
Globant
we are ready

UI Engineering Studio. Day 14

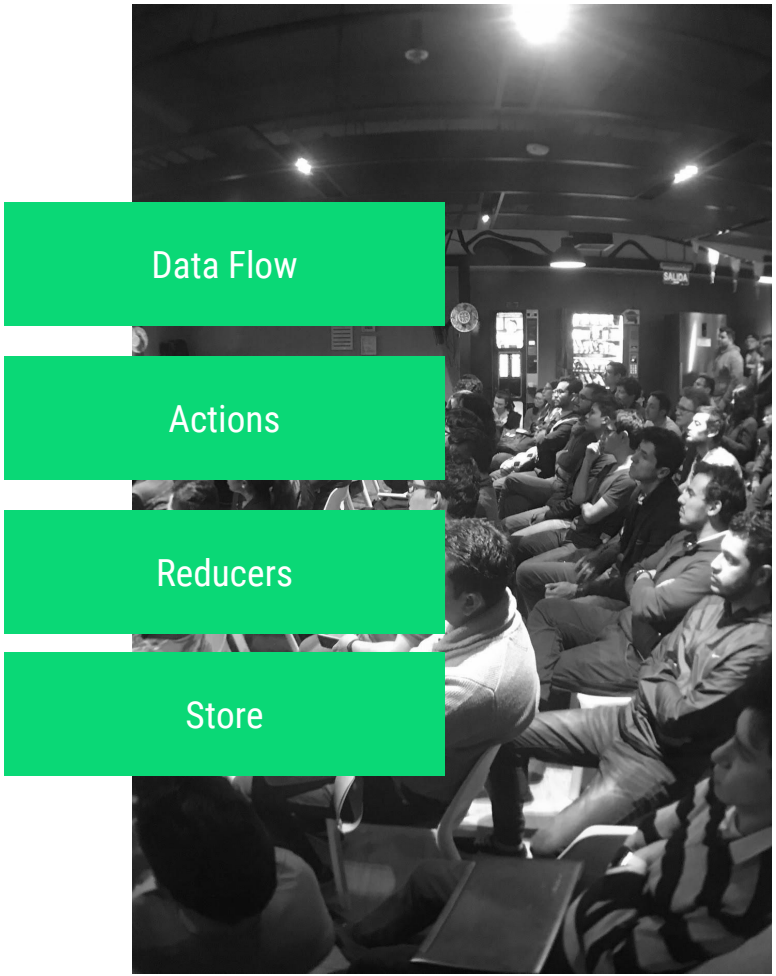
Bootcamp: Redux

Redux

Predictable state container for JS applications.

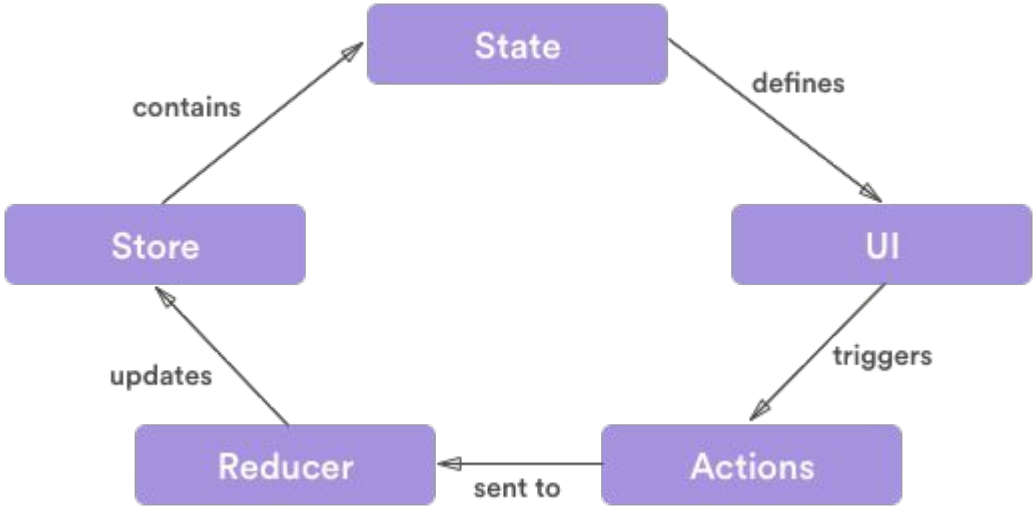
Why Redux?

- Helps us to write apps with a consistent behaviour
- It can be executed in different environments
- Easy testing
- Just 2kb



UI Boot Camp: Redux

Data Flow



UI Boot Camp: Redux

Data Flow

Redux architecture revolves around a strict unidirectional data flow. The lifecycle is:

- You call store.dispatch(action).
- The Redux store calls the reducer function you gave it.
- The root reducer may combine the output of multiple reducers into a single state tree.
- The Redux store saves the complete state tree returned by the root reducer.

UI Boot Camp: Redux

Actions

- Payloads of information
- The only source of information for the store
- Send them using store.dispatch().
- They describe WHAT happens

```
{
  type: ADD_TODO,
  text: 'Build my first Redux app'
}
```

UI Boot Camp: Redux

Reducers

Specify HOW the application's state changes in response to actions sent to the store.

```
import { VisibilityFilters } from './actions'

const initialState = {
  visibilityFilter: VisibilityFilters.SHOW_ALL,
  todos: []
}

function todoApp(state, action) {
  if (typeof state === 'undefined') {
    return initialState
  }

  // For now, don't handle any actions
  // and just return the state given to us.
  return state
}
```

UI Boot Camp: Redux

Store

The Store is the object that brings Actions and Reducers together.

```
{
  visibilityFilter: 'SHOW_ALL',
  todos: [
    {
      text: 'Consider using Redux',
      completed: true,
    },
    {
      text: 'Keep all state in a single tree',
      completed: false
    }
  ]
}
```

UI Boot Camp: Redux

Redux

https://codepen.io/dmiller5383/pen/awQaZg?editors=0010

UI Boot Camp: Redux

Homework!

We're going to continue with the TODO list.

- Let's implement Redux :D


Globant
we are ready