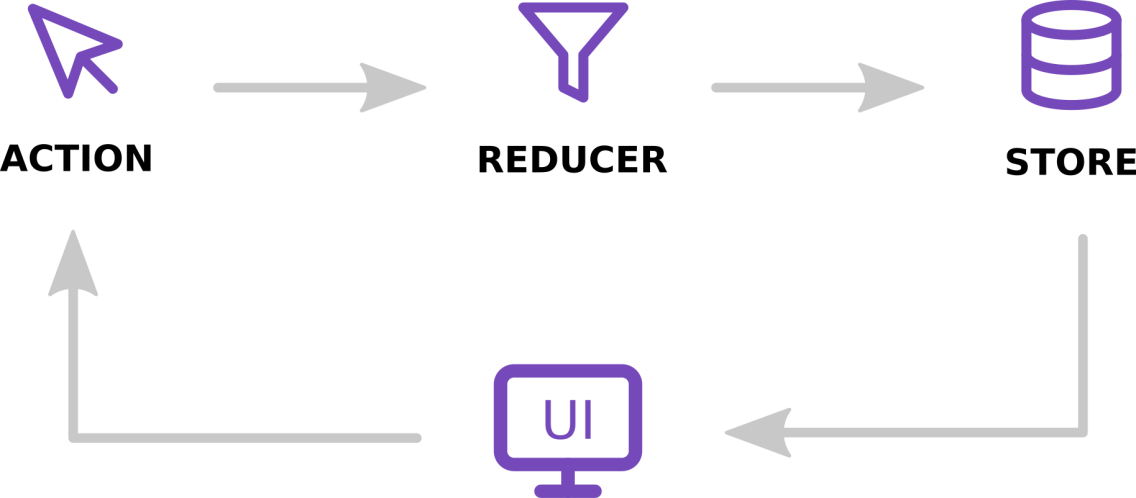
**Redux**

* Redux is a predictable state container for JavaScript apps. In other words , it’s an application data-flow architecture.
* Redux is used mostly for application state management. To summarize it, Redux maintains the state of an entire application in a single immutable state tree (object), which can’t be changed directly. When something changes, a new object is created (using actions and reducers).
* Redux follows 2 patterns
  + One pattern that Redux follows is called “Single Source Of Truth”, which means that we have only one place (called Store) where we store the only state for the whole application. In other words, one app - one store - one state.
  + Another pattern that Redux follows is called “immutability”.  Immutability means that we don’t change the state object and its properties directly. Instead, we make a new object, recalculate the new application state and update it with our newly created object.

**The three building blocks of Redux are:**



* **Redux Store:**

The store hold the state of the application. The store is actually an object, not a class. It contains a few extra things other than your application’s state as well (like functions and other objects).

Store is the object that holds the application state and provides a few helper methods to access the state, dispatch actions and register listeners. The entire state is represented by a single store. Any action returns a new state via reducers. That makes Redux very simple and predictable.

* **Redux Actions:**

Actions are plain JavaScript objects that describe “**WHAT”** happened, but don’t describe “HOW” the app state changes. One important thing to remember is that Redux requires our action objects to contain a type field. This field is used to describe what kind of action we are dispatching and it should usually be a constant that you export from a file. All other fields in the action object are optional and are up to you.

* **Redux Reducers:**

Reducers are **pure** functions that define **HOW** the app state changes. In other words, they are used to recalculate the new application state or, at least a part of it. Whenever we dispatch an action to our store, the action gets passed to the reducer.  
The reducer function takes two arguments: the previous app state, the action being dispatched and returns the new app state.