Redux - store, actions, reducers

Learning objectives

- Use store, actions, and reducers in Redux.
- Use Redux documentation.

Estimated time: 2h

You already have a general idea of what Redux is - now it's time to get to the details! Redux can be very overwhelming but once you get yourself familiar with all its components and how they can work together, you will be ready to use it in your projects. In this lesson, we will talk more about Redux but we will not try to combine Redux with React yet. First, we will give you a chance to play only with Redux. Once you get familiar with Redux itself, you will learn how to connect it with React.

Why is it important?

As mentioned in the introduction, most modern applications rely heavily on data, so having a reliable system to control the application's global state and manipulate data is a must. Redux quickly became very popular among JS developers because of its simple yet powerful concepts that bring order and predictability to working with large amounts of data in interactive applications. A great set of dedicated tools makes it an excellent choice for React applications. At the same time, Redux is not an easy concept at the first glance. Therefore, we will introduce it to you step by step.

Redux

In the Redux introduction lesson you got a high-level understanding of it. Now, please read this article about the core concepts of Redux.

It's time to watch videos created by Dan Abramov, the co-creator of Redux, explaining the concepts of Redux (no code yet):

- The Single Immutable State Tree
- Describing State Changes with Actions
- Pure and Impure Functions
- The Reducer Function

Reducer and store

In the following set of videos, Dan Abramov shows how to use Redux store and how to create reducers. You will also see a very minimalistic Counter application that uses Redux and plain JavaScript.

- Writing a Counter Reducer with Tests
- Store Methods: getState(), dispatch(), and subscribe()
- Implementing Store from Scratch

Combining reducers

Now let's see a bit more complex example of a ToDo app that will require more than one reducer:

- Redux: Writing a Todo List Reducer (Adding a Todo)
- Redux: Writing a Todo List Reducer (Toggling a Todo)
- Redux: Reducer Composition with Arrays
- Redux: Reducer Composition with Objects

Once you have seen multiple reducers working together, you can check the `combinereducers` function from Redux that will help you to hold your state in one place.

- Redux: Reducer Composition with combineReducers()

Dispatch and actions creators

Right now you should have a basic understanding of reducers and the store. Now it's time to understand `actions`.

- First, let's go back to Store Methods: getState(), dispatch(), and subscribe(). **You have watched this video before but this time - pay special attention to the usage of the `dispatch` function.**
- Then please read about Actions.
- And, as a reminder, check how reducers are handling the actions.
- As a final note - please read about Action Creators. **Please, read only the first part of the article and ignore the part after `Can we take this a step further?`**

Redux recap

In this lesson, we covered all the essential parts of Redux. We did not touch on how to use Redux with React yet and we did it on purpose. As Redux is a complex concept, first you will practice it as a standalone code.

Before you move to more practical activities:

- Please read again about Redux Terms and Concept (you have read it in the Redux intro lesson) and check if it is more clear now!
- Check this Redux cheatsheet.

If you spot any bugs or issues in this activity, you can open an issue with your proposed change.