大前端浪潮下的新前端

分享主要分 4 个部分

第一部分：传统前端的缺点

概述：众所周知，Web 前端主要围绕着 HTML / CSS / JavaScript 展开技术栈的。依靠着这三种语言，早期的 Web 前端架构实现了 MVC。但是随着 Web 前端的发展，用户交互程度越发复杂，前端的作用域被逐渐放大，各种缺点开始显现出来 ...

作用：这一部分主要跟大家讲述一下以前前端开发使用的是哪些工具，工程管理是怎么样的，有哪些缺点，从而抛出第二部分。

第二部分：传统前端迈向新前端改善了哪些东西?

概述：通过第一部分讲述到的缺点，发现前端存在着工程化不高，代码管理困难等问题。Web 前端的飞速发展，也使得标准一步步地完善，浏览器之间的斗争越发激烈。随后，出现了性能异常优越的 V8 引擎，使得 JS 像 Java 依赖 JVM 一样脱离浏览器跨平台运行（Node.js）。Node 的出现使得 JS 的技术栈发生了天翻地覆的变化...

作用：这一部分讲述前端的作用域开始拓展放大，传统应用的架构也开始发生变化，分工更加明确。引出第三部分 MVVM 框架的内容。

第三部分：三大数据驱动型 MVVM 框架的驱动原理剖析与性能差异

概述：从 Angular.js 开始，前端进入了 MVVM 时代，使得 `数据控制` - `交互控制` - `视图展示` 能准确分离，极大改善了前端代码的可读性和拓展性。目前的 MVVM 框架多多少少都有 Angular.js 的影子。驱动原理上来说：Angular.js 使用的是脏检查机制、Vue.js 使用的是依赖收集机制、React 使用的 VDom Diff Patch 机制。不同的实现造成了渲染性能上的差异...

作用：这一部分会以图形化加上源码阅读去理解数据驱动的原理，还实现了一个小型的双向绑定 demo （在 `example/vue.js` 目录下）。讲述完原理之后，会有一个对于三大框架性能的测试，在 `framework-test` 目录下，对 Angular.js、Angular 5、React、Vue 进行了性能测试比较。

第四部分：DOM 抽象思想对原生开发的影响

概述：现在大部分的前端框架都对 DOM 对象进行了抽象，其中最典型的是 React，Vue 是后来才使用了 VDom。Dom 对象的虚拟化，使得对用户界面的描述可以通过对象来抽象表示。发挥一下脑洞，只要实现了不同平台的特殊渲染层，就可以用一系列对界面的描述对象来实现跨平台界面显示，于是出现了 Weex 和 React Native 甚至是 Native Script 之类的跨平台开发框架。

作用：目前技术创新这边在推 Creator 的方案，时间如果允许的话读一下 Weex 的源码