React State Management

CS5610: Web Development

Joydeep Mitra

Pre-class Activity

- Follow the instructions in the README to setup the repo

  https://github.com/CSE-316-Software-Development/learn-react/tree/main

State in React Components

- What do we mean by state in a component and why do we need it?
  - Let’s look at a motivating example in “src/pages/components/whyState.js” .
  - The page has a next button, which when clicked should replace the current image with another image. However, this does not happen!

State in React Components

- What is state?
  - **state** is any data that is persistent across multiple renders of the component
  - each time the **state** is updated with new data, the component is re-rendered such that the new data is reflected in the UI.
- Let's look at the component in "src/pages/components/state.js" and consider the following:
  - How is state defined?
  - How is state updated?
  - The use of **React hooks** (*useState*).

State in React Components

- State can be defined to have more than one property.
- In “src/pages/qcomps/state.js” we have defined state with the properties
  - *index* : controls which sculpture to show.
  - *showMore* : controls whether to show more information about the sculpture.

For You to Do

- The component defined in “src/pages/qcomps/state.js” has a bug. Identify it and fix it.

State in React Components

- The value of a state variable can be objects.
- This is especially encouraged if the state variables are related to each other and need to be updated together.
- As an example, consider “src/pages/components/stateObj.js”.
  - Note how name and age are updated together and hence are part of one object.

For You to Do

- When the Form component is rendered in “src/pages/qcomps/stuckForm.js”, it appears to be frozen. When a user enters stuff into the <input> tags or clicks on the reset button nothing happens. Identify and fix the error.
- “src/pages/qcomps/thankYouCrash.js” is an example of a form that takes user feedback and displays the message “Thank you!” after the user has submitted the form. Detect the bug in this code and fix it.

Rendering Components

- Calling the *set()* method triggers React to take a snapshot of the component's current state, update the state, and re-render the component with the updated state.
- A component is not re-rendered until the current render is complete.
- React prevents change to state variables in the current render.
- Consider the component "src/pages/components/snapshot.js".
  - If we press the button named "+3" once, the counter increments by 1. Why?
- A more realistic example of the benefits of snapshotting is in "src/pages/components/snapshot2.js"

Queueing Updates

- Sometimes we may have to make several updates to the same property in a state.
  - We use an arrow function to queue several updates to the same property.
  - The arrow function returns the new state, which can be used by the next updater function.
- The component in “src/pages/components/queueUpdates.js” demonstrates this concept.

For You to Do

- Suppose you are working on implementing a shopping cart. When a user clicks on “buy”, the pending counter is incremented by 1. After three seconds, the pending counter becomes 0 and the completed counter is incremented by 1. For some reason, the counters work weirdly. Fix the bug in the code in “src/pages/qcomps/shoppingCart.js”.

Updating State Objects

- When updating a state object, React recommends that we create a new object and replace the old object with the new one. Why?
  - Because we want to re-render the component after updating state.
- We can use *spread syntax* for this purpose.
- Consider the example in “src/pages/components/updObjects.js” .

For You to Do

The form in “src/pages/qcomps/updObjectsForm.js” has a few bugs. Click the button that increases the like score a few times. Notice that it does not increase. Then edit the first name and notice that the score has suddenly “caught up” with your changes. Finally, edit the last name, and notice an error in console. Identify and fix the bugs.

Arrays in React State

- We should treat state as a read-only property, which can only be changed through setter methods. Why?
  - Because we want to re-render the component after a state update.
- What if we define state to have arrays? arrays are mutable objects.
  - We must avoid operations that mutate the array.

Arrays in React State

	avoid (mutates the array)	prefer (returns a new array)
adding	push, unshift	concat, [...arr] spread syntax
removing	pop, shift, splice	filter, slice
replacing	splice, arr[i] = ... assignment	map
sorting	reverse, sort	copy the array first

Let’s consider the React component, ”src/pages/components/artistsArr.js” as example of how to update Arrays in state.

For You to Do

- The React component, ”src/pages/qcomps/artistsRemoveArr.js”, renders a list of artists with a delete button for each artist. When the delete button is pressed, the item should be deleted. However, this does not happen. Identify and fix the bugs.

Updating Array of objects

- Recall when we copy an array, we are doing a shallow copy.
- So what?
  - The shallow copy and the original references are the same!
  - Updating one will update the other.
- As an example, consider “src/pages/qcomps/arrObj.js”.
  - There are two lists with the same items.
  - We want their checkboxes to be isolated from each other.
  - Why does this not happen?
    - because of shallow copy.
  - Note all the places where we are doing a shallow copy!

For You to Do

- Can you change the code “src/pages/qcomps/arrObj.js” such that the lists are decoupled, that is, checking a box in one list should have no impact on the other list.

Additional Reading

- Recommended for Final Project:
  - Reducers and Context for shared state management.
  - React Router for easy conditional routing.