Week 6 + 7

Welcome!

Goals

- [ ] Get an intuition for how state dictates behavior in an app
- [ ] Ponder some examples/use cases
- [ ] Complete this week's challenge!

A Quick Recap...

Last week was awesome!!! Evevryone stepped up and took a leap of faith by styling a component all on your own... Kudos (□□) to you!

Big shoutout to Palak for catching the W last week!

Understanding React State

A recurring question that has been asked for the last few weeks (and rightfully so) is how the form submission works. In other words, if someone is actually using the app on the web, how is their response to the form reflected in our project.

Currently, as you may have noticed in your code, we are hardcoding cards, which obviously isn't how an actual app works.

We want our app to be able to recieve user inputs (kudos) in real-time and update our card layout accordingly.

And you guessed it, this is state!

What is State?

Remember our house analogy?

- HTML is the walls (structure)
- CSS is the paint (styling)
- TypeScript is the door (logic)

State is the furniture inside the house - it can change and move around dynamically!

State is data that can change over time in your React component. When state changes, React automatically re-renders the component to show the updated information.

In effect, this is what gives **functionality** and actual usefulness to your code.

Some Real-World Examples...

There are a few ways to think about state in everyday terms:

A light switch - It has two states: ON or OFF

- Current state: The light is currently ON
- *Action: You flip the switch*
- New state: The light is now OFF

A shopping cart - It has a list of items

- Current state: 3 items in cart
- *Action: You add a new item*
- New state: 4 items in cart

Our Kudos App - It will have a list of kudos cards

- Current state: 1 kudos card displayed
- *Action: User submits the form*
- New state: 2 kudos cards displayed

This is exactly what we're building today!

Hooks

Hooks are functions that let you "hook into" and control React states.

Since we can use hooks for each individual (functional) component, it is actually possible to write entire React applications using only functional components, which in turn HIGHLY simplifies the component model that we discussed last week.

Before we dive into how they are used in our project, let's go over some properties of hooks that will help you understand them better

General Properties

- They can only be used with functional components. You CANNOT use them in class components.
- Every time our function runs/is called, its hooks must run in the exact same order. In other words, we cant really have a hook inside of a conditional statement, since if the statement didn't run, we would have an error. This follows that we really can't have hooks nested inside of anything (loops, conditionals, functions, etc.). They must be at the top level of our function, and always called in the exact same order.
- Lastly, there are a few important types of hooks that React allows us to use. In the next section, I will go over one of these types.

The useState Hook

React gives us a special function called `useState` to manage state. The pattern looks like this:

```
const [currentValue, functionToUpdateIt] = useState(initialValue);
```

Think of it like this:

- **currentValue** - What the state is right now
- **functionToUpdateIt** - How to change it
- **initialValue** - What it starts as

Don't worry if this seems abstract - it'll make perfect sense once we start building!

Your Objective

Last week, you successfully built the static UI for our Kudos Board!

There's just one problem... they don't talk to each other. □□

These components aren't connected yet; **they don't share any data.**

When you fill out the form and hit "Send Kudos," nothing happens. The form data just disappears, and no new card appears.

Your goal this week is to fix this. You need to use React's `useState` hook to capture the form submissions and display them as new cards.

Task Outline

Make the app fully functional. When a user fills out the `KudosForm` and clicks "Send Kudos," a new `KudosCard` should appear in the `cards-grid`.

To solve this, you will need to:

1. **Find the Right Home for State:** Where does the list of all kudos need to live so that *both* the form (to add kudos) and the card display (to read kudos) can access it?
2. **Initialize the State:** In the correct component (if you cant figure which, let us know), use the `useState` hook to create a state variable that holds an *array* of `Kudos` objects. You can start it with an empty array `[]` or use the example kudos from the lesson plan.
3. **Create an "Add" Function:** In the *same component* where you created your state, write a function (like `handleAddKudos`) that takes a new `Kudos` object as an argument and updates the state by adding this new object to the array.
4. **Pass the Function:** Pass your new "add" function down to the `<KudosForm />` as a prop (e.g., `onSubmit`).
5. **Render the List:** Use your state array to render the list of cards. Instead of hard-coding a `<KudosCard />`, you'll need to *map* over your state array and return a `<KudosCard />` for each item.

Bonus (If ur trying to flex)

Once it's working, try implementing a "Empty State" message. Your app should show a friendly message like "No kudos yet. Be the first!" *only* when the `kudosList` array is empty.

Resources Some Hints...

This is a great resource to get you started. There is a also a really good example implementation:

Managing State

- Don't forget to import `useState` from `'react'` (this is likely why your code might be not working).
- Remember the `Kudos` interface from Week 4? It will help TypeScript know what your state looks like (`useState<Kudos[]>(...)`).
- **IMPORTANT:** You must **never** modify state directly (like using `.push()`). You must always create a *new* array using the spread operator (`...`) and pass that to your `setKudosList` function.
- When you `.map()` over your array to create cards, React will need a unique `key` prop for each `<KudosCard />` (if unsure/unfamiliar with what a key is or what this means, ask us!).

As always, Derrick and I are here for you!!! Just ask if you get stuck.

Revision #10
Created 28 October 2025 00:53:06 by Emilio Cardillo Schrader
Updated 19 November 2025 19:14:29 by Emilio Cardillo Schrader