1. What is HTTP protocol? What are contents of request and response?

HTTP (Hypertext Transfer Protocol) is a protocol used for sending and receiving data over the internet. It is the foundation of data communication for the World Wide Web. The request and response are the two main components of HTTP. The request consists of a request method, a URL, HTTP version, and optional headers, while the response consists of HTTP version, status code, status message, and optional headers.

2. Why HTTP is called as connection-less protocol? Why it is called as stateless protocol?

HTTP is called a connection-less protocol because each request-response pair is independent and unrelated to any previous request-response pair. The server does not keep track of the requests and responses that have been sent and received previously. HTTP is called a stateless protocol because it does not maintain any information or state about the client between requests.

3. What is state management? Explain server side and client side state management objects.

State management refers to the mechanism used to store and retrieve data between different requests made by a client to the server. In server-side state management, the server stores the state information of the client and sends it back on every subsequent request from the same client. In client-side state management, the state information is stored on the client side and sent to the server with each request.

4. Which are HTTP request methods? Explain difference between GET and POST methods.

The HTTP request methods are GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE. The GET method is used to retrieve data from the server, while the POST method is used to send data to the server for processing. The main difference between GET and POST methods is that GET requests are used to retrieve data and do not change the state of the server, while POST requests are used to send data and change the state of the server.

5. What is difference between HTTP and HTTPS? What is SSL?

HTTP is an unsecured protocol, while HTTPS is a secured version of HTTP. HTTPS uses SSL (Secure Socket Layer) or TLS (Transport Layer Security) to encrypt data sent over the internet. SSL (Secure Sockets Layer) is a security protocol that provides secure communication over the internet.

6. What is AJAX? How it is used with and without jQuery?

AJAX (Asynchronous JavaScript and XML) is a technique used for updating a web page without reloading the entire page. It allows the user to interact with the web application without any page refresh. AJAX can be used with or without jQuery. With jQuery, AJAX can be implemented using the $.ajax() method, while without jQuery, AJAX can be implemented using the XMLHttpRequest object.

7. What is DOM? Explain in context of HTML page.

DOM (Document Object Model) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. In the context of an HTML page, the DOM represents the HTML document as a tree structure, with each HTML element represented by a node in the tree.

8. What is jQuery? How to call REST services using jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It provides a set of features and utilities that make it easier to work with HTML documents, handle events, create animations, and more. To call REST services using jQuery, we can use the $.ajax() method, which is used to perform an asynchronous HTTP request.

9. What do you mean by responsive web UI? What is grid system in bootstrap?

A responsive web user interface (UI) is a design approach that enables a website or web application to adapt to different screen sizes, resolutions, and device types. With a responsive UI, the layout, typography, and content of the website can adjust automatically based on the size and orientation of the user's device, ensuring that the user experience is consistent and optimal regardless of the device being used. Bootstrap is a popular front-end development framework that includes a grid system as one of its core features. The Bootstrap grid system is based on a 12-column layout, with CSS classes that allow developers to specify how many columns a particular UI element should occupy at different screen sizes.

10. What are key components of bootstrap?

Some of the key components of Bootstrap include:

Grid System: As I mentioned earlier, Bootstrap's grid system provides a flexible and customizable layout that makes it easy to create responsive designs.

Typography: Bootstrap includes a set of typography styles and classes that allow developers to easily format text and headings with consistent and professional-looking styles.

Forms: Bootstrap includes pre-built form controls and layouts, as well as validation styles and feedback messages, to make it easier to create user-friendly forms.

Buttons: Bootstrap provides a set of CSS classes that make it easy to create different button styles and sizes, including dropdown buttons and button groups.

Navigation: Bootstrap includes navigation components like navbar, tabs, and pagination that make it easy to create user-friendly and intuitive navigation menus.

Icons: Bootstrap includes a set of scalable vector icons, called Glyphicons, as well as support for other popular icon libraries like Font Awesome.

Modals: Bootstrap's modal component provides a flexible and customizable way to display content in a popup window, which can be useful for things like sign-up forms or image galleries.

Utilities: Bootstrap includes a range of utility classes for things like spacing, text alignment, and visibility, which can help developers customize their designs and layouts quickly and easily.

What is React? What are the major features of React?

React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components that can be composed to build complex UIs. Some of the major features of React include:

Virtual DOM: React uses a virtual representation of the DOM to minimize the number of actual DOM manipulations and improve performance.

Component-based architecture: React encourages developers to break UIs down into smaller, reusable components that can be composed to build complex UIs.

Declarative syntax: React uses a declarative syntax to define how the UI should look at any given time, rather than requiring developers to write imperative code to update the UI.

One-way data flow: React uses a unidirectional data flow, where data flows down from parent components to child components, to make it easier to reason about how data is being used in the application.

What are the advantages and limitations of React?

Advantages of React include:

Reusability: React allows developers to create reusable UI components, which can save time and effort in building complex UIs.

Performance: By using a virtual representation of the DOM, React minimizes the number of actual DOM manipulations and improves performance.

Declarative syntax: The declarative syntax used in React makes it easier to reason about how the UI should look at any given time, and reduces the amount of imperative code that developers need to write.

Large developer community: React has a large and active developer community, which means that there are many resources available for learning and troubleshooting.

Limitations of React include:

Steep learning curve: React has a relatively steep learning curve, especially for developers who are not familiar with functional programming concepts.

Overhead: React can add some overhead to the development process, since it requires developers to write more code than they might with a simpler solution.

Integration with legacy code: If a project already has a large codebase, integrating React can be difficult and time-consuming.

What is JSX? How to loop inside JSX?

JSX is a syntax extension for JavaScript that allows developers to write HTML-like code in their JavaScript files. It is used in React to define the structure and content of UI components.

To loop inside JSX, you can use the map function on an array and return a new array of JSX elements. For example:

```
const myArray = ["apple", "banana", "cherry"];

const myJSX = (
  <ul>
    {myArray.map((item) => (
      <li>{item}</li>
    ))}
  </ul>
);
```

This code will loop through the myArray array, and for each item in the array, it will return a new JSX element that contains the item as text. The resulting myJSX variable will contain a JSX element that renders an unordered list with three list items, one for each item in the array.

What is the difference between state and props? Why you can't update props in React?

In React, both state and props are used to manage and pass data between components. However, there are some key differences between them.

State is an internal data storage mechanism in a component that allows the component to keep track of its own data and re-render based on changes to that data. State is initialized within the constructor of the component and can be updated using the setState() method, which triggers a re-render of the component and its children.

Props, on the other hand, are external data passed to a component by its parent component. Props are read-only and cannot be modified by the component that receives them. When props are updated in the parent component, the child component is re-rendered with the new props.

The reason why you can't update props in React is that React follows a one-way data flow, where the parent component passes data down to its children components via props. The child component can use the props to render itself, but it cannot change them. If a child component needs to update data, it should do so by updating its own state, which will trigger a re-render of the component and its children.

What is the difference between HTML and React event handling?

In HTML, event handling is done using inline event handlers or by adding event listeners to the DOM element. For example, you can add an onclick attribute to an HTML button element to handle the click event:

```
<button onclick="handleClick()">Click me</button>
```

In React, event handling is done by passing a callback function to the onClick (or other event) prop of a React component. For example, to handle the click event of a button in React, you would define a handler function and pass it to the button component as a prop:

```
function handleClick() {
  console.log('Button clicked');
}

function App() {
  return (
    <button onClick={handleClick}>Click me</button>
  );
}
```

The main difference is that in React, event handling is done using JavaScript rather than inline HTML attributes, which allows for more flexibility and better separation of concerns.

What is Virtual DOM? How it works?

The Virtual DOM is a lightweight, in-memory representation of the actual DOM (Document Object Model) in a web page. It was created by React to optimize the performance of updates to the UI.

When a component's state or props change, React creates a new Virtual DOM tree that represents the updated UI. The new tree is then compared to the previous tree to determine the differences (or "diffs") between them. React then calculates the minimum number of changes required to update the actual DOM to match the new Virtual DOM, and applies those changes to the DOM.

By using the Virtual DOM, React avoids the need to update the entire DOM tree when a component's state or props change, which can be time-consuming and expensive. Instead, it only updates the parts of the DOM that have changed, resulting in faster and more efficient updates.

How to create components in React? When to use a Class Component over a Function Component? What are Pure Components?

In React, a component is a reusable piece of code that represents a specific part of a user interface. Components can be created in two ways: using class components and using functional components.

A class component is a JavaScript class that extends the React.Component class. It provides access to more features, such as state, lifecycle methods, and other advanced features. Class components are used when you need more features than just rendering data.

A functional component is a JavaScript function that returns a React element. It is simpler to write and easier to understand than class components. Functional components are used when you just need to render data and don't need access to state or lifecycle methods.

A Pure Component is a type of React component that is optimized to prevent unnecessary re-renders. It is similar to a regular component, but it implements shouldComponentUpdate() method with a shallow comparison of current and next props and state. Pure components are used when you need to improve performance by avoiding unnecessary re-renders.

18. What are the lifecycle methods of React?

The React component lifecycle methods can be divided into three main phases: mounting, updating, and unmounting. Here are the methods for each phase:

Mounting phase:

constructor(): This is called when a component is first created, and it's used to initialize state and bind methods.

getDerivedStateFromProps(): This is called when a component is first created and whenever it receives new props, and it's used to update state based on prop changes.

render(): This is called whenever a component needs to be rendered, and it's used to return a description of the component's user interface.

componentDidMount(): This is called after a component has been rendered for the first time, and it's used to initialize anything that requires the DOM, like setting up event listeners or fetching data from an API.

Updating phase:

getDerivedStateFromProps(): This method is also called during the updating phase, as well as during the mounting phase.

shouldComponentUpdate(): This is called before a component is updated, and it's used to decide whether the component should be re-rendered or not.

render(): This is called again to re-render the component if it's determined that it needs to be updated.

getSnapshotBeforeUpdate(): This is called right before the component's changes are committed to the DOM, and it's used to capture some information about the current state of the DOM before it changes.

componentDidUpdate(): This is called after the component's changes have been committed to the DOM, and it's used to update any other parts of the component's state that depend on the changes that were made.

Unmounting phase:

componentWillUnmount(): This is called just before a component is removed from the DOM, and it's used to clean up any resources that the component is using, like event listeners or timers.

These lifecycle methods can be used to control the behavior of React components and manage their state over time, making it easier to create complex and interactive user interfaces.

19. What are Higher-Order components?

Higher-Order components (HOCs) are functions that take a component as input and return a new component with enhanced functionality. HOCs allow you to reuse code and share functionality between multiple components. Some examples of HOCs include:

WithAuth: a HOC that adds authentication capabilities to a component

WithLogging: a HOC that logs component lifecycle events

WithRouter: a HOC that adds routing capabilities to a component

20. What are render props?

Render props is a pattern in React where a component receives a function as a prop that it can use to render its content. The function is typically called with some data or state from the component, and the returned JSX is used to render the component's content. Render props allow for more flexible and reusable components, and are often used in conjunction with other patterns like HOCs and context.

21. What is React Router?

React Router is a library for managing routing in a React application. It allows you to define routes and map them to specific components, and provides features like URL parameters, nested routes, and programmatic navigation. React Router is commonly used in single-page applications and other applications where the URL should reflect the current state of the application.

22. How to perform automatic redirect after login in React?

One way to perform automatic redirect after login in React is to use the Redirect component from React Router. After a successful login, you can set a flag in the component's state to indicate that the user is authenticated, and conditionally render a Redirect component to the desired route.

23. What is Redux? What are the core principles of Redux?

Redux is a state management library for JavaScript applications. It provides a predictable and centralized way to manage application state, and is commonly used in React applications. The core principles of Redux include:

Single source of truth: the entire state of the application is stored in a single object tree

State is read-only: the state can only be changed by dispatching actions

Changes are made with pure functions: reducers are pure functions that take the current state and an action, and return a new state

24. What is an action in Redux? Can I dispatch an action in reducer?

Redux is a state management library for JavaScript applications. It provides a predictable and centralized way to manage application state, and is commonly used in React applications. The core principles of Redux include:

Single source of truth: the entire state of the application is stored in a single object tree

State is read-only: the state can only be changed by dispatching actions

Changes are made with pure functions: reducers are pure functions that take the current state and an action, and return a new state

25. What do you mean by reactive programming? How it works?

Reactive programming is a programming paradigm that emphasizes asynchronous data streams and the propagation of change. In reactive programming, data streams are treated as first-class citizens, and operations can be performed on them as they are emitted. This allows for more declarative and concise code, as well as better scalability and performance. Reactive programming is often used in front-end web development, as well as in other areas like IoT and data analysis.

26. How to invoke REST services in React?

In React, you can invoke REST services using the fetch() API, which allows you to send HTTP requests and receive responses. fetch() returns a Promise that resolves to the response from the server, which you can then use to update the component's state or perform other operations. Alternatively, you can use third-party libraries like Axios or jQuery to handle REST requests.

27. How to implement REST services in Express? Explain CRUD operations (on database) using REST services.

To implement REST services in Express, you can define routes that map to specific CRUD operations on a database. The four basic CRUD operations are: Create: POST request that adds a new record to the database

Read: GET request that retrieves one or more records from the database

Update: PUT or PATCH request that modifies an existing record in the database

Delete: DELETE request that removes a record from the database

To handle these operations in Express, you can define routes with the appropriate HTTP method and endpoint, and use a database library like Sequelize or Mongoose to interact with the database. For example, a route to create a new user might look like this:

```
app.post('/users', async (req, res) => {
  const newUser = req.body;
  const result = await User.create(newUser);
  res.json(result);
});
```

This route handles a POST request to the /users endpoint, creates a new user in the database using the User model, and sends back the result as JSON.

28. What is event loop in Node JS? Why Node JS is single-threaded?

The event loop is a key feature of Node.js that allows for asynchronous I/O and non-blocking operations. It is a loop that constantly checks for new events to process, and executes them one at a time. Each event is processed in a separate turn of the loop, and callbacks are used to signal when an operation is complete. The event loop is single-threaded, meaning that all events are processed on a single thread of execution, but it is highly efficient and can handle many concurrent requests.

Node.js is single-threaded because it was designed to optimize for scalability and performance in server-side applications. By using asynchronous I/O and non-blocking operations, Node.js can handle many concurrent connections and requests without blocking the main thread of execution. This allows for highly scalable and performant applications, especially in cases where I/O operations are a bottleneck. However, it also means that Node.js is not well-suited for CPU-bound tasks, as these can block the event loop and reduce overall performance.