Ajax with Redux

Let's say the user clicks a button which gets data from the server and displays it.

When should the dispatch happen?

- On click? Yes, b/c we need to send the request
- But ....

The problem is that it is async

- Dispatching has to run synchronously
- Why? b/c rendering runs immediately after a dispatch
- But the population of data takes a while to return from the server
- So the page section re-renders before the new data arrives. User never sees it.

- If only we could dispatch an action, but delay the reducer until we get a return from the server.
- Hmm. Is there a thing that allows us to run asynchronously and have access to the dispatch function?

Middleware!

- So no matter how you do it, middleware must be involved when you make Ajax calls.
- Here's one method...

1. Create the middleware function that will eventually run

```
const getPersonMiddleware =
{getState, dispatch} => next => action => {
  if (action.type === FETCH_PERSON) {
    fetch(`/api/person/${action.personId}`)
      .then(res => res.json())
      .then(person => dispatch({
        type: SET_PERSON, person
      })
    );
  }
  next(action);
}
```

2. Register the middleware

```
const middleware =
  applyMiddleware(getPersonMiddleware);
const store =
  createStore(
    rootReducer,
    initialState,
    middleware);
```

3. Run the middleware by dispatching an action

```
// Do this on, say, a button click or whatever
this.handleClick = evt =>
 getPerson(personId) {
   store.dispatch({
     type:FETCH_PERSON,
     personId
   });
 }
```

- Note the absence of any response or anything. It is just set it and forget it. We rely on the middleware to dispatch a synchronous SET_PERSON action and the subscription to refresh the UI when an Ajax response is received.

Hands on fetching Ajax data