComponent extends React.Component
{
    render() {
        return <div>{this.props.title}</div>;
    }
}
```

- **Functional components example:**

```
const StatelessComponent =
        props => <div>{this.props.title}</div>;
```

30. Explain the lifecycle methods of components.

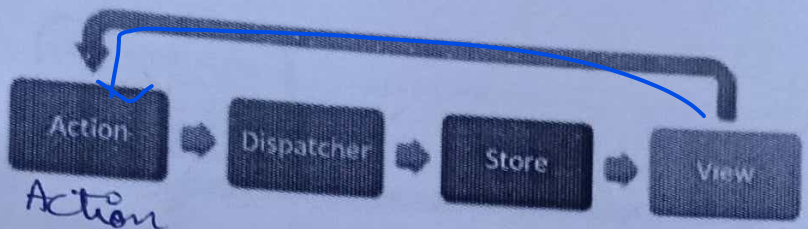
- **getInitialState():** This is executed before the creation of the component.
- **componentDidMount():** Is executed when the component gets rendered and placed on the DOM.
- **shouldComponentUpdate():** Is invoked when a component determines changes to the DOM and returns a "true" or "false" value based on certain conditions.
- **componentDidUpdate():** Is invoked immediately after rendering takes place.
- **componentWillUnmount():** Is invoked immediately before a component is destroyed and unmounted permanently. (this method is cld when a component is being removed from the DOM.)

31. What is Redux?

Redux is an open-source, JavaScript library used to manage the application state. React uses Redux to build the user interface. It is a predictable state container for JavaScript applications and is used for the entire application's state management.

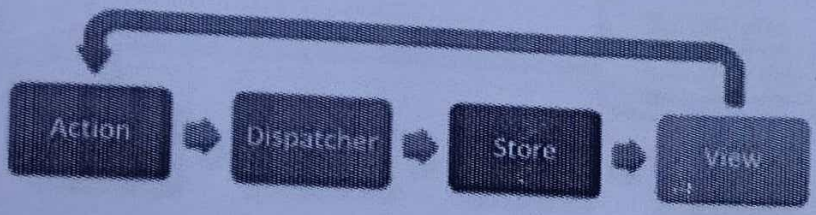
32. What are the components of Redux?

- **Store:** Holds the state of the application.
- **Action:** The source information for the store.
- **Reducer:** Specifies how the application's state changes in response to actions sent to the store.

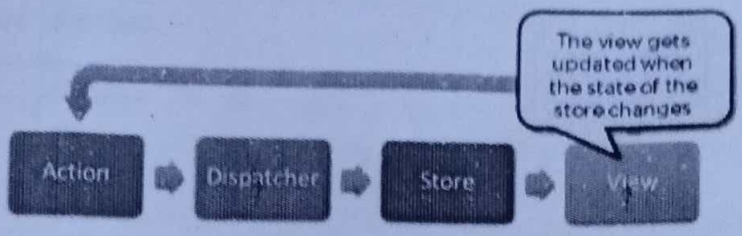


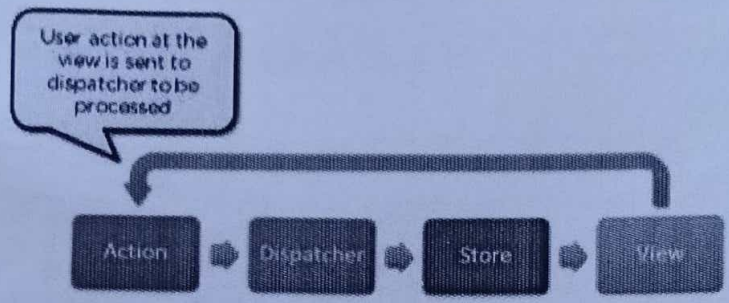
33. What is the Flux?

- Flux is the application architecture that Facebook uses for building web applications. It is a method of handling complex data inside a client-side application and manages how data flows in a React application.

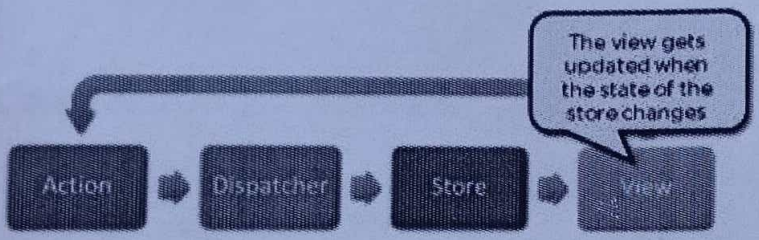


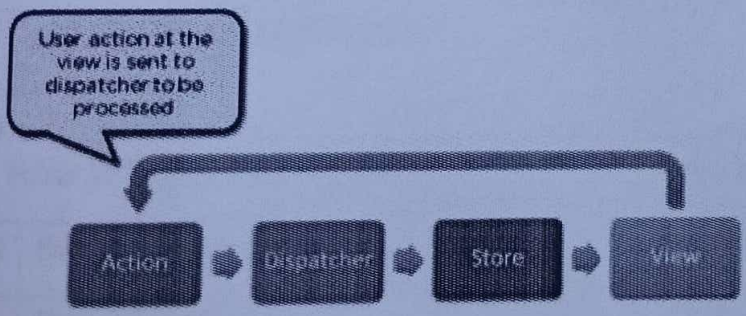
- There is a single source of data (the store) and triggering certain actions is the only way way to update them.The actions call the dispatcher, and then the store is triggered and updated with their own data accordingly.





- When a dispatch has been triggered, and the store updates, it will emit a change event that the views can rerender accordingly.





34. How is Redux different from Flux?

SN	Redux	Flux
1.	Redux is an open-source JavaScript library used to manage application State	Flux is an architecture and not a framework or library
2.	Store's state is immutable	Store's state is mutable
3.	Can only have a single-store	Can have multiple stores
4.	Uses the concept of reducer	Uses the concept of the dispatcher

35. What is React Router?

React Router is a routing library built on top of React, which is used to create routes in a React application. This is one of the most frequently asked react interview questions.

36. Why do we need to React Router?

- It maintains consistent structure and behavior and is used to develop single-page web applications.
- Enables multiple views in a single application by defining multiple routes in the React application.

37. How is React routing different from conventional routing?

SN	React Routing	Conventional routing
1.	Single HTML page	Each view is a new HTML file
2.	The user navigates multiple views in the same file	The user navigates multiple files for each view
3.	The page does not refresh since it is a single file	The page refreshes every time user navigates

4.	Improved performance	Slower performance

38. How do you implement React routing?

We can implement routing in our React application using this method:

Considering we have the components **App**, **About**, and **Contact** in our application:

```
const routing = (
  <Router>
    <div>
      <h1>React Router Example</h1>
      <Route path="/" component={App} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
```

39. How do you style React components?

There are several ways in which we can style React components:

- **Inline Styling**

```
class Simple extends React.Component {
  render() {
    return (
      <div>
      <h1 style={{color: "blue"}}>Hello Simple!</h1>
      </div>
    );
  }
}
```

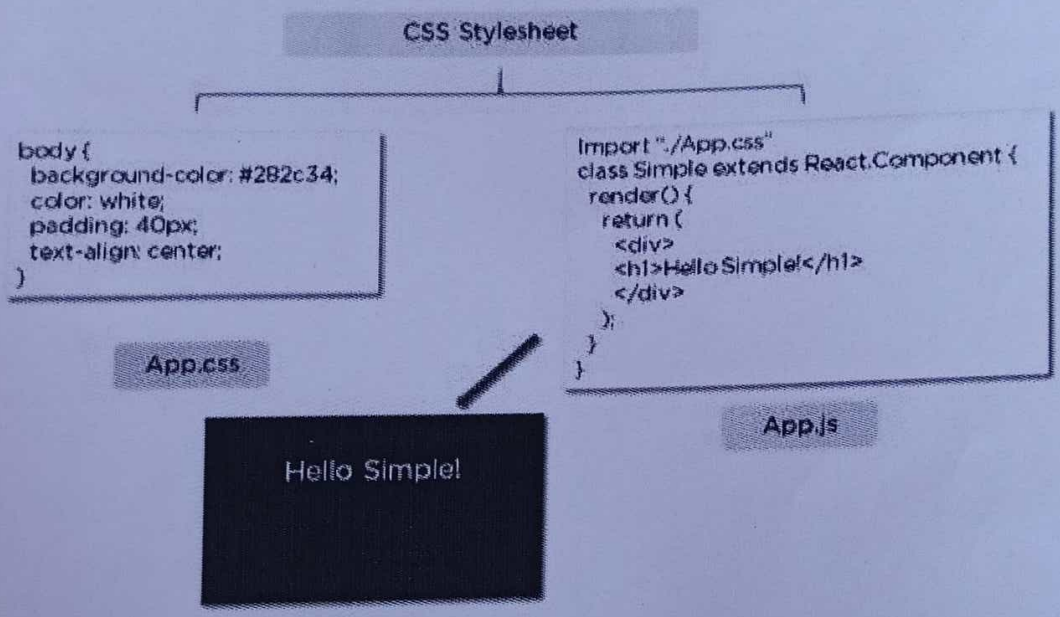
Hello Simple!

- **JavaScript Object**

```
class Simple extends React.Component {
 render() {
  const simpleStyle = {
   color: "white",
   backgroundColor: "Green",
   margin: "8px",
   fontFamily: "Open Sans"
  };
  return (
   <div>
   <h1 style={simpleStyle}>Hello Simple!</h1>
   </div>
  );
 }
}
```

Hello Simple!

- **CSS Stylesheet**



40. Explain the use of CSS modules in React.

- The CSS module file is created with the **.module.css** extension
- The CSS inside a module file is available only for the component that imported it, so there are no naming conflicts while styling the components.