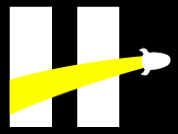
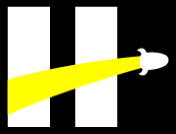
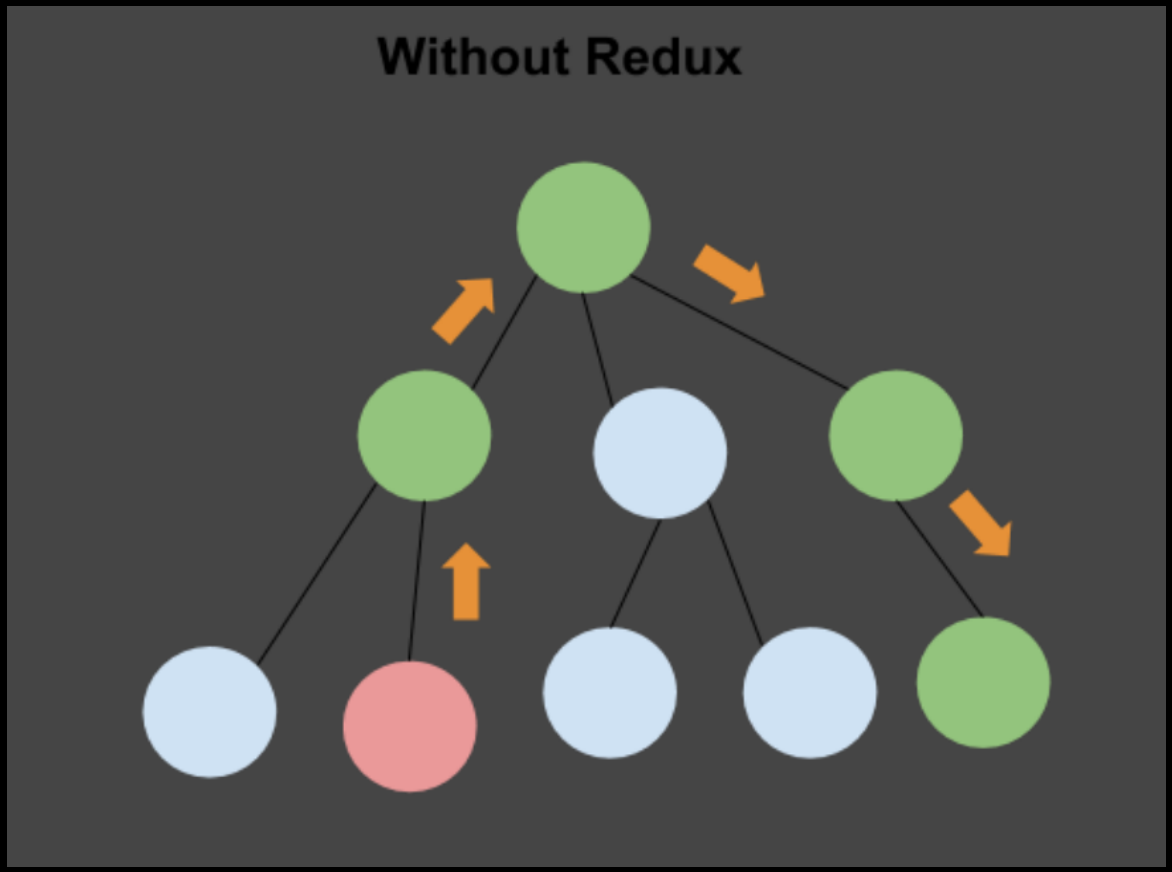

HENRY



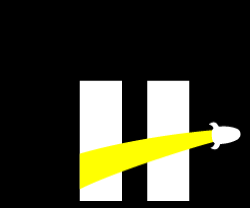
Redux



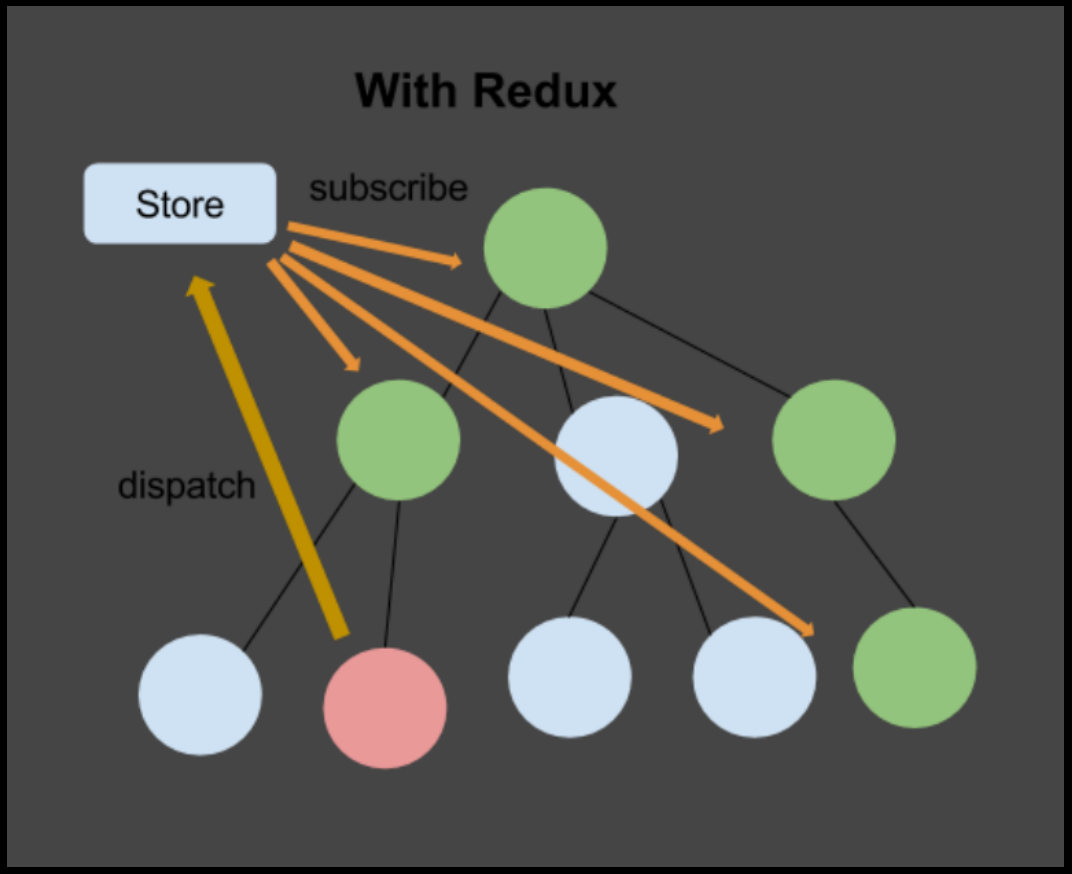
Redux



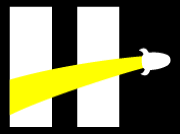
Podría generar ciertos problemas.



Redux



¿Qué Componentes *suscribimos* al store?



Componentes Presentacionales VS Containers

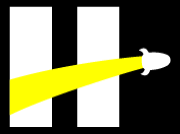
Tambien llamados smarts y dumbs components.

Smart:

- Cómo funcionan las cosas.
- Poco o nada de DOM.
- Sin estilos
- Provee datos
- Invoca las acciones

Dumb:

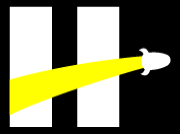
- Cómo se ven las cosas.
- Trabaja con sus props.
- Generalmente no tienen estado propio. (Algunos sí!)



Componentes Presentacionales VS Containers

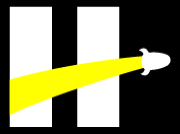
Separarlos de esta manera traer varias ventajas:

- Separa los problemas de la lógica de lo presentacional.
- Obtenemos componentes reutilizables (los presentacionales mayormente).
- Localizamos la complejidad en los containers



Componentes Presentacionales VS Containers

	Presentacionales	Containers
Próposito	Cómo se ven las cosas (markup, estilos)	Cómo funcionan las cosas (traer datos, actualizar estados)
Sabe de Redux	NO	SI
Para leer datos	Lee de props	Se suscribe a los estados de Redux
Para cambiar datos	Invoca callbacks de sus props	Envía acciones a Redux



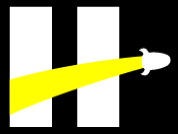
React-Redux

Vamos a conectar a redux a los Containers.

```
1  import { bindActionCreators } from 'redux';
2  import { connect } from 'react-redux';
3  import * as actionsCreators from '../actions/actionCreators.js';
4
5  import Main from './Main.js';
6
7  function mapStateToProps(state) {
8    return {
9      posts: state.posts,
10     comments: state.comments
11   }
12 }
13
14 function mapDispatchToProps(dispatch) {
15   return bindActionCreators(actionsCreators, dispatch);
16 }
17
18 const App = connect(mapStateToProps, mapDispatchToProps)(Main);
19
20 export default App;
```

mapStateToProps: Recibe el estado de la apliación y lo mapea a props de react.

mapDispatchToProps: Recibe el método dispatch y retorna callbacks props que vamos a poder pasar a los Componentes presentacionales



React-Redux

Todos los Componentes Containers deben tener acceso al Store para que puedan suscribirse a ella.

```
...
import { Provider } from 'react-redux'; //Bindings from redux and React
import store, { history } from './store.js';

const router = (
  <Provider store={store}>
    <Router history={ history }>
      <Route path="/" component={App}>
        <IndexRoute component={PhotoGrid}/>
        <Route path="view/:postId" component={Single}/>
      </Route>
    </Router>
  </Provider>
)
...
```

Lo que nos recomienda redux es usar un Componente especial de react-redux llamado <Provider> que mágicamente hace que el Store esté disponible para todos los Container de nuestra app, sin pasarla explícitamente.