Redux学习摘要

学习网址：<http://cn.redux.js.org/docs/introduction/ThreePrinciples.html>

Redux 是 JavaScript 状态容器，提供可预测化的状态管理。

Redux 除了和 [React](https://facebook.github.io/react/) 一起用外，还支持其它界面库。  
它体小精悍（只有2kB）且没有任何依赖

应用中所有的 state 都以一个对象树的形式储存在一个单一的 store 中。  
惟一改变 state 的办法是触发 action，一个描述发生什么的对象。  
为了描述 action 如何改变 state 树，你需要编写 reducers。

Flux和redux的区别

1. 而不同于 Flux ，**Redux 并没有 dispatcher 的概念**。原因是它依赖纯函数来替代事件处理器。
2. 是 **Redux 设想你永远不会变动你的数据**。你可以很好地使用普通对象和数组来管理 state ，而不是在多个 reducer 里变动数据。

**三大原则**

### 一、单一数据源

**整个应用的**[**state**](http://cn.redux.js.org/docs/Glossary.html#state)**被储存在一棵 object tree 中，并且这个 object tree 只存在于唯一一个**[**store**](http://cn.redux.js.org/docs/Glossary.html#store)**中。**

### 二、State 是只读的

**惟一改变 state 的方法就是触发**[**action**](http://cn.redux.js.org/docs/Glossary.html#action)**，action 是一个用于描述已发生事件的普通对象。**

### **三、**使用纯函数来执行修改

**为了描述 action 如何改变 state tree ，你需要编写**[**reducers**](http://cn.redux.js.org/docs/Glossary.html#reducer)**。**

**Action**

**Action** 是把数据从应用传到 store 的有效载荷。它是 store 数据的**唯一**来源。一般来说你会通过[store.dispatch()](http://cn.redux.js.org/docs/api/Store.html#dispatch) 将 action 传到 store。

Action 本质上是 JavaScript 普通对象。我们约定，action 内必须使用一个字符串类型的 type 字段来表示将要执行的动作。多数情况下，type 会被定义成字符串常量。当应用规模越来越大时，建议使用单独的模块或文件来存放 action。

**Action 创建函数** 就是生成 action 的方法。

function firstAction (test){

return {

type: ‘first click’,

text:test

}

}

Action 创建函数也可以是异步非纯函数。

reducer 就是一个纯函数，接收旧的 state 和 action，返回新的 state。

state 为 undefined，此时我们可借机设置并返回应用的初始 state。

reducer 纯净非常重要。**永远不要**在 reducer 里做这些操作：

* 修改传入参数；
* 执行有副作用的操作，如 API 请求和路由跳转；
* 调用非纯函数，如 Date.now() 或 Math.random()。
* **不要修改 state。** 使用 [Object.assign()](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Object/assign" \t "_blank) 新建了一个副本。不能这样使用 Object.assign(state, {visibilityFilter: action.filter })，因为它会改变第一个参数的值。你**必须**把第一个参数设置为空对象。你也可以开启对ES7提案[对象展开运算符](http://cn.redux.js.org/docs/recipes/UsingObjectSpreadOperator.html)的支持, 从而使用 { ...state, ...newState } 达到相同的目的。
* **在 default 情况下返回旧的 state。**遇到未知的 action 时，一定要返回旧的 state。

[combineReducers()](http://cn.redux.js.org/docs/api/combineReducers.html) 所做的只是生成一个函数，这个函数来调用你的一系列 reducer，每个 reducer **根据它们的 key 来筛选出 state 中的一部分数据并处理**，然后这个生成的函数再将所有 reducer 的结果合并成一个大的对象。

# Store

Store 就是用来维持应用所有的 [state 树](http://cn.redux.js.org/docs/Glossary.html#state) 的一个对象。 改变 store 内 state 的惟一途径是对它 dispatch 一个[action](http://cn.redux.js.org/docs/Glossary.html#action)。

Store 不是类。它只是有几个方法的对象。 要创建它，只需要把根部的 [reducing 函数](http://cn.redux.js.org/docs/Glossary.html#reducer) 传递给[createStore](http://cn.redux.js.org/docs/api/createStore.html)。

* 提供 [getState()](http://cn.redux.js.org/docs/api/Store.html" \l "getState) 方法获取 state；
* 提供 [dispatch(action)](http://cn.redux.js.org/docs/api/Store.html#dispatch) 方法更新 state；
* 通过 [subscribe(listener)](http://cn.redux.js.org/docs/api/Store.html#subscribe) 注册监听器;
* 通过 [subscribe(listener)](http://cn.redux.js.org/docs/api/Store.html#subscribe) 返回的函数注销监听器。

### [getState()](http://cn.redux.js.org/docs/api/Store.html#getState)

返回应用当前的 state 树。  
它与 store 的最后一个 reducer 返回值相同。

谨记

Redux只有一个store

永远不要更改reducer里原有的state，当 state 变化时需要返回全新的对象，而不是修改传入的参数。

Action中至少有一个type

改变 state 的惟一方法是 [dispatch](http://cn.redux.js.org/docs/api/Store.html#dispatch) action

# Middleware

**提供的是位于 action 被发起之后，到达 reducer 之前的扩展点。** 你可以利用 Redux middleware 来进行日志记录、创建崩溃报告、调用异步接口或者路由等等。