Quiz

This quiz will test what you have learned about strongly-typing class components.

Creating strongly-typed class components

1

We have a `CountDown` component that has no props but has a `count` state. How can we declare the `CountDown` component with strongly-typed `count` state and initialize it to `10`?

2

We have a `List` component that renders an array of strings in an unordered HTML list. Below is an example consumption of the component:

```
<List data={["fred", "bob"]} />
```

How can we define the type for the `data` prop in the `List` component?

3

We have a `List` component that renders an array of strings in an unordered HTML list like in the last question. This component allows the consumer to optionally render the list items though.

Below are example consumptions of the component:

```
<List data={["fred", "bob"]} />

<List data={["fred", "bob"]} renderItem={item => <i>{item}</i
>} />
```

How can we define the type for the `data` and `renderItem` props in the `List` component?

4

We have a `Loading` component that renders a loading spinner and an

optional text message.

Below are example consumptions of the component:

```
<Loading />
<Loading message="Please wait ..." />
```

How can we define the type for the `message` prop and default it to `"Loading ..."` if it isn't passed into the component?

5

We have a `CountDown` component that has an internal `decrementCount`

method:

```
class CountDown extends React.Component<...> {
  ...
  decrementCount = () => {
    ...
  };
}
```

Is the `decrementCount` available to consumers in the type system?

i.e. would the following raise a type error?

```
const counter = new CountDown(...);
counter.decrementCount();
```

Check Answers

In the next chapter, we will learn how to strongly type component events and handlers that listen to them.