React Lecture - 9

More on Events

Passing Event Handlers as Props

Passing Event Handlers as Props

Often you'll want the parent component to specify a child's event handler.

Consider buttons: depending on where you're using a Button component, you might want to execute a different function—perhaps one plays a movie and another uploads an image.

To do this, pass a prop the component receives from its parent as the event handler like so:

Example

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

function PlayButton({ movieName }) {
  function handlePlayClick() {
    alert(`Playing ${movieName}!`);
  }

  return (
    <Button onClick={handlePlayClick}>
      Play "{movieName}"
    </Button>
  );
}

function UploadButton() {
  return (
    <Button onClick={() => alert('Uploading!')}>
      Upload Image
    </Button>
  );
}

export default function Toolbar() {
  return (
    <div>
      <PlayButton movieName="Kiki's Delivery Service" />
      <UploadButton />
    </div>
  );
}

```

Naming event handler props

Built-in components like <button> and only support browser event names like onClick

However, when you're building your own components, you can name their event handler props any way that you like.

By convention, event handler props should start with on, followed by a capital letter.

For example, the Button component's onClick prop could have been called onSmash

Event Bubbling

Similar to Javascript events in React also bubbles.
If in case you want to stop the bubbling you can event.stopPropagation() method

Preventing Default Behaviour

Similar to javascript to prevent default behaviour in React we can use the event object method preventDefault