Specifying a type for an object prop

React component props aren't always based on primitive types. How can we specify types for props that are objects? We'll find out in this lesson.

WE'LL COVER THE FOLLOWING

Specifying an object prop

The props we have implemented so far have been primitive types. What about a prop that has a complex object as its type? Well, a prop type can be an interface or type alias to define a complex type.

In the CodeSandbox project, we were working on in the last lesson, let's change the `who` prop to be an object that has a `name` property and `friend` `boolean` property. The `Hello` component will be as follows:

```
const Hello = ({ who, message = "How are you?" }: Props) => (
  <React.Fragment>
    <p>
      Hello, {who.name}
      {who.friend && " my friend"}
    </p>
    {message && <p>{message}</p>}
  </React.Fragment>
);
```

So, the `who` prop is now an object containing `name` and `friend` properties. What should the definition of the `Props` type be?

Show Answer

Specifying an object prop in an object prop

Let's implement another property in the `who` property called `address`. This is an object containing `line1`, `line2`, `state`, and `zipcode` properties. Let's make this `address` property optional.

Show Answer

So, objects that are graphs many levels deep can still be represented with types in TypeScript. Neat!

Wrap up

Well done! We can now create strongly-typed props that have a deep object graph using type aliases or interfaces.

In the next lesson, we'll learn how to type a React component prop that is a function with TypeScript. We'll continue to use the CodeSandbox project we have been working on, so don't lose it.