Redux

Clara Miranda García (UO264958)
Daniel Rückert García (UO236405)
Óscar Sánchez Campo (UO265078)

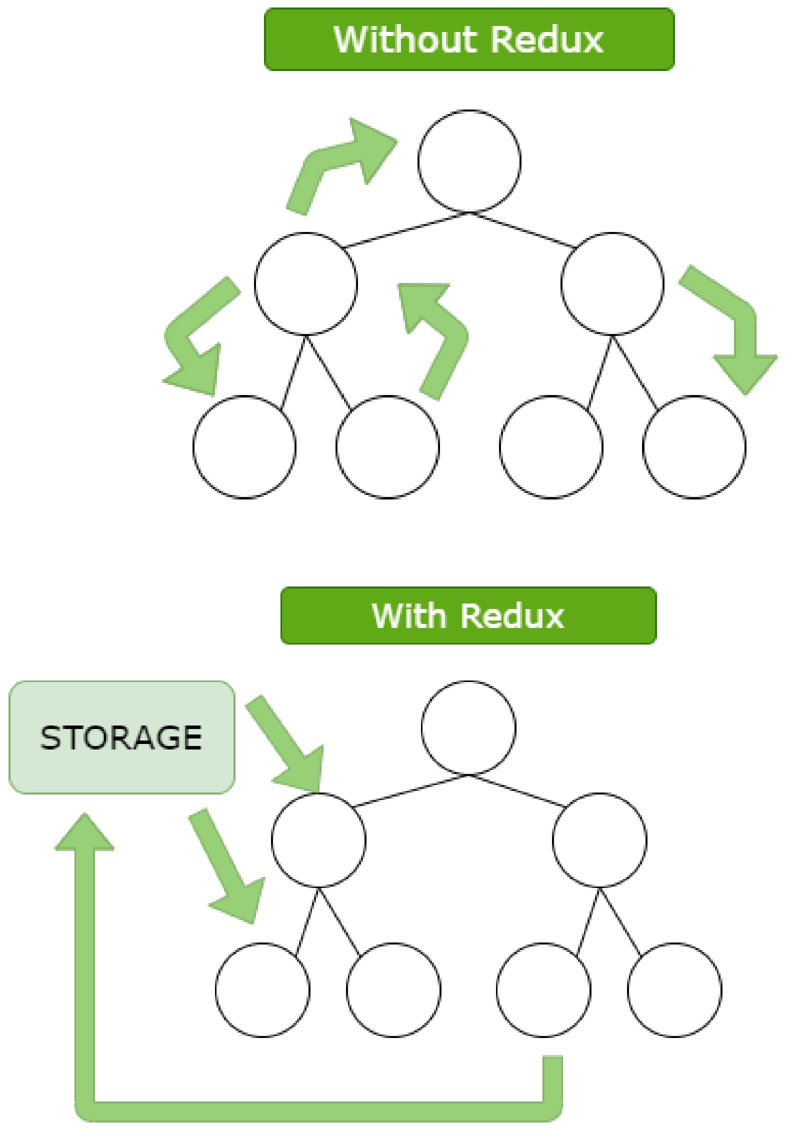
- What is Redux

What is Redux?

- **"Redux is a predictable state container for JavaScript applications"**
  - Why do we need a predictable state container?
  - What is a state container?
  - How does that fit into modern web applications?

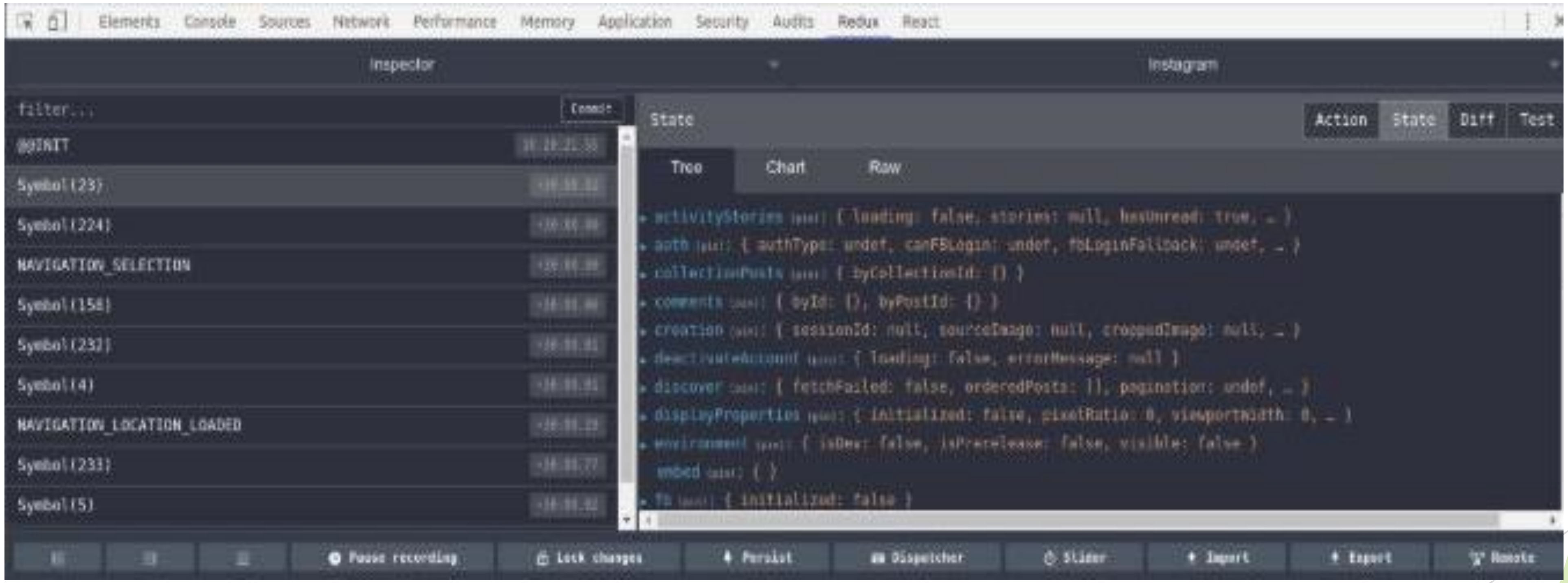
What is Redux?

- Why do we need to use it?



What is Redux?

- Time-traveling debugger



```
@@INIT
Symbol(23)
Symbol(224)
NAVIGATION_SELECTION
Symbol(158)
Symbol(232)
Symbol(4)
NAVIGATION_LOCATION_LOADED
Symbol(233)
Symbol(5)
```

```
▸ activityStories: { loading: false, stories: null, hasUnread: true, … }
▸ auth: { authType: undef, canFBLogin: undef, fbLoginFallback: undef, … }
▸ collectionPosts: { byCollectionId: {} }
▸ comments: { byId: {}, byPostId: {} }
▸ creation: { sessionId: null, sourceImage: null, croppedImage: null, … }
▸ deactivateAccount: { loading: false, errorMessage: null }
▸ discover: { fetchFailed: false, orderedPosts: [], pagination: undef, … }
▸ displayProperties: { initialized: false, pixelRatio: 0, viewportWidth: 0, … }
▸ environment: { isDev: false, isPrerelease: false, visible: false }
  embed: { }
▸ fb: { initialized: false }
```

- Stakeholders

Stakeholders

- Co-authors

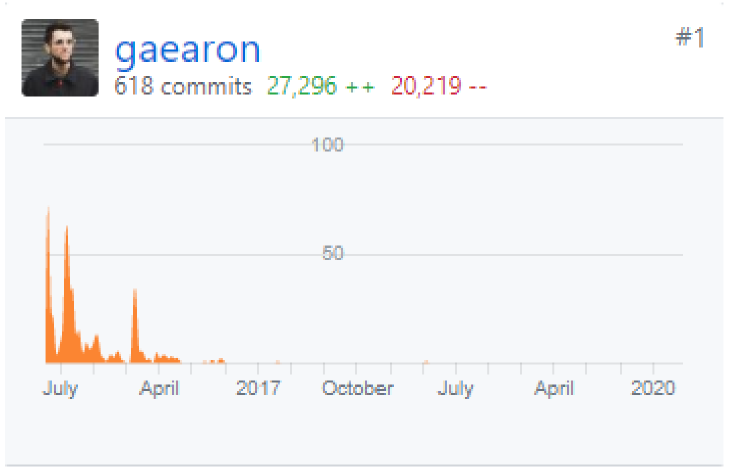
Stakeholders

- Open-source community

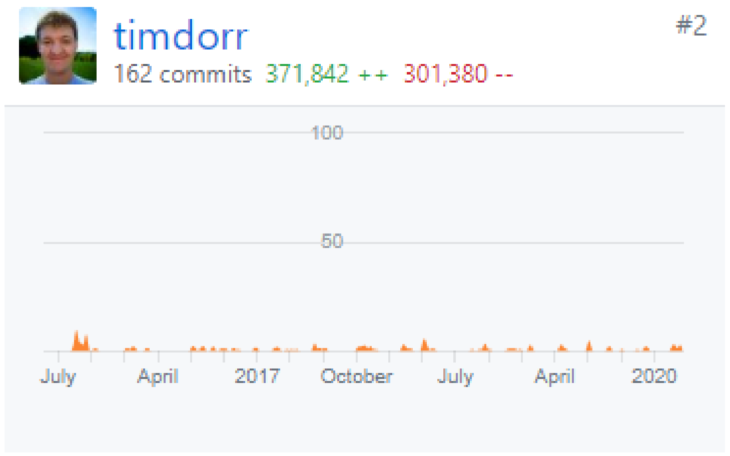
723 contributors

- Developers

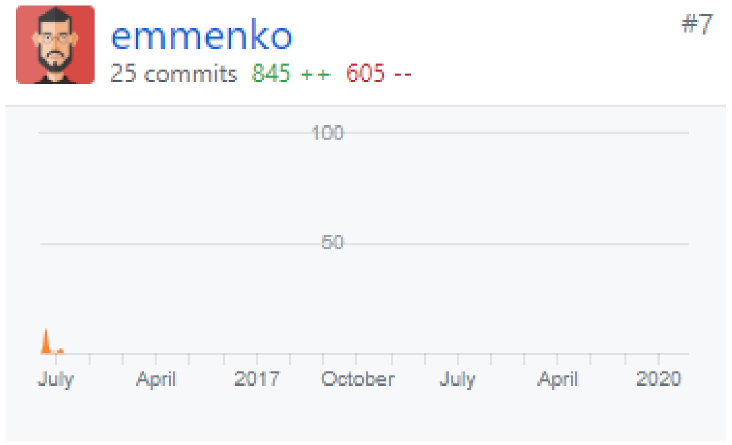
- Facebook



gaearon #1
618 commits 27,296 ++ 20,219 --



timdorr #2
162 commits 371,842 ++ 301,380 --



emmenko #7
25 commits 845 ++ 605 --

Stakeholders Contributors

Stakeholders
How to contribute

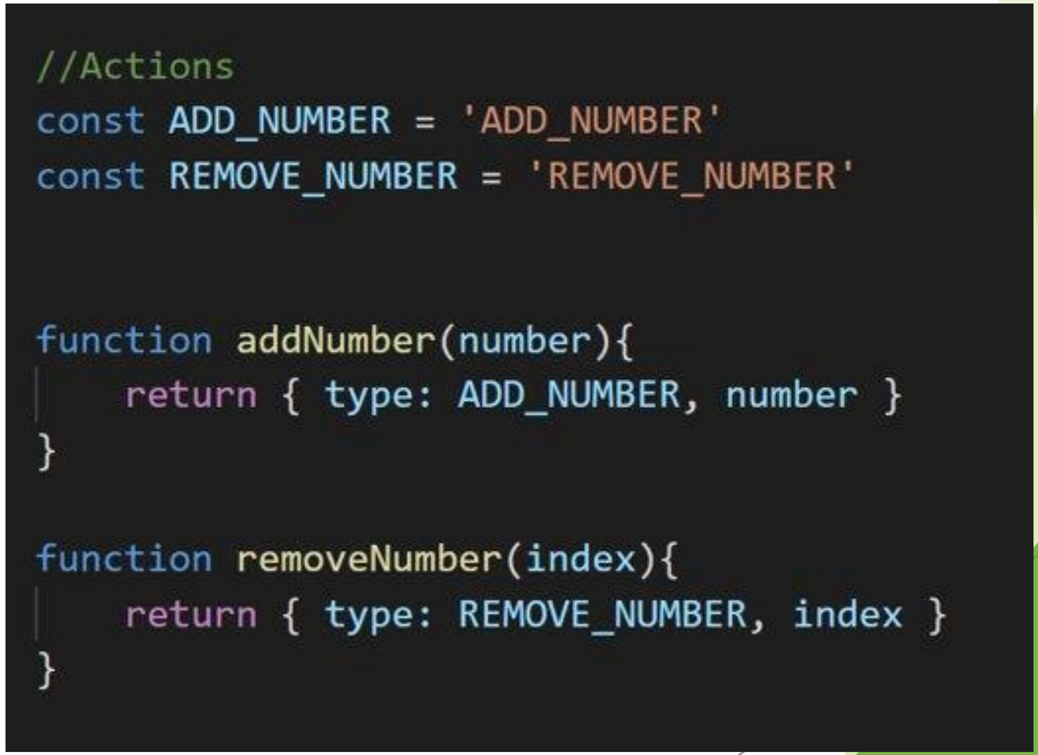
- Stable software
- Issues
- Pull request to "next" branch

- Components

Components

- Actions
- Reducers
- Store

Action:
(p1, p2, ...) => {type , p1, p2, ...}



```
//Actions
const ADD_NUMBER = 'ADD_NUMBER'
const REMOVE_NUMBER = 'REMOVE_NUMBER'


function addNumber(number){
    return { type: ADD_NUMBER, number }
}

function removeNumber(index){
    return { type: REMOVE_NUMBER, index }
}
```

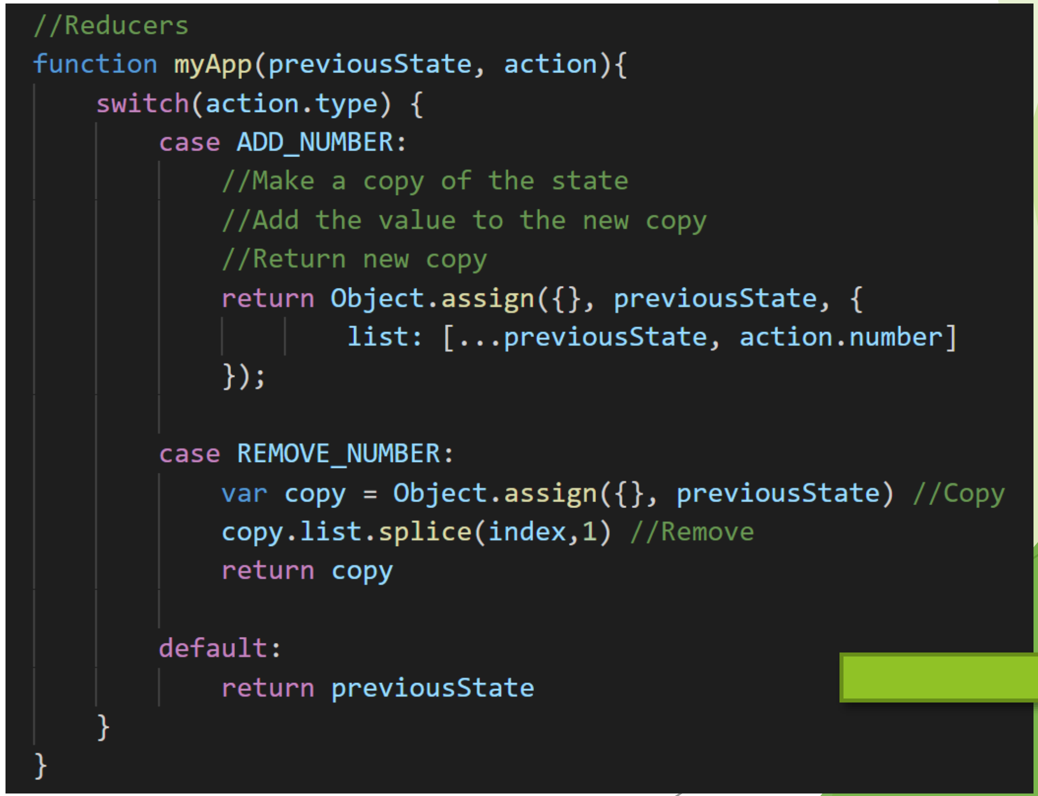
Actions

- Says "something will be done“
- Stores the parameters

Reducers

Action:
(p1, p2, ...) => {type , p1, p2, ...}

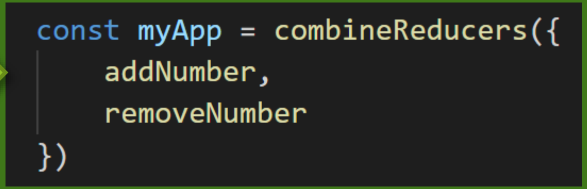
Reducer:
(previousState, action) => newState



```
//Reducers
function myApp(previousState, action){
    switch(action.type) {
        case ADD_NUMBER:
            //Make a copy of the state
            //Add the value to the new copy
            //Return new copy
            return Object.assign({}, previousState, {
                    list: [...previousState, action.number]
            });

        case REMOVE_NUMBER:
            var copy = Object.assign({}, previousState) //Copy
            copy.list.splice(index,1) //Remove
            return copy

        default:
            return previousState
    }
}
```

- Selects which code will be executed
- Gets the previous state
- Returns a new modified state

- Extracting common code
    - Reducer composition
- Get rid of the switch



```
const myApp = combineReducers({
    addNumber,
    removeNumber
})
```

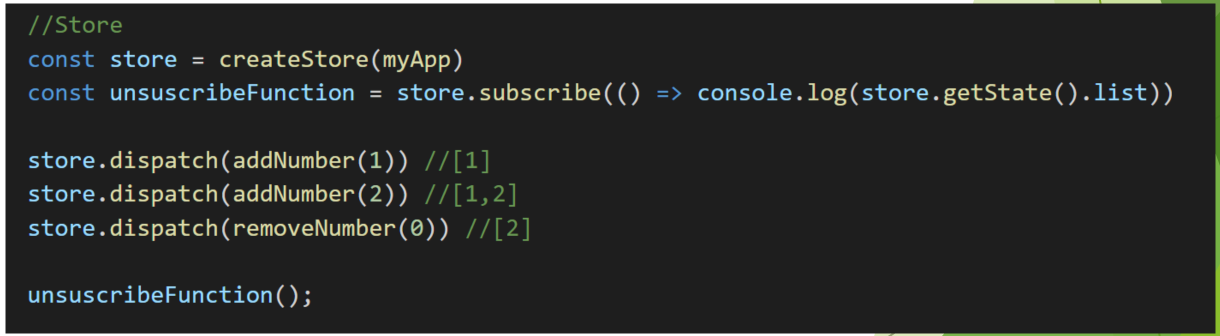
Store

- Initiates the execution of an action
- Maintains the current state
- Notifies changes to the components

Action:
(p1, p2, ...) => {type , p1, p2, ...}

Reducer:
(previousState, action) => newState

Store:
- subscribe: () => unsubscribe()
- dispatch: (action) => ()
- getState: () => currentState



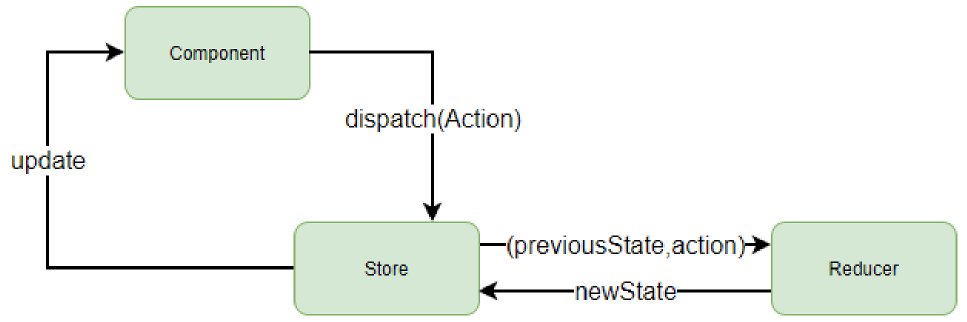
```
//Store
const store = createStore(myApp)
const unsuscribeFunction = store.subscribe(() => console.log(store.getState().list))

store.dispatch(addNumber(1)) //[1]
store.dispatch(addNumber(2)) //[1,2]
store.dispatch(removeNumber(0)) //[2]

unsuscribeFunction();
```

- Architectural style

Architectural Style
The Redux Pattern

- Based on Flux
- 3 Principles
  - Single source of truth
    - Store
  - Read-only state
    - Manipulated by actions, never directly!!!
  - Changes made with pure functions
    - Never change the input!!!

Unidirectional Data Flow

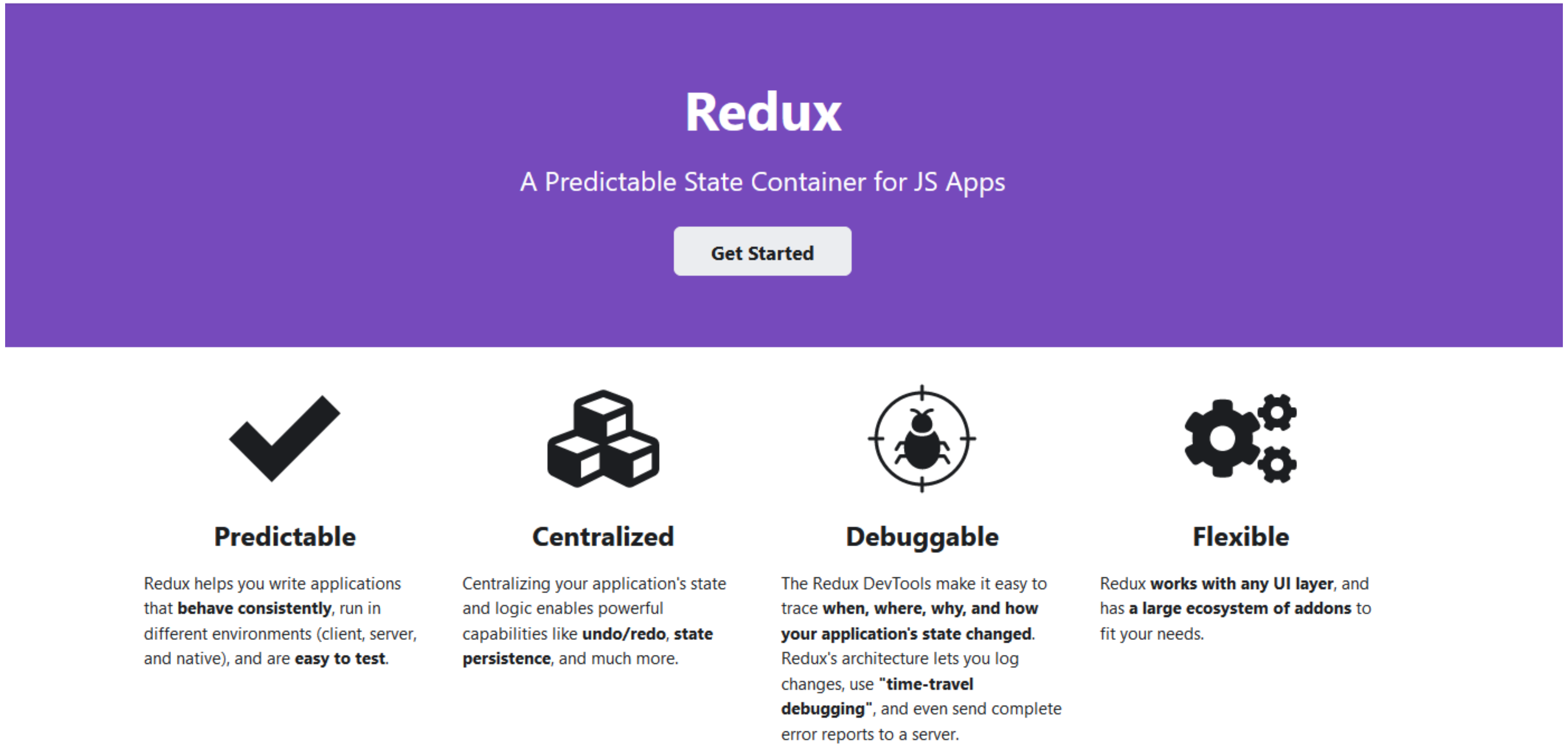


- Predictable
  - Easier to debug
  - Easier to modify
  - Less error prone

- Quality Attributes

Quality Attributes

- Lets see what they tell us on their official website:



Redux

A Predictable State Container for JS Apps

Predictable

Redux helps you write applications that **behave consistently**, run in different environments (client, server, and native), and are **easy to test**.

Centralized

Centralizing your application's state and logic enables powerful capabilities like **undo/redo**, **state persistence**, and much more.

Debuggable

The Redux DevTools make it easy to trace **when, where, why, and how your application's state changed**. Redux's architecture lets you log changes, use **"time-travel debugging"**, and even send complete error reports to a server.

Flexible

Redux **works with any UI layer**, and has **a large ecosystem of addons** to fit your needs.

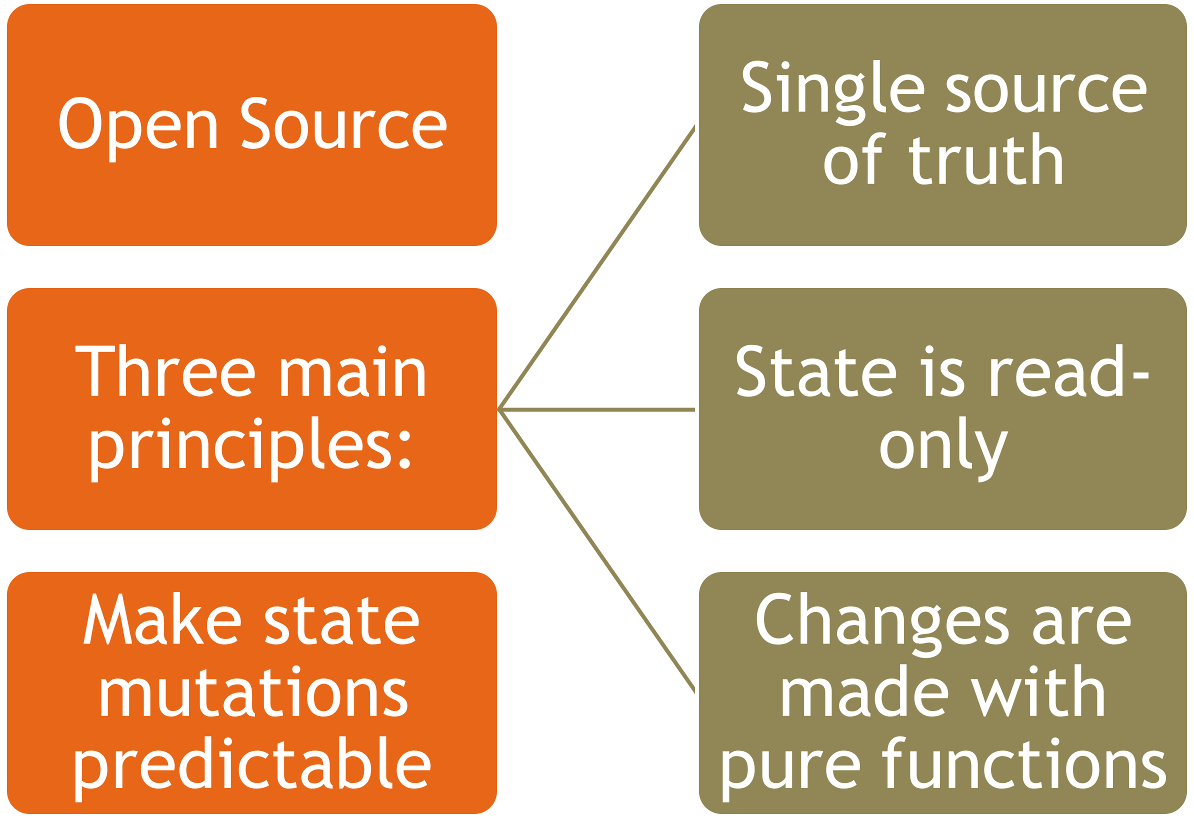
Quality Attributes

From this we can conclude that the quality attributes they value the most are:

- Reliability
- Reusability
- Supportability
- Interoperability

- Constraints

Constraints



- Development Aspects

Development Aspects

Written purely on TypeScript and Javascript

Modular

- Any Questions?