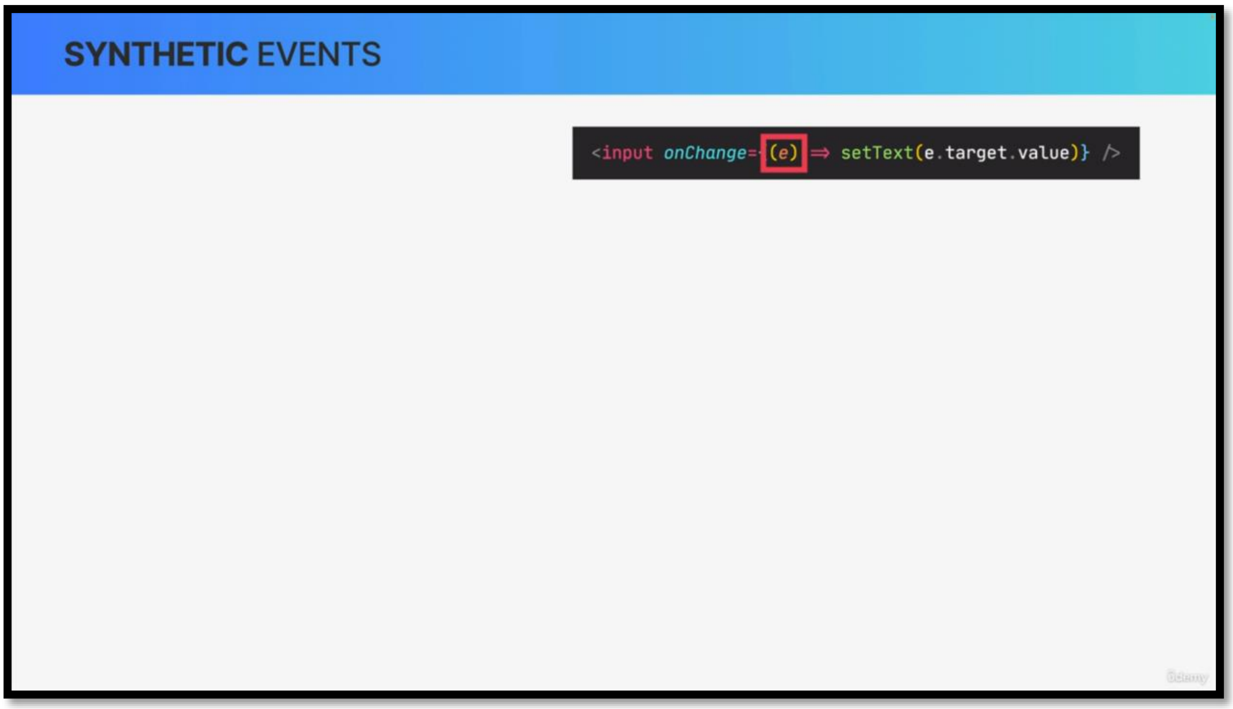
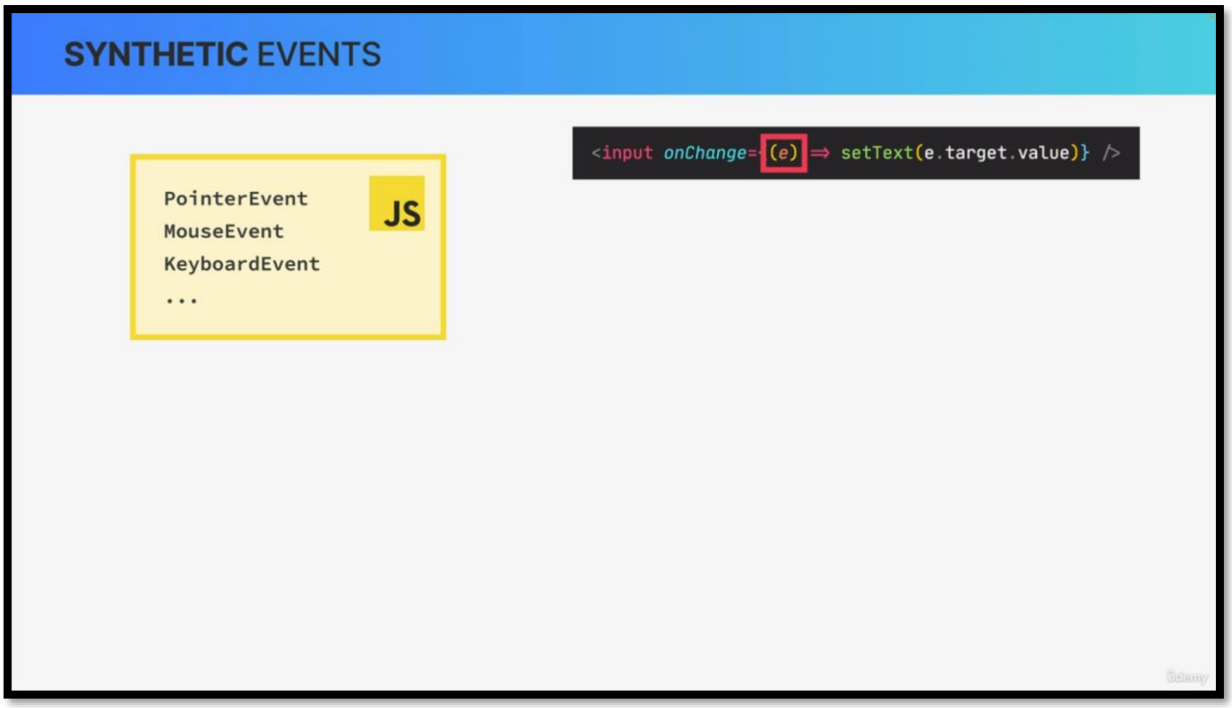

SYNTHETIC EVENTS

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

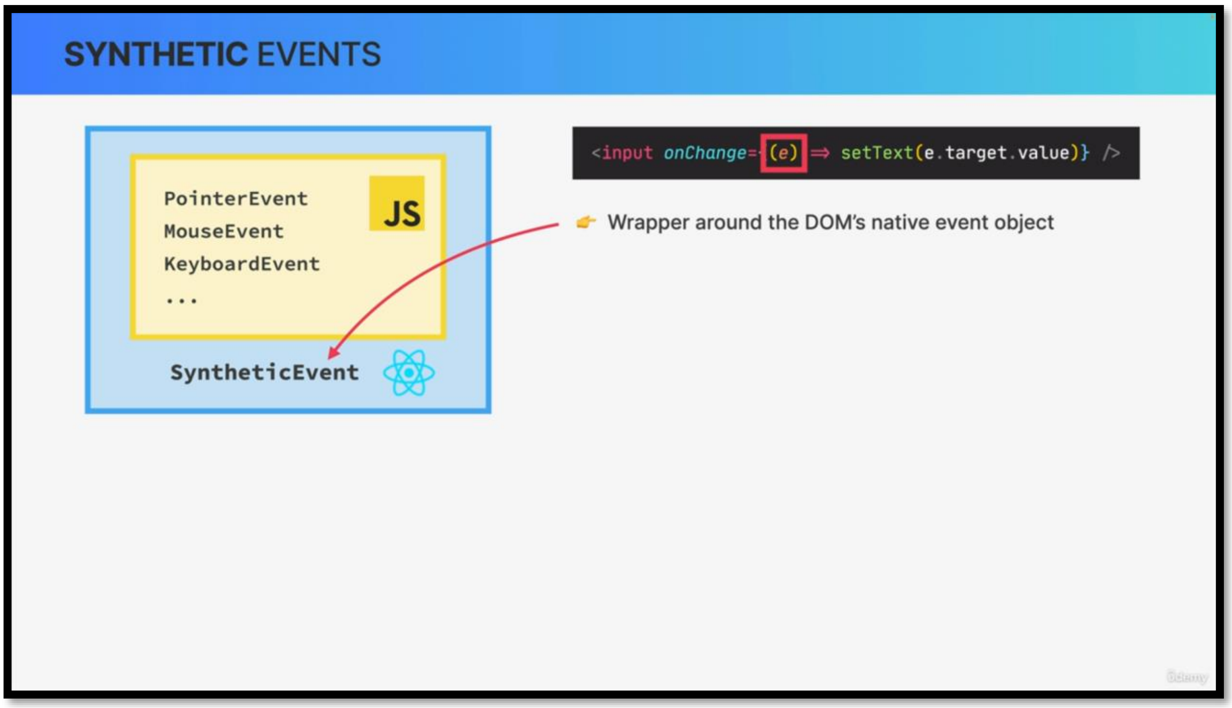
Whenever we declare an event handler like this one, React gives us access to the event object that was created, just like in vanilla JavaScript.


SYNTHETIC EVENTS

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

However, in React, this event object is actually different. So, in vanilla JavaScript, we simply get access to the native DOM event object, for example, pointer event, mouse event, keyboard event, and many others. React, on the other hand, will give us something called a synthetic event which is basically a thin wrapper around the DOM'S native event object.

So, these synthetic events have the same interface as native event objects
and that includes the important methods stopPropagation and
preventDefault.

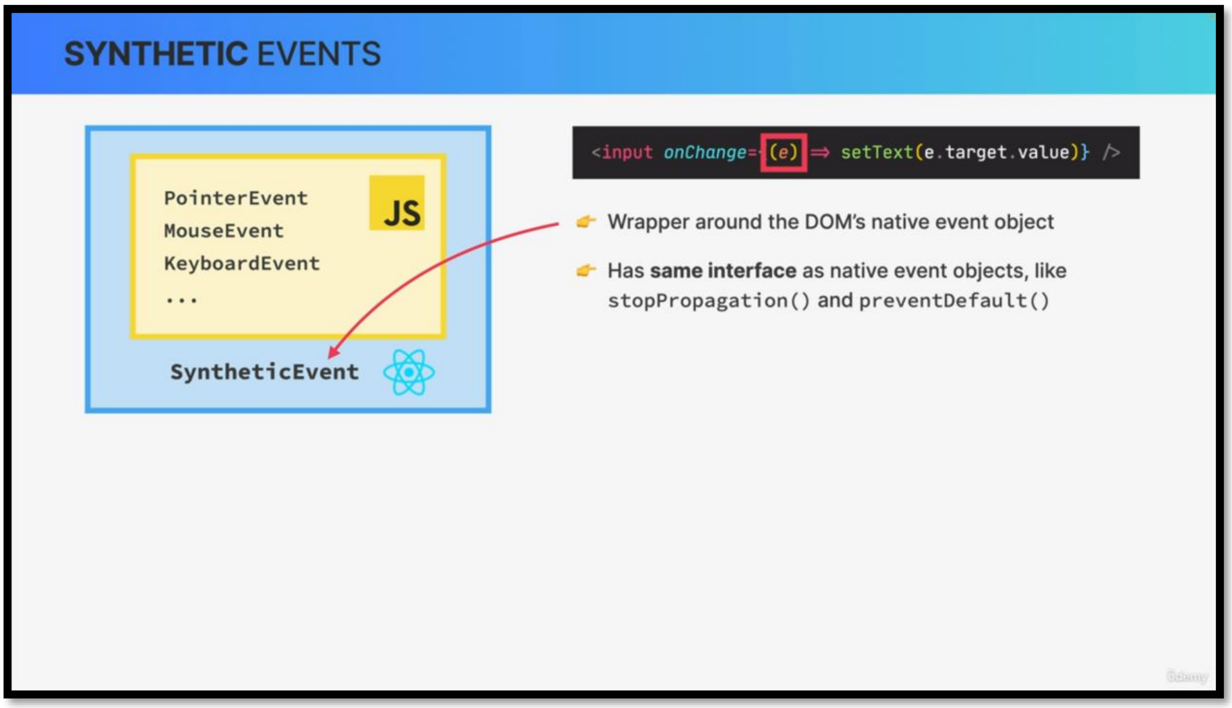


SYNTHETIC EVENTS

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

- Wrapper around the DOM's native event object

By wrapper we simply mean that synthetic events are pretty similar to
native event objects, but they just add or change some functionalities on
top of them.

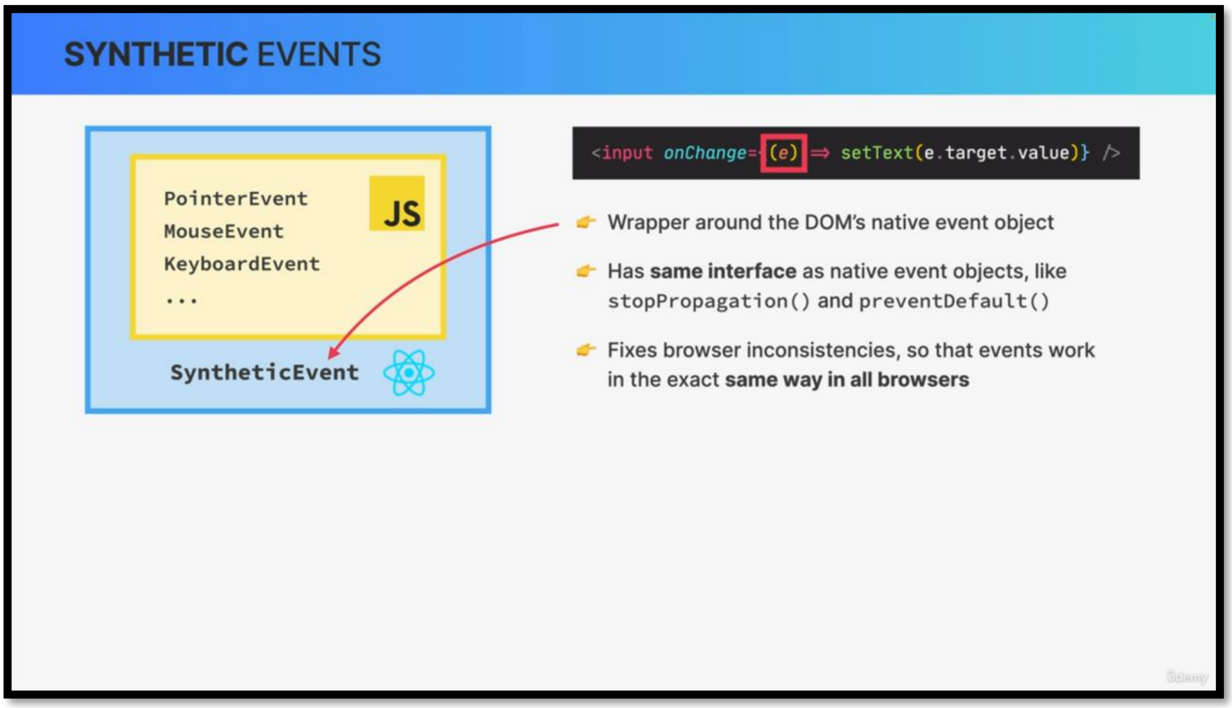


SYNTHETIC EVENTS

```
<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()`

So, these synthetic events have the same interface as native event objects
and that includes the important methods stopPropagation and
preventDefault.



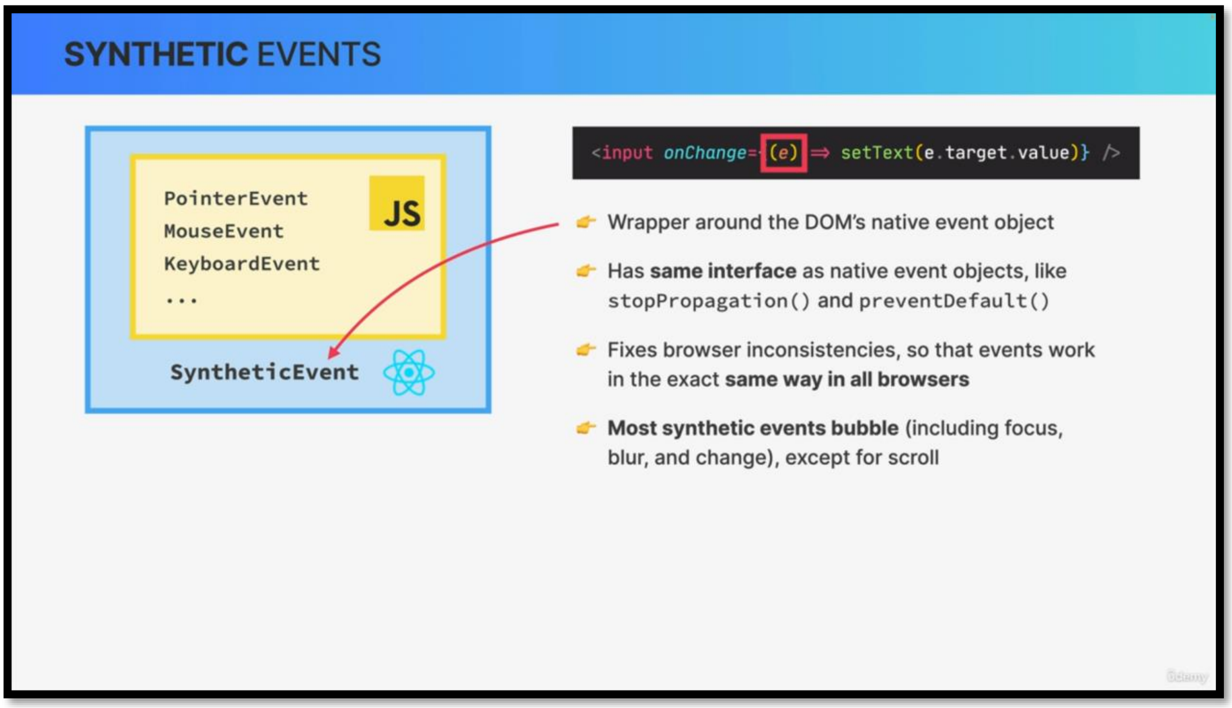
SYNTHETIC EVENTS

```
PointerEvent
MouseEvent
KeyboardEvent
...
```

```
<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**

What's special about synthetic events though and one of the reasons why the React team decided to implement them is the fact that they fix some browser inconsistencies making it so that events work in the exact same way in all browsers.



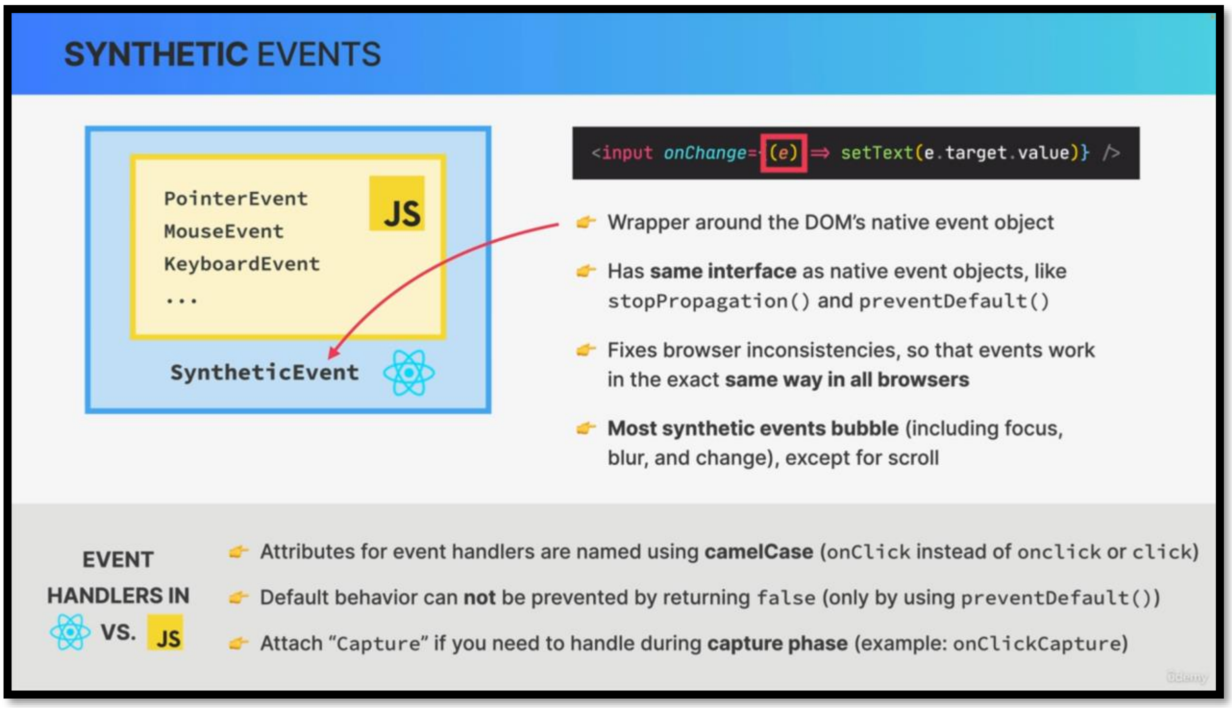
SYNTHETIC EVENTS

```
PointerEvent
MouseEvent
KeyboardEvent
...
```

```
<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

The React team also decided that all of the most important synthetic events actually bubble, including the focus, blur, and change events which usually do not bubble. The only exception here is the scroll event which does also not bubble in React.



SYNTHETIC EVENTS

PointerEvent
MouseEvent
KeyboardEvent
...

JS

SyntheticEvent

```
<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`)