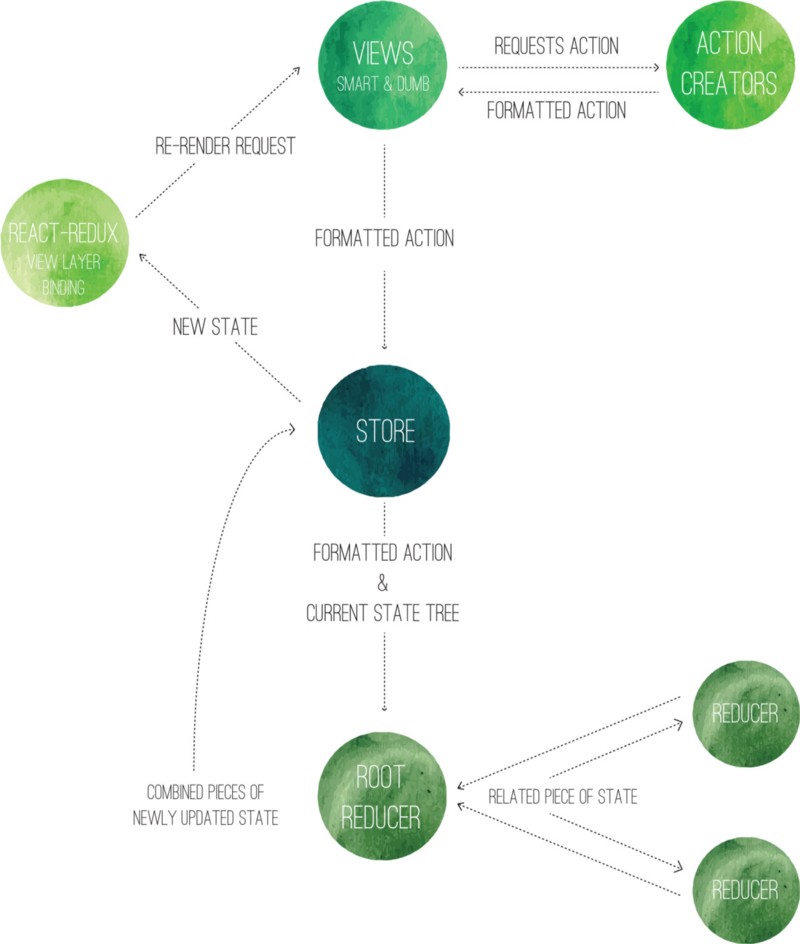
**Redux vs Flux**

Going into Redux, I wanted to see the difference in patterns between the two.

The general gist of Redux is that the entire state of your app is stored in an object tree inside a single store. The only way to change the state tree is to emit an action (object). Pure reducers will then specify how those actions transform the state tree.

The single major difference between Flux and Redux is that Redux doesn’t have a Dispatcher, or support many stores. Instead, there is just a single store with a single root reducer.



Action creators in Redux differ from action creators in Flux by forgoing sending the action to the dispatcher, and instead returning a formatted action object to the store.

The store delegates the work to the reducers by sending them what specific state change they need to update.

The root reducer will split the actions up and send it to the respective reducer. Instead of changing the state completely, they make a copy of the current state, and send back the state changes made on that copy.

The smart components are in charge of the actions. They essentially tell the dumb components what to do by passing a function to them via props. The dumb components, or presentational components are mainly in charge of laying out DOM elements.

The view layer binding connects the store to the views by introducing three concepts:

* The Provider component — uses the connect() method to link up the component’s children to the store.
* Connect() — is a function that will set up all the wiring to get state updates using the selector.
* Selector — specifies what parts of the state a component needs as properties.

It will then trigger a re-render updating all the state changes in the views.