React Concepts Guide

Handling Events in React

Question 1: How are events handled in React compared to vanilla JavaScript? Explain the concept of synthetic events.

In vanilla JavaScript, events are handled directly on DOM elements using methods like `addEventListener` or inline event handlers. React uses **Synthetic Events**, which are React's wrapper around native DOM events.

Synthetic Events:

- Provide consistent behavior across different browsers
- Have the same interface as native events but work uniformly
- Automatically handle browser compatibility issues
- Are pooled for performance optimization
- Can access the original native event through `nativeEvent` property

Question 2: What are some common event handlers in React.js? Provide examples of onClick, onChange, and onSubmit.

Common Event Handlers:

- **onClick**: Triggered when an element is clicked (buttons, divs, etc.)
- **onChange**: Triggered when form input values change (input fields, textareas, selects)
- **onSubmit**: Triggered when a form is submitted
- **onFocus/onBlur**: Triggered when elements gain/lose focus
- **onMouseOver/onMouseOut**: Triggered on mouse hover events
- **onKeyDown/onKeyUp**: Triggered on keyboard events

Question 3: Why do you need to bind event handlers in class components?

In class components, event handlers need to be bound because:

- JavaScript class methods don't automatically bind `this` context
- Without binding, `this` becomes `undefined` in strict mode
- Binding ensures the method can access component's state and props
- Can be done in constructor, using arrow functions, or with property initializer syntax

Conditional Rendering

Question 1: What is conditional rendering in React? How can you conditionally render elements in a React component?

Conditional Rendering is the ability to render different UI elements based on certain conditions or state values. It allows components to display different content dynamically.

Methods to conditionally render:

- If-else statements (outside JSX)
- Ternary operators (inside JSX)
- Logical AND (&&) operator
- Switch statements
- Immediately Invoked Function Expressions (IIFE)

Question 2: Explain how if-else, ternary operators, and && (logical AND) are used in JSX for conditional rendering.

If-else: Used outside JSX return statement to determine what to render based on conditions.

Ternary Operator: Used directly inside JSX for inline conditional rendering. Format: `condition ? elementIfTrue : elementIfFalse`

Logical AND (&&): Used when you want to render something or nothing. Format: `condition && elementToRender`. If condition is false, nothing renders.

Forms in React

Question 1: How do you handle forms in React? Explain the concept of controlled components.

Form Handling in React: Forms are handled by managing form data through component state rather than DOM manipulation.

Controlled Components:

- Form elements whose values are controlled by React state
- Input values are set by state and updated through event handlers
- React state becomes the "single source of truth"
- Changes trigger re-renders with updated values

- Provides better control and validation capabilities

Question 2: What is the difference between controlled and uncontrolled components in React?

Controlled Components:

- Values managed by React state
- Require onChange handlers to update state
- Provide immediate validation and formatting
- Better for complex forms with interdependent fields

Uncontrolled Components:

- Values managed by DOM itself
- Use refs to access values when needed
- Simpler to implement for basic forms
- Less control over user input and validation

Lifecycle Methods (Class Components)

Question 1: What are lifecycle methods in React class components? Describe the phases of a component's lifecycle.

Lifecycle Methods are special methods that run at specific points in a component's existence.

Three Main Phases:

1. **Mounting**: Component is being created and inserted into DOM
2. **Updating**: Component is being re-rendered due to state/props changes
3. **Unmounting**: Component is being removed from DOM

Each phase has specific methods that can be overridden to perform actions.

Question 2: Explain the purpose of componentDidMount(), componentDidUpdate(), and componentWillUnmount().

componentDidMount():

- Runs after component is mounted to DOM
- Perfect for API calls, subscriptions, timers
- Equivalent to useEffect with empty dependency array in hooks

componentDidUpdate():

- Runs after component updates (re-renders)
- Used for operations based on prop/state changes
- Receives previous props and state as parameters
- Must include condition checks to avoid infinite loops

componentWillUnmount():

- Runs just before component is removed from DOM
- Used for cleanup activities
- Cancel timers, remove subscriptions, cleanup resources
- Prevents memory leaks

Routing in React (React Router)

Question 1: What is React Router? How does it handle routing in single-page applications?

React Router is a standard routing library for React applications that enables navigation between different components/pages without full page reloads.

How it handles SPA routing:

- Uses browser's History API to manipulate URL
- Matches current URL to specific components
- Renders appropriate components based on route
- Maintains application state during navigation
- Enables back/forward button functionality
- Supports nested routing and route parameters

Question 2: Explain the difference between BrowserRouter, Route, Link, and Switch components in React Router.

BrowserRouter:

- Top-level router component that wraps entire application
- Uses HTML5 history API for clean URLs
- Enables routing functionality for child components

Route:

- Defines mapping between URL paths and components
- Renders specific component when path matches current URL
- Supports exact matching and dynamic parameters

Link:

- Creates navigational links without page reloads
- Replaces traditional anchor tags in SPAs
- Updates URL and triggers route changes

Switch:

- Renders only the first matching Route
- Prevents multiple routes from rendering simultaneously
- Ensures exclusive route matching
- Note: Replaced by Routes component in React Router v6