**Calling Async Function in Action with Redux-Thunk**

[Redux-thunk](https://github.com/reduxjs/redux-thunk) is a simple middleware that enables you to call functions in redux action.

Thunk means function returning function. In JavaScript programming, we use thunk all the time although we may not use the name. [Wikipedia](https://en.wikipedia.org/wiki/Thunk) has a simple example of thunk if you are interested. As the name suggests, redux-thunk is a thunk used as a middleware to inject the logic into action dispatching.

The use case is very simple. Use it when you want to call async function before dispatching the action. See the example below to retrieve the account information from a remote endpoint then update the store. Wouldn’t it be nice if we can do this in action?

<pre class=" toolbar:2 lang:js font-size:12px decode:true">

</pre>

By default, redux dispatch method only takes object. See the source code for the dispatch method below. The function above will cause error. It will tell you that only object can be dispatched and even tell you to use custom middleware for async action.

<pre class=" toolbar:2 lang:js font-size:12px decode:true">

</pre>

Redux-thunk is basically a custom middleware that allows function to be passed to dispatcher and get rid of the error that you get from the native dispatch function in redux.

Redux-thunk source code is famous for its brevity. Size does not matter here. It does the magic of bringing async functions into action.

This is the current redux-thunk source code.

<pre class=" toolbar:2 lang:js font-size:12px decode:true">

</pre>

Awesome, now you know what redux-thunk is and when to use it. It’s time to set it up

**Setting up redux-thunk**

Setting up is as easy as the module itself. All you need to do is to install the redux-thunk module and apply it as a middleware in the index file.

**Installation**

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

npm i –save redux-thunk

[/cc]

**Index.js**

<pre class=" toolbar:2 lang:js font-size:12px decode:true">

</pre>

Let’s start writing action with async functions. Here are some examples. The argument can be just dispatch or dispatch + getState or dispatch + getState + your custom arguments. This gives a lot of flexibility and can add cool logic to your action.

**Action.js**

<pre class=" toolbar:2 lang:js font-size:12px decode:true">

</pre>