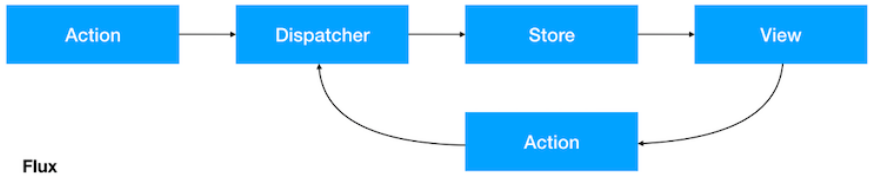
一、什么是Flux

Flux 是一种架构思想，专门解决软件的结构问题。它跟MVC 架构是同一类东西，但是更加简单和清晰。
Flux存在多种实现（至少15种），redux是其中一种。

二、Flux基本概念

首先，Flux将一个应用分成四个部分。

```
View： 视图层
Action（动作）：视图层发出的消息（比如mouseClick）
Dispatcher（派发器）：用来接收Actions、执行回调函数
Store（数据层）：用来存放应用的状态，一旦发生变动，就提醒Views要更新页面
```



Flux 的最大特点，就是数据的"单向流动"。

```
用户访问 View
View 发出用户的 Action
Dispatcher 收到 Action，要求 Store 进行相应的更新
Store 更新后，发出一个"change"事件
View 收到"change"事件后，更新页面
```

上面过程中，数据总是"单向流动"，任何相邻的部分都不会发生数据的"双向流动"。这保证了流程的清晰。

三、Redux

React 只是 DOM 的一个抽象层，并不是 Web 应用的完整解决方案。有两个方面，它没涉及。

```
代码结构
组件之间的通信
```

2013年 Facebook 提出了 Flux 架构的思想，引发了很多的实现。2015年，Redux 出现，将 Flux 与函数式编程结合一起，很短时间内就成为了最热门的前端架构。

但是，值得注意的是，Redux并非必须

```
如果你不知道是否需要 Redux，那就是不需要它
```

只有遇到 React 实在解决不了的问题，你才需要 Redux
简单说，如果你的UI层非常简单，没有很多互动，Redux 就是不必要的，用了反而增加复杂性。

不需要使用Redux的项目：

- 用户的使用方式非常简单
- 用户之间没有协作
- 不需要与服务器大量交互，也没有使用 WebSocket

- 视图层（View）只从单一来源获取数据

需要使用Redux的项目：

- 用户的使用方式复杂
- 不同身份的用户有不同的使用方式（比如普通用户和管理员）
- 多个用户之间可以协作
- 与服务器大量交互，或者使用了WebSocket
- View要从多个来源获取数据

从组件层面考虑，什么样子的需要Redux：

- 某个组件的状态，需要共享
- 某个状态需要在任何地方都可以拿到
- 一个组件需要改变全局状态
- 一个组件需要改变另一个组件的状态

Redux的设计思想：

- Web 应用是一个状态机，视图与状态是一一对应的。
- 所有的状态，保存在一个对象里面（唯一数据源）。

注意：flux、redux都不是必须和react搭配使用的，因为flux和redux是完整的架构，在学习react的时候，只是将react的组件作为redux中的视图层去使用了。

Redux的使用的三大原则：

1. Single Source of Truth(唯一的数据源)
2. State is read-only(状态是只读的)
3. Changes are made with pure function(数据的改变必须通过纯函数完成)

四、Redux使用流程