18. States, Props, and Events in React

Table of Content

1. React states

The state is a built-in React object that contains data or information about the component.

A component's state can change over time; whenever it changes, the component re-renders. The change in state can happen as a response to user action or system-generated events and these changes determine the component's behavior and how it will render.

The useState hook

The React useState Hook allows us to track state in a function component.

The `useState` Hook can be used to keep track of strings, numbers, booleans, arrays, objects, and any combination of these!

We could create multiple state Hooks to track individual values.

- **initialise state**

  We initialize our state by calling `useState` in our function component.

  `useState` accepts an initial state and returns two values:

- The current state.
- A function that updates the state.

```
import { useState } from "react";

function FavoriteColor() {
  const [color, setColor] = useState("");

  return <h1>My favorite color is {color}!</h1>
}
```

The [and] syntax here is called array destructuring and it lets you read values from an array. The array returned by useState always has exactly two items.

▼ update state

To update our state, we use our state updater function.

We should never directly update state. Ex: `color = "red"` is not allowed.

```
<button
  onClick={() => setColor("blue")}>
  Blue
</button>To update the state based on the previous state, use this format instead:
```

▼ useState with Objects and Arrays

Instead of typing this:

```
  const [brand, setBrand] = useState("Ford");
  const [model, setModel] = useState("Mustang");
  const [year, setYear] = useState("1964");
  const [color, setColor] = useState("red");
```

You can use an object that includes the all 4 properties, like this:

```
const [car, setCar] = useState({
    brand: "Ford",
    model: "Mustang",
    year: "1964",
```

```
    color: "red"
  });
```

But,

If we only called `setCar({color: "blue"})`, this would remove the brand, model, and year from our state.

We can use the JavaScript spread operator to help us.

```
setCar(previousState => {
      return { ...previousState, color: "blue" }
    });
```

Similar operations should be performed with the arrays.

2. Passing data in between the states using props

1. Define a callback in the parent which takes the data you need in as a parameter.
2. Pass that callback as a prop to the child.
3. Call the callback using props.[callback] in the child (insert your own name where it says [callback] of course), and pass in the data as the argument.

3. Event Handling

React lets you add *event handlers* to your JSX. Event handlers are your own functions that will be triggered in response to user interactions like clicking, hovering, focusing on form inputs, and so on.

Built-in components like `<button>` only support built-in browser events like `onClick`. However, you can also create your own components, and give their event handler props any application-specific names that you like.

```
function Toolbar({ onPlayMovie, onUploadImage }) {
  return (
    <div>
```

```
      <Button onClick={onPlayMovie}>
        Play Movie
      </Button>
      <Button onClick={onUploadImage}>
        Upload Image
      </Button>
    </div>
  );
}
```

and then inside the parent:

```
export default function App() {
  return (
    <Toolbar
      onPlayMovie={() => alert('Playing!')}
      onUploadImage={() => alert('Uploading!')}
    />
  );
}
```

4. Lists in React

In React, you will render lists with some type of loop. The JavaScript `map()` array method is generally the preferred method.

```
const cars = ['Ford', 'BMW', 'Audi'];
<ul>
    {cars.map((car) => <Car brand={car} />)}
</ul>
```

💡 **Logical && Operator for conditional rendering**

```
{cars.length > 0 &&
  <h2>
    You have {cars.length} cars in your garage.
  </h2>
}
```

If cars.length > 0 is equates to true, the expression after && will render.

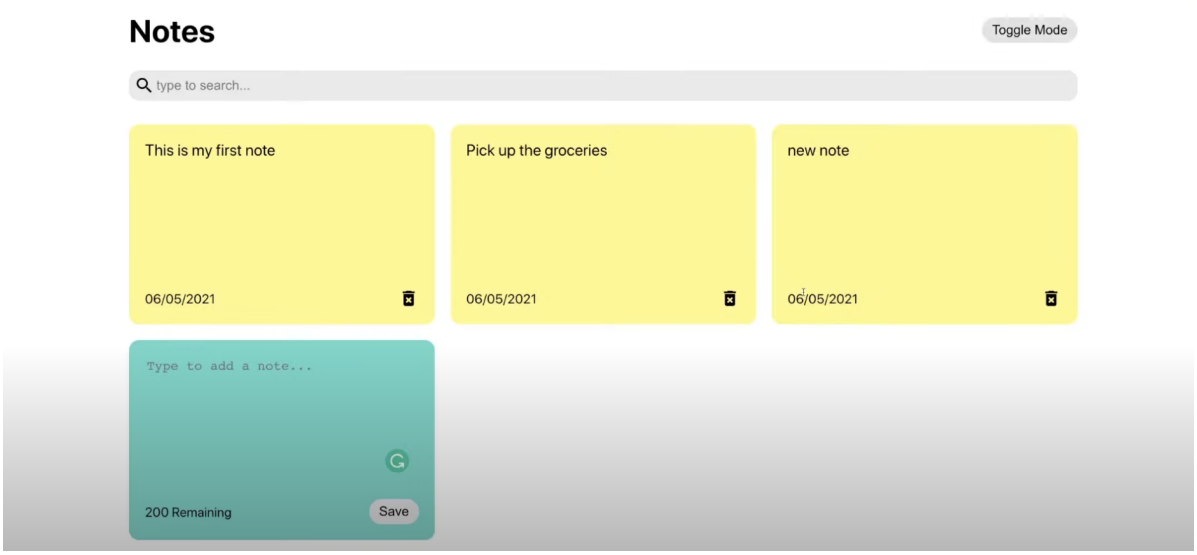
You should pass the key attribute which should be unique to every list element.

Assignments

1. Create a Unit Converter Application

   https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c8bc19a6-02c0-458c-a63d-93e11c753497/Screen_Recording_2022-12-07_at_8.47.10_PM.mov

2. Create a Notes Taking App with the following UI. Here you use the green card to create new Notes. Add the functionality to filter the Notes by typing to search.


Notes
Toggle Mode
type to search...
This is my first note
06/05/2021
Pick up the groceries
06/05/2021
new note
06/05/2021
Type to add a note...
200 Remaining
Save