Redux, beyond React - <https://www.youtube.com/watch?v=dbyQynY7zDw>

**Three core components to Redux:**

* Actions – Plain JavaScript objects that represent something happened
  + Something has happened
  + Some kind of unique identifier
  + Basically, an event
    - Const echoAction = (value) => ({

type: ‘ECHO’

Payload: value

})

* Reducer – A function that specifies how the state is changed in response to an action
  + Listen for actions
  + Does not modify the state
  + Take existing state
  + Combine with action
  + Produce new state
  + It’s a pure function – receives everything it needs to do, no external dependencies, doesn’t mutate anything it doesn’t own
    - Const reducer = (state, action) => {

If (action.type === ‘ECHO’) {

Return action.payload;

}

Return state;

}

* Store – Single JavaScript Object
  + Cornerstone of your data flow
  + Only place where state lives
  + Passes actions to reducer
  + Apply middleware
  + Notify subscribers
    - Const store =

createStore(reducer, ‘’);

store.subscribe(() =>

console.log(store.getState())

);

Store.dispatch(

echoAction(‘Hello’)

);

**Pure Functions:**

* Same input -> same output, doesn’t modify its argument
* Returns a new object with the updated value from the argument, so you don’t modify the existing value
* No side effects
  + Function reducer (state, action) {

Switch (action.type) {

Case ‘INCREMENT’:

Return { count: state.count + 1 };

}

}

* Benefits
  + Easy testability
  + Easy undo/redo
  + Time travel debugging
  + This is only beneficial if it’s a large application

**Interesting Quirks:**

* Actions need a type
* Reducers can’t change the provided state
* Reducers can’t return *undefined* (state of application should never be undefined)
* Store dispatching is synchronous – everything happens synchronously (when you dispatch an action is goes straight to the reducer)

**Redux is Simple:**

* Only a few core concepts (create store, create middleware, combine multiple reducers)
* Mostly how thing interact with Redux though (describes how an action should look and how you provide that to a reducer)
* The ‘core’ is pretty small

**Observable Redux:**

* Redux stores are observable objects
  + Implements Symbol.Observable
* Combine with Rx (JS)
  + Conditional subscription
  + Map state from the stream (can re-map data)
  + Combine streams with stores (firing data to the UI)

**Only one big JavaScript object, that’s the store:**

* Never mutate this object, only create new versions of the object
  + Allows for a full history of the exact state of your application
  + Multiple reducers, each change part of the store

https://github.com/angular-redux/platform/tree/master/packages/example-app