Introducción a Redux

Hola!

Carlos Muiño

Fullstack Developer at
@String-Projects.

@camumino

Redux: Un enfoque ‘sano’ para gestionar el flujo del estado en aplicaciones react complejas

CONTENIDO

- Historia
- ¿por qué Redux?
- ¿qué es Redux?
- Caso práctico

1. Historia

Javascript Frameworks

“the hardest part of writing a webapp is $X, so here's some code to make that easier”

¿Por que existen los frameworks?

- Llevan estado fuera del DOM
- Proporcionan alto nivel de abstracción
- Organizan el código

Pros

- Comparten conceptos comunes entres Apps y desarrolladores
- Grandes comunidades detrás que evolucionan, documentan y arreglan bugs
- Mejores aplicaciones gracias a sus convenios y guías de estilo

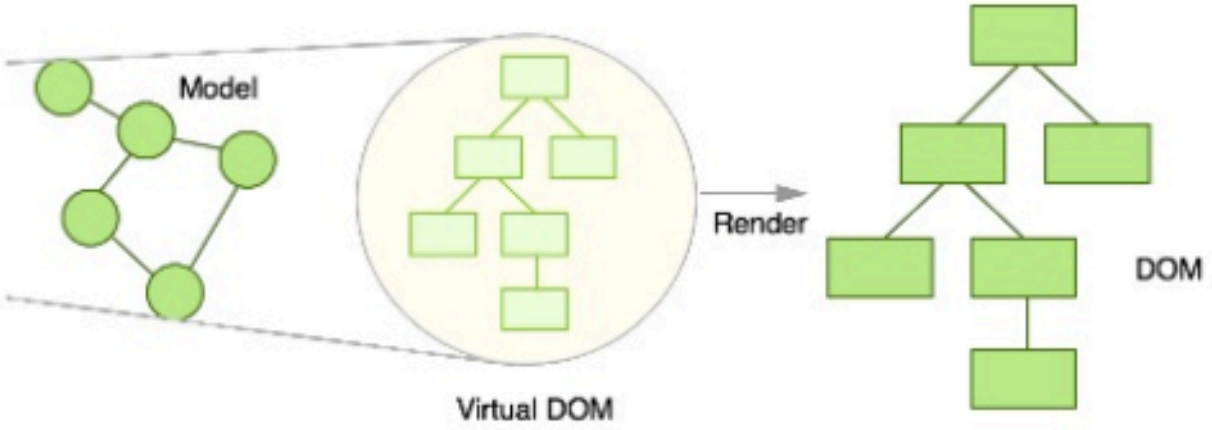
Cons

- Curva de aprendizaje
- Configuración e infraestructura

React

- Librería Javascript enfocada a UI
- Flujo de datos Reactivo unidireccional
- Virtual DOM

Virtual DOM

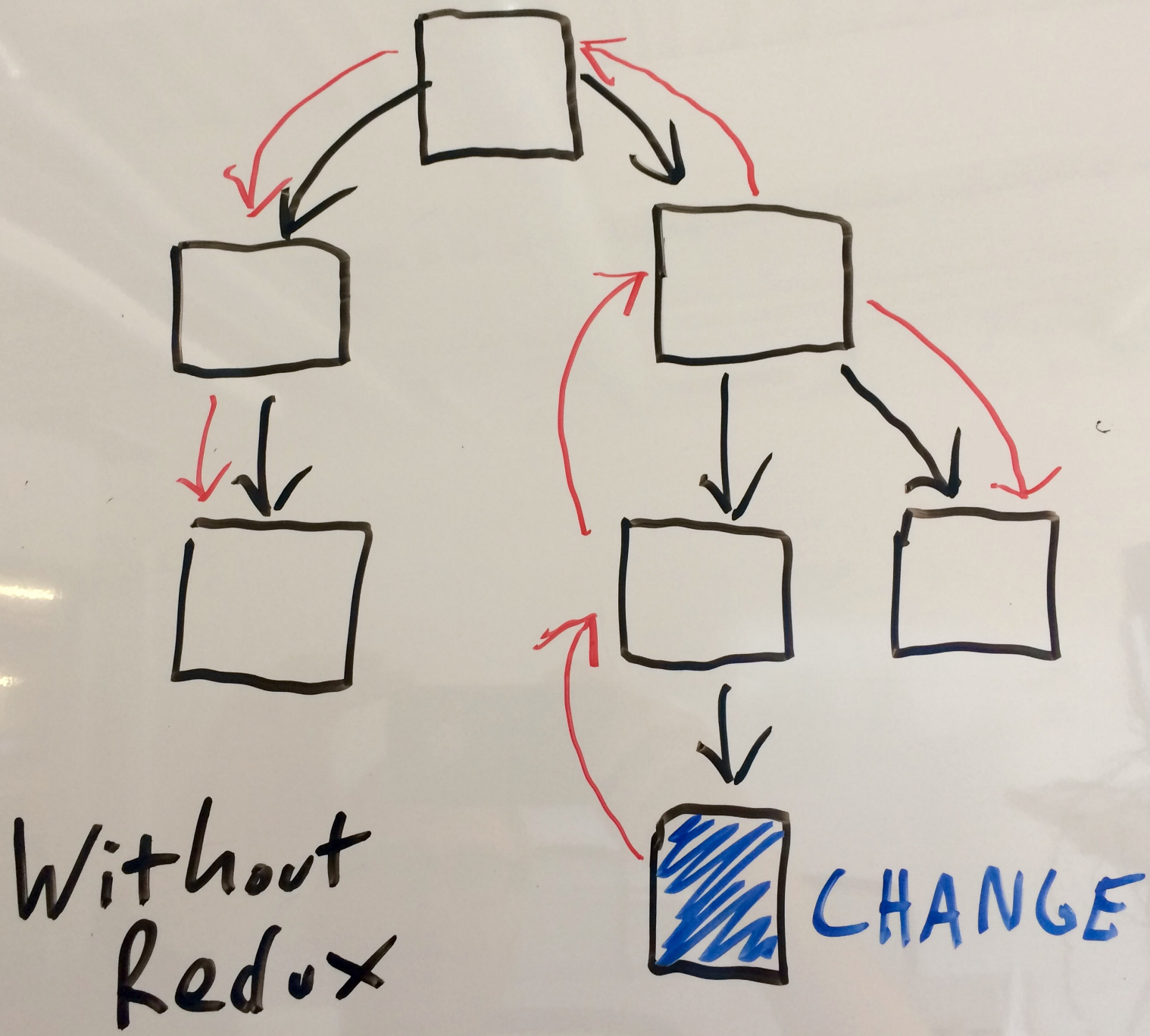


2. ¿Por que Redux?

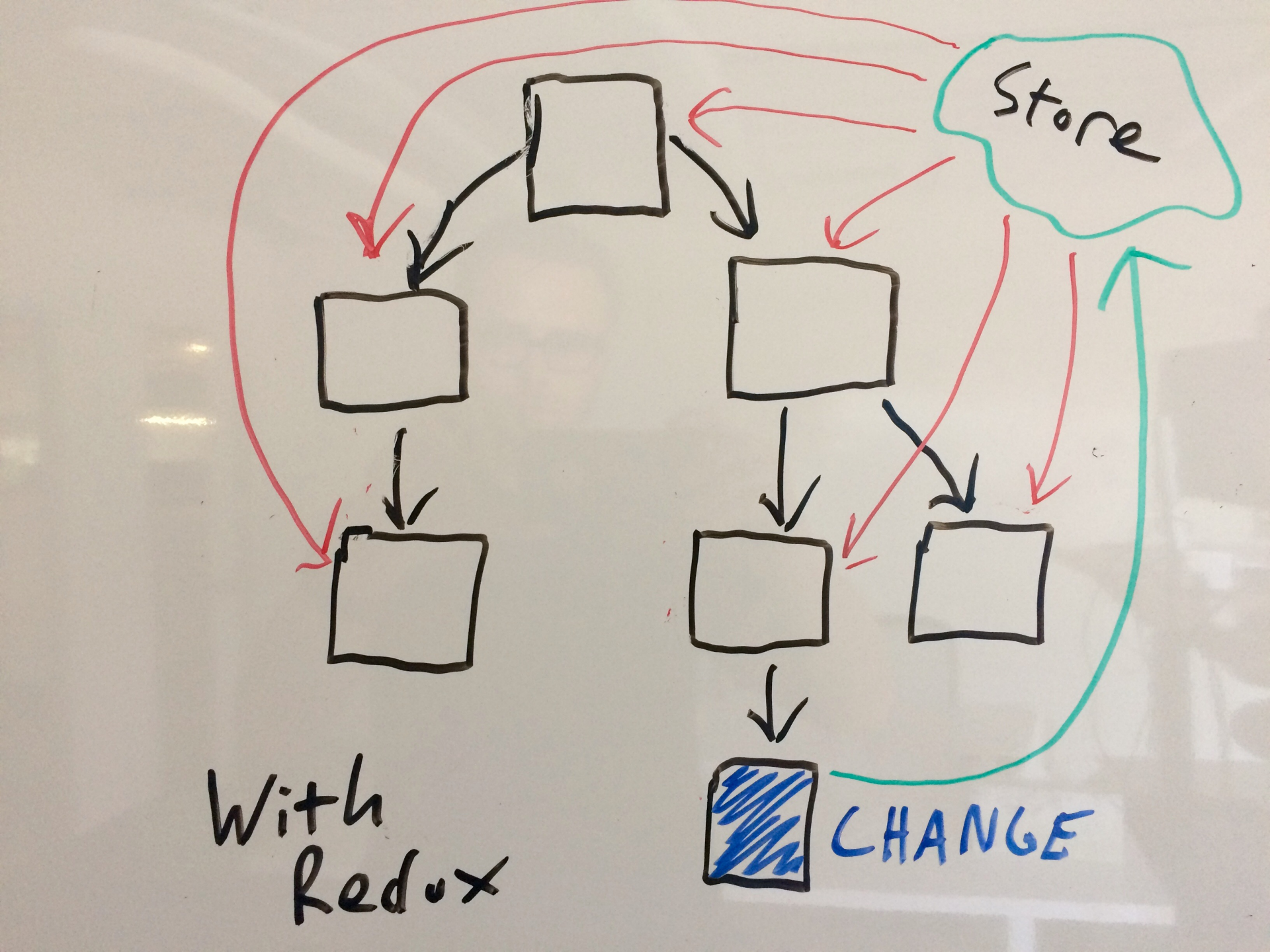
Consecuencias del crecimiento de aplicaciones

El estado se va propagando entre los diferentes componentes, provocando:

- Desorden
- Dificultad para seguir, por qué y cómo se actualiza el estado
- Estado muy acoplado a la lógica de la vista


Without
Redux
CHANGE

Redux al rescate


Store
With Redux
CHANGE

Único estado

Redux maneja el estado en una única pieza fuera de React.

3. ¿Que es Redux?

Orígenes

- Herramienta basada en el patron Flux, para manejar el estado de una aplicación.
- Creado por Dan Abramov para una charla en React Europe 2015.
- Actualmente es la herramienta mas utilizada para gestionar el estado en aplicaciones React.

Patrón Flux

"Flux eschews MVC in favor of a unidirectional data flow"

Conceptos clave

Única fuente de la "verdad"

El estado de toda la aplicación se almacena en un árbol de objetos dentro de un único almacén.

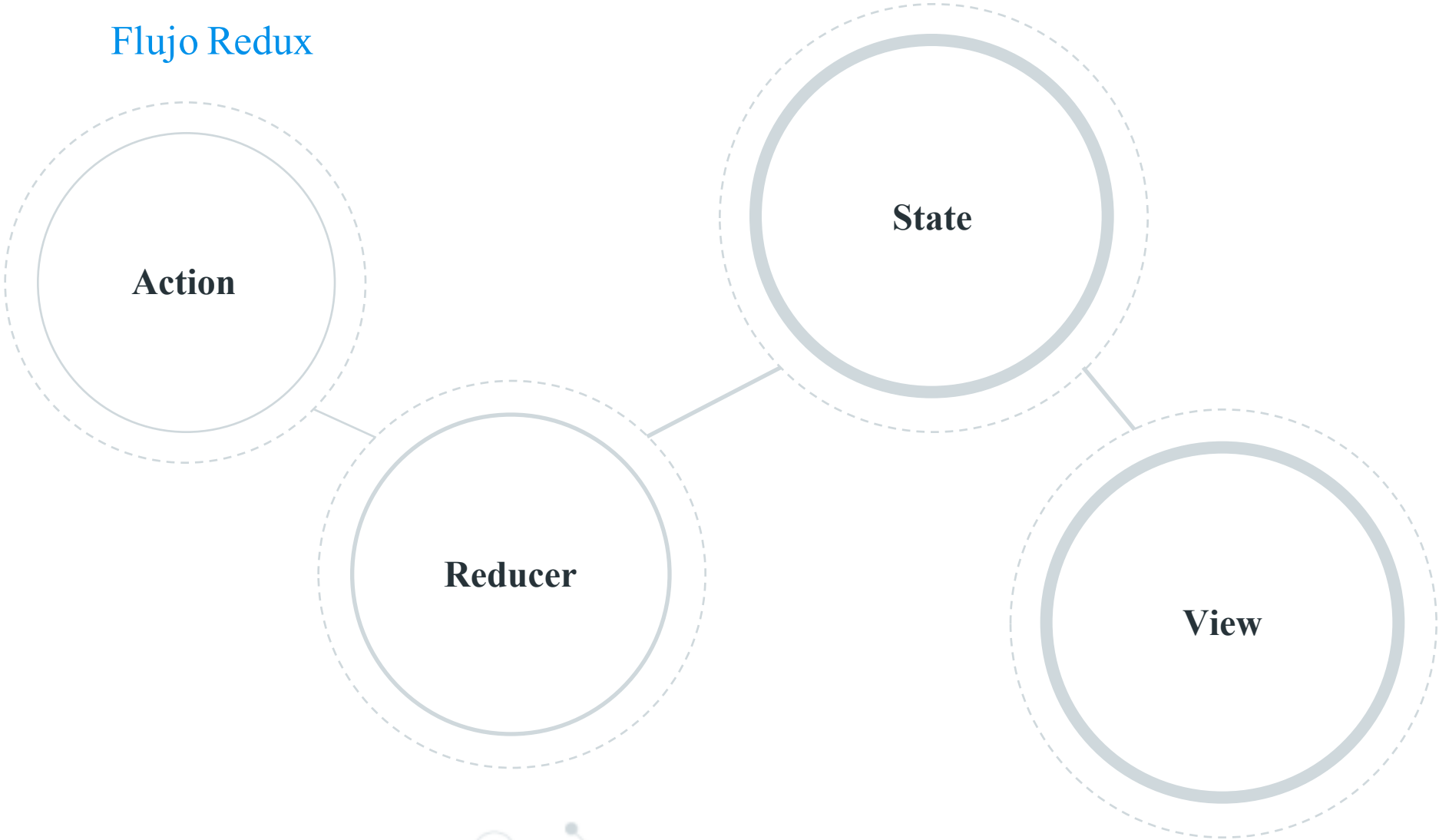
Estado de solo lectura

La única forma de cambiar el estado es lanzando una acción, que es un objeto que describe lo que ocurrió.

Cambios mediante funciones puras

Todas las actualizaciones de estado se realizan mediante funciones puras llamadas reducers.

Flujo Redux



Acción

Describe la acción que fue lanzada mediante el type y declara los parámetros a ser enviados al reducer

```
function addTodo(text) {
  const trimmedText = text.trim();
  return {
    type: 'ADD_TODO',
    text: trimmedText,
  }
}
```

Reducer

Funciones sencillas que manejan la lógica de actualización del estado. Reciben el estado actual y su único objetivo es devolver un nuevo estado. (state, action) => newState

```
const todos = (state = [], action) => {
  switch (action.type) {
    case 'ADD_TODO':
      return [
        ...state,
        {
          text: action.text,
          completed: false
        }
      ]
    default:
      return state
  }
}
```

Store

Lugar donde ‘vive’ el estado, es encargado de definir el estado inicial y conectar con el reducer

```
import { createStore } from 'redux'
import todoReducer from '../reducers'
let store = createStore(todoReducer);
```

Juguemos!

Gracias!

¿Preguntas?

@camumino

camumino@gmail.com

Código demo: https://github.com/StringProjects/introduccion_redux