2)Write a program to use and update the state object?

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  changeColor = () => {
    this.setState({color: "blue"});
  }
  render() {
    return (
      <div>
        <h1>My {this.state.brand}</h1>
        <p>
          It is a {this.state.color}
          {this.state.model}
          from {this.state.year}.
        </p>
        <button
          type="button"
          onClick={this.changeColor}
        >Change color</button>
      </div>
    );
  }
}
```

3) Write a program to create an error boundary to handle errors in the render phase:

Error boundaries are React components that **catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI** instead of the component tree that crashed. Error boundaries catch errors during rendering, in lifecycle methods, and in constructors of the whole tree below them.

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { error: null, errorInfo: null };
  }
  componentDidCatch(error, errorInfo) {
    // Catch errors in any components below and re-render with error message
    this.setState({
      error: error,
      errorInfo: errorInfo
    })
    // You can also log error messages to an error reporting service here
  }
  render() {
    if (this.state.errorInfo) {
      // Error path
      return (
        <div>
```

```
          <h2>Something went wrong.</h2>
          <details style={{ whiteSpace: 'pre-wrap' }}>
            {this.state.error && this.state.error.toString()}
            <br />
            {this.state.errorInfo.componentStack}
          </details>
        </div>
      );
    }
    // Normally, just render children
    return this.props.children;
  }
}
class BuggyCounter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { counter: 0 };
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState(({counter}) => ({
      counter: counter + 1
    }));
  }
  render() {
    if (this.state.counter === 5) {
      // Simulate a JS error
      throw new Error('I crashed!');
    }
    return <h1 onClick={this.handleClick}>{this.state.counter}</h1>;
  }
}


function App() {
  return (
    <div>
      <p>These two counters are each inside of their own error boundary. So if one crashes, the other is not affected.</p>
      <ErrorBoundary><BuggyCounter /></ErrorBoundary>
      <ErrorBoundary><BuggyCounter /></ErrorBoundary>
    </div>
  );
}


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

4) Write down the Program to prevent re-rendering.

App.js

```
import React from 'react';
import ReactDOM from 'react-dom/client'


const Greeting = React.memo(props => {
  console.log("Greeting Comp render");
  return <h1>Hi {props.name}!</h1>;
});

function App() {
  const [counter, setCounter] = React.useState(0);
  React.useEffect(() => {
    setInterval(() => {
      setCounter(counter + 1);
    }, 2000);
  }, []);

  console.log('App render')
  return <Greeting name="Ruben" />
}
export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import './App.js'
import App from './App.js';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />)
```

5) What are the different ways to style a React component? Define them with example?

There are about eight different ways to styling React Js components, there names and explanations of some of them are mentioned below.

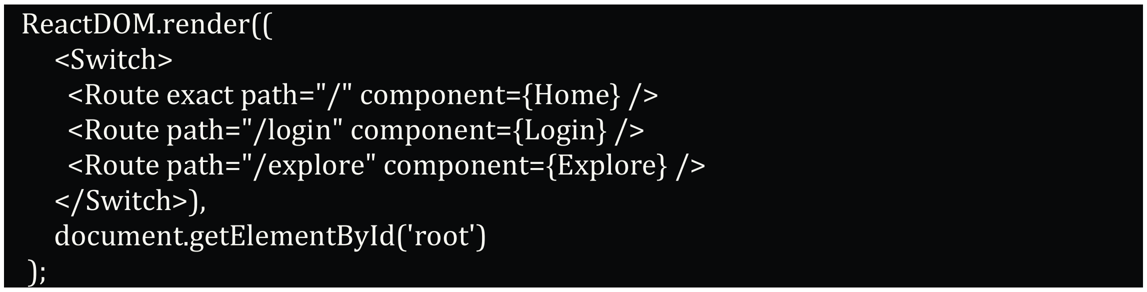
1. **Inline CSS**
2. **Normal CSS**
   Normal CSS**:** In the external CSS styling technique, we basically create an external CSS file for each component and do the required styling of classes. and use those class names inside the component.
3. **CSS in JS**
   CSS in JS: The'react-jss' integrates JSS with react app to style components. It helps to write CSS with Javascript and allows us to describe styles in a more descriptive way.
4. **Styled Components**
   Style components is a third party package using which we can create a component as a JavaScript variable that is already styled with CSS code and used that styled component in our main component.
5. **CSS module**
   CSS Modules: A CSS module is a simple CSS file but a key difference is by default when it is imported every class name and animation inside the CSS module is scoped locally to the component that is importing it also CSS file name should follow the format **'filename.module.css'**.
6. **Sass & SCSS**

Sass is the most stable, powerful, professional-grade CSS extension language. It’s a CSS preprocessor that adds special features such as variables, nested rules, and mixins or syntactic sugar in regular CSS.

7. **Less**

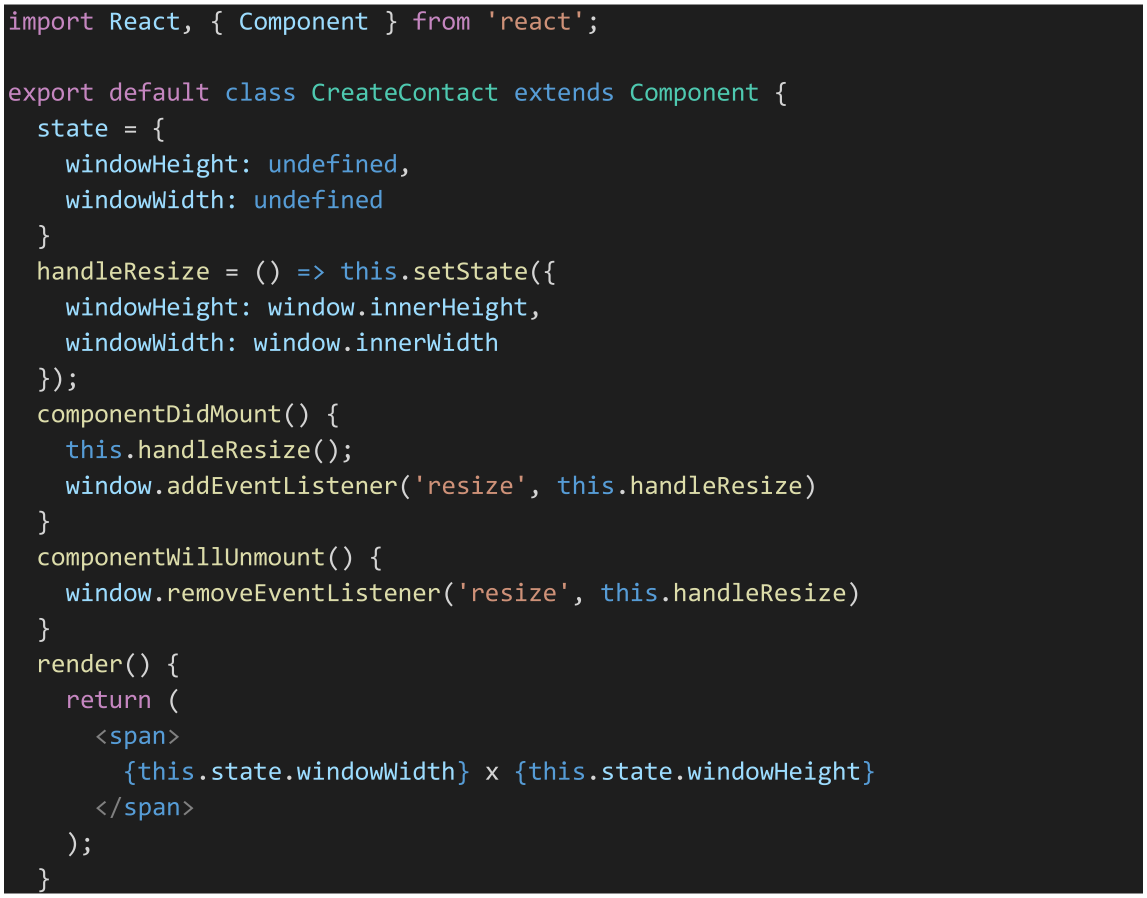
8. **Stylable**

6) Write down the program to create a switching component for displaying different pages?



```
ReactDOM.render((
   <Switch>
    <Route exact path="/" component={Home} />
    <Route path="/login" component={Login} />
    <Route path="/explore" component={Explore} />
   </Switch>),
   document.getElementById('root')
 );
```

7)How to re-render the view when the browser is resized?



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

export default class CreateContact extends Component {
  state = {
    windowHeight: undefined,
    windowWidth: undefined
  }
  handleResize = () => this.setState({
    windowHeight: window.innerHeight,
    windowWidth: window.innerWidth
  });
  componentDidMount() {
    this.handleResize();
    window.addEventListener('resize', this.handleResize)
  }
  componentWillUnmount() {
    window.removeEventListener('resize', this.handleResize)
  }
  render() {
    return (
      <span>
        {this.state.windowWidth} x {this.state.windowHeight}
      </span>
    );
  }
}
```

8) Write down the program to perform automatic redirect after login?

```
const nav = useNavigate();
export default function Login() {
async function handleSubmit(event) {
  event.preventDefault();
  try {
    await Auth.signIn(email, password);
    userHasAuthenticated(true);
    nav("/");
  } catch (e) {
    alert(e.message);
  }
}
```

9)Write down the program to create forms in React?

```
export default class NameForm extends React.Component {
    constructor(props) {
      super(props);
      this.state = {value: ''};

      this.handleChange = this.handleChange.bind(this);
      this.handleSubmit = this.handleSubmit.bind(this);
    }
    handleChange(event) {
      this.setState({value: event.target.value});
    }
    handleSubmit(event) {
      alert('A name was submitted: ' + this.state.value);
      event.preventDefault();
    }
    render() {
      return (
        <form onSubmit={this.handleSubmit}>
          <label>
            Name:
```

```
            <input type="text" value={this.state.value}
onChange={this.handleChange} />
          </label>
          <input type="submit" value="Submit" />
        </form>
      );
    }
}
```

10) Why do class methods need to be bound to a class instance, and how can you avoid the need for binding?

Part B:

1) What are the features of React?

Features of React:

- JSX (JavaScript Syntax Extension)

  JSX(JavaScript Syntax Extension): JSX is a combination of HTML and JavaScript. You can embed JavaScript objects inside the HTML elements.

- Virtual DOM

  React uses virtual DOM which is an exact copy of real DOM. Whenever there is a modification in the web application, the whole virtual DOM is updated first and finds the difference between real DOM and Virtual DOM. Once it finds the difference, then DOM updates only the part that has changed recently and everything remains the same.

- One-way data binding

  One-way data binding, the name itself says that it is a one-direction flow. The data in react flows only in one direction i.e. the data is transferred from top to bottom i.e. from parent components to child components.

- Performance

  React uses virtual DOM and updates only the modified parts. So , this makes the DOM to run faster. DOM executes in memory so we can create separate components which makes the DOM run faster.

- Extensions

  React has many extensions that we can use to create full-fledged UI applications. It supports mobile app development and provides server-side rendering.

- Conditional statements

  JSX allows us to write conditional statements. The data in the browser is displayed according to the conditions provided inside the JSX.

- Components

  React.js divides the web page into multiple components as it is component-based. So the component logic which is written in JavaScript makes it easy and run faster and can be reusable.

- Simplicity

  React.js is a component-based which makes the code reusable and React.js uses JSX which is a combination of HTML and JavaScript. This makes code easy to understand and easy to debug and has less code.

2) What is JSX?

3) Why use React instead of other frameworks, like Angular?

React is better than Angular **due to it's virtual DOM implementation and rendering optimizations**. Migrating between React's versions is quite easy, too; you don't need to install updates one by one, as in the case of Angular. Finally, with React, developers have myriads of existing solutions they can use.

Compared to other frontend frameworks, the React code is **easier to maintain and is flexible due to its modular structure**. This flexibility, in turn, saves huge amount of time and cost to businesses.

4) What are synthetic events in React?

In order to work as a cross-browser application, React has created a wrapper same as the native browser in order to avoid creating multiple implementations for multiple methods for multiple browsers, creating common names for all events across browsers. Another benefit is that it increases the performance of the application as React reuses the event object.

It pools the event already done hence improving the performance.

5) What are keys in React?

A key is a unique identifier. In React, it is used to identify which items have changed, updated, or deleted from the Lists. It is useful when we dynamically created components or when the users alter the lists. It also helps to determine which components in a collection needs to be re-rendered instead of re-rendering the entire set of components every time.

6)What are the differences between functional components and class components?

Functional Components	**Class Components**
A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX).	A class component requires you to extend from React. Component and create a render function which returns a React element.
There is no render method used in functional components.	It must have the render() method returning JSX (which is syntactically similar to HTML)
Functional component run from top to bottom and once the function is returned it cant be kept alive.	Class component is instantiated and different life cycle method is kept alive and being run and invoked depending on phase of class component.
Also known as Stateless components as they simply accept data and display them in some form, that they are mainly responsible for rendering UI.	Also known as Stateful components because they implement logic and state.
React lifecycle methods (for example, componentDidMount) cannot be used in functional components.	React lifecycle methods can be used inside class components (for example, componentDidMount).
Hooks can be easily used in functional components to make them Stateful.	It requires different syntax inside a class component to implement hooks.

example: const [name,SetName]= React.useState(‘ ‘)	example: constructor(props) {
	super(props);
	this.state = {name: ‘ ‘}
	}

Constructors are not used.	Constructor are used as it needs to store state.

7) What is the virtual DOM? How does react use the virtual DOM to render the UI

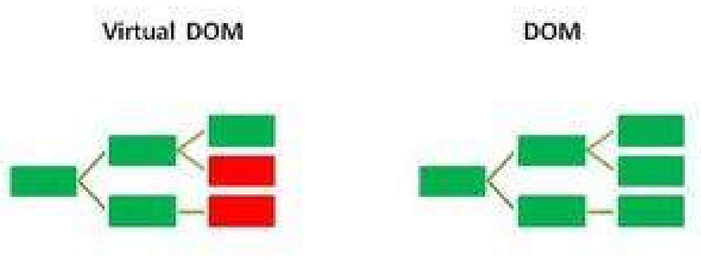
DOM: DOM stands for ‘Document Object Model’. In simple terms, it is a structured representation of the HTML elements that are present in a webpage or web-app. DOM represents the entire UI of your application. The DOM is represented as a tree data structure. It contains a node for each UI element present in the web document.

Updating DOM

Every time there is a change in the state of your application, the DOM gets updated to reflect that change in the UI. Though doing things like this is not a problem and it works fine, consider a case where we have a DOM that contains nodes in a large number, and also all these web elements have different styling and attributes.

Virtual DOM:

It is a copy of the Real DOM. Whenever there is a change in the state of any element, a new Virtual DOM tree is created. This new Virtual DOM tree is then compared with the previous Virtual DOM tree and make a note of the changes. After this, it finds the best possible ways to make these changes to the real DOM. Now only the updated elements will get rendered on the page again.



8) What are the differences between controlled and uncontrolled components?

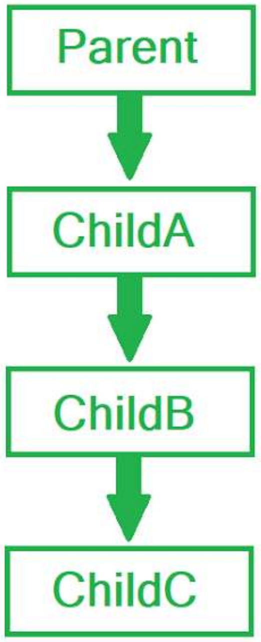
- In a controlled component, form data is handled by a React component. Whereas in uncontrolled components, form data is handled by the DOM itself.
- Usage of Component State is a must for controlled components. Use of state is completely optional for uncontrolled components, but one must use Refs in it.
- With controlled components, we can validate as the input is being changed but the same is not possible with uncontrolled components.

9)Explain react state and props.

SN	Props	State
1.	Props are read-only.	State changes can be asynchronous.
2.	Props are immutable.	State is mutable.
3.	Props allow you to pass data from one component to other components as an argument.	State holds information about the components.
4.	Props can be accessed by the child component.	State cannot be accessed by child components.
5.	Props are used to communicate between components.	States can be used for rendering dynamic changes with the component.
6.	Stateless component can have Props.	Stateless components cannot have State.
7.	Props make components reusable.	State cannot make components reusable.
8.	Props are external and controlled by whatever renders the component.	The State is internal and controlled by the React Component itself

10) What is prop drilling in React?

Prop drilling is basically a situation when the same data is being sent at almost every level due to requirements in the final level. Here is a diagram to demonstrate it better. Data needed to be sent from *Parent* to *ChildC.* In this article different ways to do that are discussed.



12) What is React Hooks? Explain it?

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

Hooks are backward-compatible, which means it does not contain any breaking changes. Also, it does not replace your knowledge of React concepts.

1. Only call Hooks at the top level

Do not call Hooks inside loops, conditions, or nested functions. Hooks should always be used at the top level of the React functions. This rule ensures that Hooks are called in the same order each time a components renders.

2. Only call Hooks from React functions

You cannot call Hooks from regular JavaScript functions. Instead, you can call Hooks from React function components. Hooks can also be called from custom Hooks.

13) Explain Strict Mode in React.

StrictMode is a tool for highlighting potential problems in an application. Like Fragment, StrictMode does not render any visible UI. It activates additional checks and warnings for its descendants.

14) How to prevent re-renders in React?

Using React.memo

15) Name a few techniques to optimize React app performance.

1. Using Immutable Data Structures
2. Function/Stateless Components and React.PureComponent
3. Multiple Chunk Files
4. Use React.Fragments to Avoid Additional HTML Element Wrappers
5. Avoid Inline Function Definition in the Render Function.
6. Avoid using Index as Key for map
7. Avoiding Props in Initial States
8. Memoize React Components

16) What are the different phases of the component lifecycle?

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.

The three phases are: **Mounting**, **Updating**, and **Unmounting**.

Mounting means putting elements into the DOM.

React has four built-in methods that gets called, in this order, when mounting a component:

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()

The next phase in the lifecycle is when a component is *updated*.

A component is updated whenever there is a change in the component's `state` or `props`.

React has five built-in methods that gets called, in this order, when a component is updated:

1. `getDerivedStateFromProps()`
2. `shouldComponentUpdate()`
3. `render()`
4. `getSnapshotBeforeUpdate()`
5. `componentDidUpdate()`

The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

React has only one built-in method that gets called when a component is unmounted:

- `componentWillUnmount()`

18) What is React Router?

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.