Congratulations! In this lesson you’ve built a strong conceptual foundation of Redux and built a state object, some actions, and a reducer along the way. Here’s what else you learned:

* Redux is a library for managing and updating application state based on the Flux architecture
* Redux makes code more predictable, testable, and maintainable by consolidating state in a single object. Components are just given data to render and can request changes using events called actions.
* In a Redux application, *data flows in one direction*: from state to view to action back to state and so on.
* *State* is the current information behind a web application.
* An *action* is an object describing an event in the application. It must have a type property and it typically has a payload property as well.
* A *reducer* is a function that determines the application’s next state given a current state and a specific action. It returns a default initial state if none is provided and returns the current state if the action is not recognized
* A reducer must make follow these three rules:
  1. They should only calculate the new state value based on the existing state and action.
  2. They are not allowed to modify the existing state. Instead, they must copy the existing state and make changes to the copied values.
  3. They must not do any asynchronous logic or other “side effects”.
* In other words, a reducer must be a *pure* function and it must update the state *immutably*.
* The *store* is a container for state, it provides a way to dispatch actions, and it calls the reducer when actions are dispatched. Typically there is only one store in a Redux application.