Initial State and Reducer

In this lesson, we will define the initialState and reducer for a simple Todo app with useReducer.

WE'LL COVER THE FOLLOWING

Initial state

We are developing a ToDo app. Our state has a list of ToDo items. Let's put three items initially so that we see them when the app starts.

```
const initialState = {
  todos: [
    { id: 1, title: 'Wash dishes' },
    { id: 2, title: 'Study JS' },
    { id: 3, title: 'Buy ticket' },
  ],
  query: '',
};
```

The IDs are almost always recommended when we have a list in a state. We use them to specify the `key` prop in React.

Right now, an item has only `title` in addition to `id`. We will be using another property, `completed`, that is undefined initially.

There is also `query` which we will use to highlight ToDo titles.

Now, let's define a reducer function to manipulate this state immutably.

Reducer

Reducer

A **reducer** is a function to update a state with an action. An action is an object, typically with a `type` property. A reducer returns a new state without mutating any objects in it.

As we need to keep track of `nextId`, we define it in advance as an easy solution.

```
let nextId = 4;
```

The solution is `4` because we have three items initially.

We then define our reducer.

```
const reducer = (state, action) => {
  switch (action.type) {
    case 'ADD_TODO':
      return {
        ...state,
        todos: [...state.todos, { id: nextId++, title: action.title }],
      };
    case 'DELETE_TODO':
      return {
        ...state,
        todos: state.todos.filter(todo => todo.id !== action.id),
      };
    case 'TOGGLE_TODO':
      return {
        ...state,
        todos: state.todos.map(todo =>
          todo.id === action.id
            ? { ...todo, completed: !todo.completed }
            : todo,
        ),
      };
    case 'SET_QUERY':
      return {
        ...state,
        query: action.query,
      };
    default:
      return state;
  }
```

```
};
```

It is a bit long, but there is a pattern. For each action type, it creates a new state. Let's summarize what each action type means.

- `ADD_TODO` creates a new item and appends it to the end of the list. It requires the title.
- `DELETE_TODO` removes an existing item from the list. It requires the `id`.
- `TOGGLE_TODO` changes the `completed` flag of the item. It requires the `id`.
- `SET_QUERY` sets `query` to highlight ToDo titles.

Store file

We create a file named `store.js` and export some hooks and a component.

Using `initialState` and `reducer`, the file is as follows:

```
// store.js

import { useReducer } from 'react';
import { createContainer } from 'react-tracked';

const initialState = ...;

let nextId = 4;

const reducer = ...;

const useValue = () => useReducer(reducer, initialState);

export const {
  Provider,
  useTrackedState,
  useUpdate: useDispatch,
} = createContainer(useValue);
```

That's all we need to define our global state.

Next

In the next lesson, we will create a TodoList component.