Interview Assessment: Web Development & React:

**Part 1: Web Development Basics**

1. **What are HTML, CSS, and JavaScript? Briefly explain their roles in web development.**

**Answer:**

* + **HTML (HyperText Markup Language)** is the standard language used to create and structure content on the web. It provides the structure (headings, paragraphs, links, etc.).
  + **CSS (Cascading Style Sheets)** is used for styling HTML elements (colors, fonts, layouts).
  + **JavaScript** is a programming language used to create dynamic content on web pages, allowing interaction and changes without reloading the page.

1. **What is the purpose of the DOCTYPE declaration in HTML?**

**Answer:**  
The DOCTYPE declaration tells the browser which version of HTML the page is written in, ensuring that the browser renders the page correctly. For modern web development, <!DOCTYPE html> is used for HTML5.

1. **Describe the difference between id and class attributes in HTML.**

**Answer:**

* + **id:** Used to uniquely identify an element within a page. Each id must be unique on a page.
  + **class:** Used to assign an element to a group of elements that share common styles. Multiple elements can share the same class.

1. **What are semantic HTML elements? Give three examples.**

**Answer:**  
Semantic HTML elements are elements that have meaningful names and describe their content. Examples:

* + <article>
  + <section>
  + <header>

1. **Explain the CSS box model and its components.**

**Answer:**  
The CSS box model describes the rectangular boxes that elements generate on a page. The components are:

* + **Content:** The actual content (text, images, etc.).
  + **Padding:** Space between the content and the border.
  + **Border:** The border surrounding the padding (optional).
  + **Margin:** Space outside the border.

1. **How does flexbox differ from grid in CSS layout?**

**Answer:**

* + **Flexbox** is designed for one-dimensional layouts (either horizontal or vertical).
  + **Grid** is designed for two-dimensional layouts (both rows and columns).

1. **What is event bubbling in JavaScript?**

**Answer:**  
Event bubbling is a concept where an event triggered on an element propagates (bubbles) up to its ancestors in the DOM tree. This allows for event delegation and handling events on parent elements instead of individual child elements.

1. **What is the difference between == and === in JavaScript?**

**Answer:**

* + == is a loose equality operator that compares values, allowing type coercion.
  + === is a strict equality operator that compares both values and types, without type coercion.

1. **Explain the concept of responsive design. How can you implement it?**

**Answer:**  
Responsive design is a web design approach that ensures web pages look good and are usable across all devices and screen sizes. It can be implemented using:

* + **Media queries** to apply different styles based on screen size.
  + **Flexible grid layouts** and **images** that adjust based on the viewport.

1. **What is AJAX, and why is it used?**

**Answer:**  
AJAX (Asynchronous JavaScript and XML) allows web pages to retrieve data from a server asynchronously (without reloading the page), making web applications faster and more interactive.

**Part 2: React**

1. **What is React, and what problem does it solve?**

**Answer:**  
React is a JavaScript library used to build user interfaces, particularly single-page applications. It solves the problem of efficiently updating and rendering UI components when data changes, using a virtual DOM.

1. **What is a component in React? Explain the difference between class and functional components.**

**Answer:**  
A **component** is a self-contained unit of UI in React. Components can be either:

* + **Class components:** Define a component using a class and extend React.Component. They have access to lifecycle methods and can manage state.
  + **Functional components:** Defined using functions and can use hooks (like useState and useEffect) to manage state and side effects.

1. **How does state management work in React?**

**Answer:**  
State in React is an object that determines the behavior and rendering of a component. When the state changes, the component re-renders to reflect the changes. State can be managed using useState in functional components or this.state in class components.

1. **Explain the purpose of props in React.**

**Answer:**  
Props (short for properties) are used to pass data from a parent component to a child component. Props are immutable in the child component.

1. **What is JSX?**

**Answer:**  
JSX (JavaScript XML) is a syntax extension for JavaScript that allows you to write HTML-like code within JavaScript. JSX is then compiled into React elements.

1. **How do you pass data from a parent component to a child component in React?**

**Answer:**  
Data can be passed from a parent component to a child component using **props**. The parent component provides the data, and the child component receives it as an argument.

1. **What are React hooks? Name at least three commonly used hooks.**

**Answer:**  
React hooks are functions that let you use state and other React features in functional components. Common hooks:

* + useState: Manages state.
  + useEffect: Handles side effects like data fetching.
  + useContext: Accesses context values.

1. **What does the useEffect hook do? Give a basic usage example.**

**Answer:**  
useEffect is used to handle side effects in functional components, such as data fetching, subscriptions, or manual DOM manipulations.  
Example:

javascript

Copy

useEffect(() => {

fetchData();

}, []); // Empty dependency array means it runs once after the component mounts

1. **How do you handle events in React?**

**Answer:**  
In React, events are handled similarly to DOM events, but with slight differences:

* + React uses camelCase syntax for events (e.g., onClick, onChange).
  + You pass a function to the event handler.  
    Example:

javascript

Copy

function handleClick() {

console.log('Button clicked!');

}

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

1. **What is the virtual DOM and how does React use it?**

**Answer:**  
The virtual DOM is an in-memory representation of the real DOM. React uses it to optimize updates by first updating the virtual DOM, comparing it to the real DOM (reconciliation), and then updating only the changed parts of the real DOM.

**Part 3: Advanced React**

1. **What is context in React? When should you use it?**

**Answer:**  
Context allows you to share values (like themes, authentication status) across your component tree without having to pass props down manually at every level. Use context when you need to pass data deeply in the component tree.

1. **Explain the concept of higher-order components (HOCs).**

**Answer:**  
HOCs are functions that take a component and return a new component with additional props or functionality. They are used for code reuse, such as adding authentication or logging features.

1. **What are render props in React?**

**Answer:**  
Render props is a pattern where a component takes a function as a prop and calls it to know what to render. It allows for greater flexibility and reusable logic.

1. **How would you optimize performance in a large React application?**

**Answer:**

* + Use **React.memo** to prevent unnecessary re-renders.
  + Implement **code-splitting** using React.lazy and Suspense.
  + Optimize components with **shouldComponentUpdate** or **React.PureComponent**.
  + Use **useCallback** and **useMemo** hooks to memoize functions and values.

1. **What are React portals, and when would you use them?**

**Answer:**  
React portals allow you to render children into a different DOM node outside of the current component's hierarchy. It's useful for modals, tooltips, and overlays.

1. **Explain error boundaries in React.**

**Answer:**  
Error boundaries are React components that catch JavaScript errors in any child component during rendering, lifecycle methods, and constructors, and display a fallback UI instead of crashing the component tree.

1. **How does React handle reconciliation?**

**Answer:**  
React uses an efficient diffing algorithm to compare the virtual DOM and the real DOM. It finds the minimal number of changes needed to update the real DOM.

1. **What is code-splitting and how can it be implemented in React?**

**Answer:**  
Code-splitting is the practice of breaking down large bundles into smaller chunks, which can be loaded on demand. It can be implemented using React.lazy and Suspense.

1. **Describe how custom hooks work and give an example use case.**

**Answer:**  
Custom hooks are JavaScript functions that can use React hooks and share logic between components. Example:

javascript

Copy

function useLocalStorage(key, initialValue) {

const [value, setValue] = useState(() => {

const saved = localStorage.getItem(key);

return saved ? JSON.parse(saved) : initialValue;

});

useEffect(() => {

localStorage.setItem(key, JSON.stringify(value));

}, [value]);

return [value, setValue];

}

1. **What is server-side rendering (SSR) in React, and why would you use it?**

**Answer:**  
SSR is the process of rendering React components on the server and sending the fully rendered page to the client. It improves initial page load speed and SEO.

**Part 4: Scenario-Based Questions**

1. **A user submits a form, but nothing happens. Walk me through how you would debug this issue in a React app.**

**Answer:**

* + Check if the form has a submit event handler attached.
  + Ensure that the onSubmit handler is properly bound to the form.
  + Check the onClick or onSubmit function for errors or missing logic.
  + Check the console for any errors.
  + If it's an API request, ensure the API endpoint is correct and reachable.
  + If using state to capture form data, check if the state updates correctly.

1. **You notice a component is re-rendering excessively and impacting performance. What steps would you take to identify and resolve the issue?**

**Answer:**

* + Use React DevTools to inspect why the component re-renders.
  + Check if the component's parent is re-rendering unnecessarily, causing child components to re-render.
  + Optimize re-renders by using **React.memo** or **PureComponent**.
  + Use **useMemo** and **useCallback** to memoize values and functions.

1. **You need to share state between components that are not directly related in the tree. What approaches would you consider?**

**Answer:**

* + Use **Context API** to share state globally across the component tree.
  + Consider using a **state management library** like Redux or Recoil for larger-scale state sharing.

1. **Describe how you would implement authentication in a React application.**

**Answer:**

* + Store authentication data (e.g., token) in **localStorage** or **sessionStorage**.
  + Use React context to share the authentication state across the application.
  + Redirect the user to a login page if they are not authenticated using **React Router**.

1. **A new feature requires fetching data from an API and updating the UI, but the data can also change in real-time (e.g., via WebSockets). How would you design this in React?**

**Answer:**

* + Use useEffect to fetch data on initial mount.
  + Use **WebSocket** (via useEffect with a clean-up function) to receive real-time updates.
  + Use useState to store the data and update the UI as the data changes.

1. **You are handed a legacy React codebase with class components and need to migrate it to hooks. What steps would you follow?**

**Answer:**

* + Identify components that manage state and refactor them to functional components using useState.
  + Replace lifecycle methods like componentDidMount and componentDidUpdate with useEffect.
  + Refactor class methods to hooks where applicable.
  + Test the functionality after each step.

1. **Your app needs to support both dark and light themes. How would you approach this in React?**

**Answer:**

* + Use React context to manage theme state globally.
  + Store the current theme in **localStorage** to persist the user's choice across sessions.
  + Apply different styles based on the theme state, either by toggling class names or using CSS variables.

1. **There’s a bug that only appears on mobile devices. How would you go about diagnosing and fixing it?**

**Answer:**

* + Use browser dev tools to simulate mobile devices and check if the issue can be reproduced.
  + Check for CSS media query issues that might be specific to mobile.
  + Debug JavaScript logic that might behave differently on mobile, such as touch events.

1. **A client wants SEO optimization for their React single-page application. What would you recommend?**

**Answer:**

* + Use **server-side rendering (SSR)** to ensure that search engines can crawl the content.
  + Alternatively, use **React Helmet** to manage meta tags and improve SEO.
  + Implement **dynamic rendering** for SEO purposes.

1. **Describe how you would handle and display global error messages throughout your React app.**

**Answer:**

* Use **React context** to store and manage error messages.
* Display global error messages in a consistent UI component, such as a banner or modal.
* Provide an error handling boundary using **error boundaries** in React.