**Basic React Js I Q**

1. What problem is React solving?

Earlier, when doing web development, in order to remove, add, or change an element on the page, the entire page had to be re-rendered. This massively impacted the application’s performance, slowing down and degrading the user experience. That is where React came into the picture. With the concept of its virtual DOM, now, whenever there is a change, only a part of the page that changes is updated with the help of Virtual DOM. This significantly reduces the page load time, enhancing the app’s performance and giving a great user experience.

2. How is the Virtual DOM different from the Real DOM?

Virtual DOM is a lightweight Javascript object that is more like a skeletal representation of the Document Object Model(DOM), which is also known as the Real DOM. The virtual DOM, like the real DOM, comprises various nodes, which are actually the elements on the webpage. However, what makes it so light is the fact that it is just a set of nodes, and styling and JavaScript functionalities are not yet applied to it like the Real DOM.

3. How is Virtual DOM enhancing performance?

Since Virtual DOM is a lightweight Javascript Object, calculations on it are less expensive. Whenever any changes are made on the page, the changes are first made on the Virtual DOM. After that, a patchwork of the change is done on the Real DOM, which means only the specific part is re-rendered. This saves a lot of time, as the entire page does not have to be redone again.

4. What is a Diffing Algorithm? How does it work?

Diffing algorithm is the core of React’s enhanced user experience. React has two virtual DOMs: one is the DOM after the update and the other one before the update. The diffing algorithm compares the two virtual DOMs, analyzing the differences between them at each level. It then comes up with the minimum number of changes that are required to be done on the Real DOM. This leads to a faster user experience due to the fast and efficient rendering of the webpage.

5. Is React a library or Framework?

React is a library and not a framework. It only caters to the view layer of the MVC(Model-View-Controller) architecture. It helps in constructing the user interface of the application from scratch.

6. What features of React place it ahead in the game?

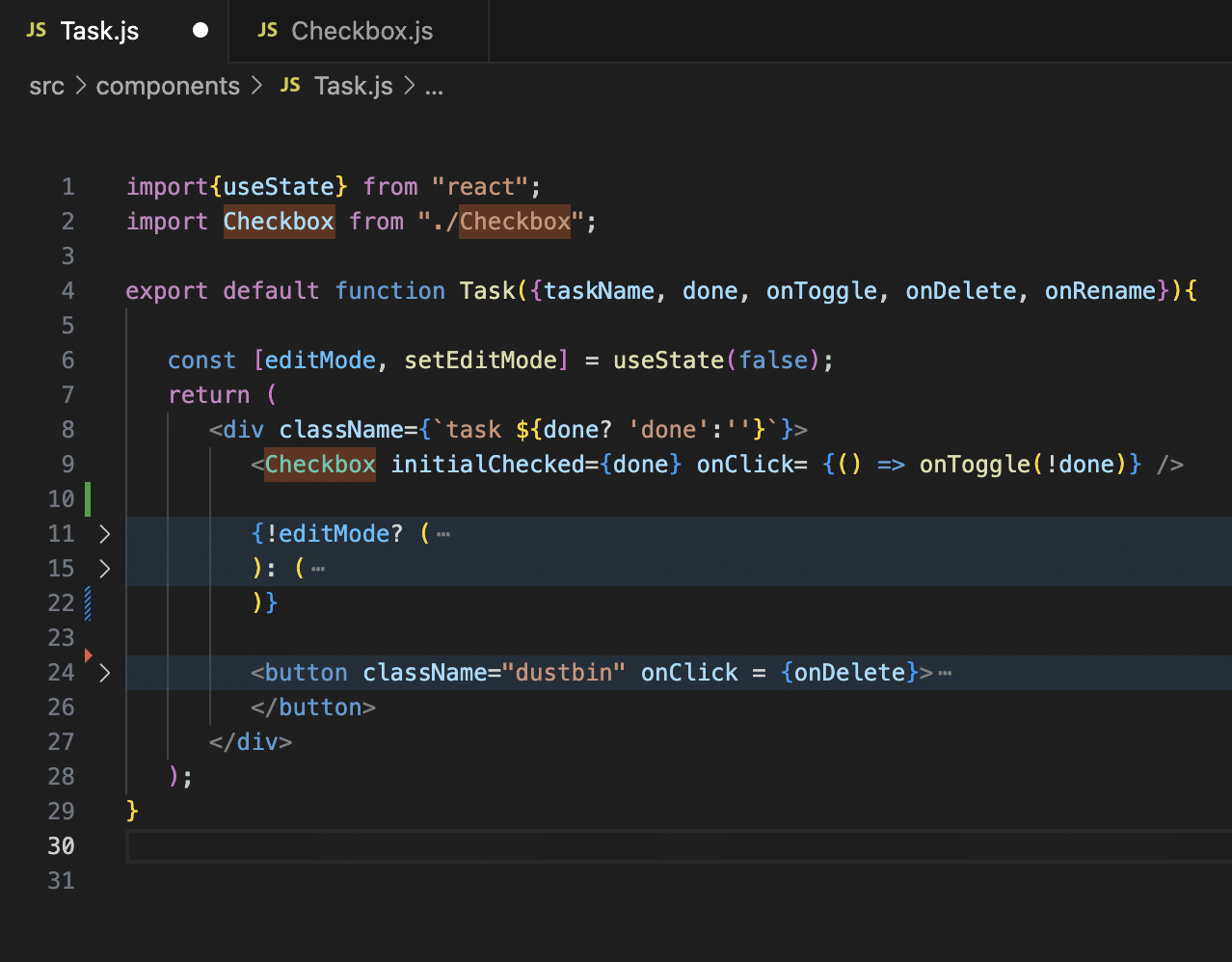
React comes loaded with features that set it apart and make it the most chosen library for developing user interfaces. Its top 5 features are:

1. Virtual DOM   
   Virtual DOM is a twin of the Real DOM. It identifies the point of change and does a patchwork at that point on the Real DOM, improving the UI updates and making its performance skyrocket.
2. React’s Components  
   React follows a component approach where all its significant code chunks can be segregated into different components instead of putting them all together in one place. This helps in making the code base more manageable and saves a lot of time for developers when any changes are to be made.
3. Parent-Child Relationship  
   In React, data flows in a Unidirectional Pattern, i.e., from Parent to Child components via Props. This makes the codebase more understandable for existing contributors as well as new ones.
4. JSX  
   Another cool feature of React is JSX, which is known as Javascript XML. This feature helps you to write Javascript and HTML within the same component file in the form of Javascript code.
5. CSR and SSR  
   React works well with both Client-Side Rendering(CSR) and Server-Side Rendering(SSR).

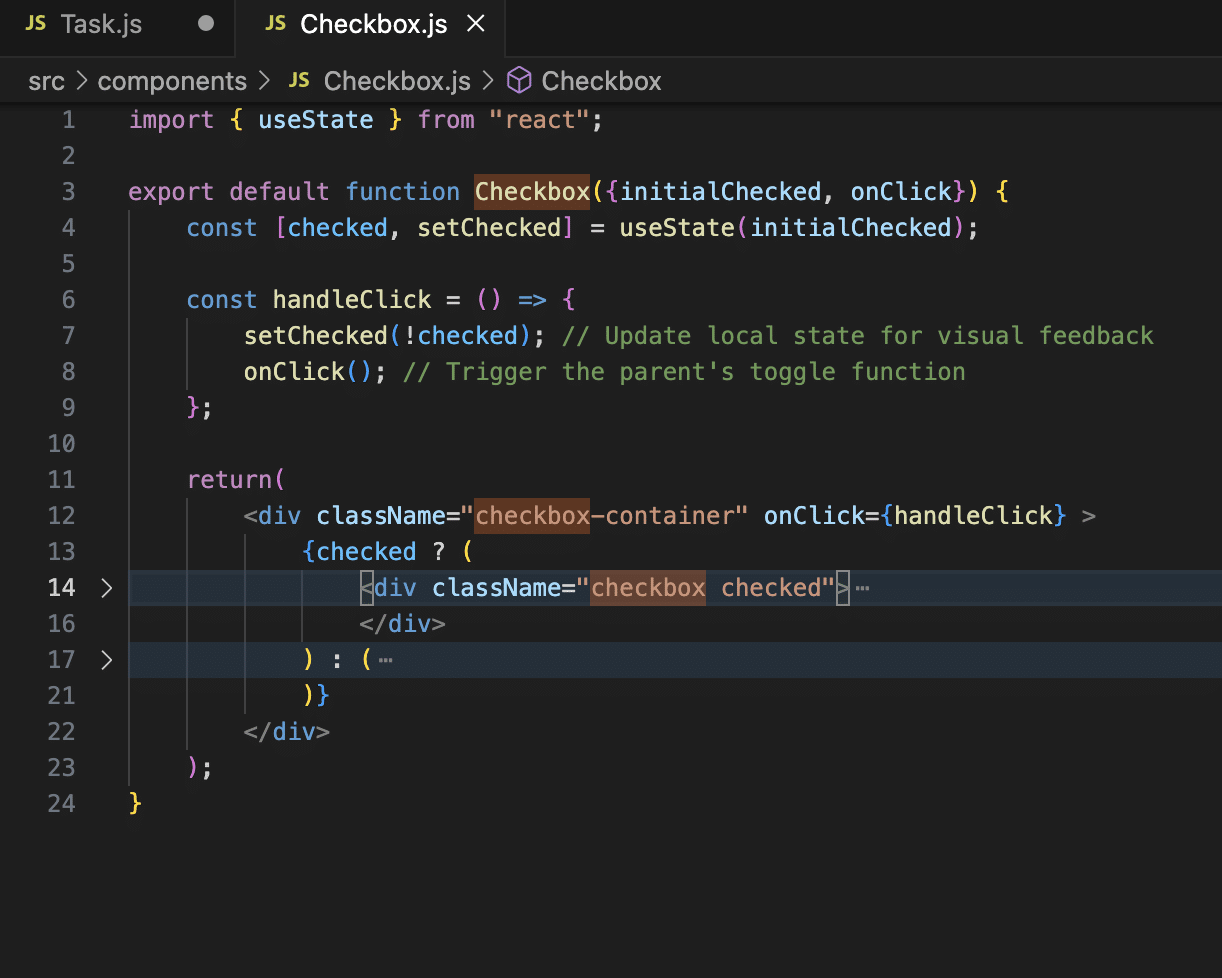
7. Parent component vs child component in React?

A parent component is a component that has one or more child components. The parent component is responsible for passing down data to its children via props. It can also conditionally render either of the child components. While each parent can have more than one child component, each child component can only have one parent.

Here Task.js is a parent component, which imports a child component named Checkbox.



Here Checkbox.js is a child component.



8. What is the difference between State and Props in React?

In  React, State and Props both store crucial Data, and based on these two data, the changes are made to the Virtual DOM and the Real DOM. Let us understand them in more detail.

The state can be understood as the Heart of each component. Whenever there is a change in any component in react, its state gets updated. Whenever there is a change in state it will cause the child components to re-render. However, these child components cannot directly access a parent’s state. How does the data transfer happen?

There comes Props into the picture. The state data from each parent is passed to the child component in the form of props. Each child received the prop, and if there are any changes based on this new data, it is reflected and the child component is re-rendered.

9. What is the entry point of your react project?

In any react project, index.js is the most significant Javascript file in the entire application. It is the entry point of your React application. It imports all the necessary libraries like React, ReactDOM, and index.css. It also imports the App component from the App.js file that contains the application’s logic. Additionally, in this file, you can set any other configurations that might be needed in the application.

10. What is JSX? How do browsers read it?

JSX stands for JavaScript XML. It enables us to write HTML and Javascript together in React. While learning JSX can be slightly intimidating at the beginning, once you get the hang of it, it feels like a cakewalk. It makes the react code base much more simpler, elegant, and concise.

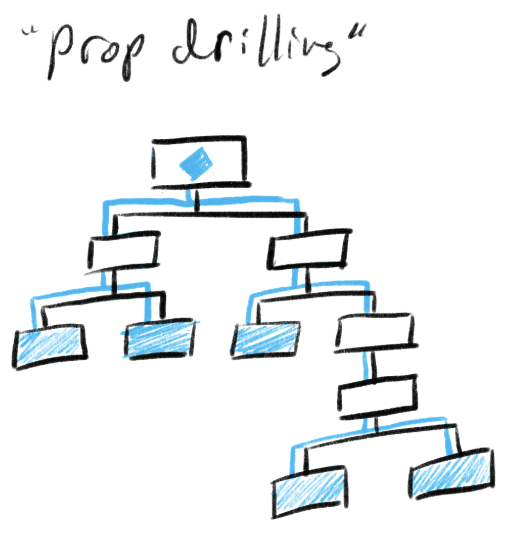
There is a catch! Browsers are not equipped to understand JSX. So, we need tools like Babel. Babel is a free, open-source Javascript compiler and transpiler. It converts modern JS and JSX into standard Javascript that can be understood across all browsers.

11. Why do we need to have keys in React?

Keys play a vital role when we are using list items. It serves as a unique identifier for each list element, helping React understand which item was modified and also optimizes the code.

**Intemediate React Js Interview Questions**

12. What is prop drilling in React? What are its downsides?

Prop drilling in React is the method of passing data from a parent to its child components which ensures uni-directional flow of data  
As the application grows in size, data passing via prop drilling can become more complicated. Say there is a parent component and has a child; the child also has a child, and it keeps going on. In this case, passing the data from the parent to the ‘nth’ child component will become a major hassle.

To solve this problem, it is recommended that state management solutions, like context API, Redux, etc can be used. These solutions make sure that all the data is centralized in one place. This helps eliminate the need to pass the data as a prop from one component to another.

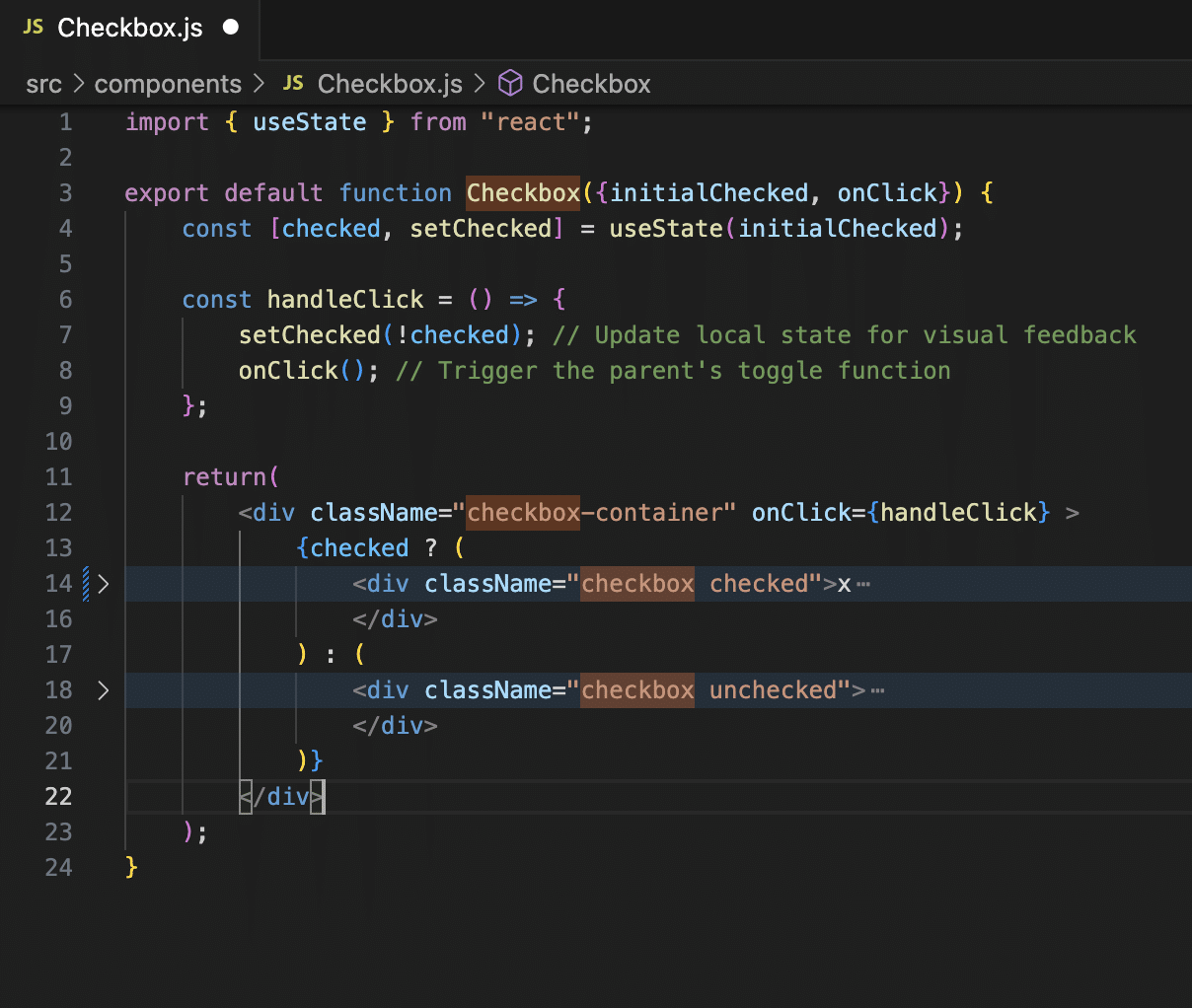
13. What is a render() in React? How many times does it get called?

Rendering is the process of converting the react code into an actual DOM representation that is displayed on the browser. This is done with the help of render() method. This method has a mandatory return statement inside it, which stores the entire UI of the page in the form of JSX.

The render() method is called when the page loads for the very first time, also known as the initial render, and subsequently every time the state of the component updates.

14. What is Conditional rendering in React?

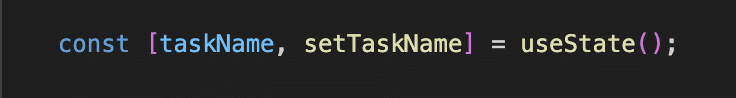
Conditional rendering is the process of rendering a specific child component from the parent if it matches certain conditions. This can be done in React with the help of if/else statements, Switch Statements, ternary operators, IIFE(Immediately invoke functions), etc.

The following image shows components being conditionally rendered with the help of ternary operators. 

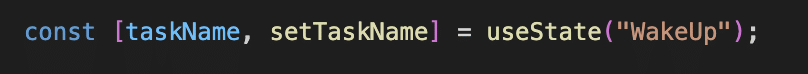
If the value of ‘checked’ is True, it will render the first component, else it will render the second component.

15. What is useState() Hook?

The useState() Hook in React is a function in React version 16.8 that lets one add state to functional components.



It declares a state variable (taskName in this case). The value of the variable task name initially is undefined. Additionally, we can also assign some initial value to the variables like this –



The second parameter is a function(setTaskName in this case) that is called when the value of the variable needs to be updated.

16.What is the use of react-router?

React comes with a library known as [react-router](https://intellipaat.com/blog/react-router/) that is used to create Single Page Web Applications. It helps define various routes within an application and renders different components in each route without refreshing the page.

17. What is a React Fragment?

To return multiple elements they have to be wrapped in a root element like a <div>. React Fragment is a feature in React that will help club all the elements inside one parent node. The bonus of using React Fragment is that it does not add any extra nodes to the DOM like <div>.

18. What is Axios?

Axios is a promise-based HTTP client, a widely used Javascript library that handles HTTP requests(network calls). It makes the process of sending and receiving data very effortless. It also transforms the received data to JSON automatically.

19. What is the use of React Developer Tools?

To return multiple elements they have to be wrapped in a root element like a <div>. React Fragment is a feature in React that will help club all the elements inside one parent node. The bonus of using React Fragment is that it does not add any extra nodes to the DOM like <div>.

20. What is the difference between Dependencies and Dev Dependencies?

Dependencies and devDependencies are packages that are present in the React project to help you create the React application. These dependencies can be found inside the package.json file in your React project.

The table below talks about the difference between the two.

|  |  |
| --- | --- |
| Dependencies | Dev Dependencies |
| Packages needed to run the app in Production | Packages needed during local development |
| Example – react, axios, redux, etc. | Example – webpack, babel, eslint, etc. |

21. What is Webpack?

Webpack is a powerful code bundler that converts all the files inside your React project, like JS files, CSS, images, assets, etc, into a single file for for sending to the browser. It automatically transpiles the code with the help of Bable, takes care of internal code splitting, and enables lazy loading of components. It also takes care of the optimization process by offering features like code minification, tree shaking, etc, that help bring down the size of the application, thereby improving the load time.

Advanced React JS I Q

22. What is event handling in react? Give an example of handling form input.

Event Handling is basically how the app responds to user interaction. React efficiently handles various input events like mouse events(onClick, onMouseEnter, etc.), keyboard events(onKeyUp,onKeyDown, etc.), Form events(onChange, onSubmit, etc.), and many more.

function App() {

  const handleChange = (event) => {

    console.log(event.target.value); // Logs the input value

 };

  return <input type="text" onChange={handleChange} />;

}

23. What is the use of e.preventDefault()?

When working with forms in React, upon form submission, the general behavior of the browser is to reload the webpage. If you are trying to track the form data, you wouldn’t be able to see anything on the console because of the page reload. To stop this, you can use e.preventDefault(). This helps in handling the form submission without reloading the page.

Code snippet for understanding usage –

function handleSubmit(e) {

e.preventDefault();

console.log("Form submitted!");

}

24. Different Between React-Redux and React Context API.

React-Redux is React’s state management library. It focuses on having a centralized state, also known as a redux store. It requires the setup of the store, reducers, actions, and middleware(if handling async calls). It is recommended for use in complex applications. React Context API is a built-in react feature, it helps in state sharing without the need to pass props from parent to child via prop drilling. It is great when having a small application as it requires minimal setup and no extra libraries to be installed.

25. What is a Callback function in React?

A callback function in React is a function that is passed to another function as an argument. This function is executed once the parent function has finished executing.

function App() {

  const handleClick = () => {

    alert("Button clicked!");

 };

  return <button onClick={handleClick}>Click Me</button>;

}

Here, handleClick is a callback function passed to onClick event. This function will be called once the event happens.

Example 2 –

import { useEffect } from "react";

function App() {

  useEffect(() => {

    const timer = setTimeout(() => {

 console.log("Timeout triggered");

    }, 3000);

  }, []);

return <div>Check the console after 3 seconds</div>;

}

In this example, setTime is a callback function that executes after 3 seconds.

26. How can re-renders in React be prevented using shouldComponentUpdate()?

To enhance the performance of your web application it is vital to cut down on unnecessary re-renders. It can be done in two ways –

1. Use lifecycle method shouldComponentUpdate()

It returns a boolean value, true or fals,e to communicate whether the component needs to update or not.

Example -

shouldComponentUpdate(nextProps) {

nextProps.value !== this.props.value;

}

This will stop the component from re-rendering if the ‘value’ hasn’t changed

27. What is a dependency array?

Dependency array is a feature that comes with hooks. It is denoted as []. It helps in optimizing the application by rendering any particular component only when the specified value changes. If an empty dependency array is passed([]), it will execute the hook only once.

Example –

import React, { useState, useEffect } from "react";

function App() {

  const [count, setCount] = useState(0);

  useEffect(() => {

    console.log(`Count is ${count}`);

  }, [count]);

  return (

    <div>

      <button onClick={() => setCount(count + 1)}>Increment</button>

    </div>

  );

}

useEffect() will run on every render by default. But here, we have specified that it must run only when the value of ‘count’ changes.

Alternatively, if we want the hook to run on only once, we can send an empty array like this-

 useEffect(() => {

    console.log(`Count is ${count}`);

  }, []);

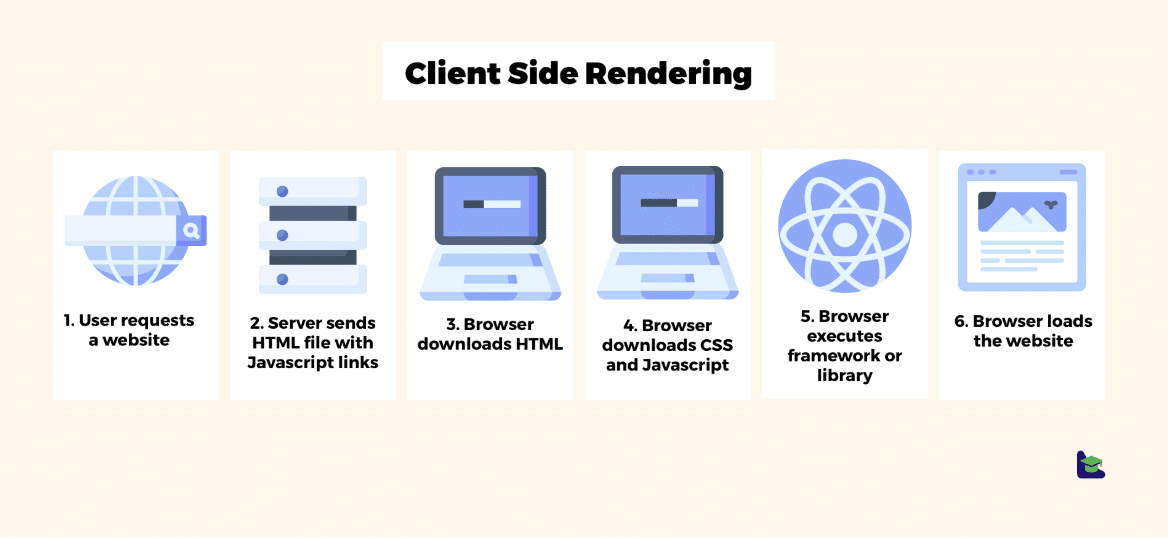
28. How to Optimise a React Application?

A React application can be optimized in various ways. Some of the ways to achieve the same are-

1. Use lazy loading for code splitting using React.lazy and Suspense.
2. Using IntersectionObserverAPI
3. Minimizing HTTP requests using concepts of Throttling and Debouncing
4. Using dependency array with useeffect() hook
5. Using useMemo() and useCallback() hooks
6. By implementing Treeshaking feature of webpack to reduce the bundle size
7. Using the concepts of Code Splitting and minification.
8. Using Pure components instead of class components wherever possible
9. Efficient implementation of the life cycle method  shouldComponentUpdate(), if following the class-based.

29. Server Side Rendering vs Client Side Rendering?

In Server Side Rendering, when the user requests a website

1. A ready-to-be-rendered HTML response is built on the server along with JavaScript functionalities and is sent to the browser.
2. The browser renders the page which is now viewable (but not interactable).
3. Browser downloads and executes JS and React.
4. The page becomes fully interactable.  
   

In Client Side Rendering, when the user requests a website

1. The server sends an empty HTML file with JS links to the browser.
2. Browser downloads the HTML.
3. Browser then downloads CSS and JavaScript.

30. What can cause shouldComponentUpdate() to get into an infinite loop?

If we update the state inside shouldComponentUpdate(), it will always result in a re-render, which will, in turn, call the shouldComponentUpdate() method, and again, the state will be updated and, like this, get into an infinite loop and lead to crashing of the web page.

Bonus Question

* What problems are Hooks solving?

Hooks were introduced in React version 16.8. They help get rid of hefty class components and enable functional components to access the state. Hooks also help in writing minimal code to implement the same logic. For eg. We can use the useEffect() hook and implement both componentDidUpdate and shouldComponentUpdate life cycle methods without having to declare these two different lifecycle methods.

* What is Mounting?

Mounting is the process of converting the virtual representation of a component into a final UI representation that is visible to users.

* What are Pure Components in React?

Pure Components in React are functional components that prevent unnecessary re-renders and optimize performance.

* What happens in the lifecycle method componentWillUnmount()?

The lifecycle method componentWillUnmount() is used to take care of cleanup tasks. It is invoked just before the component is removed from the DOM. In this lifecycle method, tasks like – canceling any existing timers, cleaning network requests, etc.

* What is Next JS?

Next, JS is a full-blown React framework. It is open-source and used to create powerful web applications with ease. It handles Server Side Rendering (SSR) and Static Site Generation (SSG).