Composing Reducers

Reminder: What do these do?

```
const newObj = {...oldObj, prop1:val1, prop2:val2};

const obj2 = { key1: someFunc(someval) };

store.dispatch({type:"notarealtype"});
```

The more complicated state becomes, the more complicated the reducer must become

For instance ...

Simple

- To set data which is immediately inside the state object:

```
case SET_GENDER:
return {...state, gender: action.gender};
```

More complex

- To set data which is inside an embedded object in state:

```
case SET_FIRST_NAME:
return {...state, name: {...state.name, first:
action.first}};
```

Even more complex

- To set data which is 2 levels deep:

```
case SET_COMPANY_CEO_NAME:
return {...state, company: {...state.company,
{ ceo: {...state.company.ceo, name:
action.name } } } } };
```

Solution: Create sub-reducers

- Sub-reducer is a reducer function that handle a slice of state.

- Remember that a reducer is a function that returns a state object so ...

1. We'll split the state into sub-states
2. We'll handle each sub-state with a sub-reducer
3. We'll combine the sub-reducers to form the main reducer

For example:

- Pseudo-code ...

```
rootReducer = (state, action) => return {
  name: nameReducer(state.name, action),
  location: locationReducer(state.location, action),
  picture: pictureReducer(state.picture, action),
  ...
};
```

- Remember, name, location, and picture are just objects.

Two options

1. Use combineReducers
   - Constrained to a square-shaped data
   - Easier for another dev to understand
2. Do it ourselves manually
   - More control and flexibility
   - More code to write

Using Redux's combineReducers function

combineReducers() is a function built-in to Redux

- It's easy to use but limited.
- It assumes you're naming the sub-reducer function the same as its key in the state object.
- For example ...

```
import { createStore, combineReducers } from 'redux';

const name = (state, action) => {
  // Here, "state" is only the object under
  // the "name" key.
}
const location= (state, action) => {
  // Here, "state" is only the object under
  // the "location" key.
}
const picture= (state, action) => {
  // Here, "state" is only the object under
  // the "picture" key.
}

const rootReducer = combineReducers(name, location,
picture);
```

Manually combining reducers

If that is too limiting, we can do the same thing manually

We will do essentially the same thing as combineReducers() except that we have more control and flexibility.

Combining manually

```
const rootReducer = (state, action) => {
  return {
    ...personReducer(state, action),
    name: nameReducer(state.name, action),
    location: locationReducer(state.location, action),
    picture: pictureReducer(state.picture, action)
  }
}
```

- And you could do this multiple levels deep.
- As many nested levels as you like.