Eventful issues

You're now aware that React can work with most of the same events found in HTML, although React handles them differently.

This means that you may encounter unfamiliar errors when you run your event-driven React code. However, in this reading, you'll learn about some of the most common errors associated with events and how you can deal with them.

Event Errors

When you work in any programming environment, language, or framework, you are bound to write code that throws errors, for a variety of reasons.

Sometimes it's just about writing the wrong syntax. Other times it's about not thinking of all the possible scenarios and all the possible ways that things can go wrong in your code.

Regardless of what causes them, errors are a part of everyday life for a developer.

The JavaScript language comes with a built-in error handling syntax, the **try...catch** syntax.

Let's examine an example of an error in JavaScript:

```
(5).toUpperCase()
```

Obviously, you cannot uppercase a number value, and thus, this throws the following error:

```
Uncaught TypeError: 5.toUpperCase is not a function
```

To handle this TypeError, you can update the code with a try...catch block that instructs the code to continue running after the error is encountered:

```
try {
    (5).toUpperCase();
}
catch(e) {
    console.log(`Oops, you can't uppercase a number.
        Trying to do it resulted in the following`, e);
}
```

The try-catch block will output some text in the console:

Oops, you can't uppercase a number. Trying to do it resulted in the following TypeError: 5.toUpperCase is not a function

It is assumed that if you are taking this course that you are already familiar with how the try...catch syntax works, so I won't go into any details after this quick refresher.

Back to React, here's an example of a simple error in a React component:

```
function NumBillboard(props) {
    return (
      <>
        <h1>{prop.num}</h1>
      </>
    )
  }

export default NumBillboard;
```

In React, an error in the code, such as the one above, will result in the error overlay showing in the app in the browser.

In this specific example, the error would be:

> ReferenceError
>
> prop is not defined

Note: *You can click the X button to close the error overlay.*

Since event-handling errors occur after the UI has already been rendered, all you have to do is use the error-handling mechanism that already exists in JavaScript – that is, you just use the **`try...catch`** blocks.