Section 4 Quiz - Working with Events & State

1. How should you NOT listen to events when working with React?
    a. **Adding an event listener (e.g. via "addEventListener") manually.**
    b. Setting an event handler function via props (e.g. onClick={...})
    c. You can't listen to events, React is about outputting data only.

2. Which value should you pass to event listener props like onClick?
    a. The code that should execute when that event occurs.
    b. The result of calling a function that should execute when the event occurs.
    c. **A pointer at the function that should execute when the event occurs.**

3. How can you communicate from one of your components to a parent (i.e. higher level) component?
    a. **You can accept a function via props and call it from inside the lower-level (child) component to then trigger some action in the parent component (which passed the function).**
    b. You can accept an event via props and trigger it from inside the lower-level (child) component to then trigger some action in the parent component (which passed the function).
    c. You can't communicate up, only down - i.e. you can only pass props down to pass data down to a component. You can't trigger an action in a higher-level component.

4. How can you change what a component displays on the screen?
    a. Use a regular JavaScript variable, change the value and output the variable's value in JSX.
    b. You can't change the output - components are static in React apps.
    c. **Create some "state" value (via useState) which you can then change and output in JSX.**

5. Why do you need this extra "state" concept instead of regular JS variables which you change and use?
    a. Because it's less code
    b. **Because standard JS variables don't cause React components to be re-evaluated**
    c. Because standard JS variables are not supported in React components

6. Which statement about useState is NOT correct?
    a. It receives an (optional) initial state value as an argument
    b. **Calling useState again will update the state value**
    c. It returns an array with exactly two elements

7. How can you update component state (created via useState)?
    a. You can assign a new value to the state variable.
    b. **You can call the state updating function which useState also returned.**

c. You can call useState again.

8. How much state may you manage in one single component?
    a. **You can have as many state slices as you need / want.**
    b. You should at most have one state (merge multiple states into a state object).
    c. You can have multiple state slices if at least one of them is an object.

9. What's wrong about this code snippet? const [counter, setCounter] = useState(1); ... setCounter(counter + 1);
    a. There's nothing wrong about it.
    b. State can't be a number.
    c. **If you update state that depends on the previous state, you should use the "function form" of the state updating function instead.**