Specifying a prop as optional

Sometimes we want a prop to be optional. In this lesson, we'll learn how to do this for React components that have their props strongly-typed with TypeScript.

WE'LL COVER THE FOLLOWING

It is possible to do this in TypeScript by putting a `?` before the type annotation.

In our CodeSandbox project that we were working on in the last lesson, add an optional `message` prop to the `Hello` component. Render the value of the `message` prop on the line after the hello message in a `p` tag. Also, remove the `FC` type and use a type annotation on the `props` parameter instead.

Show Answer

What is the value of the `message` prop when it isn't passed by a consuming component?

Show Answer

Notice in the above implementation, the short circuit operator (`&&`) is used to render the message only if it has been passed into the component. This is a typical pattern when rendering elements driven by optional props.

Notice also that we use React's `Fragment` component around the two `p` tags because the function component expects us to output a single top-level element. `Fragment` allows a single top-level element without it being output to the DOM.

the DOM.

Wrap up

Excellent! We can now create strongly-typed props for function components and specify some props as optional.

Not requiring all the props to be passed can make components easier to consume. Often, we will want to provide a default value for an optional prop. We will learn how to do this in the next lesson. We will use the CodeSandbox project we have been working on, so keep it safe.