React

1. Explain React Events

In React, **events** are actions or occurrences triggered by user interactions or browser behavior, such as clicking a button, hovering over an element, typing in a text field, or submitting a form. React wraps these browser events in its own event system to ensure consistent behavior across all browsers.

React events are similar to native DOM events but follow a different syntax and use a cross-browser wrapper known as a **SyntheticEvent**. These events can be handled using event listeners that are assigned directly in JSX using camelCase syntax.

Example:

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

Here, onClick is the event and handleClick is the function that will be executed when the event occurs.

---

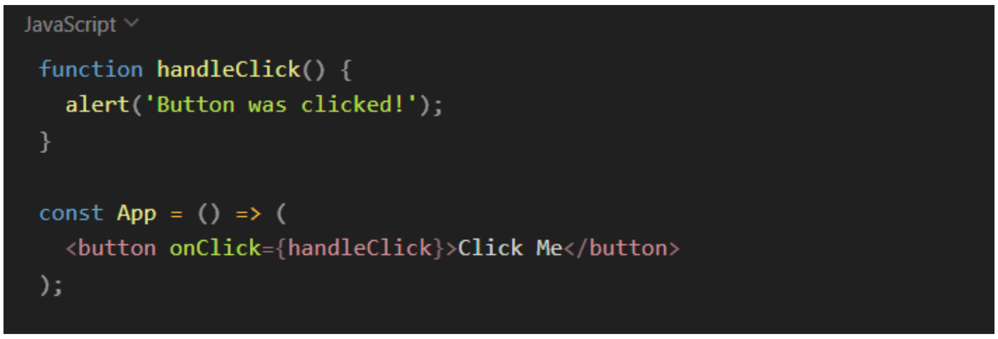
2. Explain About Event Handlers

Event handlers in React are functions that are invoked when a specific event occurs on a component. They are used to define how the application should respond to user actions.

Key characteristics of React event handlers:

- Defined as functions (either traditional or arrow functions).
- Passed as values to event attributes in JSX.
- Can be written inline or defined separately in the component logic.

Example:



```javascript
function handleClick() {
  alert('Button was clicked!');
}

const App = () => (
  <button onClick={handleClick}>Click Me</button>
);
```

Alternatively, inline:

<button onClick={() => alert('Inline click handler')}>Click Me</button>

Event handlers can also accept parameters, typically with an arrow function to avoid immediate execution:

<button onClick={(e) => handleClick(e, 'Hello')}>Click Me</button>

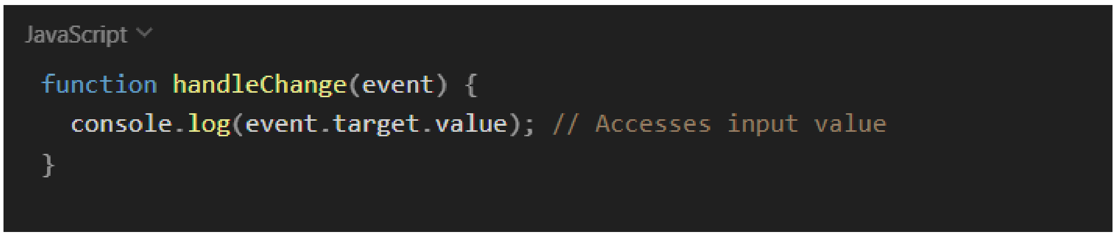
---

3. Define Synthetic Event

A **SyntheticEvent** is a cross-browser wrapper around the native browser event, provided by React. It combines the behavior of different browsers into a single consistent API, ensuring compatibility and predictable results.

Synthetic events are part of React's internal event delegation system, meaning all event handling is managed at the root of the DOM tree, improving performance and consistency.

Example usage:



```javascript
function handleChange(event) {
  console.log(event.target.value); // Accesses input value
}
```

Although it's synthetic, you can still use standard properties like event.target, event.preventDefault(), and event.stopPropagation(). React also pools synthetic events for performance reasons, which means the event object may be reused. If you need to retain it asynchronously, call event.persist().

---

4. Identify React Event Naming Convention

React follows specific **naming conventions** for event attributes to distinguish them from native HTML event attributes:

- Event names use **camelCase** instead of lowercase.
    - Example: onClick instead of onclick, onChange instead of onchange.

- The event handler value must be a **function reference**, not a string.
  <button onClick={handleClick}>Click</button>

This naming convention ensures clarity and consistency in React code and allows for easier integration with JavaScript logic.

Common event attributes include:

- onClick
- onChange
- onSubmit
- onMouseEnter
- onKeyDown
- onFocus
- onBlur

These conventions help in writing clean, readable, and bug-free React components.