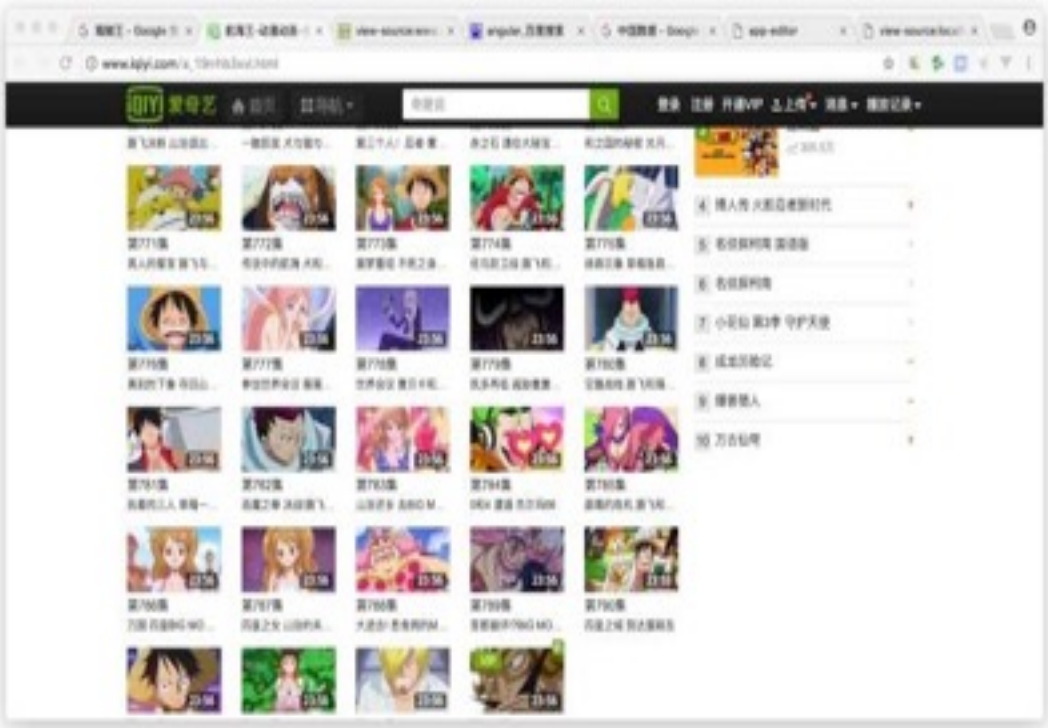
此次呢，就由我给大家分享前端的一个概念。此次的分享是想让大家对整体的web这块有一个全面的了解，后面大家可以针对自己的兴趣或者自己的喜好来进行针对性的研究 。

那我们就说说web前端到底有什么吧 ，简单点来说，就是html css还有javascript。一个网页，基本上都是由这三点构成。而前端所做的最重要的事情，就是将这三点去更好，更加方便，更加好维护的方式去展现出来。在10年之前，还是没有前端这个概念，为什么，因为那个时候，前端的职责很少，少到一般的ui设计师借助像dw这样的软件也能够胜任 。简单来说，就是前端的职责就是把原型图用html复现出来，我们俗称叫做切图。就是用上面所说的html css js去完成一个网页，一般ui也能够用dw画出页面，然后去网上找一些jquery插件拼拼凑凑也能够完成。所以前端的职责就是写好页面，后端的同学呢，拿到前端的页面，然后工作的核心内容的就被拿到后端了。由后端完成整个web项目的业务逻辑。想想看，我们所做的大部分的都是动态网页吧。动态的意思不是页面上的动画效果，而是里面的内容很多都是从后端获取到，然后去放在页面里面去输出的。

一般的流程是这样的：后端拿到数据库里面的内容，后端去将这些内容 + 前端给的静态页面拼凑在一起，然后将这些拼凑好的内容去发送给客户端让浏览器去解析这部分内容。如下图：



如上图,这些动画的内容包括，旁边的推荐导航肯定是从后端拿过来的，肯定不是个静态的，如果这样的话，每次改动都要去改页面代码，那岂不是要有无数个页面代码？这些个东西都是通过查询后端数据库里面的内容，然后在后端完成拼凑而得到了，其本质上，只有一个页面。

那么我们从这个点入手，后端同学拿到的页面，然后在后端把这页面去输出出来，这样的工程在以前是很好的解决方案，但是这就导致了一个问题，就是后端里面夹杂着前端的代码，这样在后期维护的时候会很麻烦，你想想嘛，一团乱麻怎么改嘛。工程师们就想，那么我们把这个做个分离吧，不让前后端夹杂在一起。就像我们在写页面的时候，以前都是htmlcssjs混在一个页面里面，后来把 html css js做了分离 ，js和css都用标签导入。那么前后端分离也就是这么个概念。前端的同事写的东西，不希望要夹杂在后端里面，后端的同学呢，也只想着做好后端的接口 ，然后前端需要什么，你自己去用这个接口去调就是了，大家互不干扰。这样对大家都好。

到这里，我没有说到任何的框架，任何的类库，前后端的分离，根本上来说就不是语言的变化，而是将很多由本来可以在前端完成的事情，原本是放在服务端完成的，为什么当时是由服务端完成的？

是因为服务端有良好的模块加载机制。什么意思呢 。

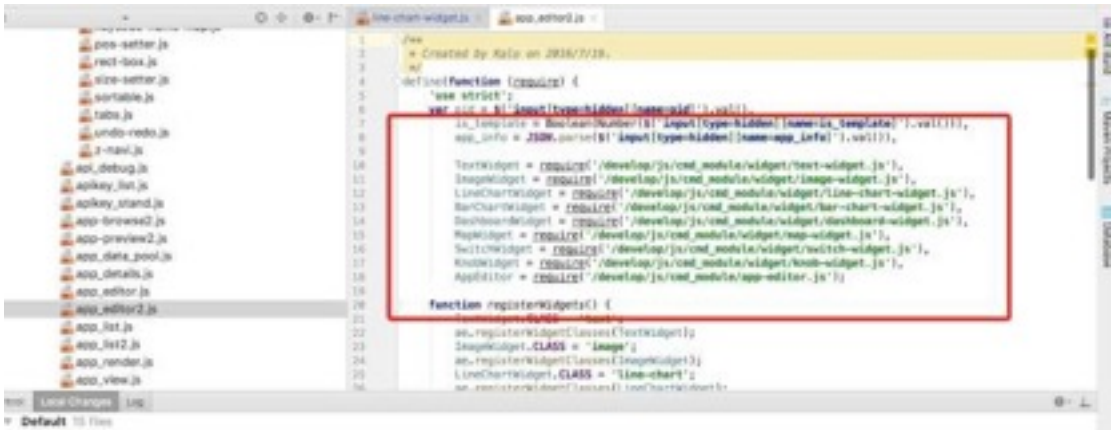
比方说，我们前端在使用一个库的时候，都是用script标签去引入js，根本就没法去组织这些js。举个简单的例子，你想引入jquery，jquery是一个操作dom非常方便的一个js库，然后又想引入一个jquery的一个插件，然后又想引入一个插件，这样一个接一个的，页面中就出现了各种各样的script标签，很不方便管理，再一个，就是各种库之间的依赖关系，比如jquery的插件是依赖的jquery的，无法很好的处理这些依赖，在做大型项目的时候，很容易就报错。

就是这样的，页面中引入大量的js文件，项目大，当做到应用级别的，可就不值这么多了

那模块加载机制是什么呢？

那么官方提供两种规范，我说的是规范，AMD规范和CMD规范，就是模块和模块之间的输入输出。比方说，我们在引入jquery之后为什么能在全局使用jquery，是因为在jquery里面，它会将jquery注册在window上面，变成全局的了，所以才能使用，那么全局都可以使用，显然不是按需索取，那么AMD和CMD规范就是用来规范如何去实现这样的规范，那衍生出来的框架比如requireJS seaJS。这是当时前端模块化的先驱了。

我就拿我以前的项目截个图举个例子



上图中我们用require去导入的，就是是各种功能的模块 。可以这么说，没有模块的这种组织方式，就没有应用级别的产品产生，应用级别的业务逻辑是很复杂的。那有了模块化，那么我们的前端项目就有了做大做复杂的基础了。

但是require还有sea这些当时的模块加载机制还是在前端完成的，什么意思？意思就是说，在页面中需要引入seaJS和requireJS是必须的，虽然在ES2015发布的es6中已经将import和exprot纳为了规范，但是就现在浏览器大部分还是不支持的。

图里面，上面是几个全局的库 ，剩下就是一个seajs的模块的函数库 ，整个应用的借口，就一个 seajs.use("develop/js/app_editor2")，也就是说还是要引入seajs这个文件 。

那如何解决现在前端模块的加载的这些弊端呢？

那就出现在后端进行打包这个概念。什么意思？意思就是，既然我在浏览器中无法直接实现模块的加载机制，我拿到后端去间接的做这些事情，最终输出一个浏览器能直接识别的js或者其他各式各样的文件，就是说原本需要引入seajs和requirejs这写的作用，我拿到后端去完成。比如说webpack。离开了浏览器的宿主环境，可以做的事情就很多了。webpack是什么呢，就是一个打包工具，也

是实现模块化的一个方案。那模块化也有了，前端好像没缺什么了，大家就开始做开发呗。做着做着，厉害的工程师们在大量的代码中又抽象出一个概念，叫做mvvm model view model- view 这样的一种模式。是将对js在浏览器中的dom操作进行一次分层，还是和html css js分层一样，这些分层就是更方便的维护与开发，让开发者们都关注的层面更加的清晰。这个就偏向与要有一部分经验的人来说了，因为每次的动态交互，都会存在一个数据的变化，这个变化呢，有可能导致其他部件的状态变化。所以之前我们都是再处理之后，再去初始化化一下rander层，重新渲染一遍 。现在有了vue react angular，这样的，我们只要专注于数据层就好了，渲染层面，就不要我们去考虑了。本质上这类框架都是为了再将js对dom的操作再进行一次分离 ，

当然大家也还是比较关心nodejs，那么node是什么？node不是一门语言，是一个运行环境。什么意思呢 ，js使用时借助宿主环境的，不信你可以试试写一个js文件，然后是没法去执行的，以前js的宿主环境在我们脑海里只有浏览器，当然还有其他比较小众的地方。那么也就是说js在我们的意识里面职能完成页面的事情，其实就是这么个道理。js是一门语言，而那些dom操作，就是宿主给js提供的各种各样的api。

而node呢，就是给了js很多在后端的api。这样大家就理清了这些大部分的概念了吧。

js + 后端api = = nodejs

js + 浏览器的api = = 传统js

这里的js就是js核心的东西，语言的语法，变量，数据类型，oop。当然现在随着前端圈子的发展，浮躁之气是难免的，百花齐放嘛，当然有这个忧虑，就是各种层出不穷的东西，学着这个，又有新的东西要学，所以做技术的，还是沉下心来，把基础打牢吧 。html css js，这三个内容你精通一个就能成为业界大师，但是大部分都是熟悉会用，仿佛就以为精通 。