Redux Combined Reducers

Code 401 (Class 32)

Any “active” reducer can hear a dispatch

“foo” reducer:

```
export const (state=initialState, action) {
    let {type, payload} = action;
    switch(type) {
        case “DO_FOO”:
            return state;
        …
    }
}
```

“bar” reducer

```
export const (state=initialState, action) {
    let {type, payload} = action;
    switch(type) {
        case “DO_FOO”:
            return state;
        …
    }
}
```

When the action of type “DO_FOO” is dispatched, both of these reducers would fire and potentially run some code

But … an app can have only 1?

RULE: An app can have only 1 store … and a store can have only 1 reducer

Redux gives us a “combineReducers” method that we can use to marry these, each managing their own slice of the overall state, which is still a single thing.

state.foo and state.bar are now things …

```
import {combineReducers} from ‘redux’;
import {fooReducer} from ‘./path/to/fooReducer;
import {barReducer} from ‘./path/to/barReducer;

export default combineReducers({
    foo: fooReducer,
    bar: barReducer
});

… elsewhere…

import reducer from ‘./path/to/thisCombinedReducer’
let store = createStore(reducer);
```

Picking state for your component

Now, when you are mapping stateToProps, in your container modules, you reference your module state either as the combined state of the 2 reducers or you can cherry pick the one you want.

e.g. In the “foo” container component, you might only need to operate on the “foo” part of the state.

```
mapStateToProps = state => ({
   foo: state.foo
});
```