Saga

1. <https://www.imooc.com/article/29441>

redux-saga常用方法解释

<https://blog.csdn.net/jiangbo_phd/article/details/54026751>(例子可能有问题，没太看懂)

redux Effects

Effect 是一个 javascript 对象，可以通过 yield 传达给 sagaMiddleware 进行执行在， 如果我们应用redux-saga，所有的 Effect 都必须被 yield 才会执行。

1. 问题

没有用到cancel啊，而且这个fork的参数为啥这么奇怪呢



官网

指南

1. 你可以很容易地测试异步流程并保持你的 action 是干净的。
2. takeEvery 允许并发 即：同时处理多个相同的 action
3. takeLatest 不允许并发 即：如果在这之前已经有一个 `USER\_FETCH\_REQUESTED` action 在处理中，

那么处理中的 action 会被取消，只会执行当前的

1. 为了运行我们的 Saga，我们需要：

创建一个 Saga middleware 和要运行的 Sagas（目前我们只有一个 helloSaga）

将这个 Saga middleware 连接至 Redux store.

注意这里不是仅仅引入中间件就可以了，区别于trunk



首先我们引入 ./sagas 模块中的 Saga。然后使用 redux-saga 模块的 createSagaMiddleware 工厂函数来创建一个 Saga middleware。

运行 helloSaga 之前，我们必须使用 applyMiddleware 将 middleware 连接至 Store。然后使用 sagaMiddleware.run(helloSaga) 运行 Saga。

5.saga:

非官网（为了快速上手）

<https://www.jianshu.com/p/6f96bdaaea22>

1.

redux-saga是一个用于管理redux应用异步操作的中间件，redux-saga通过创建sagas将所有异步操作逻辑收集在一个地方集中处理，可以用来代替redux-thunk中间件

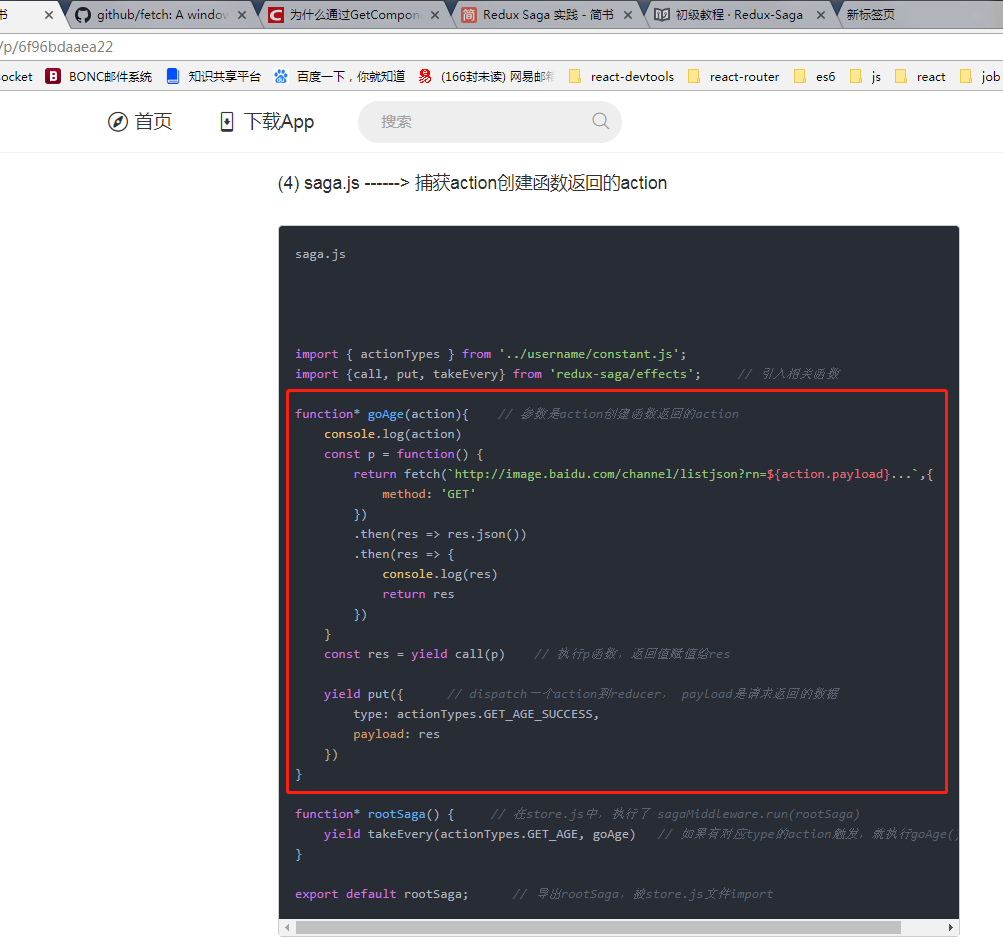
2. 这意味着应用的逻辑会存在两个地方

(1) reducer负责处理action的state更新

(2) sagas负责协调那些复杂或者异步的操作

1. 这个函数就是saga的精华：

注意：put call 都是在这个generator函数里实现的



1. 注意这个函数的意思



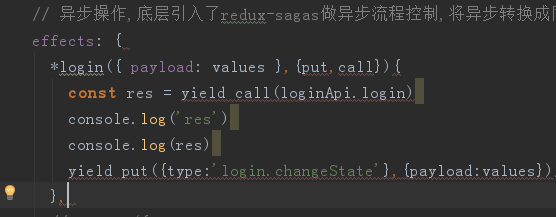
5. call(fn, ...args) 阻塞执行，call()执行完，才会往下执行 执行 fn(...args)

对比 fork(fn, ...args) 无阻塞执行



Call的形式：





6. take(pattern)

即事件监听