Questions

Answering these questions aren't required, but correct answers will help.

Part 1

When setting the state within the toggleItem function, I used this code to modify the Array:

```
this.setState(({ items }) => ({
 items: items.map((item, i) => (index === i ? { ...item, complete: !item.complete } : item)),
}));
```

Questions

Why would I have passed a function to the setState call, rather then passing an Object?

By passing a function, you are guaranteed to receive the previous state. And in this case, we only use 'items' (via destructuring assignment) from the state object.

We can also pass a function to a 'useState' hook in the function component. It makes sure the initial state is only set once which only happens in the initial rendering of the component. We normally do this when the initial state is a result from a 'thick' computing so that the same computation won't be run again, say, in a re-render caused by a side effect as long as the state is not set to something else.

What is the reason for me mapping over the entire array rather than modifying the array directly IE. items[index].complete = !item[index].complete ? And is there any possible reasons for using the object spread syntax within the loop?

The only way to change an internal state is via setState (or useState when using the hook). Both setState or useState are expected to receive a new value which will then replace the 'previous state' in the next rerender. In this case, we should always return a new object instead of changing the existing object (which works the same for redux states). The spread syntax here also makes sure a new object is always created.

Changing the value of an internal state or a redux state directly will never trigger a rerender. The only way is to create a new value and then set the new state via setState, useState or via reducers accordingly.

Part 2

Currently, we use the array index as the way to know what to-do item we are toggling, as well as using them for the key prop when rendering the list.

Questions

Generally, what drawbacks—if any—does this have and how would you solve them?
The drawback here is the item key is not consistent (can be changed when the list is changed since it is just an array index). Even most of the time, we rarely use the key to access a single item in the list, it's always better to use a consistent and unique value as the key (e.g. the id of the item).