Method 1: static getDerivedStateFromError

Let's define the getDerivedStateFromError method for the ErrorBoundary component in this lesson.

WE'LL COVER THE FOLLOWING

- Updating the State

Updating the State

Whenever an error is thrown in a descendant component, this method is called first, and the error thrown is passed as an argument.

Whatever value is returned from this method is used to update the state of the component.

Let's update the `ErrorBoundary` component to use this lifecycle method.

```
import React, { Component } from "react";
class ErrorBoundary extends Component {
  state = {};

  static getDerivedStateFromError(error) {
    console.log(`Error log from getDerivedStateFromError: ${error}`);
    return { hasError: true };
  }

  render() {
    return null;
  }
}

export default ErrorBoundary;
```

Right now, whenever an error is thrown in a descendant component, the error will be logged to the console through `console.error(error`), and an object is returned from the `getDerivedStateFromError` method. This will be used to update the state of the `ErrorBoundary` component with value `hasError: true`.

> The given code will not display anything as `render` method returns `null` but the `getDerivedStateFromError` component is added in the `ErrorBoundary` component.

Let’s merge this in our React app:

```
import React, { Component } from "react";
class ErrorBoundary extends Component {
  state = {};

  static getDerivedStateFromError(error) {
    console.log(`Error log from getDerivedStateFromError: ${error}`);
    return { hasError: true };
  }

  render() {
    return null;
  }
}

export default ErrorBoundary;
```

In the next lesson, we’ll update our app with a new method called `componentDidCatch`.