Congratulations! You were able to apply a couple of Redux Toolkit’s essential methods in order to refactor and simplify some existing Redux logic.

* **R**edux **T**ool**k**it (RTK) contains packages and functions that build in suggested best practices, simplifies most Redux tasks, prevents common mistakes, and makes it easier to write Redux applications.
* RTK has a createSlice() function that will help us simplify our Redux reducer logic and actions.
* createSlice() has one parameter, options. In this lesson, we covered three of option‘s properties: name, initialState, and reducers. options has more properties which will be covered in the next lessons.
* A case reducer is a method that can update the state, and will be executed when the corresponding action type is dispatched. This is similar to a case in a switch statement.
* You can write code that “mutates” the state inside the case reducers passed to createSlice(), and Immer will safely and accurately return an immutably updated state.
* createSlice() returns an object with the following properties: name, reducer, actions, and caseReducers.
* We typically use a Redux community code convention called the “ducks” pattern when exporting the action creators and the reducer.
* RTK has a configureStore() function that simplifies the store setup process. configureStore() wraps around the Redux core createStore() function and the combineReducers() function, and handles most of the store setup for us automatically.