<TeachMeSkills/>

45. Redux-middleware

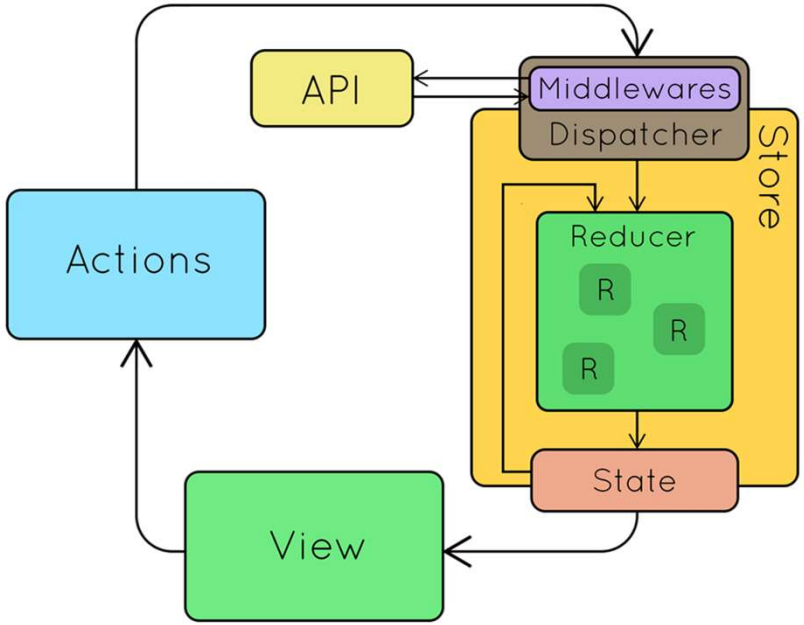
Цель:

Познакомиться с понятием redux-middleware:

- архитектура приложения с асинхронными операциями
- что такое middleware

Middleware:

Архитектура приложения с асинхронными операциями



Redux-middleware:

- redux-saga
- redux thunk

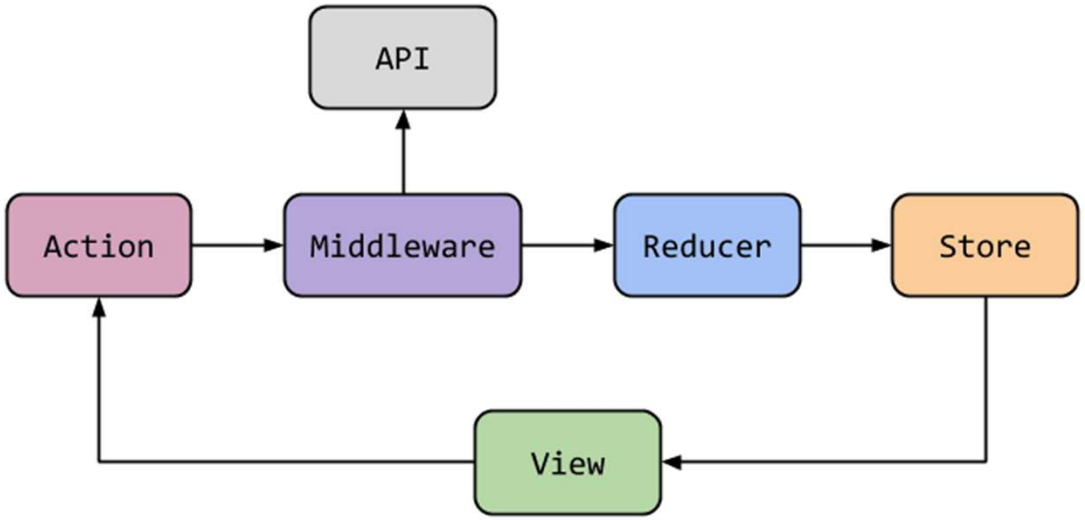
Redux-middleware:

Redux-middleware используют для:

- логирования
- обработки ошибок
- общения с асинхронным API и т.д.

Особенностью мидлвара является то, что они компонуемы. Можно объединить несколько мидлваров вместе, где каждый мидлвар будет независимым. Каждый мидлвар не будет знать и влиять на то, что происходит до или после него в цепочке.

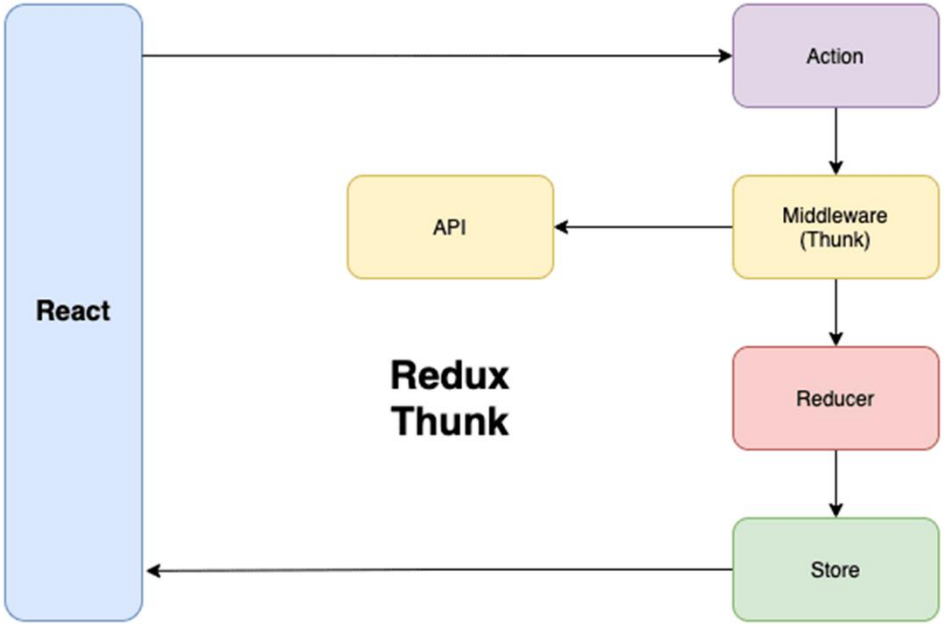
Middleware:



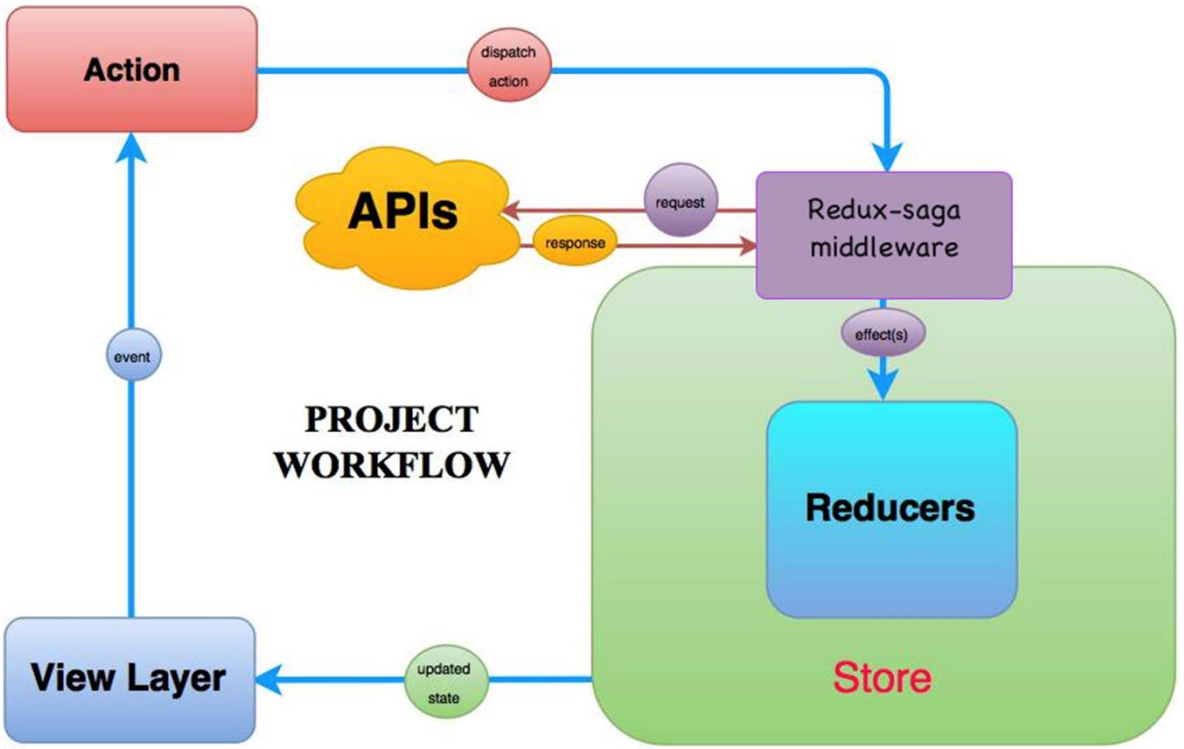
Redux-thunk:

Redux Thunk это middleware библиотека, которая позволяет вам вызвать action creator, возвращая при этом функцию вместо объекта. Функция принимает метод dispatch как аргумент, чтобы после того, как асинхронная операция завершится, использовать его для диспатчинга обычного синхронного экшена, внутри тела функции.

Redux-thunk:


React
Action
API
Middleware
(Thunk)
Redux
Thunk
Reducer
Store

Redux-saga:


Action
dispatch action
APIs
request
response
Redux-saga middleware
effect(s)
event
PROJECT WORKFLOW
Reducers
View Layer
updated state
Store

Генераторы:

Генераторы (Generators) это функции которые могут быть остановлены и продолжены, вместо выполнения всех выражений в один проход.

Когда мы вызываем функцию-генератор, она возвращает объект-итератор. И с каждым вызовом метода итератора next() тело функции-генератора будет выполняться до следующего yield выражения и затем останавливаться.

Генераторы:

```
> var foo,
      f;
  foo = function * () {
      console.debug('generator 1');
      console.debug('yield 1', yield 'A');
      console.debug('generator 2');
      console.debug('yield 2', yield 'B');
      console.debug('generator 3');
  }
  f = foo();

  console.log('tick 1');
  console.log(f.next('a'));
  console.log('tick 2');
  console.log(f.next('b'));
  console.log('tick 3');
  console.log(f.next('c'));
  console.log('tick 4');
  console.log(f.next('d'));
```

Saga helpers:

takeEvery — позволяет одновременно запускать несколько тасков одновременно.

takeLatest — позволяет получать ответ только от последнего запроса (например, чтобы всегда показывать последнюю версию данных).

Call - выполняет функцию. Если он возвращает promise, то приостанавливает сагу до тех пор, пока promise не вызовет resolve.

Put — диспатчит экшн.

Saga helpers: