

Web Application Development

Module 4 QB Solutions

Part A

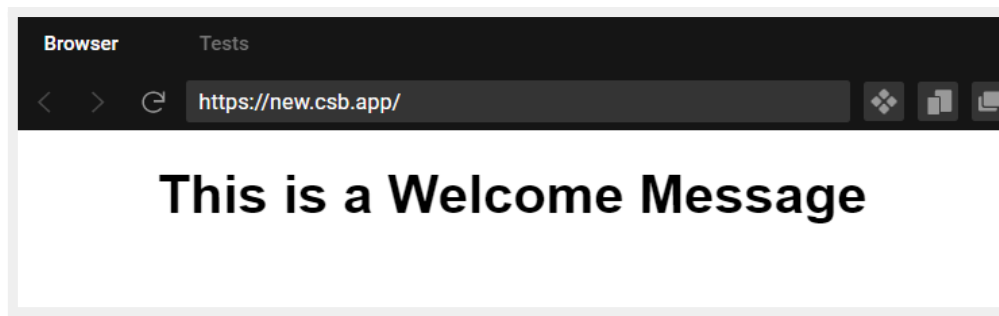
By Rishi Raj

THIS IS THE INDEX.JS FILE FOR ALL ANSWERS IN PART A

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import App from "./App";
const rootElement = document.getElementById("root");
const root = createRoot(rootElement);
root.render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

1. Develop the functional component file and display the welcome message ?

```
import "./styles.css";
export default function App() {
  return (
    <div className="App">
      <h1>This is a Welcome Message</h1>
    </div>
  );
}
```



2. Develop the class component file and display the welcome message ?

```
import React, { Component } from "react";
class ClassComponent extends React.Component{
  render(){
    return(
      <h1>This is a welcome message</h1>
    )
  }
}
export default ClassComponent
```

Same output from q1

3. How to run the JSX file and display the message

Example

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

In the example below, we embed the result of calling a JavaScript function, `formatName(user)`, into an `<h1>` element.

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};
```

```
const element = (  
  <h1>  
    Hello, {formatName(user)}!  
  </h1>  
)  
);
```

4. Develop a Program to pass the props in functional components?

```
import './styles.css';  
const Character=(props)=>{  
  return(  
    <>  
      <h1>{props.name}</h1>  
      <h1>{props.branch}</h1>  
    </>  
  )  
}  
export default function App() {  
  return (  
    <Character name="Rishi" branch="CSE" />  
  );  
};
```

Output:



5. Develop a Program to pass the props in class components ?

```
import './styles.css';  
import React, { Component } from "react";  
class Character extends Component {  
  render() {  
    return <h1>{this.props.name},{this.props.branch}</h1>;  
  }  
}
```


```
export default class App extends React.Component{
  render(){return (
    <Character name="Rishi" branch="CSE" />
  )};}
```

6. Write down the program to create a state component and subscribe button , when clicked the button to display a thank you message?

```
import React, { Component } from "react";
class Greetings extends React.Component {
  state = {
    word: ""};
  updateName() {
    this.setState({word:"Thank You"})}
  render() {
    return(
      <div>
        <h1>{this.state.word}</h1>
        <button onClick={()=>{this.updateName()}}>SUBSCRIBE</button>
      </div>)}
  }
export default Greetings;
```

Unsubscribe

SUBSCRIBE



```

import React, { Component } from "react";

class subscribe extends Component {
  state = {
    message: "",
  };
  name() {
    this.setState({ message: "Thank You" });
  }
  render() {
    return (
      <div>
        <h1>Subscribe by clicking button</h1>
        <button onClick={() => { this.name() }}>
          Subscribe
        </button>
        <h2>{this.state.message}</h2>
      </div>
    );
  }
}

export default subscribe;

```

7. Compare the difference between the props and states in react js?

<https://www.geeksforgeeks.org/what-are-the-differences-between-props-and-state/>

PROPS	STATE
The Data is passed from one component to another.	The Data is passed within the component only.
It is Immutable (cannot be modified).	It is Mutable (can be modified).

Props can be used with state and functional components.	State can be used only with the state components/class component (Before 16.0).
Props are read-only.	State is both read and write.

8. Make use of life cycle methods in class components?

[Refer to this link](#)

Lifecycle of Components

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

Mounting means putting elements into the DOM.

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

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

The render() method is required and will always be called, the others are optional and will be called if you define them.

Updating

The next phase in the life cycle 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:

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

The `render()` method is required and will always be called, the others are optional and will be called if you define them.

Unmounting

The next phase in the life cycle 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()`

Part B

Ujjwal Acharya

1. What are the features of React?

React is a JavaScript Library created by Facebook for creating dynamic and interactive applications and building better UI/UX design for web and mobile applications. React is an open-source and component-based front-end library.

React is responsible for the UI design. React makes code easier to debug by dividing them into components.

Features of React:

- JSX (JavaScript Syntax Extension)
- Virtual DOM
- One-way data binding
- Performance
- Extensions
- Conditional statements
- Components
- Simplicity

Let's understand each of them in detail.

1. JSX(JavaScript Syntax XML): JSX is a combination of HTML and JavaScript. You can embed JavaScript objects inside the HTML elements. JSX is not supported by the browsers, as a result Babel compiler transpile the code into JavaScript code. JSX makes codes easy and understandable.

```
const name="QB Solutions";  
const ele = <h1>Welcome to {name}</h1>;
```

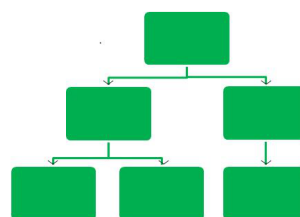
2. Virtual DOM: DOM stands for Document Object Model. It is the most important part of the web as it divides into modules and executes the code. Usually, JavaScript Frameworks updates the whole DOM at once, which makes the web application slow. But react uses virtual DOM which is an exact copy of real 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.



In the above-shown figure, when the whole virtual DOM has updated there is a change in the child components. So, now DOM finds the difference and updates only the changed part.

3. 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. The properties(props) in the child component cannot return the data to its parent component but it can have communication with the parent components to modify the states according to the provided inputs. This is the working process of one-way data binding. This keeps everything modular and fast.

One-way Data Binding



As shown in the above diagram, data can flow only from top to bottom.

4. Performance: As we discussed earlier, react uses virtual DOM and updates only the modified parts. So , this makes the DOM run faster. DOM executes in

memory so we can create separate components which makes the DOM run faster.

5. Extension: React has many extensions that we can use to create full-fledged UI applications. It supports mobile app development and provides server-side rendering. React is extended with Flux, Redux, React Native, etc. which helps us to create good-looking UI.

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

Syntax:

```
const age = 12;
if (age >= 10) {
  return (<p> Greater than { age } </p>)
}
else {
  return (<p> { age } </p>)
}
```

7. Components: React.js divides the web page into multiple components as it is component-based. Each component is a part of the UI design which has its own logic and design as shown in the below image. So the component logic which is written in JavaScript makes it easy and runs faster and can be reusable.

8. Simplicity: React.js is 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?

JSX(JavaScript XML), is a React extension which allows writing JavaScript code that looks like HTML. In other words, JSX is an HTML-like syntax used by React that extends ECMAScript so that HTML-like syntax can co-exist with JavaScript/React code. The syntax is used by preprocessors (i.e., transpilers like babel) to transform HTML-like syntax into standard JavaScript objects that a JavaScript engine will parse.

JSX allows you to write HTML/XML-like structures (e.g., DOM-like tree structures) in the same file where you write JavaScript code, then the preprocessor will transform these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have a tag name, attributes, and children.

Advantages of JSX:

- It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.
- Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.
- It is type-safe, and most of the errors can be found at compilation time.
- It makes it easier to create templates.

3. Classify conditional rendering with examples?

In React, we can create multiple components which encapsulate behaviour that we need. After that, we can render them depending on some conditions or the state of our application. In other words, based on one or several conditions, a component decides which elements it will return. In React, conditional rendering works the same way as the conditions work in JavaScript. We use JavaScript operators to create elements representing the current state, and then React Components update the UI to match them.

From the given scenario, we can understand how conditional rendering works. Consider an example of handling a login/logout button. The login and logout buttons will be separate components. If a user logged in, render the logout component to display the logout button. If a user is not logged in, render the login component to display the login button. In React, this situation is called conditional rendering.

Example:

```
function UserLogin(props) {  
  return <h1>Welcome back!</h1>;  
}  
  
function GuestLogin(props) {  
  return <h1>Please sign up.</h1>;  
}  
  
function SignUp(props) {  
  const isLoggedIn = props.isLoggedIn;  
  if (isLoggedIn) {  
    return <UserLogin />;  
  }  
  return <GuestLogin />;  
}  
  
ReactDOM.render(  
  <SignUp isLoggedIn={false} />,  
  document.getElementById('root')  
);
```

4. What is a short circuit operator in conditional rendering ?

In React, we can create multiple components which encapsulate behaviour that we need. After that, we can render them depending on some conditions or the

state of our application. In other words, based on one or several conditions, a component decides which elements it will return. In React, conditional rendering works the same way as the conditions work in JavaScript. We use JavaScript operators to create elements representing the current state, and then React Components update the UI to match them.

Logical && Operator (Short-Circuit Operator)

Another way to conditionally render a React component is by using the && operator. This operator is the extended version of the short circuit operator. It is a special case and simplified version of the Ternary operator which uses && operator. It is not like the & operator which evaluates the right side expression first. The && operator evaluates the left side expression and evaluates the result.

So, if the statement includes `false && expr`, it will not evaluate `expr` as the statement always returns false.

```
function Garage(props) {
  const cars = props.cars;
  return (
    <>
      <h1>Garage</h1>
      {cars.length > 0 &&
        <h2>
          You have {cars.length} cars in your garage.
        </h2>
      }
    </>
  );
}

const cars = ['Ford', 'BMW', 'Audi'];
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage cars={cars} />);
```

5. What are keys in React?

A key is a special string attribute you need to include when creating lists of elements in React. Keys are used in React to identify which items in the list are changed, updated, or deleted. In other words, we can say that keys are used to give an identity to the elements in the lists.

It is recommended to use a string as a key that uniquely identifies the items in the list. Below example shows a list with string keys:

```
const num = [ 1, 2, 3, 4, 5 ];
const updatedNums = num.map((number)=>{
return <li key={index}> {number} </li>;
});
```

6. What is a list with examples in conditional rendering?

Lists are very useful when it comes to developing the UI of any website. Lists are mainly used for displaying menus in a website, for example, the navbar menu. In regular JavaScript, we can use arrays for creating lists. We can create lists in React in a similar manner as we do in regular JavaScript.

```
<script type="text/javascript">
  var numbers = [1,2,3,4,5];

  const updatedNums = numbers.map((number)=>{
    return (number + 2);
  });

  console.log(updatedNums);
</script>
```

Example for Lists with Conditional Rendering:

```
function List() {
```

```

const messages = [
  { content: "Lorem", id: 1 },
  { content: "Ipsum", id: 2 },
  { content: "dolor", id: 3 },
  { content: "Sit", id: 4 },
  { content: "Amet", id: 5 }
];

return (
  <ul>
    {messages
      .filter(({ content }) => content[0] !==
content[0].toLowerCase())
      .map(message => (
        <li key={message.id}>{message.content}</li>
      ))}
  </ul>
);
}

```

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

React uses Virtual DOM which is like a lightweight copy of the actual DOM (a virtual representation of the DOM). So for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen. So each time there is a change in the state of our application, the virtual DOM gets updated first instead of the real DOM.

In react, everything is treated as a component be it a functional component or class component. A component can contain a state. Each time we change something in our JSX file or let's put it in simple terms, whenever the state of any component is changed, react updates its Virtual DOM tree. Though it may

sound that it is ineffective, the cost is not much significant as updating the virtual DOM doesn't take much time. React maintains two Virtual DOM at each time, one contains the updated Virtual DOM and one which is just the pre-update version of this updated Virtual DOM. Now it compares the pre-update version with the updated Virtual DOM and figures out what exactly has changed in the DOM like which components have been changed.

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

Controlled Components	Uncontrolled Components
The parent component holds control over the data.	DOM itself holds control over the data.
There is validation control.	There is no validation control.
The internal state is not maintained.	The internal state is maintained.
Store the current value in the form of the prop .	Use ref for the current values.
Predictable because the component handles the form elements state.	Not predictable because the form elements can lose their reference and may be changed / affected by other sources during the lifecycle of a component.
Allow substantial control over the elements and data of the form.	Allow limited control over the elements and data of the form.

9. Explain React state and props.

State	Props
<code>this.state.name</code>	<code>this.props.name</code>
State are mutable	Props are immutable
You can define states in the component itself	You can pass properties from parent components
The state is set and updated by the object.	determine the view upon creation, and then they remain static
Both are accessible as attributes of the component class and compose components with a different representation (view)	Both are accessible as attributes of the component class and compose components with a different representation (view)

10. What is client side routing?

Client-side routing

Client-side routing is the internal handling of a route inside of your JS file that is rendered to the front end (or client). The reason client-side routing has become something more developers have been considering when creating their apps is due to the popularity of creating single-page applications (SPAs).

With an SPA, we don't require multiple pages to load, just the original request with our initial HTML, CSS, and JS files from the server. Because of this, client-side routing is used to create that SPA experience while making the routes more uniform and organised for our users to see.

The routing is to give our users a better experience overall because they are able to pinpoint with the route in the URL bar where they are at in the application, all without us needing to make multiple server requests.

11. Compare the difference between Routing and Rendering in react js?

Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application. When a user types a specific URL into the browser, and if this URL

path matches any 'route' inside the router file, the user will be redirected to that particular route. React Router is a standard library system built on top of the React and used to create routing in the React application using React Router Package. It provides the synchronous URL on the browser with data that will be displayed on the web page. It maintains the standard structure and behaviour of the application and is mainly used for developing single page web applications.

In React, Render is the technique that can redirect a page with the help of function `render()`. Most importantly, render a function we can use to define the HTML code within the HTML element. It helps to display certain views in the UI using certain logic defined in the render function and returns the output. React renders HTML to the web page by using a function called `ReactDOM.render()`.

12. What is React Hooks?explain it?

- Hooks were added to React in version 16.8.
- Hooks allow function components to have access to state and other React features. Because of this, class components are generally no longer needed.
- Hooks help us to "hook" into React features such as state and lifecycle methods.
- We can also create custom hooks for our unique use cases like data fetching, logging to disk, timers, and many more.

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

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
```

```

    <p> You clicked {count} times </p>
    <button onClick={() => setCount(count + 1)}>
      Click me
    </button>
  </div>
);
}

```

13. Explain Strict Mode in React.

StrictMode is a React Developer Tool primarily used for highlighting possible problems in a web application. It activates additional depreciation checks and warnings for its child components. One of the reasons for its popularity is the fact that it provides visual feedback (warning/error messages) whenever the React guidelines and recommended practices are not followed. Just like the React Fragment, the React StrictMode Component does not render any visible UI.

The React StrictMode can be viewed as a helper component that allows developers to code efficiently and brings to their attention any suspicious code which might have been accidentally added to the application. The StrictMode can be applied to any section of the application, not necessarily to the entire application. It is especially helpful to use while developing new codes or debugging the application.

Example:

```

import React from 'react';

function StrictModeDemo() {
  return (
    <div>
      <Component1 />
      <React.StrictMode>
        <React.Fragment>
          <Component2 />

```

```
        <Component3 />
      </React.Fragment>
    </React.StrictMode>
    <Component4 />
  </div>
);
}
```

14. What is server side routing in react js?

When browsing, the adjustment of a URL can make a lot of things happen. This will happen regularly by clicking on a link, which in turn will request a new page from the server. This is what we call a server-side route. A whole new document is served to the user.

A server-side request causes the whole page to refresh. This is because a new GET request is sent to the server which responds with a new document, completely discarding the old page altogether.

Pros

- A server-side route will only request the data that's needed. No more, no less.
- Because server-side routing has been the standard for a long time, search engines are optimised for web pages that come from the server.

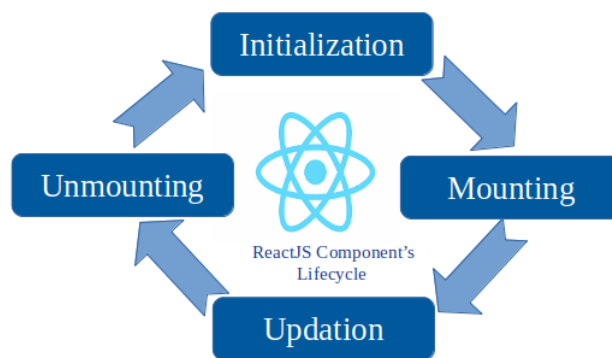
Cons

- Every request results in a full-page refresh. That means that unnecessary data is being requested. A header and a footer of a webpage often stays the same. This isn't something you would want to request from the server again.
- It can take a while for the page to be rendered. However, this is only the case when the document to be rendered is very large or when you have slow internet speed.

15. Compare the difference between Routing and Rendering.

Refer 11th Question

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



In ReactJS, the development of each component involves the use of different lifecycle methods. All the lifecycle methods used to create such components, together constitute the component's lifecycle. They are defined as a series of functions invoked in various stages of a component. Each phase of the lifecycle components includes some specific lifecycle methods related to that particular phase. There are primarily 4 phases involved in the lifecycle of a reactive component, as follows.

- Initializing
- Mounting
- Updating
- Unmounting

Phase 1: Initialising

This is the initial phase of the React component lifecycle. As the name suggests, this phase involves all the declarations, definitions, and initialization of properties, default props as well as the initial state of the component required by the developer. In a class-based component, this is implemented in

the constructor of the component. This phase occurs only once throughout the entire lifecycle. The methods included in this phase are:

- `getDefaultProps()`: The method invoked immediately before the component is created or any props from the parent component are passed into the said (child) component. It is used to specify the default value of the props.
- `getInitialState()`: The method invoked immediately before the component is created and used to specify the default value of the state.

Phase 2: Mounting

The second phase of the React component lifecycle, followed by the initialization phase, is the mounting phase. It commences when the component is positioned over the DOM container (meaning, an instance of the component is created and inserted into the DOM) and rendered on a webpage. It consists of 2 methods, namely:

- `componentWillMount()`: The method invoked immediately before the component is positioned on the DOM, i.e. right before the component is rendered on the screen for the very first time.
- `componentDidMount()`: The method invoked immediately after the component is positioned on the DOM, i.e. right after the component is rendered on the screen for the very first time.

Phase 3: Updating

The third phase of the ReactJS Component Lifecycle is the Updation phase. Followed by the mounting phase, it updates the states and properties that were declared and initialised during the initialization phase (if at all any changes are required). It is also responsible for handling user interaction and passing data within the component hierarchy. Unlike the initialization phase, this phase can

be repeated multiple times. Some of the lifecycle methods falling into this category are as follows:

- `componentWillReceiveProps()`: The method invoked immediately before the props of a mounted component get reassigned. It accepts new props which may/may not be the same as the original props.
- `shouldComponentUpdate()`: The method invoked before deciding whether the newly rendered props are required to be displayed on the webpage or not. It is useful in those scenarios when there is such a requirement not to display the new props on the screen.
- `componentWillUpdate()`: The method invoked immediately before the component is re-rendered after updating the states and/or properties.
- `componentDidUpdate()`: The method invoked immediately after the component is re-rendered after updating the states and/or properties.

Phase 4: Unmounting

Unmounting is the last phase of the ReactJS component lifecycle. This phase includes those lifecycle methods which are used when a component is getting detached from the DOM container (meaning, the instance of the component being destroyed and unmounted from the DOM). It is also responsible for performing the required cleanup tasks. Once unmounted, a component can not be re-mounted again.

- `componentWillUnmount()`: The method invoked immediately before the component is removed from the DOM at last, i.e. right when the component is completely removed from the page and this shows the end of its lifecycle.

17. What are the lifecycle methods of React?

Write Method names from previous question

18. What is a 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.

19. Can React Hook replace Redux?

Redux is a predictable state management library and architecture which easily integrates with React.

The primary advantages of Redux are:

- Deterministic state resolution (enabling deterministic view renders when combined with pure components).
- Transactional state.
- Isolate state management from I/O and side-effects.
- Easily share state between different components.
- Transaction telemetry (auto-logging action objects).

In other words, Redux gives you code organisation and debugging superpowers. It makes it easier to build more maintainable code, and much easier to track down the root cause when something goes wrong.

React hooks let you use state and React lifecycle features without using class and React component lifecycle methods. They were introduced in React 16.8.

The primary selling points of React hooks are:

- Use state and hook into the component lifecycle without using a class.
- Colocate related logic in one place in your component, rather than splitting it between various lifecycle methods.
- Share reusable behaviours independent of component implementations (like the render prop pattern).

React Hooks vs Redux

- Both of them handle state management, but with several differences. There is a lot of abstraction into the following sentence, but this seems like a golden rule to know when you should use Redux into your application:
- Redux should be used in applications that have several features. With these features sharing chunks of the same information.
- Redux offers some free tools that help us to manage the application without reinventing the wheel. It also offers us the following:
 - Saving the actual state of our entire application;
 - Tools to debug while developing like redux-devtools;
 - Change the actual state without triggering one re-render of the entire application, by using the connect() function.

20. Classify the Router component and Develop the Navbar components?

```
import { BrowserRouter as Router, Switch, Route } from
"react-router-dom";
import navbar from "../components/navbar"
import Dogs from "../pages/Dogs"
import Cats from "../pages/Cats"
import Sheeps from "../pages/Sheeps"
import Goats from "../pages/Goats"
function App() {
  return (
    <Router>
      <navbar />
      <Switch>
        <Route path="/" exact component={Dogs} />
        <Route path="/cats" component={Cats} />
        <Route path="/sheeps" component={Sheeps} />
        <Route path="/goats" component={Goats} />
      </Switch>
    </Router>
  );
}
```

Web Application Development

Module 5 QB Solutions

Part A

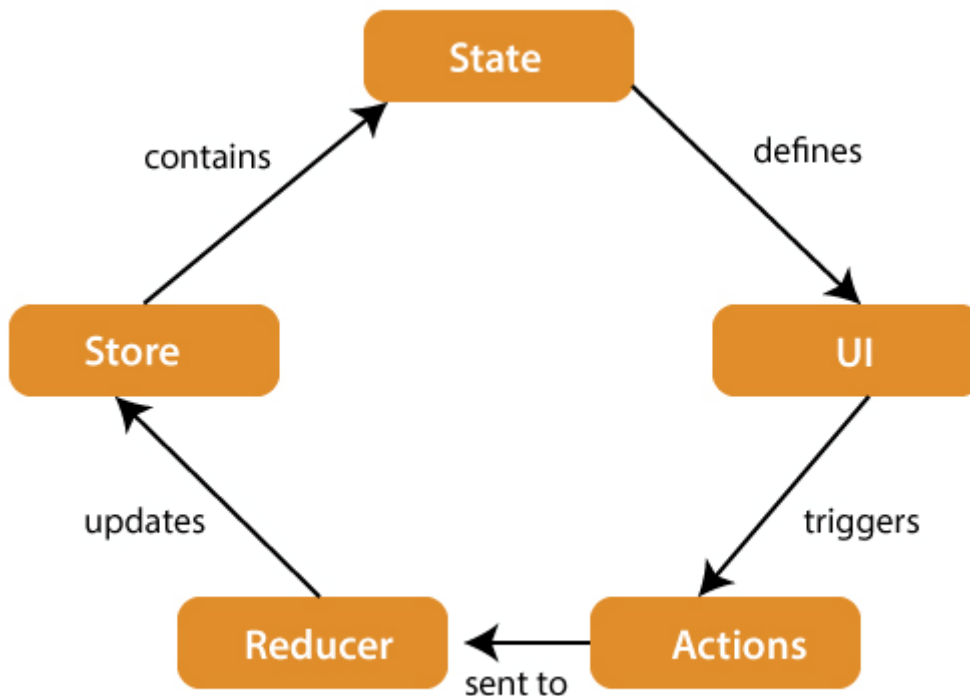
1) Explain the pillars of Redux?

Redux is a JavaScript library to manage application state. It follows three basic principles, they are:

1. Redux is a Single Source of Truth: The global state of an app is stored within an object tree in a single store. As a result, it becomes easier to build universal apps due to the server's state being serialized into the client without any extra codes. A single state tree, in turn, makes it easy to inspect an app. Besides this, it also enables a faster and shorter development cycle.
2. The State is Read-only State: There is only one way for changing the state—emit an action or an object that describes what happened. As per the second principle, neither the network nor the views callbacks would ever write to the state. Instead of it, these express intent for the transformation of the form. Since all of these changes are centralized and these can happen only in a strict order, there are no conditions to look for. Since actions are plain objects, these can be serialized, logged, stored, and then replayed to debug or test.
3. Changes are made with pure functions: In order to specify how the state tree can be transformed by actions, we can write pure reducers. The reducers are merely pure functions, which take the previous state as well as action and move it to the next state. We should remember that we should return to new state objects other than mutating to the last state. First, we should start with one reducer. Now, while our application grows, we can split it off into small reducers, which can handle specific parts of the state

tree. Since reducers are merely functions, we should control the order, wherein the order can turn into reusable reducers for the common tasks.

2) Explain React + Redux with a suitable diagram.



React Redux is the official React binding for Redux. It enables React components to read data from a Redux store and send updates to the store. Redux helps apps scale by providing a sensible way to manage state through a unidirectional data flow model. React Redux is conceptually straightforward. It subscribes to the Redux store, checks to see if the data that the component wants has changed, and re-renders the component.

3) Give a code example to demonstrate embedding two or more components into one.

Child1.js

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

class Child1 extends Component {
  render() {
    return (
      <li>
        This is a child component 1.
      </li>
    );
  }
}
```

```
    </li>
  );
}
}

export default Child1;
```

Child2.js

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

class Child2 extends Component {
  render() {
    return (
      <li>
        This is a Child component 2.
      </li>
    );
  }
}

export default Child2;
```

App.js

```
import React, { Component } from 'react';
import Child1 from './components/child1';
import Child2 from './components/child2';

class App extends Component {
  render() {
    return (
      <div>
        <div>This is a parent component</div>
        <Child1 />
        <Child2 />
      </div>
    );
  }
}

export default App;
```

4) Give a code example to modularize code in react.

Filename: index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
// Importing default export
import File from "./DefaultExport";
// Importing named exports
import { NamedExport } from "./NamedExport";
ReactDOM.render(
  <React.StrictMode>
    <File />
    <NamedExport />
  </React.StrictMode>,
  document.getElementById("root")
);
```

Filename:DefaultExport.js

```
import React from "react";
const DefaultExport = () => {
  return (
    <div>
      <h1>This is from default export</h1>
      <h2>Hello Coders</h2>
    </div>
  );
};
// Default export
export default DefaultExport;
```

Filename:NamedExport.js

```
import React from "react";
const NamedExport = () => {
  return (
    <div>
      <h1>This is from named export</h1>
      <h2>Nice to see you</h2>
    </div>
  );
};
// Named Export
```

```
export { NamedExport };
```

5) Apply the concept of redux with real time examples.

Redux **allows you to manage your app's state in a single place and keep changes in your app more predictable and traceable.** It makes it easier to reason about changes occurring in your app.

A few real life examples are as follows:

- User Login:

In this situation the program should be able to remember who has logged in. In these cases we can use so that the whole program will know the state of user login with other required details.

- Switch from light mode to dark mode:

If a user wants to change from light mode to dark mode then the whole application will be notified that the user wants the website in dark mode thus changing the state will change the whole website.

6) How do you implement React routing (The extension isn't working work on the code later)

We can implement react routing with the help of two tags according to the latest version of React:

- Routes
- Route

```
import React, { Component } from "react";
import { Routes, Route } from 'react-router-dom';
class App extends Component {
```

```

render() {
  return (
    <div>
      <Routes>
        <Route path = "/" element = {<HomePage
        />} />
        <Route path = "/cars" element =
        {<CarPage />} />
        <Route path = "/show" element =
        {<ShowPage />} />
        <Route path = "/signin" element = {<Auth
        />} />
      </Routes>
    </div>
  );
}
}

```

7) Web application without redux

Yes, you can build web applications without using Redux. Redux has nothing to do with React. They can work together, but it's not a rule of thumb. Although Redux is really helpful when managing states on large React applications, you can choose to use it or not.

8) Web Application with redux (Using Store and reducers)

Redux is a state management library that can be used to manage the state of any application, not just React. When using Redux, you will see that we may need to write more code to get the same things done. That is by

design, but your application state is more manageable, and testing becomes easy.

Three Pillars of Redux

- Store
- Action
- Reducer

9) List some of the cases when you should use Refs.

Refs provide a way to access DOM nodes or React elements created in the render method.

In the typical React dataflow, props are the only way that parent components interact with their children. To modify a child, you re-render it with new props. However, there are a few cases where you need to imperatively modify a child outside of the typical dataflow. The child to be modified could be an instance of a React component, or it could be a DOM element. For both of these cases, React provides an escape hatch.

When to Use Refs

There are a few good use cases for refs:

- Managing focus, text selection, or media playback.
- Triggering imperative animations.
- Integrating with third-party DOM libraries.
- Avoid using refs for anything that can be done declaratively.

For example, instead of exposing `open()` and `close()` methods on a `Dialog` component, pass an `isOpen` prop to it.

Part B

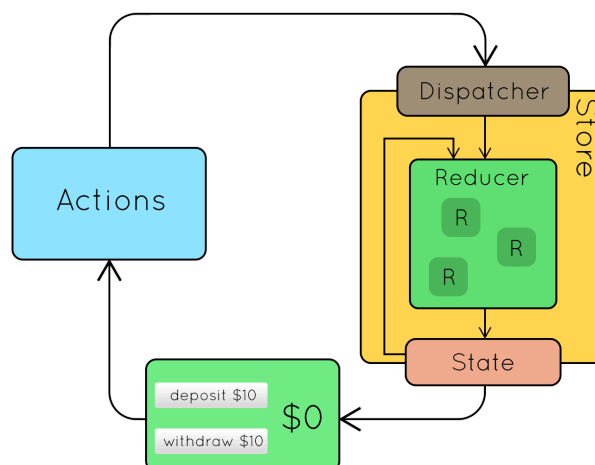
1) What is an event in React?

An event is an action that could be triggered as a result of a user action or a system-generated event. For example, a mouse click, the loading of a web page, pressing a key, window resizing, and other interactions are called events. React has its own event handling system, which is very similar to handling events on DOM elements.

The React event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event. Handling events with React has some syntactic differences from handling events on the DOM. These are:

1. React events are named in camel case instead of lowercase.
2. With JSX, a function is passed as the event handler instead of a string.
3. In react, we cannot return false to prevent the default behaviour. We must call `preventDefault` event explicitly to prevent the default behaviour.

2) Draw a diagram showing how data flows through Redux.



3) How will you distinguish Redux from Flux?

Redux: Redux is a predictable state container for JavaScript apps. Redux is a library that can be used with any UI layer or framework, including React, Angular, Ember, and vanilla JS. Redux can be used with React; both are independent of each other. Redux is a state-managed library used in

JavaScript apps. It simply manages the state of your application, or, in other words, it is used to manage the data of the application. It is used with a library like React.

Flux: Flux is the application architecture, or we can say JavaScript architecture, that is used for building client-side web applications or UI for client applications. You can begin using Flux without having to write a lot of new code. Flux overcomes the drawbacks of MVC, such as instability and complexity.

4) What are the advantages of using React?

1. A centralized state management system, i.e., store
2. Performance Optimizations
3. Pure reducer functions
4. Storing long-term data
5. Time-travel Debugging
6. Great supportive community

5) Where would you put AJAX calls in your React code?

APIs are used for fetching data from the server and using AJAX and API we call data asynchronously and show it in our HTML. You can make API requests by using browser build in fetch functions or third party libraries like Axios. You can make API requests anywhere but it is totally recommended that you make an API call in **componentDidMount()** life cycle method.

Reasons to use componentDidMount(): So, this can be really simple by just putting the API request in componentDidMount.

Using componentDidMount makes sure that data won't be loaded until after the initial render which is really important.

Using componentDidMount makes sure that the data is only fetched from the client.

6) You must've heard that "In React, everything is a component". What do you understand from that statement?

The building blocks of a React application's UI are called components. Any app UI created using React is divisible into a number of small, independent, and reusable pieces known as components. React renders each of the components independent of each other. Hence, there is no effect of rendering a component on the rest of the app's UI.

7) Can a browser read JSX?

Browsers can't read JSX because there is no inherent implementation for the browser engines to read and understand it. JSX is not intended to be implemented by the engines or browsers; it is intended to be used by various transpilers to transform this JSX into valid JavaScript code.

8) Define HOC in React. What are the benefits of HOC?

A higher-order component (HOC) is an advanced technique in React for reusing component logic. HOCs are not part of the React API per se. They are a pattern that emerges from React's compositional nature.

Concretely, a higher-order component is a function that takes a component and returns a new component.

9) What is a store in Redux?

A store is an immutable object tree in Redux. A store is a state container that holds the application's state. Redux can have only a single store in your application. Whenever a store is created in Redux, you need to specify the reducer.

The only way to change the state inside it is to dispatch an action on it.

A store is not a class. It's just an object with a few methods on it. To create it, pass your root-reducing function to createStore.

10) What is an arrow function and how is it used in React?

With arrow functions, there is no binding for this. In regular functions, this keyword represented the object that called the function, which could be the

window, the document, a button, or whatever. With arrow functions, this keyword always represents the object that defined the arrow function.

11) How is React different from React Native?

SN	ReactJS	React Native
1.	The ReactJS initial release was in 2013.	The React Native initial release was in 2015.
2.	It is used for developing web applications.	It is used for developing mobile applications.
3.	It can be executed on all platforms.	It is not platform independent. It takes more effort to be executed on all platforms.
4.	It uses a JavaScript library and CSS for animations.	It comes with built-in animation libraries.
5.	It uses React-router for navigating web pages.	It has built-in Navigator library for navigating mobile applications.
6.	It uses HTML tags.	It does not use HTML tags.
7.	It can use code components, which saves a lot of valuable time.	It can reuse React Native UI components & modules which allow hybrid apps to render natively.
8.	It provides high security.	It provides low security in comparison to ReactJS.
9.	In this, the Virtual DOM renders the browser code.	In this, Native uses its API to render code for mobile applications.

12) How is React different from Angular?

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook. React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It Create simple views for each state of your application, and React will efficiently update and render the appropriate component whenever your data changes. The declarative view makes your code more predictable and easier to debug.

Angular is a popular open-source JavaScript framework created by Google for developing web applications. Front-end developers use frameworks like Angular or React for presenting and manipulating data efficiently. Updated Angular is much more efficient compared to the older version of Angular, especially since the core functionality was moved to different modules.

That's why it becomes so much faster and smoother compared to the older one. newly added angular CLI. With that package, you can create a scaffolding for your Angular project.

13) What are the components of Redux?

STORE: A store is a place where the entire state of your application is listed. It manages the status of the application and has a dispatch (action) function. It is like a brain responsible for all the moving parts in Redux.

ACTION: Action is sent or dispatched from the view, which are payloads that can be read by reducers. It is a pure object created to store the information of the user's event. It includes information such as the type of action, time of occurrence, location of occurrence, its coordinates, and which state it aims to change.

REDUCER: Reducer reads the payloads from the actions and then updates the store via the state accordingly. It is a pure function to return a new state from the initial state.

14) What are pure components?

Pure Components in React

Pure Components in React are the components which do not re-renders when the value of state and props has been updated with the same values. If the value of the previous state or props and the new state or props is the same, the component is not re-rendered. Pure Components restricts the re-rendering ensuring the higher performance of the Component

Features of React Pure Components

Prevents re-rendering of Component if props or state is the same

Takes care of "shouldComponentUpdate" implicitly

State and Props are Shallow Compared

Pure Components are more performant in certain cases

Similar to Pure Functions in JavaScript, a React component is considered a Pure Component if it renders the same output for the same state and props value. React provides the PureComponent base class for these class

components. Class components that extend the `React.PureComponent` class are treated as pure components.

15) How would you create a form in React?

In React forms are created with the help of form tag from HTML.

But in react we can use the state to store the data and use it to either send / push the data to the database or just to display even when the user refreshes the page.

Example code:

Import React from "react";

```
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>
  );
}
}
}

```

16) What are the advantages of using Redux?

1. Centralized state management system i.e. Store
2. Performance Optimizations
3. Pure reducer functions
4. Storing long-term data
5. Time-travel Debugging
6. PoGreat supportive community

17) What is Redux?

Redux is a predictable state container for JavaScript apps. As the application grows, it becomes difficult to keep it organized and maintain data flow. Redux solves this problem by managing the application's state with a single global object called Store. Redux fundamental principles help in maintaining consistency throughout your application, which makes debugging and testing easier.

18) How are forms created in React? (Repeated)

19) What are the three principles that Redux follows?

Single source of truth

The global state of your application is stored in an object tree within a single store.

State is read-only

The only way to change the state is to emit an action, an object describing what happened.

Changes are made with pure functions

To specify how the state tree is transformed by actions, you write "pure reducers."

20) What do you understand by “Single source of truth”?

Single source of truth

The global state of your application is stored in an object tree within a single store.

Because the state from your server can be serialized and hydrated into the client with no additional coding, it is simple to build universal apps. A single state tree also

makes it easier to debug or inspect an application; it also enables you to persist your app's state in development for a faster development cycle. Some functionality that has been traditionally difficult to implement—undo and redo, for example—can suddenly become trivial to implement if all of your states are stored in a single tree.

Part C

- 1) How are actions defined in Redux?**
- 2) Explain the role of Reducer.**
- 3) What are the advantages of Redux?**
- 4) What is a React Router?**
- 5) Why is the switch keyword used in React Router v4?**
- 6) Why do we need a Router in React?**
- 7) List down the advantages of React Router.**
- 8) What do you understand about Refs in React?**
- 9) How is React Router different from conventional routing?**
- 10) What is NPM?**
- 11) What is Prop Drilling and How can you avoid it?**
- 12) What is React Context?**
- 13) What is a React Map?**
- 14) What is React conditional rendering?**

- 15) How is React Router different from conventional routing? (repeated)**
- 16) Why Switch keyword used in React Router V4?**
- 17) How many ways can we style the React Component?**
- 18) What is the significance of stores in Redux?**
- 19) What are Redux Devtools?**
- 20) Are there any similarities between Redux and RxJs?**

Redux uses the Reactive paradigm because the Store is reactive. The Store observes actions from a distance, and changes itself. RxJS also uses the Reactive paradigm, but instead of being an architecture, it gives you basic building blocks, Observables, to accomplish this pattern.