No Named exports

In this lesson, we'll learn the alternative way for using named exports with React.lazy.

WE'LL COVER THE FOLLOWING

If you remember from the previous lesson, I mentioned that `React.lazy` expects the dynamic import statement to include a module with a **default export** being a React component.

A Slight Limitation

At the moment, `React.lazy` doesn't support named exports. That's not entirely a bad thing, as it keeps tree shaking working to ensure you don't import unused modules.

Consider the following module:

```
// MyAwesomeComponents.js
export const AwesomeA = () => <div> I am awesome </div>
export const AwesomeB = () => <div> I am awesome </div>
export const AwesomeC = () => <div> I am awesome </div>
```

Now, if you attempt to use `React.lazy` with a dynamic import of the module above, you'll get an error.

```
// SomeWhereElse.js
const Awesome = React.lazy(() => import('./MyAwesomeComponents'))
```

That won't work since there's no default export in the `MyAwesomeComponents.js` module.

A Possible Solution

A workaround would be to create some other module that exports one of the components as a default.

For example, if I was interested in lazy loading the `AwesomeA` component from the `MyAwesomeComponents.js` module, I could create a new module like this:

```
// AwesomeA.js
export { AwesomeA as default } from './MyAwesomeComponents'
```

Then I can go ahead effectively use `React.lazy` as follows:

```
// SomeWhereElse.js
const AwesomeA = React.lazy(() => import('AwesomeA'))
```

Problem solved!

In the next lesson, we'll learn how to split routes to transfer a large chunk of code.