REACT INTERVIEW QUESTIONS

What Is React?

React is a popular JavaScript library used for building user interfaces (UIs) for web applications. It's known for its component-based architecture and efficient updates, making it a great choice for creating dynamic and interactive web pages. React allows developers to create reusable UI components, which can then be combined to build complex applications.

Key Features of React:

* **Component-Based:**

React applications are built using components, which are independent and reusable pieces of the UI.

* **Virtual DOM:**

React uses a Virtual DOM, which is an in-memory representation of the actual DOM. This allows for efficient updates by only modifying the parts of the UI that need to change.

* **JSX:**

React uses JSX, a syntax extension that allows developers to write HTML-like structures within their JavaScript code, making it easier to define UI components.

* **One-Way Data Flow:**

React enforces a unidirectional data flow, meaning data flows from parent components to child components, making it easier to understand and debug the application.

Benefits of Using React:

* **Improved User Experience:**

By efficiently updating only the necessary parts of the UI, React can improve the performance and responsiveness of web applications.

* **Reusability:**

Components can be reused across different parts of an application, reducing code duplication and development time.

* **Scalability:**

React's component-based architecture makes it easier to build large and complex applications.

In summary, React is a powerful and versatile JavaScript library that provides a declarative and efficient way to build user interfaces for web applications.

Lazy Loading

Lazy loading in React is a technique used to improve the performance of React applications by deferring the loading of non-essential components or resources until they are actually needed. It helps to reduce the initial load time of the application, especially when dealing with large components, images, or other media.

React provides built-in support for lazy loading using React.lazy() and <Suspense>. React.lazy() is a function that lets you dynamically import a component, while <Suspense> is a component that allows you to display a fallback UI (e.g., a loading spinner) while waiting for the lazy-loaded component to load.

Here's a basic example of how to implement lazy loading in React:

A screenshot of a computer program

AI-generated content may be incorrect.

In this example, LazyComponent is only loaded when it is about to be rendered. The <Suspense> component wraps LazyComponent and displays "Loading..." while it is being loaded. Once LazyComponent is loaded, it will be rendered in place of the fallback UI.

Lazy loading can be used for various types of resources, such as:

* **Components:** Defer loading components that are not immediately needed, such as components in different routes or tabs.
* **Images:** Load images only when they are visible in the viewport.
* **Other media:** Defer loading videos, audio, or other large media files until they are needed.

Benefits of lazy loading:

* **Improved initial load time:**

By loading only the necessary resources initially, the application can load and become interactive faster.

* **Reduced bandwidth usage:**

Unnecessary resources are not downloaded, saving bandwidth for both the user and the server.

* **Better user experience:**

Users can start interacting with the application sooner, leading to a more positive experience.

* **Optimized resource usage:**

Lazy loading can help to reduce the workload on the browser and improve overall performance, especially in low bandwidth situations.

**How to implement the lazy loading on the images?**

Modern browsers support lazy loading natively using the loading="lazy" attribute.

Code Spliting?

Code splitting is a technique used in React to divide the application's code into smaller chunks, which can then be loaded on demand. This approach reduces the initial load time of the application, leading to improved performance and a better user experience, especially on slow or unreliable networks.

Implementation

One way to implement code splitting in React is by using dynamic imports with React.lazy and Suspense. React.lazy allows you to render a dynamic import as a regular component, while Suspense lets you display a fallback UI (like a loading spinner) while the component is loading.

A screenshot of a computer program

AI-generated content may be incorrect.

Benefits

* **Reduced initial load time**: Only the necessary code is loaded initially.
* **Improved performance**: Faster page loads and smoother user interactions.
* **Optimized caching**: Code chunks can be cached by the browser, improving subsequent loads.

Best Practices

* Analyze your bundles using tools like webpack-bundle-analyzer to identify splitting opportunities.
* Focus on splitting out large libraries and features that aren't immediately needed.
* Use code splitting for larger features or sections of your application and lazy loading for individual components.
* Handle loading errors using error boundaries.

How They Work Together

* **Code splitting** creates smaller chunks of code.
* **Lazy loading** loads those chunks only when needed.

So, **code splitting is the mechanism**, and **lazy loading is the strategy** that uses it.

Imagine a **bookstore**:

* **Code Splitting** is like organizing books into different shelves by genre (smaller, manageable sections).
* **Lazy Loading** is like only bringing a book to the counter when a customer asks for it (on-demand access).

Debouncing and Throttling

**Debouncing and throttling are techniques used to limit the frequency of function execution,** especially in response to events like scrolling, resizing, or keypresses. They help improve performance and user experience by limiting the rate at which functions are called.

**Debouncing**

Debouncing delays a function call until a period of inactivity, ensuring it executes only once after a sustained period of the triggering event.It's useful for scenarios like search input fields, where you want to wait until the user has stopped typing.

A computer code with colorful text

AI-generated content may be incorrect.

**Throttling**

**Throttling** ensures that a function is executed **at most once every specified interval**, no matter how many times the event is triggered. It's useful for things like window resizing or scroll events.

A screen shot of a computer code

AI-generated content may be incorrect.

**Key Differences**

| **Feature** | **Debouncing** | **Throttling** |
| --- | --- | --- |
| Execution | After delay since last call | At regular intervals |
| Use Case | Search input, auto-save | Scroll, resize, mouse move |
| Frequency | Executes once after inactivity | Executes at most once per interval |

**How to concatenate two objects?**

**1. Using the Spread Operator (...)**

A close-up of a code

AI-generated content may be incorrect.

If both objects have the same key, the value from the second object (obj2) will overwrite the one from the first.

**2. Using Object.assign()**

A close-up of a code

AI-generated content may be incorrect.

This also overwrites values from obj1 with those from obj2 if keys overlap.

**3. Merging Deeply (for nested objects)**

For deep merging (merging nested objects), you’ll need a custom function or a library like Lodash:

A close-up of a number

AI-generated content may be incorrect.

**How to convert object to array?**

A computer code with green text

AI-generated content may be incorrect.

const obj = { a: 1, b: 2, c: 3 };

// Convert to array of values

const values = Object.values(obj); // [1, 2, 3]

// Convert to array of keys

const keys = Object.keys(obj); // ['a', 'b', 'c']

// Convert to array of key-value pairs

const entries = Object.entries(obj); // [['a', 1], ['b', 2], ['c', 3]]