COLLEGIO TECHNOLOGIES INC.

# REACT NATIVE: CLEANUP AND DATA PERSISTENCE

# INTRODUCTION

- We have some animations to work with for swiping left and right in our app, but it still looks a bit odd. Let’s continue using animations to clean up the look and give our player cards a ‘stacked’ feel!
- Let’s get some basics out of the way and clean up some of our look:
  - Fix up the list of players to show all that are past the current index, and not loop amongst players.
  - Create a display for if there are no more players present.

# EXERCISE: 'TINDER' APP CLEANUP

- We want to give our cards a 'stacked' feel, instead of showing like a list.
- In order to relatively display components, we can use the 'absolute' position style, and set the width to the screen width
  - **Note:** Do not set 'left' and 'right' to 0; this will cause unexpected results with any gesture animations!
- When you perform this change, there are some additional things that you must keep in mind:
  - Should your cards stay in the same place when the list is updated, or should they move?
  - How would you handle new incoming cards?

## DATA PERSISTENCE IN REACT NATIVE

## INTRODUCTION TO PERSISTING APPLICATION STATE

- By design, when an app is restarted or closed, any state stored in Redux is automatically dumped and unrecoverable.

- It is a good idea to have some form of data persistence in any app that handles data changes to prevent data loss in the case where you may not have an internet connection.

- Luckily, there is a way that we can do this by using ***Redux Persist***.

# REDUX PERSIST

- You can install Redux Persist via NPM:
  - **npm install --save redux-persist**

- Redux Persist is a package that we can use to persist (hold onto data) and rehydrate (restore saved state values) a redux store.

- In order to use Redux Persist, you need to include it when you are declaring your redux store, and additionally, should specify how you are planning on storing the data.
  - React Native has a **AsyncStorage** module that can be used for this.

## REDUX PERSIST

- Redux persist essentially is used as middleware to fetch data from your specified source, and update the redux state accordingly.

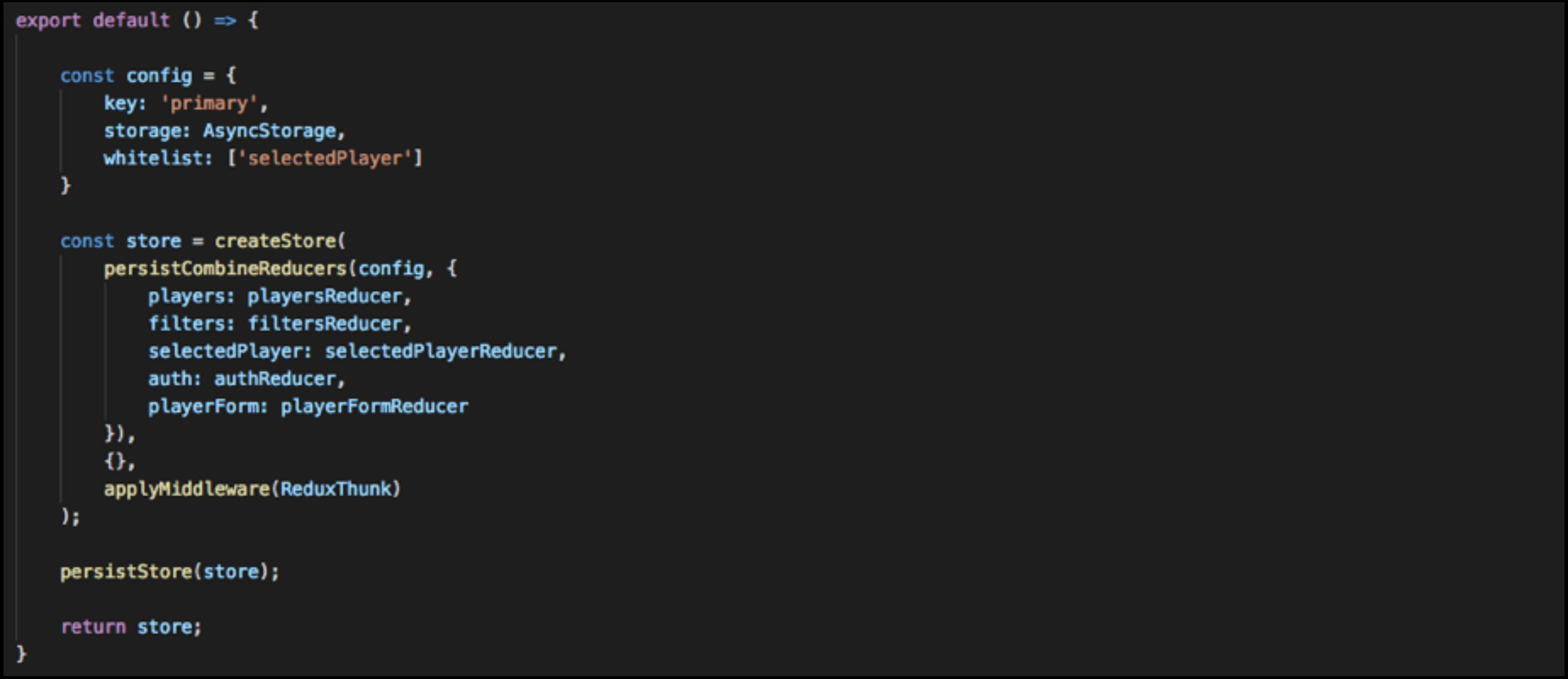
- The process of updating data with stored values is called **Rehydration**.

- In order to set your app up to use Redux Persist, you will need to replace your store's **combineReducers** function with **persistCombineReducers**. This function accepts the persist configuration as the first argument, and your reducers as the second argument.

# REDUX PERSIST

- The Redux Persist configuration will require you to specify a key, as well as your storage type. In addition, you will want to specify which data you want to store by adding it to a **whitelist** array.

- Finally, you call **persistStore()** and pass your store to apply Redux Persist.



```
export default () => {

    const config = {
        key: 'primary',
        storage: AsyncStorage,
        whitelist: ['selectedPlayer']
    }

    const store = createStore(
        persistCombineReducers(config, {
            players: playersReducer,
            filters: filtersReducer,
            selectedPlayer: selectedPlayerReducer,
            auth: authReducer,
            playerForm: playerFormReducer
        }),
        {},
        applyMiddleware(ReduxThunk)
    );

    persistStore(store);

    return store;
}
```

## RESOURCES

- **Redux Persist:**
  - **Official:** https://github.com/rt2zz/redux-persist

## DISCLAIMER