Redux Thunk

Download Demo Code

Goals

- Understand what ordinary actions & action creators can/cannot do
- Introduce "thunk" middleware as method to have more powerful actions
- Dispatch actions asynchronously

Actions

What's an action?

An object with key of ***type***

```
{
  type: "LOAD_TODOS",
  todos
}
```

Action Creators

What's an action creator?

A function that returns an action

actionCreators.js

```
function gotTodos(todos) {
  return {
    type: "LOAD_TODOS",
    todos
  };
}
```

Can an action creator return multiple actions?

actionCreators.js

```
function gotTodos(todos) {
  return {
    type: "LOAD_TODOS",
    todos
  };
}
```

No.

It can only return a single action.

Can an action creator do something asynchronous?

actionCreators.js

```
function gotTodos(todos) {
  return {
    type: "LOAD_TODOS",
    todos
  };
}
```

No — it needs to directly return a single action!

(Redux doesn't ***await*** action creators when it calls them)

Doing Async Stuff With Redux

Loading Todos

Imagine we need to get todos from an API and then get them into Redux

We can't have our action creator handle the async stuff

So we'd have to do this in our component

src/TodoList.js

```
function TodoList({ todos }) {
  const dispatch = useDispatch();

  useEffect(() => {
    async function getTodos() {
      let res = await axios.get("/api/todos");
      // dispatch once we get results from API
      dispatch(gotTodos(res.data.todos));
    }
    getTodos();
  }, [dispatch]);

  // render todos

}
```

This is *okay* ... but:

- It's not great to have this logic in our component
- Would be nicer to extract this and have a smarter action creator...
  - "I can retrieve from API and *then* dispatch the data!"

Redux-Thunk

- We want another type of action — one that is more flexible
  - It can dispatch multiple times
  - It can dispatch asynchronously
- To do this, we'll add *middleware* to Redux — ***redux-thunk***

What's a "thunk"?

A thunk is a function that wraps an expression to delay its evaluation.

```
// calculation of 1 + 2 is immediate (ie, x === 3)

let x = 1 + 2;
```

```
// calculation of 1 + 2 is delayed; foo can be called later
// to perform calculation --- foo is a "thunk"

let foo = () => 1 + 2;
```

Thunk Actions

So, instead of having an action that is a simple object with a ***type*** ...

We can have actions that are "thunks"

- They can be dispatched normally
- But they're functions that can do whatever they want inside
  - Including dispatching multiple times/asynchronously

Setting This Up

```
$ npm install redux-thunk
```

demo/thunk-demo/src/index.js

```
import { createStore, applyMiddleware, compose } from "redux";
import thunk from "redux-thunk";
import rootReducer from "./rootReducer";

const store = createStore(
  rootReducer,
  compose(
    applyMiddleware(thunk),
    window.__REDUX_DEVTOOLS_EXTENSION__
    && window.__REDUX_DEVTOOLS_EXTENSION__()
  )
);
```

Actions

- We'll make a an action creator that returns a "thunk" action
- A "thunk" action is a function *(not an object itself)*
  - That function will make an AJAX request
  - When the request completes, it can itself dispatch!

Putting it together

src/actionCreators.js

```
// "thunk" action creator that we
// dispatch -- does async operation, then
// dispatches actions when it's done.
// the thunk middleware exposes dispatch
// to our inner function!

export function getTodosFromAPI() {
  return async function(dispatch) {
    let res = await axios.get('/api/todos');
    dispatch(gotTodos(res.data.todos));
  };
}

// normal action creator & action

function gotTodos(todos) {
  return { type: "LOAD_TODOS", todos };
}
```

src/TodoList.js

```
function TodoList({ todos }) {
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(getTodosFromAPI())
  }, [dispatch]);

  // render todos

}
```

Extending This

Can even have different dispatches in our thunk:

src/actionCreators.js

```
// "thunk" action creator

export function getTodosFromAPI() {
  return async function(dispatch) {
    dispatch(startLoad());

    try {
      let res = await axios.get(
        '/api/todos');
      dispatch(gotTodos(res.data.todos));
    }

    catch(err) {
      dispatch(showErr(err.response.data));
    }
  }
}
```

src/actionCreators.js (cont'd)

```
// normal action creator & action

function gotTodos(todos) {
  return { type: "LOAD_TODOS", todos };
}

// another normal action creator & action

function showErr(msg) {
  return { type: "SHOW_ERR", msg };
}

// another normal action creator & action

function startLoad() {
  return { type: "SHOW_SPINNER" };
}
```

Review

A "thunk" action is

- a function
  - takes ***dispatch*** as argument
  - calls ***dispatch*** when it wants

a thunk

```
async function(dispatch) {
  dispatch(startLoad());

  try {
    let res = await axios.get('/api/todos');
    dispatch(gotTodos(res.data.todos));
  }

  catch(err) {
    dispatch(showErr(err.response.data));
  }
}
```

A "thunk" action creator is

- a function *(like all action creators!)*
  - it *returns a function* (the thunk!)

action creator that returns a thunk

```
function getTodosFromAPI() {
  return function (dispatch) {
    // this is thunk === can call dispatch
    // whenever it wants
  }
}
```

Alternatives

- *redux-promise*: uses promises to handle async actions
  - Simpler in scope, but limited to single promise return
- *redux-saga*: uses generators to handle async actions
  - Can be more flexible, but is more complex

redux-thunk is the most popular & works great!