
Redux

@dan_abramov

Redux is a predictable state container for JavaScript apps.

Motivación

El frontend es muy complejo

Store

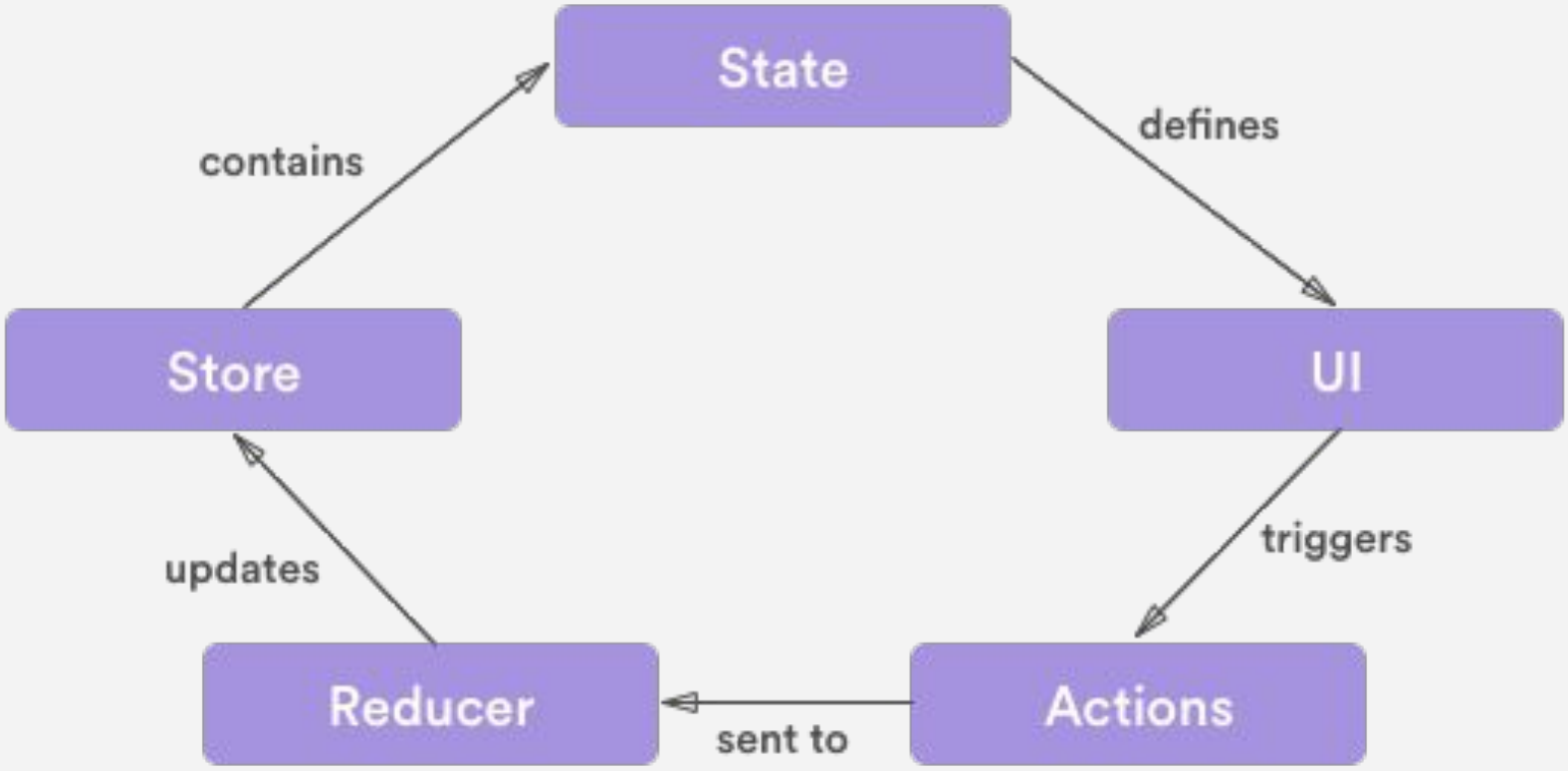
El centro y la verdad de todo, con métodos para actualizar, obtener y escuchar datos

Acciones

Las acciones son un bloque de información que envía datos desde la aplicación hacia el store

Reducers

Cambian el estado de la aplicación


State
defines
UI
triggers
Actions
sent to
Reducer
updates
Store
contains

Tres principios

Única fuente de la verdad

Single source of truth

El estado es de sólo lectura

State is read-only

Los cambios se realizan con **funciones puras**

Changes are made with pure functions

Quizá no necesites Redux

Redux <3 React

Pero puedes utilizarlo con vanilla.js o con cualquier otra librería o framework

Store

El centro y la verdad de todo, con métodos para actualizar, obtener y escuchar datos

1. Contiene el estado de la aplicación
2. Puedes acceder al estado con el método **getState()**
3. Puedes actualizar el estado con el método **dispatch(action)**
4. Escucha cambios con el método **subscribe(listener)**
5. Dejar de escuchar cambios retornando la función del método **subscribe(listener)**

Import { createStore } from ‘redux’

1. **Reducer** = función pura que retorna el próximo estado

2. **PreloadState / InitialState** = Estado inicial de la aplicación, primere carga, llamado al api. Puede ser cualquier tipo de dato

3. **Enhancer** = función que puede extender redux con capacidades añadidas por librerías externas. Es un parámetro opcional.

```
const store =  createStore(
    reducer,
    initialState,
    enhancer
)
```

Action

Bloque de información que envía datos a la aplicación

1. Se envían usando el método **dispatch()** del **store**
2. Son la única fuente de información del Store
3. Son objetos planos de JavaScript

```
{
  type: ‘ADD_SONG’,
  payload: ‘Despacito’
}
```

```
store.dispatch({
  type: 'ADD_SONG',
  payload: 'Despacito'
})
```

Reducer

Modifica el estado

1. Es una función pura
2. Puede haber más de un reducer en una aplicación pero solo debe haber un store
3. Devuelve el siguiente estado

Nunca hacer en un reducer

1. Modificar sus argumentos
2. Realizar tareas con efectos secundarios como llamados a APIs
3. Llamar a funciones no puras como Date.now() Math.random()

¿Qué es una función pura?

*“Dados los mismos **parámetros/argumentos/entradas** deben retornar el mismo resultado, sin importar el número de veces que se llame”*

“La función no debe tener efectos secundarios”

```
const reducer =  function(
    state,
    action
) {
// que hago con el estado y la acción
}
```

```
const reducer =  function(state, action) {
    switch (action.type) {
        case ‘ADD_SONG’:
            return [ ...state, { title: action.payload } ]
        default:
            return state
    }
}
```

Normalizando datos

Datos NO normalizados

Datos normalizados

```
{
  "categories": [
    {
      "title": "las más sonadas",
      "id": "1",
      "playlist": [
        {
          "title": "Despacito",
          "id": "123"
        }
      ]
    }
  ]
}
```

```
{
  "categories": {
    "1": {
      "title": "Las más sonadas",
      "playlist": [
        "123"
      ]
    }
  },
  "media": {
    "123": {
      "title": "Despacito"
    }
  }
}
```

Middlewares

Cada middleware recibe dispatch y el getState del Store como argumentos, y regresa una función. Esa función va a recibir el método para despachar el siguiente middleware, y se espera que devuelva una función que recibe action y llame next(action)