Introduction to State in React

Table of Content

Local Variables and React Rendering

1. **Local variables don't persist between renders.** When React renders this component a second time, it renders it from scratch—it doesn't consider any changes to the local variables.
2. **Changes to local variables won't trigger renders.** React doesn't realize it needs to render the component again with the new data.

> Therefore, to update a component with new data, two things need to happen:

1. **Retain** the data between renders.
2. **Trigger** React to render the component with new data (re-rendering).

React useState Hook to the rescue

The `useState` Hook provides these two things:

1. A **state variable** to retain the data between renders.
2. A **state setter function** to update the variable and trigger React to render the component again.

When you call `useState`, you are telling React that you want this component to remember something:

```
const [index, setIndex] = useState(0);
```

In this case, you want React to remember the `index`.

💡 The convention is naming this pair as `const [something, setSomething]`. You could name it anything you like, but conventions make things easier to understand across projects.

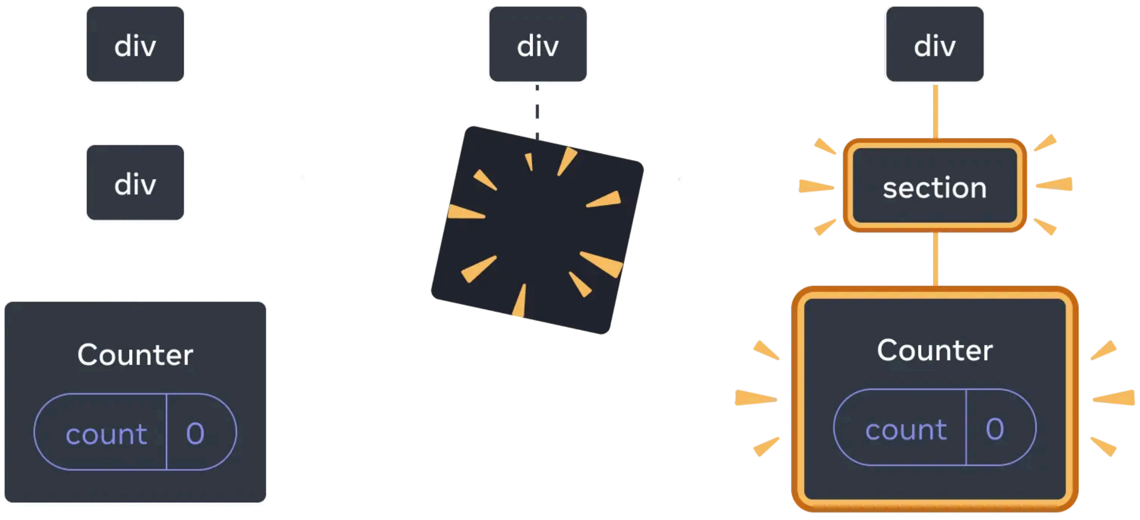
💡 Hooks can only be called at the top level of the component function. We cannot use hook in if-else, for loops or nested functions

State is isolated and private

State is local to a component instance on the screen. In other words, **if you render the same component twice, each copy will have a completely isolated state!** Changing one of them will not affect the other.

State is tied to a position in the render tree

- This means different instances of a component rendered at different positions will have separate states.
- This also means that once a component is not rendering, its state will not be preserved



Assignments

- Follow the following steps:
    - Create an array of Problems Array having question-and-answer
    - Render only one Problem Component in the Parent component. The parent component will also have a Next and Previous Button
    - Clicking on these buttons should show the next and previous Problem respectively. Hint: Use these buttons to update the `index` state variable.
- Now add a `showAnswer` state variable to show or hide the answer field inside the Problem component.