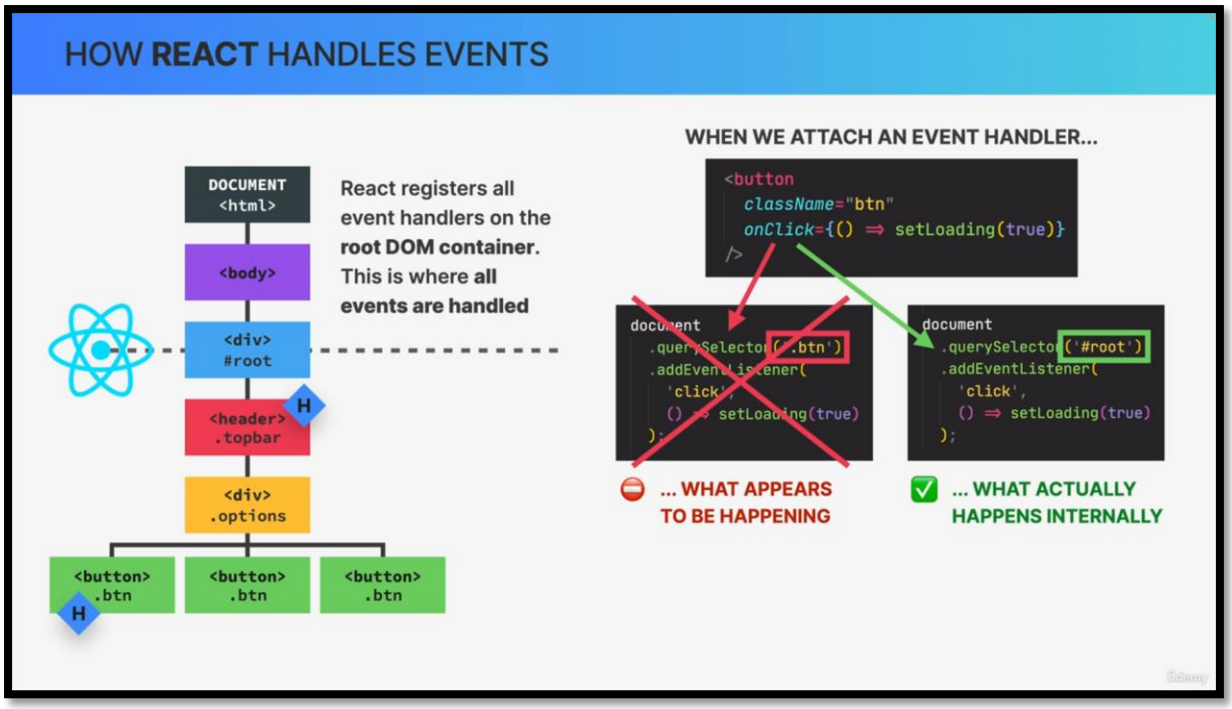


let's consider this same DOM tree and let's say again that we want to attach an event handler to one of the buttons, or even to some other DOM element and this is what that would look like in React code.

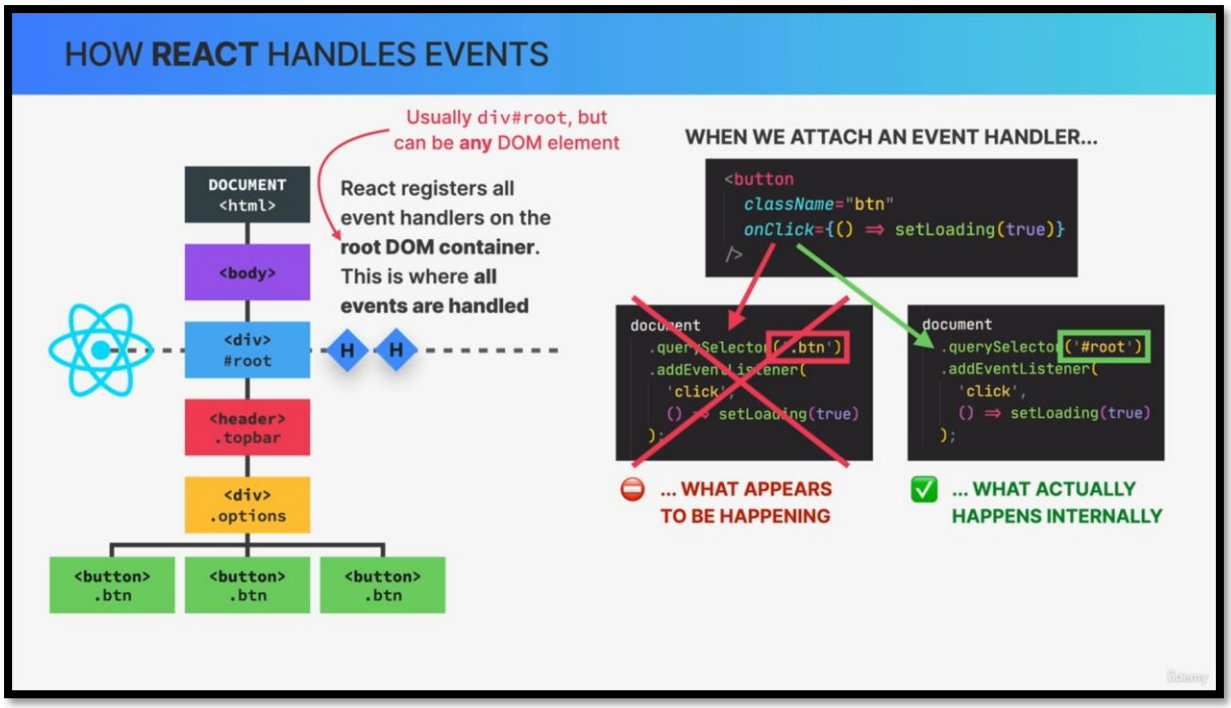
We would simply use the onClick prop to listen for click events, and then pass it a function. So that's really easy.

If we think about how React actually registers these event handlers behind the scenes, we might believe that it would look something like this.

So, React might select a button and then add the event handler to that element. So, that sounds pretty logical.

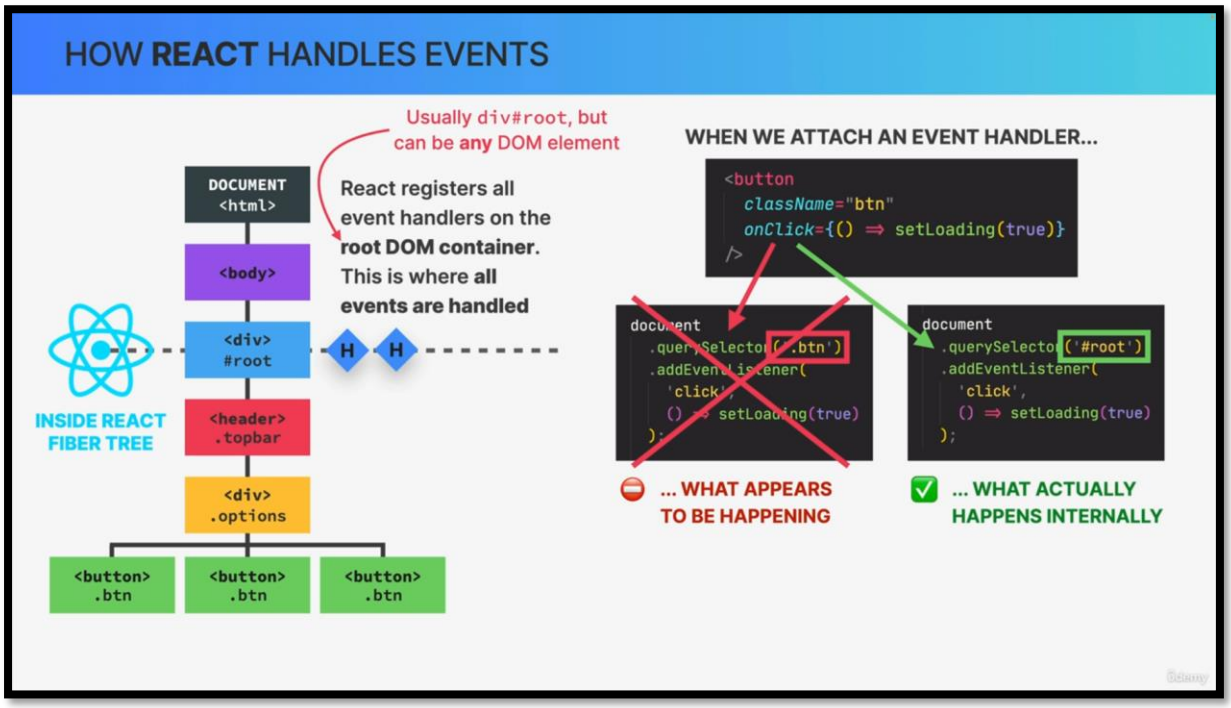
However, this is actually not what React does internally. Instead, what React actually does is to register this and all other event handler functions to the root DOM container and that root container is simply the DOM element in which the React app is displayed.

If we use the default Create React App, that's usually the div element with an ID set to root.

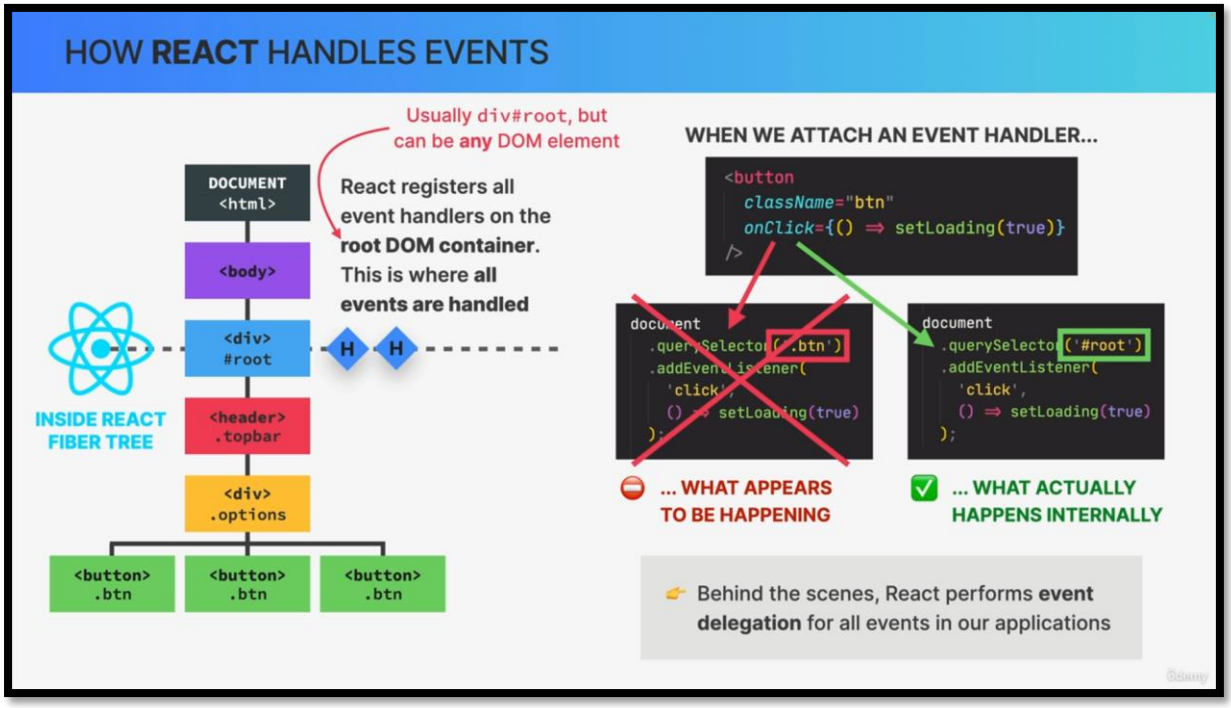


So again, instead of selecting the button where we actually placed our event handler, we can imagine that React selects the root element, and then adds all our event handlers to that element.

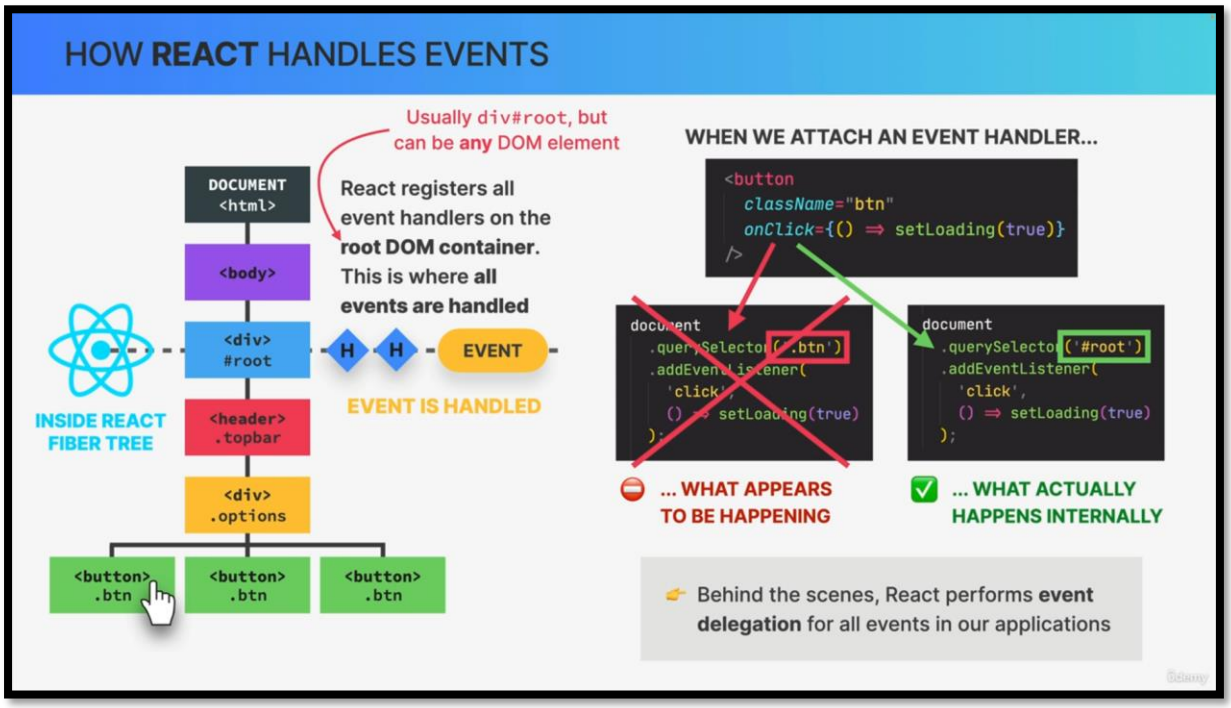
React physically registers one event handler function per event type and it does so at the root note of the Fiber tree during the render phase.



If we have multiple onClick handlers in our code React we'll actually somehow bundle them all together and just add one big onClick handler to the root node of the fiber tree and so this is yet another important function of the fiber tree.



React actually performs event delegation for all events in our applications. So, we can say that React delegates all events to the root DOM container, because that's where they will actually get handled, not in the place where we thought we registered them.



Whenever a click happens on the button, a new event object is fired off which will then travel down the DOM tree until it reaches the target element.

From there, the event will bubble back up. Then as soon as the event reaches the root container where React registered all our handlers, the event will actually finally get handled according to whatever handlers match the event and the target element.

Finally, once that's all done, the event, of course, continues bubbling up until it disappears into nowhere and the beauty of this is that it all happens automatically and invisibly just to make our React apps yet a little bit more performant.