

УПРАВЛЕНИЕ СОСТОЯНИЕМ В ANGULAR 101

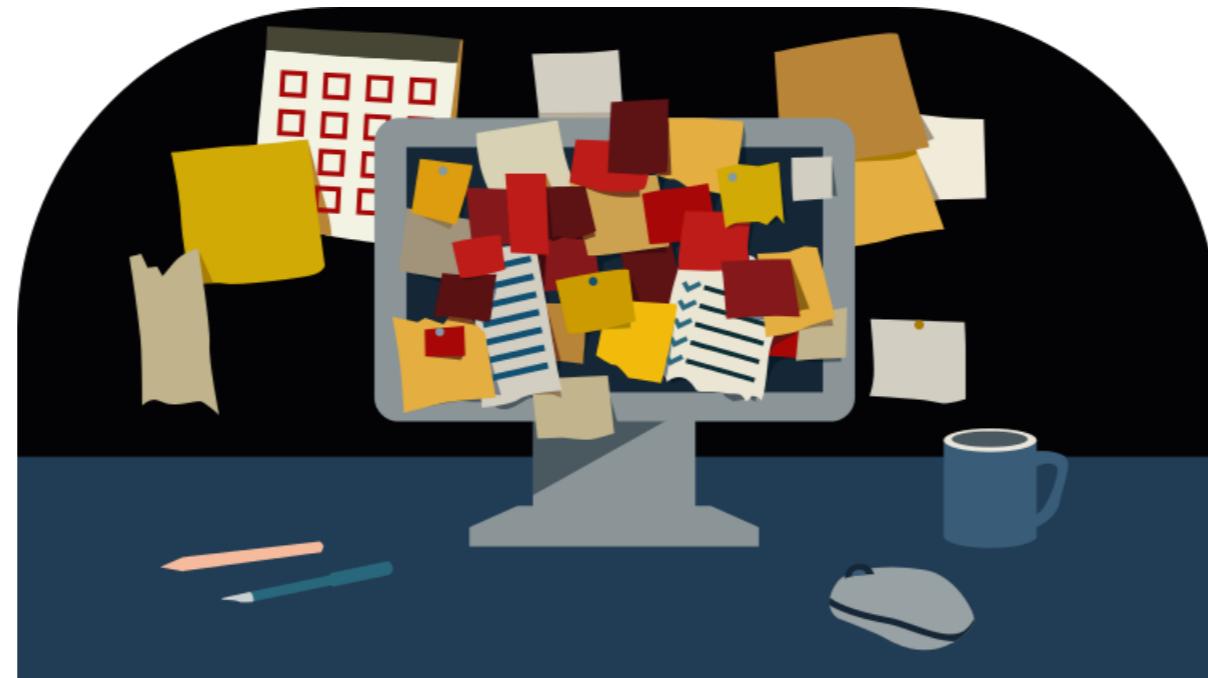
Рассказывает Черепов Антон





101 - вводный курс во что либо.*

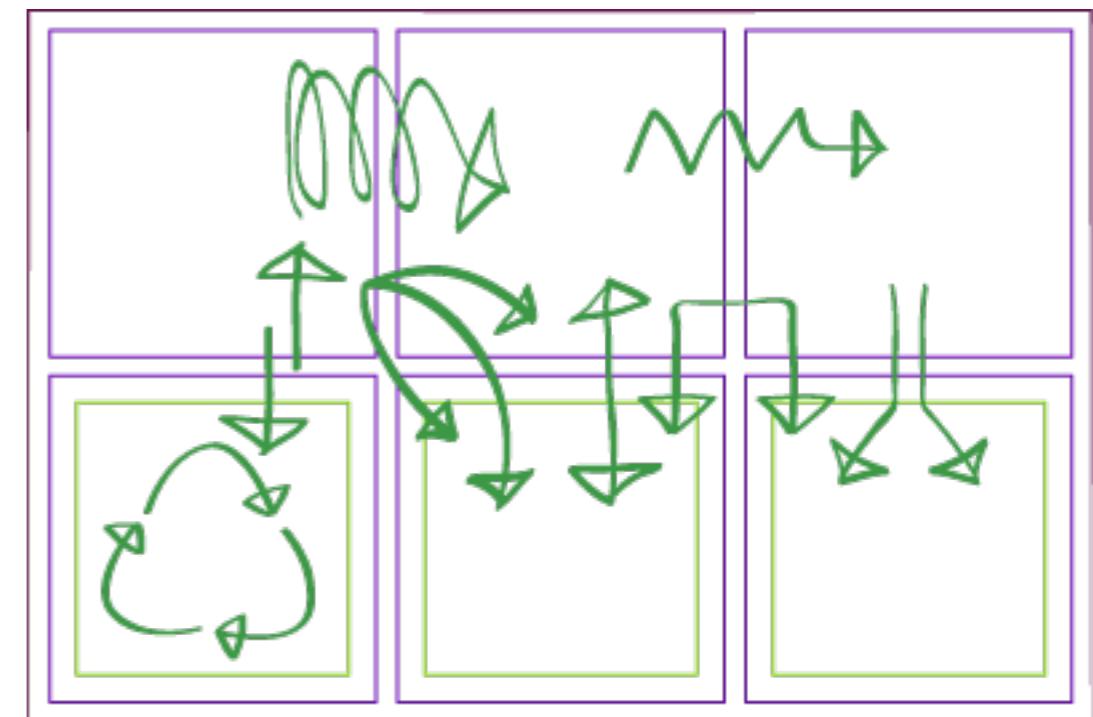
* В Американской системе образования и многих других данный код указывает, что курс носит вводный характер и не требует каких-либо знаний по теме.



Типичное приложение может работать с такими данными, как:

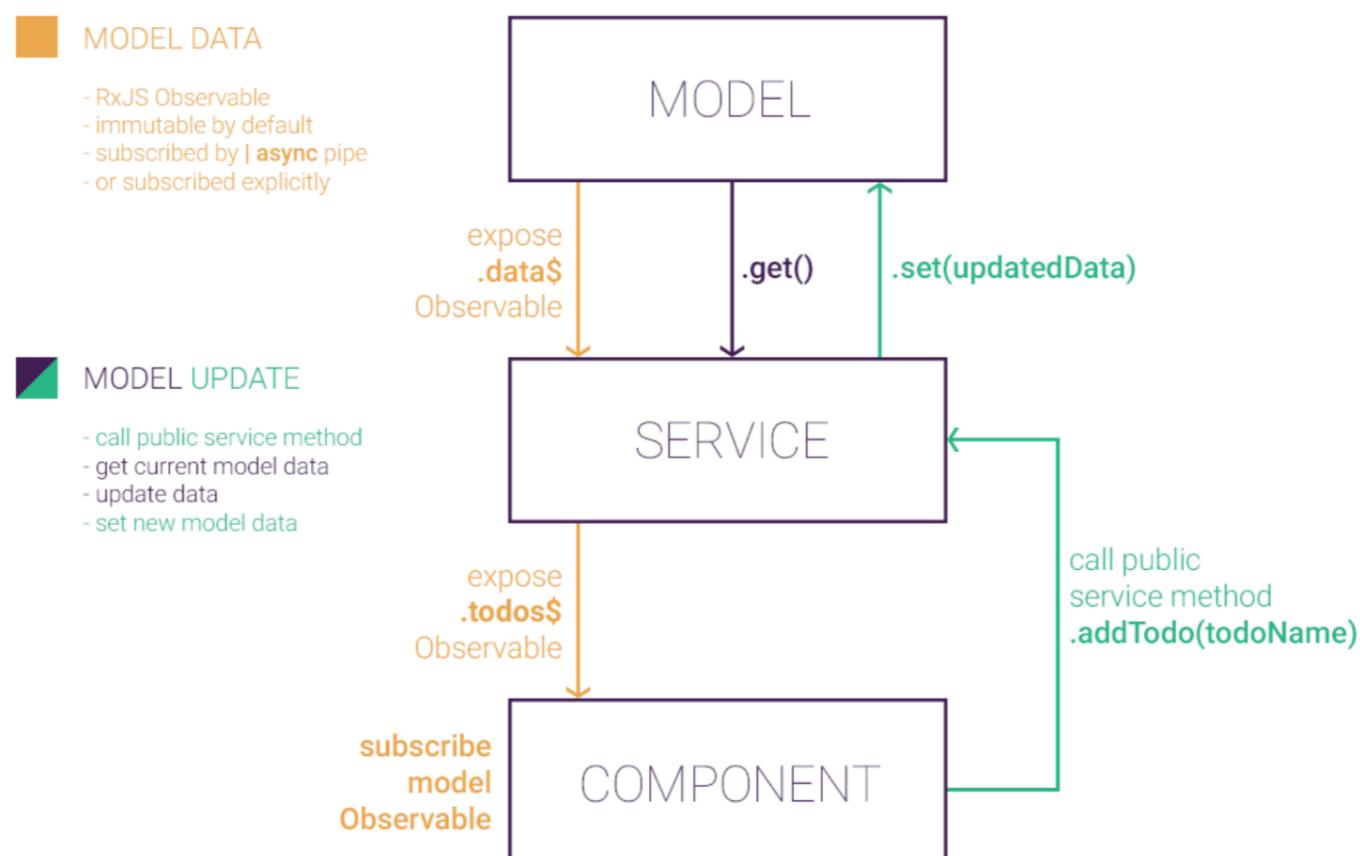
- Информация от сервера и результаты её обработки
- Состояние элементов UI
- Данные форм ввода, настройки фильтров, поисковые запросы по информации на странице
- Кастомизация элементов UI в зависимости от роли пользователя и пр.

**Чем больше приложение,
тем сложнее управлять его
данными и манипулировать
ими.**



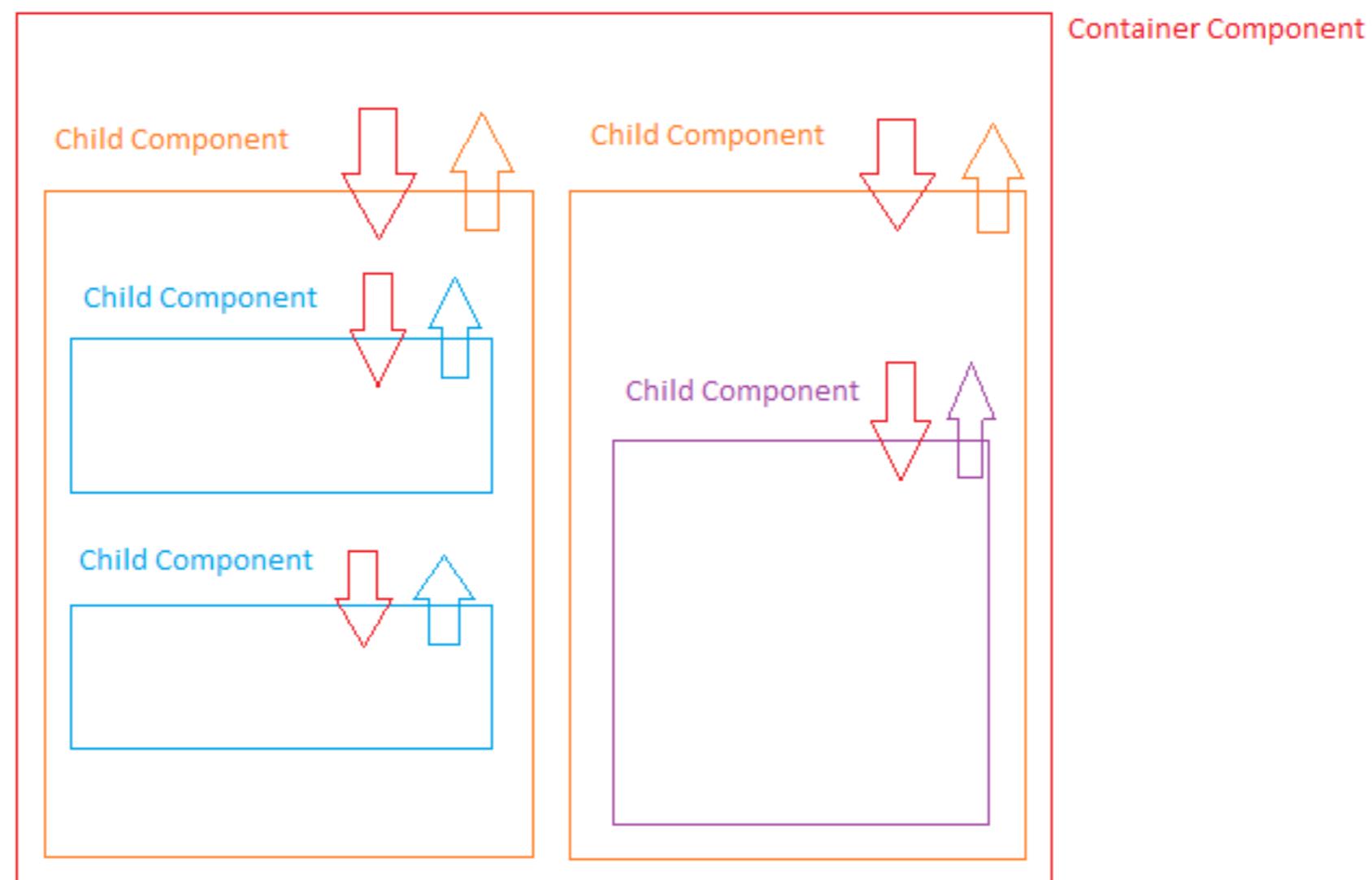
Как мы можем работать с состоянием приложения и данными?

- использовать сервисы и rxjs (ngx-model)



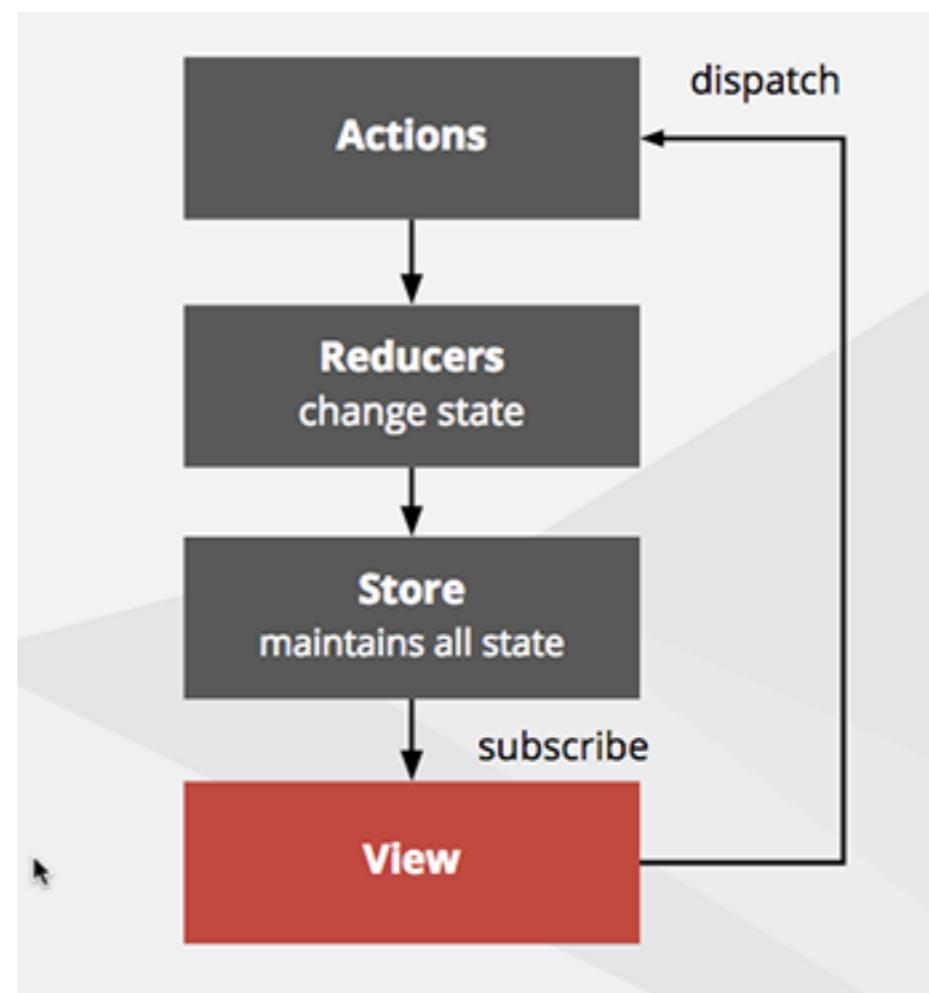
Как мы можем работать с состоянием приложения и данными?

- использовать сервисы и rxjs (ngx-model)
- передавать между компонентами общую модель данных



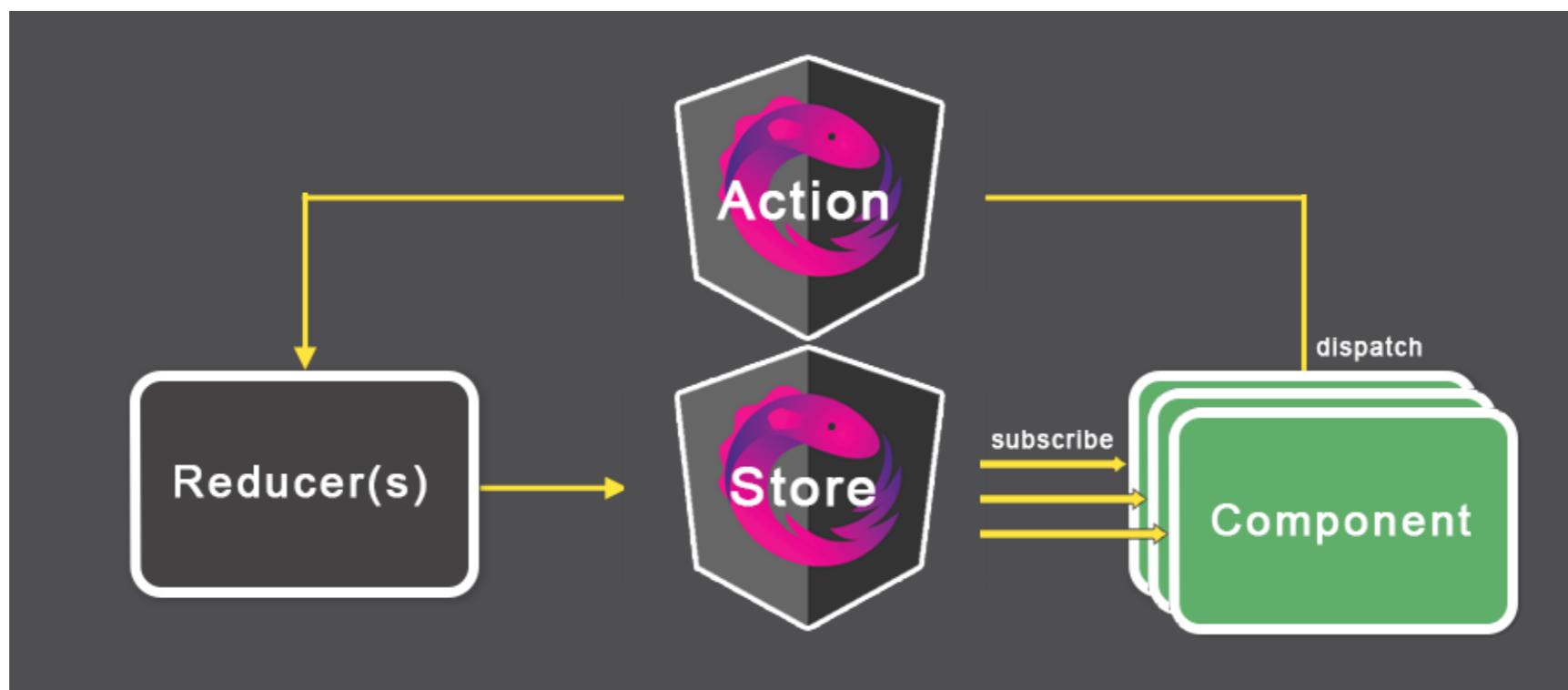
Как мы можем работать с состоянием приложения и данными?

- использовать сервисы и rxjs (ngx-model)
- передавать между компонентами общую модель данных
- использовать redux-паттерн



Как мы можем работать с состоянием приложения и данными?

- использовать сервисы и rxjs (ngx-model)
- передавать между компонентами общую модель данных
- использовать redux-паттерн
- использовать библиотеку ngrx/store



Redux



3 основных принципа:

- Единственный источник правды (Store)
- Состояние (State) неизменяемо (Immutable) и доступно только для чтения
- Обновлять состояние могут только "чистые" функции (Reducers)

Redux

Базовые принципы



Состояние, дерево состояний (State, State Tree)

- большой JavaScript объект, свойства которого содержат необходимую для приложения информацию

Redux

Базовые принципы



Действия (Actions)

- Представляют собой инструкции, которые уведомляют Store (хранилище) о наших намерениях внести изменения
- Состоят из двух свойств: тип (type), описывающий действие и дополнительные данные (payload)
- Единственный способ обновить состояние приложения

```
{  
  type: 'SHOW_ITEM',  
  payload: {  
    id: 10  
  }  
}
```

Redux

Базовые принципы



Преобразователи (Reducers)

- Чистые функции
- В зависимости от переданного типа действия реализует определённую логику и возвращает новое состояние
- Преобразователи синхронны, нужно стараться избегать асинхронных операций внутри них
- Каждый преобразователь (Reducer) отвечает за работу с определённой частью дерева состояний

```
function reducer(
  state: IItemsState,
  action: ItemsActions
) {
  switch (action.type) {
    case 'SHOW_ITEM':
      return {
        ...state,
        current: action.payload,
        loaded: true,
        loading: false
      }
    default:
      return state;
  }
}
```

Redux

Базовые принципы



Хранилище (Store)

- Контейнер для состояния, который содержит API для взаимодействия с состояниями
- Состоит из исполнителей (Reducer), состояний (States) и подписчиков (Subscribers)
- Компоненты взаимодействуют с хранилищем:
 - Подписываются на составляющие части состояния
 - Отправляют Действия в Хранилище
 - Хранилище вызывает Reducer, передавая ему на вход свое предыдущее состояние и само действие, а затем уведомляет всех подписчиков об изменении *

Redux

Базовые принципы



Неизменяемость (Immutability)

- Состояние хранится в неизменяемом объекте
- Если необходимо обновить состояние, то создаётся его копия и в ней обновляются данные

ngrx



- Основан на Redux
- Написан с помощью RxJS
- Предназначен для Angular
- * В ngrx всё построено на Observables, поэтому любые изменения сразу же "доставляются" всем подписчикам

ngrx



Преимущества

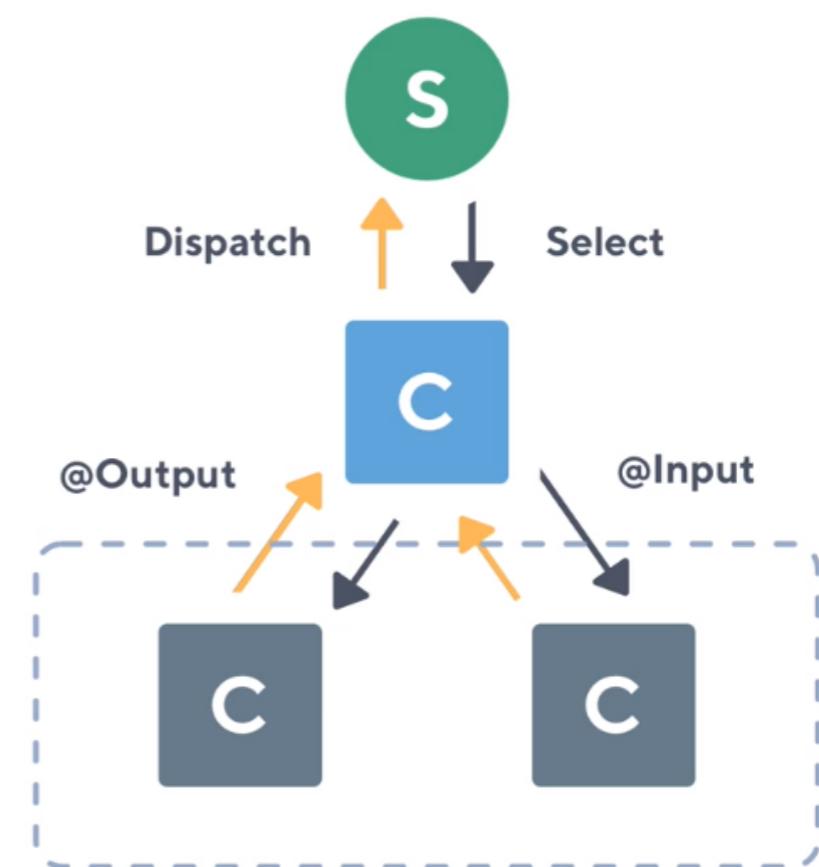
- Единственный источник правды (данных)
- Тестируемость кода
- Выигрыш в производительности
 - Возможность использовать стратегию определения изменений onPush

ngrx



При работе с ngRx используется 2 абстракции:

1. **Container component** - взаимодействует с Store по средствам методов dispatch и select
2. **Presentational component** (набор всех компонент приложения) - общается с Container component по средствам получения (@Input)/отправки значений (@Output)



ngrx



Selectors

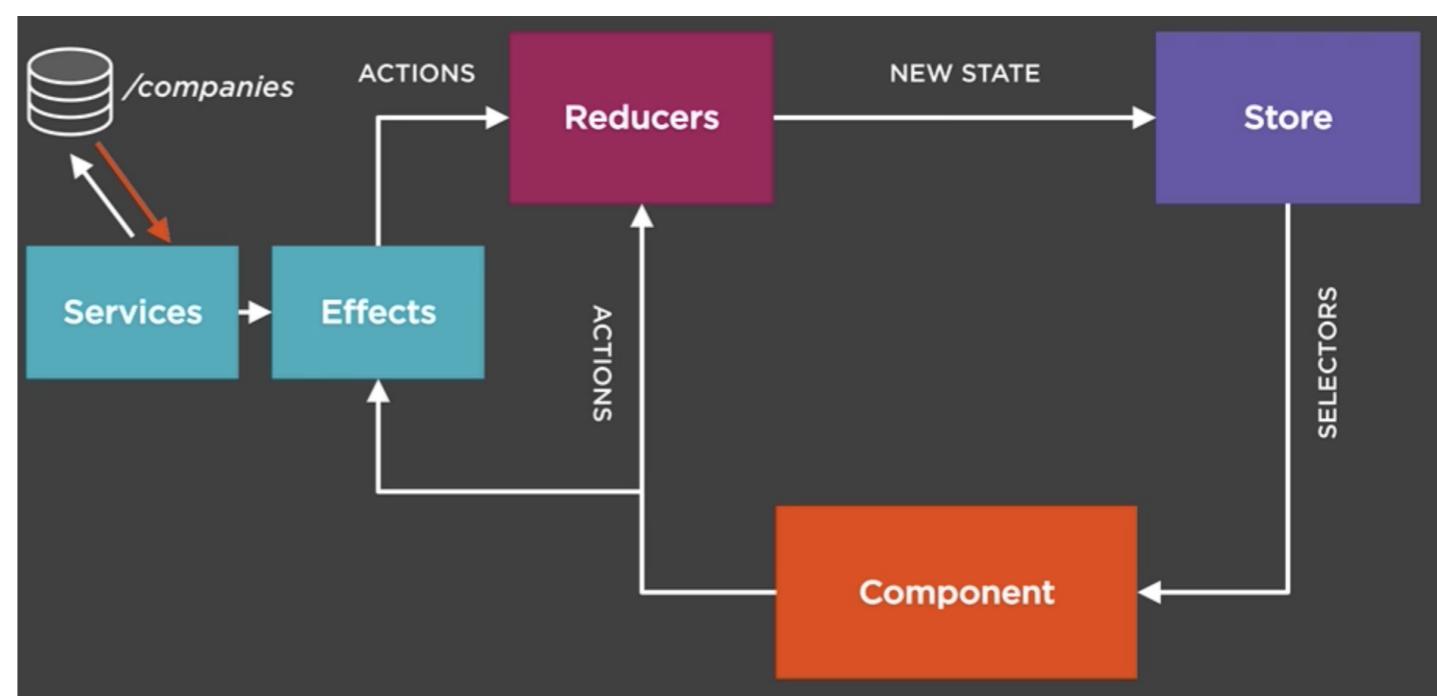
- Метод, который принимает в качестве аргумента функцию или строку и позволяет получить доступ к элементам дерева состояний (определенному состоянию).

ngrx



Effects

- наблюдает за действиями ngrx/store
- изолирует side-эффекты от компонент
- взаимодействует с системой вне Angular
- позволяет совершать асинхронные действия



DEMO TIME



Полезные ссылки

- Курс от Todd Motto по ngrx <https://ultimateangular.com/ngrx-store-effects>
- Статьи по ngrx:
 - <https://toddmotto.com/ngrx/>
 - <http://onehungrymind.com/build-better-angular-2-application-redux-ngrx/>
 - <https://gist.github.com/btroncone/a6e4347326749f938510>



Спасибо :)