Basic React+Redux App with Async call to API .

I think that in Redux one of the most challenging for a beginner task is Async call to external API. When I first started to learn Redux somehow I couldn’t find any simple example of App with API call. The [async actions tutorial](http://redux.js.org/docs/advanced/AsyncActions.html) given in the documentation is no doubt a scalable approach to handling these kind of situations, but it’s not very simple, and back then it was really over the top for me.

So, I decided to build real life React-Redux application with Async call to API.I tried

to keep application and code as simple as possible to make it’s easy for understanding.

In my App I call news API from <https://newsapi.org> . You can easily get your own API key there in case you need one.

My application description:

Application shows top 10 news from chosen channel.

-There are six buttons, each button represents one channel

-Press one channel-button to choose channel.

-Press button “Get top news” to see top news from chosen channel.

-You can choose another channel and get top news from it.

I use [Redux Thunk middleware](https://github.com/gaearon/redux-thunk) in my App to make. Main goal of my article is to show how to

deal with asynchronous calls in Redux so I will not get into a lot of details about everything else, just general explanation.

From my experience of learning React+Redux I would strongly advice to keep file structure well organized and not to keep everything in one file.

Let's have a quick look at the file structure of my example. Here I use quite popular pattern of dividing all components into two categories: Container and Presentational components. So, we have containers folder and components folder accordingly. In a few words Container components know about Redux while Presentational components don’t know nothing about Redux.

src/

components/

App.js

ChannelsField.js

NewsItem.js

containers/

Channel.js

Button.js

TopNews.js

actions/

index.js

reducers/

index.js

css/

styles.css

index.html

index.js

Now a few sentences explaining why to make Async call in Redux is not a straightforward task. When we call an asynchronous API, there are three crucial states in time: the state before we start the call, the state between the moment a call start and the moment an answer received, and the state after we receive the answer.

Each of these three states usually will be obtained by dispatching normal actions that will be processed by reducers synchronously. Usually, for any API request you'll want to dispatch at least two different kinds of actions:

**An action informing the reducers that the request began.**

The reducers may handle this action by toggling an Loading flag in the state. This way the UI knows it's time to show a spinner or on in my App I just show “Loading…”.

**An action informing the reducers that the request finished successfully.**

The reducers may handle this action by merging the new data into the state they manage and resetting Loading. The UI would hide the spinner, and display the fetched data.

The third action could be **An action informing the reducers that the request failed,**

but because I prefer to keep things as easy as possible let focus on two actions only.

Without [middleware](https://redux.js.org/advanced/middleware), Redux store only supports [synchronous data flow](https://redux.js.org/basics/data-flow). So, without any middleware, our action creator function must return plain object , but with Thunk Middleware we can write action creator functions that returns functions.  **If Redux Thunk middleware is enabled, any time you attempt to dispatch a function instead of an action object, the middleware will call that function with dispatch method itself as the first argument**.

Lets have a look at the index.js file where we are going to include the Redux Thunk middleware in the dispatch mechanism.