Responding to Events in React

1. Types of Events
2. Handling Different Events
3. Event Propagation

- Events are user interactions (click, input, hover, etc.).
- React uses **Synthetic Events** (a wrapper around native events for cross-browser compatibility).
- Event handling in React is slightly different from vanilla JS.

Types of Events

React normalizes events so that they have consistent properties across different browsers.

Here's a table of the most common React events:

Event Type	Event Names
Form Events	`onChange` , `onSubmit` , `onReset` , `onInput`
Mouse Events	`onClick` , `onDoubleClick` , `onMouseEnter` , `onMouseLeave` , `onMouseDown` , `onMouseUp` , `onMouseMove` , `onContextMenu`
Keyboard Events	`onKeyDown` , `onKeyUp`
Focus Events	`onFocus` , `onBlur`
Clipboard Events	`onCopy` , `onCut` , `onPaste`
Drag Events	`onDrag` , `onDragStart` , `onDragEnd` , `onDragOver` , `onDrop`

Here's the complete list of React Events on React official documentation.

Handling Events

Here's a code snippet that explains how you handle events in React:

```
export default function Button() {
  function handleClick() {
    alert('You clicked me!');
  }

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

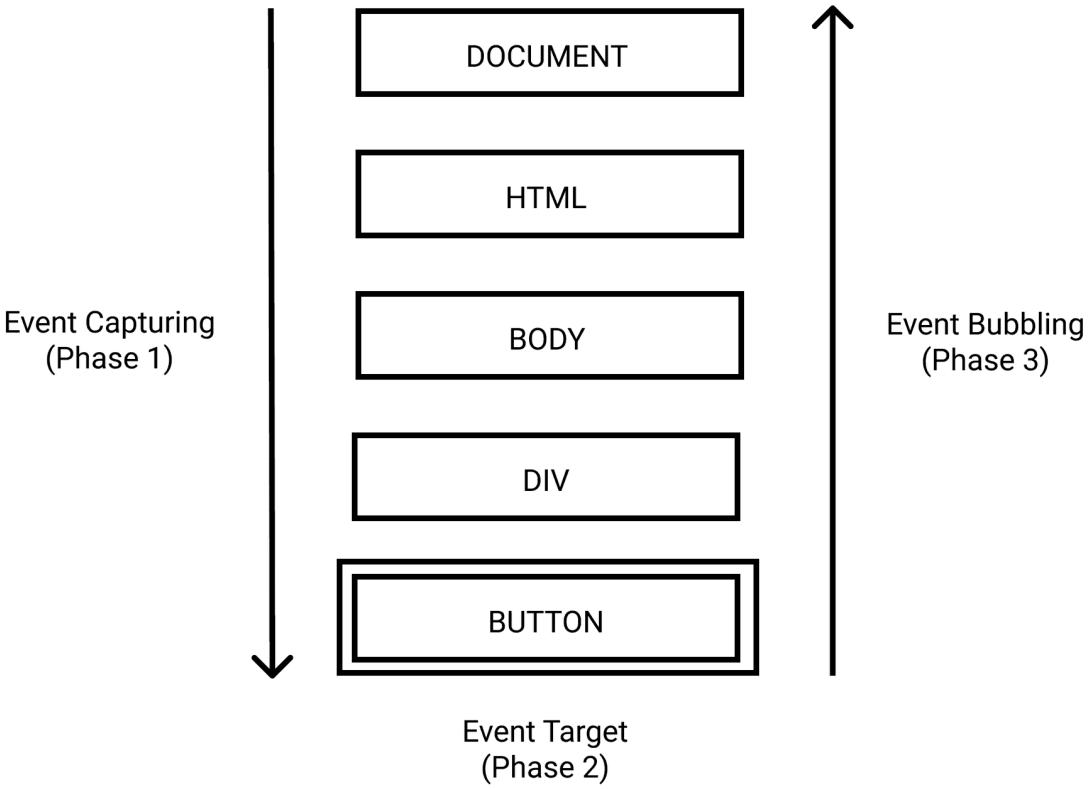
Be sure to pass the function and NOT call the function while handling the events

passing a function (correct)	calling a function (incorrect)
`<button onClick={handleClick}>`	`<button onClick={handleClick()}>`
`<button onClick={() => alert('...')}>`	`<button onClick={alert('...')}>`

This way, rather than executing the code inside with every render, this creates a function to be called later.

Event propagation

Event handlers will also catch events from any children your component might have. We say that an event "bubbles" or "propagates" up the tree: it starts with where the event happened, and then goes up the tree.



Stopping propagation

Event handlers receive an **event object** as their only argument. By convention, it's usually called `e`, which stands for "event". You can use this object to read information about the event.

That event object also lets you stop the propagation. If you want to prevent an event from reaching parent components, you need to call `e.stopPropagation()`

```
function Button({ onClick, children }) {
  return (
    <button onClick={e => {
      e.stopPropagation();
      onClick();
    }}>
      {children}
    </button>
```

```
  );
}
```

Assignment:

1. Toggle a class in the body tag when a button is clicked. Use `document.body` to access the body.
2. Follow the steps below:
    a. Create a list of items (e.g., an array of fruits).
    b. Render each item as a button inside a child component.
    c. Log the item's text to the console when a button is clicked.