Creating strongly-typed refs in function components

In this lesson, we learn how to get a strongly-typed reference to an element in a function component.

WE'LL COVER THE FOLLOWING

Understanding the `useRef` hook

The `useRef` can be used to access all the properties and methods of an element.

```
const element = React.useRef(null);
// can access all the properties and methods of `element` via `element.cur
rent`

...
return (
  <SomeComponent ref={element} />
);
```

It is commonly used when we need to invoke methods on an element imperatively.

Strongly-typing the element with `useRef`

Below is an example of using the `useRef` hook:

```
const Search: React.FC = () => {
  const input = React.useRef(null);
  React.useEffect(() => {
    if (input.current) {
      input.current.focus();
```

```
    }
  }, []);
  return (
    <form>
      <input ref={input} type="type" />
    </form>
  );
};
```

We are setting focus on an `input` when the component first renders.

What do you think the type of `input.current` is inferred as?

Show Answer

We can explicitly define the type of the element returned from `useRef` by passing a generic type parameter:

```
const element = React.useRef<ElementType>(null);
```

If we turn our attention back to the `Search` component from earlier in this lesson, what type should we define `input` as?

Show Answer

A revised, more strongly-typed version of the `Search` component is below. If you run it, you will see that the focus is set on the input after it renders.

```
import * as React from "react";
import * as ReactDOM from "react-dom";

const Search: React.FC = () => {
  const input = React.useRef<HTMLInputElement>(null);
  React.useEffect(() => {
    if (input.current) {
      input.current.focus();
    }
  }, []);
  return (
    <form>
      <input ref={input} type="type" />
    </form>
  );
```

```
};

 ReactDOM.render(
    <Search />,
    document.getElementById("root")
);
```

Wrap up

We can get a strongly typed reference to a rendered element in a function component by passing in the element type in the generic parameter in the `useRef` hook. We'll discover how to get a strongly typed reference to an element in a class component in the next lesson.