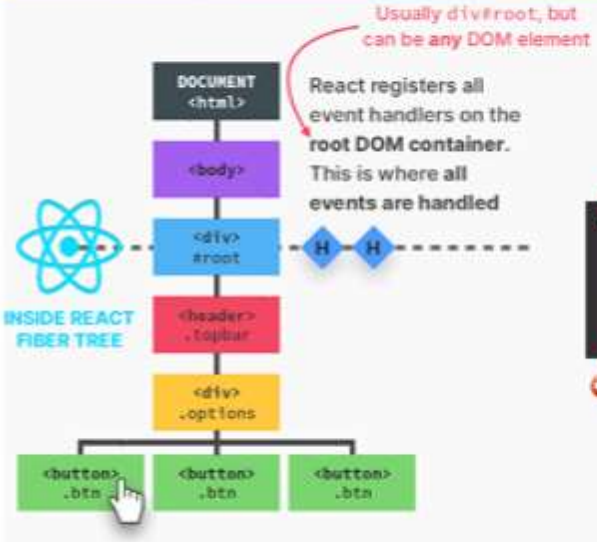

React
EVENTS
DOM REFRESHER: EVENT PROPAGATION AND DELEGATION
EVENT
DOCUMENT
<html>
DOM tree (not Fiber tree or React element tree)
EVENT DELEGATION
1 CAPTURING PHASE
3 BUBBLING PHASE
<body>

#root
<header>
.topBar

.options
<button>
.btn
<button>
.btn
<button>
.btn
H
By default, event handlers listen to events on the target and during the bubbling phase
We can prevent bubbling with e.stopPropagation()
Handling events for multiple elements centrally in one single parent element
Better for performance and memory, as it needs only one handler function
1 Add handler to parent (.options)
2 Check for target element (e.target)
3 If target is one of the <button>s, handle the event
Very common in vanilla JS apps, but not so much in React apps
2 TARGET ELEMENT

React

HOW **REACT** HANDLES EVENTS



React registers all event handlers on the **root DOM container**. This is where **all events are handled**

WHEN WE ATTACH AN EVENT HANDLER...

```
<button
  className="btn"
  onClick={() => setLoading(true)}
/>
```

```
document
  .querySelector('.btn')
  .addEventListener(
    'click',
    () => setLoading(true)
  );
```

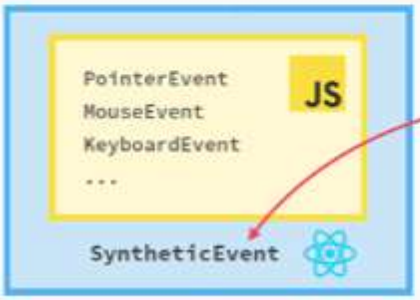
... WHAT APPEARS TO BE HAPPENING

```
document
  .querySelector('#root')
  .addEventListener(
    'click',
    () => setLoading(true)
  );
```

... WHAT ACTUALLY HAPPENS INTERNALLY

👉 Behind the scenes, React performs **event delegation** for all events in our applications

SYNTHETIC EVENTS




```
<input onChange={(e) => setText(e.target.value)} />
```

- 👉 Wrapper around the DOM's native event object
- 👉 Has **same interface** as native event objects, like `stopPropagation()` and `preventDefault()`
- 👉 Fixes browser inconsistencies, so that events work in the exact **same way in all browsers**
- 👉 **Most synthetic events bubble** (including focus, blur, and change), except for scroll

EVENT HANDLERS IN VS. JS

- 👉 Attributes for event handlers are named using **camelCase** (`onClick` instead of `onclick` or `click`)
- 👉 Default behavior can **not** be prevented by returning `false` (only by using `preventDefault()`)
- 👉 Attach "`Capture`" if you need to handle during **capture phase** (example: `onClickCapture`)

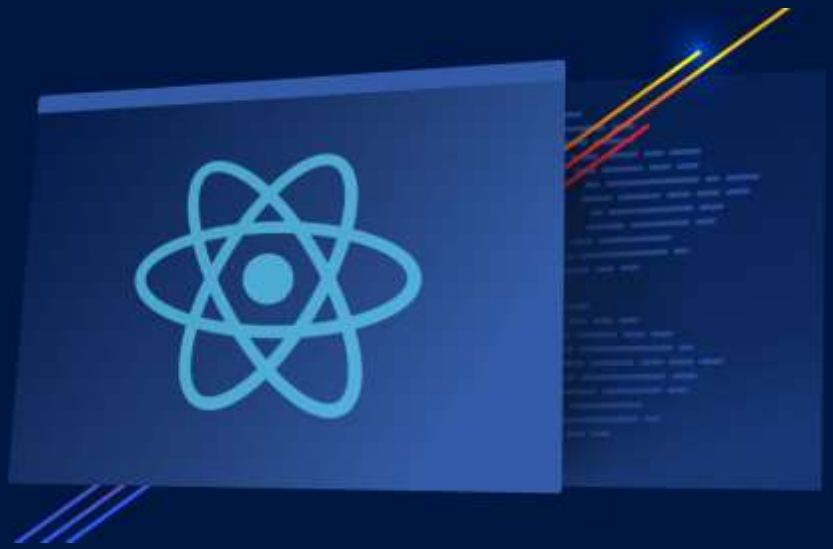
React encourages attaching event handlers to the most top-level element that makes sense. This is not necessarily the root DOM container but often a common ancestor for the elements you want to handle events on.

React uses event delegation internally to efficiently manage events on a higher level and delegate handling to the appropriate components.

React

Do you find it helpful?

Let me know down in the comments



Click To Follow For More On LinkedIN