React

React 是当前最火的前端框架，另外两个火的是 Vue 和 Angular。

关键词一：虚拟 dom

React 使用虚拟 dom，这一点和 vue 是一样的，何谓虚拟 dom 呢？也就是说 react 完全使用自己的一套机制，名为 virtual dom，然后把它映射到最终渲染层上。在此期间，一切的变化，通过一套 diff 算法，先改变 virtual dom，再把改变呈现到最终渲染层。注意，不存在什么真实 dom 到虚拟 dom 的映射（很多人还在纠结这一点）。只有虚拟 dom 映射到真实 dom，变化是虚拟 dom 的变化，操作也在虚拟 dom 上。理解这一点。

关键词二：MVVM

React 是 MVVM 吗？ 并不算是严格意义上的 MVVM 架构，Angular 才是。React 只是单向数据绑定，这点和 Vue 是一样的。不过这并不是说 React 没有借鉴 MVVM 的思想。与之前的架构相比，如 MVC 和你们所用到的 MVP，你需要同时关心两件事情：

1. 数据更新和数据更新的动作
2. 界面更新和界面更新的动作

但是如果是 React，你只需要关心一件事：

1. 数据更新和数据更新的动作

怎么说呢？总之就是编程思维上的转变吧，多敲代码很快就适应了。

关键词三：ReactNative

怎么说呢？

如果是纯 React，只用来写前端,React 那可以说是相当舒服，JSX 和 HTML 模板可以说近乎无缝对接了。什么 div，img，css 预编译，webpack 打包啊，直接写的飞起。

那现在如果换到 react-native，事情并没有想象的那么单纯。前端的 h5 标签几乎用不了，不支持直接使用 css（sass、less 党们表示很绝望）。你享受不到在前端语言里放飞自我的感觉。只能用一些 react-native 里，他们已经定义好的 View、Text 这些之类的标签。

你可能在想，为什么 React 不搞一个大一统呢？反正它最后都是要渲染到浏览器或原生的 Android 或 iOS 上。如果我做的话（作为一个前端码农），那我就让 ReactNative 内部实现一套机制，div 对应 View，img 对应 Image，以此类推，把 css 属性映射到原生 View 属性上去。。可惜 React 并没有这么做，现在我也没发现有第三方库做出相关的工作（有把 RN 映射到 Web 端的库，但是感觉违背 React 的初衷了）。ReactNative 之所以搞出这些标签，一方面是实现所有 html/css 渲染这件事确实复杂，当然不是说 facebook 没这实力，它也是为了照顾其他厂商的面子，你想我都做到这程度，不相当于实现一套浏览器了吗？是想置那些浏览器厂商于何地（Microsoft、Firefox、Google 表示有话说）？另一方面，一定意义上是想让一些原生应用开发者，也加入到 React 的阵营。通过 ReactNative，领略到 React 的强大，从此入坑 React。不得不说 facebook 还是有些心机，你看它开源的协议，BSD，这个协议是说，你的产品只要用了它的库，它就可以使用你在产品中所使用的所有专利。

这里还有个故事，facebook 之前把它产品线下的所有开源库都改成了 BSD 协议，后来遭到开发者和社区的疯狂吐槽。facebook 因此不得不服软，最终同意把 React、Jest、Flow、Immutable.js 这四个库改回 MIT license。但是很可惜，直到现在，ReactNative 依然是 BSD 协议。我们的产品涉及到了核心专利了吗？会被 facebook 盯上使用吗？是否需要权衡一下呢？

关键词四：生态

React 的生态可谓爆炸，各种第三方库层出不穷。刚才说了，我们只关心数据，其它的 React 帮我们做好。所以，React 的库，很多都是管理 React 数据的库。其中，最出名的就是 Redux。而在 Redux 和 React 之间，还有各种各样的第三方中间件库。

其他废话不多说，就说说 Redux 好在哪？

Redux 好就好在它易于管理。

让我们回顾一个需求，假如A页面要向B页面传值或者由A页面触发一个B页面的事件（两者是非嵌套组件，或者说没有引用关系），该如何处理？我估计你们要扯上异常高大上的RxJava（RxBus）,或者曾经被你们用过然后又被你们鄙视的EventBus（原生handler哭了。。），事件订阅总能解决问题。但是呢，你怎么去追踪这些玩意儿？至少在我做Android开发那会儿，貌似没有特别好的办法。无非打个Tag，定义一个标志位之类。等到万一出bug了，调试的时候还不好调，这点尤为无语。

而到了Redux这里，一切都变得简单，Redux眼里，一切皆状态，一切状态皆由行为触发，并且都是可以追踪的。

Redux保持单一数据源，并且状态由全局store统一管理。所有的变化，不管是你是什么事件，什么数据，无论什么变化，都通过Action去触发，也只能由Action触发产生相应的结果（这里说的有点绝对，但是Redux是推荐这么做的）。

全局有啥好？不相当于js把变量全挂在window下？额，这个，先不说这是不是一回事儿（这个本来就没什么好说的）。骚年，你只需要知道，这个世界永远是在变化的。比如曾经在对象里写尾逗号是不提倡的，甚至是嗤之以鼻的，但现在ES6里又提倡写尾逗号了。这就跟时尚潮流一样，是一个道理。说全局不好的，想想后台怎么写的。Redux只是把后台管理那套移到前端而已。

回到刚才的话题，A页面要跟B页面交互，只需向Redux 请求一个行为而已。而这个行为或与之对应的状态都是需要之前就在Redux的store里定义好。这一切都是可以通过调试工具追踪的，并且行为和状态变化的过程甚至可以回溯，不会放过任何可能造成变化的点，外界美其名曰“时间旅行”。

说道Redux的状态，这里还得在提一下，你可以看到整个应用完整的状态树。之前那些重复数据、重复方法balabala之类的，在这里无形可遁。至少我一眼就可以看出来整个应用哪些数据定义的不合理。

Redux另一个值得称道的地方就是，它简单，但扩展性强。它可以组合各种中间件，完成你想要的任何工作，再怎么复杂也可以搞定。你想优雅也可以，脏一点也行，随你自己喜欢。Redux即使不好，也绝对好用了。