## **行业知识总结**

## **CSS**

### **1. CSS基础知识**

* 层叠和继承
* 优先级
* 盒模型
* 定位
* 浮动

### **2. CSS进阶**

* CSS Sprite
* 浏览器兼容性
* IE HasLayout和Block Format Content
* CSS Frameworks
* CSS3
* CSS性能优化
* LESS and SASS

### **3.CSS Sprite**

CSS Sprite主要用于前端性能优化的一种技术，原理是通过将多张背景图片合成在一张图片上从而减少HTTP请求，加快载入速度。

### **4.浏览器兼容性**

绝大部分情况下我们需要考虑浏览器的兼容性，目前正在使用的浏览器版本非常多，IE6, IE7, IE8, IE9, IE10, Chrome, Firefox, Safari。

### **5.IE HasLayout和Block Format Content**

IE HasLayout是一个 Internet Explorer for Windows的私有概念，它决定了一个元素如何显示以及约束其包含的内容、如何与其他元素交互和建立联系、如何响应和传递应用程序事件、用户事件等。这种渲染特性可以通过某些 CSS 属性被不可逆转地触发。而有些 HTML 元素则默认就具有”layout”。目前只有IE6和IE7有这个概率。BFC是 W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行定位，以及与其他元素的关系和相互作用。这个其实和浏览器的兼容性有关，因为绝大部分的兼容性问题都是它们引起的。参考：[CSS BFC和IE Haslayout介绍](http://blog.csdn.net/borishuai/article/details/8127758" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **6.CSS Framework**

CSS框架是一系列CSS文件的集合体，包含了基本的元素重置，页面排版、网格布局、表单样式、通用规则等代码块,用于简化web前端开发的工作，提高工作效率。目前常见框架有：

* [960 Grid System](http://960.gs/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Blueprint CSS](http://code.google.com/p/blueprintcss/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Bluetrip](http://bluetrip.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Minimum Page](http://www.minimumpage.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)

还是一个比较出名和特殊的框架是Twitter的[Bootstrap](https://github.com/twitter" \t "http://www.cnblogs.com/doit8791/p/_blank)。Bootstrap是快速开发Web应用程序的前端工具包。它是一个CSS和HTML的集合，它使用了最新的浏览器技术，给你的Web开发提供了时尚的版式，表单，buttons，表格，网格系统等等。它是基于Less开发的。不支持IE6，在IE7和IE8里效果也不咋地。

### **7.CSS3**

虽然CSS3还没有正式成为标准，但是包括IE9+, chrome, Firefox等现代浏览器都支持CSS3。CSS提供了好多以前需要用JavaScript和切图才能搞定的功能，目前主要功能有：

* 圆角
* 多背景
* @font-face
* 动画与渐变
* 渐变色
* Box阴影
* RGBa-加入透明色
* 文字阴影

### **8.CSS性能优化**

CSS 代码是控制页面显示样式与效果的最直接“工具”，但是在性能调优时他们通常被 Web 开发工程师所忽略，而事实上不规范的 CSS 会对页面渲染的效率有严重影响，尤其是对于结构复杂的 Web 2.0 页面，这种影响更是不可磨灭。所以，写出规范的、高性能的 CSS 代码会极大的提高应用程序的效率。参考[CSS性能优化探讨](http://blog.csdn.net/borishuai/article/details/8671929" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **9.LESS和SASS**

[LESS](http://lesscss.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)和[SASS](http://sass-lang.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)都是 CSS 预处理器，用来为 CSS 增加一些编程的的特性，无需考虑浏览器的兼容性问题，例如你可以在 CSS 中使用变量、简单的程序逻辑、函数等等在编程语言中的一些基本技巧，可以让你的 CSS 更见简洁，适应性更强，代码更直观等诸多好处。

[SASS](http://sass-lang.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)基于Ruby开发。[LESS](http://lesscss.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)既可以在客户端运行，也可以借助Node.js或者Rhino在服务端运行。

## **JavaScript**

### **10. JavaScript基础知识**

* 数据类型
* 变量
* 表达式与运算符
* 控制语句
* 函数
* 异常
* OO
* 事件
* BOM
* 闭包

### **11. JavaScript进阶**

* DOM
* JSON
* AJAX
* JavaScript Frameworks
* HTML5
* 前端模板
* 前端MVC
* 模块化开发
* JavaScript单元测试
* JavaScript设计模式
* NodeJS
* ES5

### **12.DOM**

DOM即文档对象模型，HTML DOM 定义了访问和操作HTML文档的标准方法。几乎所有的现代浏览器都能很好的支持DOM了。

### **13.JSON**

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写，同时也易于机器解析和生成。是目前事实上数据交换的标准格式，几乎所有语言都支持JSON，比XML强太多了。

### **14.AJAX**

AJAX即“Asynchronous JavaScript and XML”（异步JavaScript和XML)，AJAX并非缩写词，而是由Jesse James Gaiiett创造的名词，由Google发扬光大。用于创建更好更快以及交互性更强的 Web 应用程序的技术。

### **15.JavaScript Frameworks**

JavaScript Frameworks可以极大简化我们JavaScript编程的工作量，它主要提供了以下几个主要功能： DOM操作，跨浏览器兼容性，以及程序架构。当然像jQuery它本身其实并不是一个框架，它是一个库(lib)。目前主流的框架或库有如下几个：

* [jQuery](http://www.jquery.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [YUI](http://developer.yahoo.com/yui/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [DOJO](http://dojotoolkit.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Mootoolos](http://mootools.net/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [ExtJS](http://blog.csdn.net/borishuai/article/details/www.sencha.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Prototype](http://prototypejs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)

以上都是一些重量级的框架或者库，还是小巧的库也是强烈推荐的，比如[Underscore](http://documentcloud.github.com/underscore/" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **16.HTML5**

HTML5同CSS3类似，即虽然没有成为标准，但是主流的浏览器都支持了。HTML5不是HMTL，虽然也提供了一些新标签，但是它的主要用途还是JavaScript。HTML5主要提供以下功能：

* 本地音频视频播放
* Canvas/SVG
* 地理信息
* 硬件加速
* 本地运行
* 本地存储
* 从桌面拖放文件到浏览器上传
* 语义化标签，Form表单

### **17.前端模板**

前端模板主要是为了解决复杂的数据拼接问题，可以将模板语言转换化为HTML结构，可以大大简化工作量，同时代码的可维护性得到很大的提高。目前比较主流前端模板有：

* [MustCache](http://mustache.github.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [JsRender](https://github.com/BorisMoore/jsrender" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **18.前端MVC**

Web应用的功能越来越强，Javascript代码也越来越多，大量的JS代码要以何种架构来组织就成了一个亟待解决的问题，于是就有人把传统的MVC架构移植到前端来解决这些问题。目前主流前端MVC框架主要有以下这些：

* [Backbone.js](http://backbonejs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Spine](http://spinejs.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [KnockoutJS](http://knockoutjs.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [YUI](http://yuilibrary.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Agility.js](http://agilityjs.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Ember.js](http://emberjs.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Batman.js](http://batmanjs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [AngularJS(Google)](http://angularjs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Dojo](http://dojotoolkit.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)

[TodoMVC](http://addyosmani.github.com/todomvc/" \t "http://www.cnblogs.com/doit8791/p/_blank)用上面所有的MVC框架写了同一个示例代码“Todo List”，是个学习对比MVC框架的好地方。

### **19.模块化开发**

其实现在JavaScript模块化是个很热门的东西了，主要特点是“模块化开发，按需加载“。这其中[CommonJS](http://commonjs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)组织定义了[AMD](https://github.com/amdjs/amdjs-api/wiki/AMD" \t "http://www.cnblogs.com/doit8791/p/_blank)的规范用来规范浏览器端的模块定义。[RequireJS](http://requirejs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)和[SeaJS](http://seajs.org/docs/" \t "http://www.cnblogs.com/doit8791/p/_blank)是实现了AMD的两个优秀的框架。详见：[http://www.weakweb.com/articles/341.html](http://www.weakweb.com/articles/341.html" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **20.JavaScript单元测试**

但是随着单元测试的普及，尤其是敏捷开发的推动，涌现了许多优秀的JavaScript单元测试框架，见详细列表。所有的这些框架基本上都能对Javascript代码进行很好的测试，当然UI部分的代码测试一样比较麻烦，但是我们可以通过精心构造我们的测试代码来测试部分UI代码。主流的测试框架如下：

* [QUnit](http://qunitjs.com/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [Jasmine](http://pivotal.github.com/jasmine/" \t "http://www.cnblogs.com/doit8791/p/_blank)
* [JsTestDriver](http://code.google.com/p/js-test-driver/" \t "http://www.cnblogs.com/doit8791/p/_blank)

目前jQuery的所有代码都是通过QUnit进行测试的，并且将[测试代码](https://github.com/jquery/jquery/tree/master/test/unit" \t "http://www.cnblogs.com/doit8791/p/_blank)放在Github上了，大家感兴趣可以参考一下。详见：[JavaScript单元测试框架介绍](http://www.weakweb.com/articles/255.html" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **21.JavaScript设计模式**

好多设计模式是可以应用于JavaScript的，比如经常用到的事件处理的观察者模式，因此设计模式是提升编码层次的必学技术。

### **22.NodeJS**

[NodeJS](http://nodejs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)现在是比较火热的，其最大的贡献就是把JavaScript移植到服务器端了，这样前端和后端就可以使用同样的技术，方便统一开发。而且NodeJS是非阻塞调用的，在特定领域性能是非常强劲的。而且这是前端开发人员进军后台开发的好机会，进而前后端统一开发，但又不用去学习其它后台开发语言。

### **23.ES5**

ES5就是ECMAScript 5，也就是最新的JavaScript规范，对之前的JavaScript作了很多改进，增加了好多新的特性，比如JSON，而且现代主流浏览器都开始支持ES5了，还是非常有必要学习一下的。

## **24.Others**

下面是一些和HTML, CSS, JavaScript没有直接关系，但是对于前端开发同样非常重要的一些技术。

* 响应式设计
* Http1.1
* Web移动开发
* 前端安全
* 跨域处理
* 调试工具
* SEO
* A/B test
* 可用性/可访问性
* 前端流程/部署
* 正则表达式
* 编辑器
* 浏览器插件开发
* 浏览器原理

### **25.响应式设计**

伴随着各种智能设备的流行，响应式设计现在是非常火热。以前做网页只要面向PC机的浏览器，页面直接固定宽度就行，比如960px，而现在通过手机的访问量已经超过PC机，并且设备的尺寸多种多样，未来会更多。在这种背景下，网页支持所有设备进行访问是基本要求了，而响应式设计能很好的解决这些问题。

### **26.Http1.1**

HTTP对于前端开发来说还是很重要的，比如最简单的GET，POST方式，Request/Response 头部，状态码等。

### **27.Web移动开发**

现在移动开发非常非常流行了，而开发方式一般是native的方式或者Web方式，作为前端开发人员来说自然是去学习Web移动开发了。PhoneGap是必学的，前端层面的框架如jQueryMobile, Sencha Touch, jQTouch等都是不错的选择。

### **28.前端安全**

随着前端技术的发展,安全问题已经从服务器悄然来到了每一个用户的的面前，盗取用户数据, 制造恶意的可以自我复制的蠕虫代码,让病毒在用户间传播,使服务器当掉. 更有甚者可能会在用户不知觉得情况下,让用户成为攻击者,这绝对不是骇人听闻。富客户端的应用越来越广，前端的安全问题也随之增多。常见的攻击方法有：

* **XSS，跨站脚本攻击(Cross Site Script)**。它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入的恶意html代码会被执行，从而达到恶意用户的特殊目的。
* **CSRF(Cross Site Request Forgery)，跨站点伪造请求**。顾名思义就是 通过伪造连接请求在用户不知情的情况下，让用户以自己的身份来完成攻击者需要达到的一些目的。
* **cookie劫持**，通过获取页面的权限，在页面中写一个简单的到恶意站点的请求，并携带用户的cookie 获取cookie后通过cookie 就可以直以被盗用户的身份登录站点。

### **29.跨域处理**

同源策略规定跨域之间的脚本是隔离的，一个域的脚本不能访问和操作另外一个域的绝大部分属性和方法。所谓的跨域处理就是处于不用域之间的脚步互相调用，目前有很多方法来处理它。

### **30.调试工具**

前端的调试工具很多，比如Firebug，Webkit核心的web inspector, IE的iedeveloper。HTTP相关的fiddler, httpwatch等，还有格式化代码的[jsbeatutifier](http://www.jsbeautifier.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)，它有助于阅读压缩处理过的JavaScript代码。IETester可以模拟所有的IE版本，是调试IE兼容性的好工具。

### **31.SEO**

前端开发人员很多时候还是需要了解搜索引擎优化的。

### **32.A/B test**

A / B测试的核心就是：确定两个元素或版本（A和B）哪个版本更好，你需要同时实验两个版本。最后，选择最好的版本使用。

### **33.可用性/可访问性**

可用性指的是：产品是否容易上手，用户能否完成任务，效率如何，以及这过程中用户的主观感受可好，是从用户的角度来看产品的质量。可用性好意味着产品质量高，是企业的核心竞争力。

可访问性：上网用户中那些视力受损的人，通过屏幕阅读器使用键盘命令将网页的内容读给他们听。以语义化的HTML（结构和表现相分离的HTML）编写的网页文件，就可以让此类用户更容易导航，且网页文件中的重要信息也更有可能被这些用户找到。

可以通过逐步强化你的网站功能，同时对支持性进行测试。运用“渐进增强”和“平稳退化”原则开发网站。

### **34.正则表达式**

估计绝大部分的编程语言都会用到它，当处理字符串时可以极大的简化你的工作。必学啊。

### **35.编辑器**

人人都有自己喜欢的编辑器，从前端的角度看，Eclipse + Aptana, Notepad++, VIM都是不错的选择。我个人比较喜欢用Notepad++,简洁，快。

### **36.前端流程/部署**

当前端项目比较复杂时，我们就应该考虑引入自动构建，自动化部署等技术了。可以使用JSLint来对JavaScript进行语法检查，用CSSLint或CSS Validator检查CSS语法，用JSMin或YUI Compressor对JavaScript代码进行压缩，可以使用JSDoc/YUIdoc进行文档自动化生成，使用Jasmine/JsTestDriver进行自动化单元测试，可以使用Ant/Maven/Make进行自动构建部署。不过伴随着NodeJS的流行，[Grunt](http://www.gruntjs.org/" \t "http://www.cnblogs.com/doit8791/p/_blank)，[Bower](http://bower.io/" \t "http://www.cnblogs.com/doit8791/p/_blank)和[Yeoman](http://yeoman.io/" \t "http://www.cnblogs.com/doit8791/p/_blank)现在几乎是前端最流行的自动化的项目构建工具了。详见[Web前端开发流程自动化](http://blog.csdn.net/borishuai/article/details/8477514" \t "http://www.cnblogs.com/doit8791/p/_blank)

### **37.浏览器插件开发**

浏览器是我们的工作平台，在上面开发插件是很有趣的，很多时候也是很有用的。

### **38.浏览器原理**

前端工作绝大部分都是运行在浏览器上面，所以了解浏览器原理有助于更深入的理解各种技术的原理，工作过程。

## **39.沟通能力**

优秀的前端工程师需要具备良好的沟通能力，因为你的工作与很多人的工作息息相关。在任何情况下，前端工程师至少都要满足下列四类客户的需求。

* 1. 产品经理——这些是负责策划应用程序的一群人。他们能够想象出怎样通过应用程序来满足用户需求，以及怎样通过他们设计的模式赚到钱（但愿如此）。一般来说，这些人追求的是丰富的功能。
* 2. UI设计师——这些人负责应用程序的视觉设计和交互模拟。他们关心的是用户对什么敏感、交互的一贯性以及整体的好用性。他们热衷于流畅靓丽但并不容易实现的用户界面。
* 3. 项目经理——这些人负责实际地运行和维护应用程序。项目管理的主要关注点，无外乎正常运行时间（uptime）——应用程序始终正常可用的时间、性能和截止日期。项目经理追求的目标往往是尽量保持事情的简单化，以及不在升级更新时引入新问题。
* 4. 最终用户——当然是应用程序的主要消费者。尽管我们不会经常与最终用户打交道，但他们的反馈意见至关重要；没人想用的应用程序毫无价值。最终用户要求最多的就是对个人有用的功能，以及竞争性产品所具备的功能。

那么，前端工程师应该最关注哪些人的意见呢？答案是所有这四类人。优秀的前端工程师必须知道如何平衡这四类人的需求和预期，然后在此基础上拿出最佳解决方案。由于前端工程师处于与这四类人沟通的交汇点上，因此其沟通能力的重要性不言而喻。如果一个非常酷的新功能因为会影响前端性能，必须删繁就简，你怎么跟产品经理解释？再比如，假设某个设计如果不改回原方案可能会给应用程序造成负面影响，你怎么才能说服UI设计师？作为前端工程师，你必须了解每一类人的想法从何而来，必须能拿出所有各方都能接受的解决方案。从某种意义上说，优秀的前端工程师就像是一位大使，需要时刻抱着外交官的心态来应对每一天的工作。

## **40.如何提高前端技术**

* Github是一个优秀的代码托管网站，我们可以在上创建我们个人的项目，同时也是学习的好地方，我们可以关注其它优秀的项目。JSFiddle是一个web开发人员的练习场，一个可以在很多方面应用的工具。我们可以用他来在线编辑一些HTML,CSS,javascript片段。你编辑的代码可以与其他人分享，或嵌入你的博客等
* 阅读优秀的开源代码
* 参加[前端聚会](http://blog.csdn.net/borishuai/article/details/8529507" \t "http://www.cnblogs.com/doit8791/p/_blank)
* 关注技术发展趋势，了解最新的行业技术，可以通过订阅知名博客，阅读技术新闻获取
* 写博客/记笔记，可以进行知识积累。博客可以自己买空间或者各大知名博客网站。笔记的话比如evernote，有道笔记等。

## **41.按阶段划分**

如何划分仁者见仁，智者见智，要根据项目的情况作出调整，以下是我根据我自身的经验作出的划分，给大家作个参考。

### **第一阶段**

入门，打基础同时能参与到项目中去。

* HMTL & XHTML
* CSS基础知识
* JavaScript基础知识
* DOM
* JSON
* AJAX
* JavaScript Frameworks
* 编辑器

### **第二阶段**

掌握前端核心技术，可以独立干活。

* HTML5标签，TML标签语义化
* CSS Sprite
* 浏览器兼容性
* IE HasLayout和Block Format Content
* CSS3
* 精通JavaScript Frameworks
* HTML5
* 前端模板
* 前端MVC
* 模块化开发
* Http1.1
* 调试工具
* 正则表达式
* 响应式设计

### **第三阶段**

把握整个前端项目，做整个前端项目的架构师。

* CSS性能优化
* LESS and SASS
* JavaScript单元测试
* JavaScript设计模式
* NodeJS
* ES5
* Web移动开发
* 浏览器插件开发
* 前端安全
* 跨域处理
* SEO
* A/B test
* 可用性/可访问性
* 前端流程/部署
* 浏览器原理

### **第四阶段**

一代宗师。

* 不停的学习新的技术
* 交互设计能力，管理能力

## **42前端工程师未来的路在哪？**

第一条路是走技术流路线，即深入研究前端相关的各项技术，比如浏览器原理，JavaScript本身的研究，W3C各种标准等。前端技术发展很快，各种新技术层出不穷，这条路走下去是很累的。

第二路是往交互设计方向走，前端工程师做到一定程度后交互设计能力也会得到很大的提高，对整个信息架构的把握能力也会更强。可以很好的弥补视觉设计师在交换设计上的不足。这条路其实和第一条是不冲突的，甚至可以理解为同一条路。

第三条路是往后走，即去学习后台开发的技术，比如JAVA/PHP等，其实绝大部分传统的后台开发人员就是这个状态，即前后台都做。个人认为这样的话其实已经走回老路了，毕竟前端就是从原来的后台开发那里独立出来的。

最后一条路就是往管理方向，比如项目经理，或者干脆转行，比如公务员等。

## **43.前端开发知识结构**

前端工程师

* + 浏览器
    - IE6/[7](http://www.microsoft.com/en-us/download/internet-explorer-7-details.aspx)/[8](http://windows.microsoft.com/en-US/internet-explorer/downloads/ie-8)/[9](http://windows.microsoft.com/en-US/internet-explorer/downloads/ie-9/worldwide-languages)/[10](http://windows.microsoft.com/en-US/internet-explorer/ie-10-worldwide-languages)/[11](http://windows.microsoft.com/en-US/internet-explorer/ie-11-worldwide-languages) (Trident)
    - [Firefox](http://www.mozilla.org/en-US/) (Gecko)
    - [Chrome](http://www.google.com/chrome)/[Chromium](http://www.chromium.org/) (Blink)
    - [Safari](http://www.apple.com/safari/) (WebKit)
    - [Opera](http://www.opera.com/) (Blink)
  + 编程语言
    - [JavaScript](https://developer.mozilla.org/en-US/docs/JavaScript)/[Node.js](http://nodejs.org/)
    - [CoffeeScript](http://coffeescript.org/)
    - [TypeScript](http://www.typescriptlang.org/)
  + 切页面
    - [HTML](http://www.w3.org/html/)/[HTML5](http://www.w3.org/TR/html5/)
    - [CSS/CSS3](http://www.w3.org/Style/CSS/)
    - [Sass](http://sass-lang.com/)/[LESS](http://lesscss.org/)/[Stylus](http://learnboost.github.io/stylus/)/[postCSS](https://github.com/postcss/postcss)
    - [PhotoShop](http://www.photoshop.com/products/photoshop)/[Paint.net](http://www.getpaint.net/)/[Fireworks](http://www.adobe.com/cn/products/fireworks.html)/[GIMP](http://www.gimp.org/)/[Sketch](http://bohemiancoding.com/sketch/)
  + 开发工具
    - 编辑器和IDE
      * [VIM](http://www.vim.org/)/[Sublime Text2](http://www.sublimetext.com/)
      * [Notepad++](http://notepad-plus-plus.org/)/[EditPlus](http://www.editplus.com/)
      * [WebStorm](http://www.jetbrains.com/webstorm/)
      * [Emacs](http://www.gnu.org/software/emacs/) [EmacsWiki](http://emacswiki.org/)
      * [Brackets](http://brackets.io/)
      * [Atom](https://atom.io/)
      * [Lime Text](http://limetext.org/)
      * [Light Table](http://lighttable.com/)
      * [Codebox](https://www.codebox.io/)
      * [TextMate](http://macromates.com/)
      * [Neovim](http://neovim.org/)
      * [Komodo IDE / Edit](http://www.activestate.com/komodo-edit)
      * [Eclipse](http://www.eclipse.org/)
      * [Visual Studio](http://www.visualstudio.com/)/[Visual Studio Code](https://code.visualstudio.com/)
      * [NetBeans](https://netbeans.org/)
      * [Cloud9 IDE](http://c9.io/)
      * [HBuilder](http://www.dcloud.io/)
      * [Nuclide](http://nuclide.io/)
      * [VS code](https://code.visualstudio.com/)
    - 调试工具
      * [Firebug](http://getfirebug.com/)/[Firecookie](https://addons.mozilla.org/en-US/firefox/addon/firecookie/)
      * [YSlow](http://developer.yahoo.com/yslow/)
      * [IEDeveloperToolbar](http://www.microsoft.com/en-us/download/details.aspx?id=18359)/[IETester](http://www.my-debugbar.com/wiki/IETester/HomePage)
      * [Fiddler](http://www.telerik.com/fiddler)
      * [Chrome Dev Tools](https://developer.chrome.com/devtools)
      * [Dragonfly](http://www.opera.com/dragonfly/)
      * [DebugBar](http://www.debugbar.com/)
      * [Venkman](https://developer.mozilla.org/en-US/docs/Venkman)
      * [Charles](https://www.charlesproxy.com/)
    - 版本管理
      * [Git](http://git-scm.com/)/[SVN](http://subversion.apache.org/)/[Mercurial](http://mercurial.selenic.com/)
      * [Github](https://github.com/)/[GitLab](https://about.gitlab.com/)/[Bitbucket](https://bitbucket.org/)/[Gitorious](https://gitorious.org/)/[GNU Savannah](http://savannah.gnu.org/)/[Launchpad](https://launchpad.net/)/[SourceForge](http://sourceforge.net/)/[TeamForge](http://www.collab.net/products/teamforge)
  + 代码质量
    - Coding style
      * [JSLint](http://www.jslint.com/)/[JSHint](http://www.jshint.com/)/[jscs](https://github.com/mdevils/node-jscs)/[ESLint](https://github.com/eslint/eslint)
      * [CSSLint](http://csslint.net/)
      * [Markup Validation Service](http://validator.w3.org/)
      * [HTML Validators](https://validator.whatwg.org/)
    - 单元测试
      * [QUnit](http://qunitjs.com/)/[Jasmine](http://jasmine.github.io/)
      * [Mocha](http://mochajs.org/)/[Should](https://github.com/visionmedia/should.js/)/[Chai](http://chaijs.com/)/[Expect](https://github.com/LearnBoost/expect.js/)
      * [Unit JS](http://unitjs.com/)
    - 自动化测试
      * [WebDriver](http://docs.seleniumhq.org/docs/03_webdriver.jsp)/[Protractor](https://github.com/angular/protractor)/[Karma Runner](https://github.com/karma-runner/karma)/[Sahi](http://sahi.co.in/)
      * [phantomjs](http://phantomjs.org/)
      * [SourceLabs](https://saucelabs.com/)/[BrowserStack](http://www.browserstack.com/)
  + 前端库/框架
    - [jQuery](http://jquery.com/)/[Underscore](http://underscorejs.org/)/[Mootools](http://mootools.net/)/[Prototype.js](http://www.prototypejs.org/)
    - [YUI3](http://yuilibrary.com/projects/yui3/)/[Dojo](http://dojotoolkit.org/)/[ExtJS](http://www.sencha.com/products/extjs)/[KISSY](http://docs.kissyui.com/)
    - [Backbone](http://backbonejs.org/)/[KnockoutJS](http://knockoutjs.com/)/[Emberjs](http://emberjs.com/)
    - [AngularJS](http://angularjs.org/)
      * [Batarang](https://chrome.google.com/webstore/detail/angularjs-batarang/ighdmehidhipcmcojjgiloacoafjmpfk)
    - [Bootstrap](http://getbootstrap.com/)
    - [Semantic UI](http://www.semantic-ui.com/)
    - [Juice UI](http://juiceui.com/)
    - [Web Atoms](http://webatomsjs.neurospeech.com/)
    - [Polymer](http://docs.polymerchina.org/)
    - [Dhtmlx](http://dhtmlx.com/)
    - [qooxdoo](http://qooxdoo.org/)
    - [React](http://facebook.github.io/react/)
    - [Brick](http://mozbrick.github.io/)
    - [vue.js](http://cn.vuejs.org/)
  + 前端标准/规范
    - HTTP/1.1: RFCs 7230-7235
    - [HTTP/2](https://http2.github.io/)
    - [ECMAScript3/5](http://www.ecma-international.org/publications/standards/Ecma-262.htm)
    - [ECMAScript6](http://www.ecma-international.org/ecma-262/6.0/index.html)
    - [W3C: DOM/BOM/XHTML/XML/JSON/JSONP/...](http://www.w3.org/TR/)
    - [CommonJS Modules](http://wiki.commonjs.org/wiki/Modules/1.0)/[AMD](https://github.com/amdjs/amdjs-api/wiki/AMD)
    - [HTML5](http://www.w3.org/html/wg/drafts/html/master/)/[CSS3](http://www.w3.org/Style/CSS/specs.en.html)
    - [Semantic Web](http://semanticweb.org/)
      * [MicroData](http://schema.org/)
      * [RDFa](http://www.w3.org/TR/rdfa-core/)
    - [Web Accessibility](http://www.w3.org/WAI/)
      * [WCAG](http://www.w3.org/TR/WAI-WEBCONTENT/)
      * [Role Attribute](http://www.w3.org/TR/role-attribute/)
      * [WAI-ARIA](http://www.w3.org/TR/wai-aria/)
  + 性能
    - [JSPerf](http://jsperf.com/)
    - [YSlow 35 rules](http://developer.yahoo.com/performance/rules.html)
    - [PageSpeed](https://developers.google.com/speed/pagespeed/)
    - [HTTPWatch](http://www.httpwatch.com/)
    - [DynaTrace's Ajax](http://www.compuware.com/application-performance-management/dynatrace-ajax-download.html)
    - [高性能JavaScript](http://book.douban.com/subject/5362856/)
  + SEO
  + 编程知识储备
    - [数据结构](http://zh.wikipedia.org/wiki/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84)
    - OOP/AOP
    - [原型链](http://net.tutsplus.com/tutorials/javascript-ajax/prototypes-in-javascript-what-you-need-to-know/)/作用域链
    - [闭包](http://www.jibbering.com/faq/notes/closures/)
    - [编程范型](http://zh.wikipedia.org/wiki/%E7%BC%96%E7%A8%8B%E8%8C%83%E5%9E%8B)
    - [设计模式](http://addyosmani.com/resources/essentialjsdesignpatterns/book/)
    - [Javascript Tips](http://sanshi.me/articles/JavaScript-Garden-CN/html/index.html)
  + 部署流程
    - 压缩合并
      * [YUI Compressor](http://developer.yahoo.com/yui/compressor/)
      * [Google Clousure Complier](https://developers.google.com/closure/compiler/)
      * [UglifyJS](https://github.com/mishoo/UglifyJS)
      * [CleanCSS](https://github.com/GoalSmashers/clean-css)
    - 文档输出
      * [JSDoc](https://github.com/jsdoc3/jsdoc)
      * [Dox](https://github.com/visionmedia/dox)/[Doxmate](https://github.com/JacksonTian/doxmate)/[Grunt-Doxmate](https://github.com/luozhihua/grunt-doxmate)
    - 项目构建工具
      * [make](http://www.gnu.org/software/make/)/[Ant](http://ant.apache.org/)
      * [GYP](http://code.google.com/p/gyp/)
      * [Grunt](http://gruntjs.com/)
      * [Gulp](http://gulpjs.com/)
      * [Yeoman](http://yeoman.io/)
      * [FIS](http://fis.baidu.com/)
      * [Mod](https://github.com/modulejs/modjs)
    - ES6＋ 转换器
      * [Traceur](https://github.com/google/traceur-compiler)
      * [Babel](https://babeljs.io/)
  + 代码组织
    - 类库模块化
      * [CommonJS](http://www.commonjs.org/)/AMD
      * [YUI3模块](http://yuilibrary.com/projects/yui3/)
      * [webpack](http://webpack.github.io/)
    - 业务逻辑模块化
      * [bower](https://github.com/twitter/bower)/[component](https://github.com/component/component)
    - 文件加载
      * [LABjs](http://labjs.com/)
      * [SeaJS](http://seajs.org/)/[Require.js](http://requirejs.org/)
    - 模块化预处理器
      * [Browserify](https://github.com/substack/node-browserify)
  + 安全
    - [CSRF](http://en.wikipedia.org/wiki/Cross-site_request_forgery)/[XSS](http://en.wikipedia.org/wiki/Cross-site_scripting)
    - [CSP](http://www.w3.org/TR/CSP/)
    - [Same-origin policy](https://developer.mozilla.org/docs/Web/Security/Same-origin_policy)
    - ADsafe/Caja/Sandbox
  + 移动Web
    - HTML5/CSS3
    - [响应式网页设计](http://zh.wikipedia.org/wiki/%E5%93%8D%E5%BA%94%E5%BC%8F%E7%BD%91%E9%A1%B5%E8%AE%BE%E8%AE%A1)
    - [Zeptojs](http://zeptojs.com/)/[iScroll](http://cubiq.org/iscroll)
    - V5/[Sencha Touch](http://www.sencha.com/products/touch)
    - [PhoneGap](http://phonegap.com/)
    - [jQuery Mobile](http://jquerymobile.com/)
    - [W3C Mobile Web Initiative](http://www.w3.org/Mobile/)
    - [W3C mobileOK Checker](http://validator.w3.org/mobile/)
    - [Open Mobile Alliance](http://openmobilealliance.org/)
    - [React Native](https://facebook.github.io/react-native/)
  + 前沿技术社区/会议
    - [D2](http://d2forum.org/)/[WebRebuild](http://www.webrebuild.org/)
    - NodeParty/[W3CTech](http://w3ctech.com/)/[HTML5梦工厂](http://www.html5dw.com/)
    - [JSConf](http://jsconf.com/)/[沪JS(JSConf.cn)](http://jsconf.cn/)
    - QCon/Velocity/SDCC
    - [JSConf](http://jsconf.com/)/[NodeConf](http://www.nodeconf.com/)
    - [CSSConf](http://cssconf.com/)
    - YDN/YUIConf
    - HybridApp
    - [WHATWG](http://whatwg.org/)
    - [MDN](https://developer.mozilla.org/zh-CN/)
    - [codepen](http://codepen.io/)
    - [w3cplus](http://www.w3cplus.com/)
    - [CNode](https://cnodejs.org/)
  + 计算机知识储备
    - 编译原理
    - [计算机网络](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C)
    - [操作系统](http://zh.wikipedia.org/wiki/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)
    - 算法原理
    - 软件工程/软件测试原理
    - [Unicode](http://www.unicode.org/)
  + 软技能
    - 知识管理/总结分享
    - 沟通技巧/团队协作
    - 需求管理/PM
    - 交互设计/可用性/可访问性知识
  + 可视化
    - SVG/Canvas/VML
    - SVG: [D3](http://d3js.org/)/[Raphaël](http://raphaeljs.com/)/[Snap.svg](http://snapsvg.io/)/[DataV](http://datavlab.org/datavjs/)
    - Canvas: [CreateJS](http://www.createjs.com/)/[KineticJS](http://kineticjs.com/)
    - [WebGL](http://en.wikipedia.org/wiki/WebGL)/[Three.JS](http://threejs.org/)

后端工程师

* + 编程语言
    - C/C++/Java/PHP/Ruby/Python/...
  + 网页服务器
    - [Nginx](http://nginx.org/en/)
    - [Apache](http://httpd.apache.org/)
    - [Lighttpd](http://www.lighttpd.net/)
  + 数据库
    - SQL
    - [MySQL](http://www.mysql.com/)/[PostgreSQL](http://www.postgresql.org/)/[Oracle](http://www.oracle.com/us/products/database/overview/index.html)/[DB2](http://www-01.ibm.com/software/data/db2)
    - [MongoDB](http://www.mongodb.org/)/[CouchDB](http://couchdb.apache.org/)
  + 数据缓存
    - [Redis](http://redis.io/)
    - [Memcached](http://memcached.org/)
  + 文件缓存/代理
    - [Varnish](https://www.varnish-cache.org/)
    - [Squid](http://www.squid-cache.org/)
  + 操作系统
    - Unix/Linux/OS X/Windows
  + 数据结构

## **前端工程师（技术+设计的完美结合）**

[添加评论](https://www.zhihu.com/collection/21009601) • [修改记录](https://www.zhihu.com/collection/21009601/log)

## **44.[什么是前端系统的业务逻辑？在重复开发了n个系统前端的代码之后，我们能沉淀什么？用什么沉淀？](https://www.zhihu.com/question/20159831" \t "https://www.zhihu.com/collection/_blank)**

沉淀出什么真的得看个人方向。比较典型的方向有：

* 技术专家。专注于语言细节、基础库、基础框架等的开发或研究。
* 行业专家。专注于前端在某个行业或领域内的应用问题。
* 架构师。关注如何设计架构、技术选型、等等，
* 管理层。当然做的还是前端方面的管理，要不就是转行了。
* 转行：比如转设计师、转产品经理、转测试，等等。

至于怎么沉淀，也还是得看个人。每个人对知识的学习方式都不太一样，这是属于知识管理的层面的东西了，可以找找这方面相关的资料。

## **45.[什么是前端系统的业务逻辑？在重复开发了n个系统前端的代码之后，我们能沉淀什么？用什么沉淀？](https://www.zhihu.com/question/20159831" \t "https://www.zhihu.com/collection/_blank)**

个人感觉前段代码有很强和可重用性，HTML和CSS JS 表现的很明显，我可以用一个HTML的导航栏的结构写出所有网站的单列导航，只要修改HTML文字内容和css就可以。但是我感觉这样还是不够，然后我们将HTML细分，将CSS分类。提取出那些所有网站通用的代码，将可以回要修改的CSS提取出来。建立属于自己的代码规范和库赖提到效率和方便团队开发。  
我以前一直诧异为什么想前段这样这样搞重用性的内容（HTML和css）没有一个完善和完美的库。后来我对自己的解释是，所有网站的表现都是不一样的。没有两个一样的网站表现，所以当我们写后台可配后，前台就像织毛衣一样的切图写代码。  
多火一直在思考和完善这个问题。我们在内部提出了LMC的思想，现在正在不断的完善和实践。  
将网站前台分为LMC三层，  
  
L即layout，布局层，用来布置整个页面的格局；  
M即module，组件层，用来封装网站常用的组件块；  
C即conetnt，内容层，用来给module层填充内容；  
最后还有一个combine组装层，负责调整module、content的位置。  
  
将HTML代码结构固定后用四个css来控制页面的展现。想相同的css代码提出，如css的表现属性（在了L，M,C）实现可重用，将css文本和自身的属性提出来配置不同的网站。  
在JS方面写好有各种结构的代码，实现一段代码的多种表现，如所有的图展都是一段js代码，在js有表现方式（左右滑动，上线滑动，变淡）有没有文字介绍，有没有数字小框，有没有左右的滑动的接口。和html结构半分离，用js在dom加需要的结构和内容。用html 的class来当接口实现js效果。  
我们在html，css，js分离的基础上在将内容更具可重新细分非LMC，在将LMC的css细分，利用css的属性可继承和权重做变化。实现高可重用性。  
多活还很年轻，我们也很年轻，LMC也很年轻。我们相信LMC的可行性和高效性，我们在努力建立我们的思想和代码规范，我们也在努力的实践和完善。希望有一天我们可以拿出我们的LMC同更多的交流和方便更多的前台开发。  
优秀的程序员都是努力和很懒的。同志们，快快做完那个页面我们就去喝酒。

## **46.[为什么很难聘到前端工程师？](https://www.zhihu.com/question/19923687" \t "https://www.zhihu.com/collection/_blank)**

在我离开我的老东家之前也面试了三个月的前端工程师，最后到我走都没有招到合适的。老东家那边的要求是"熟悉javascript、actionscript、 css、html、各浏览器兼容性"。   
 其实并非我要求的太高，基本的正则表达式、甚至cloneNode之类的DOM方法都不知道、更别提actionscript神马了。各种不靠谱。   
 招不到人，给钱不够多我想是其中一个原因吧，毕竟重赏之下必有勇夫。   
 你可以考虑直接招一个靠谱的前端架构师。   
 三个诸葛亮顶个臭皮匠在技术领域不怎么适合。   
 一个靠谱的前端可以顶得了一个团队。。。   
 其实最近我也在考虑转型的问题。   
 想转型其实考虑的下面几点，我想其中有一些可能能回答这个问题吧：   
 1、开发环境恶劣，各浏览器兼容性能头疼，需要不断地跟着浏览器走，兼容性涉及的不仅仅是CSS、还有各种DOM、BOM模型以及研究各种诡 异。。。   
 2、没有钱途。工作是为了爱好和生活。靠谱前端工资确实也比较高，但是擅长javascript、html、css(这些是基本功)，还捎带至少懂一门后台语 言(php/C#/java/c++)，还天天追着各种浏览器的兼容性，这个才给开到多少钱？一个擅长j2ee、熟悉linux和ssh的工程师是什么价格。我们都需 要面包。   
 3、被动。这是一个很头疼的问题。很多时候，工期不够的时候总是压缩前端这边的，我不知道为什么。我工作过的相对好一点的公司也仅仅是 一个上市集团下属的垂直门户，而在哪里也都是如此。但是，大家都是人。   
 4、学习难。相对其他几种语言，都有社区和相对全的资料。前端的参考只有W3C和ECMA那一堆鸟文。就算你看懂了，还有各种浏览器兼容性 ，你又要Mozilla、MSDN各种搜，各种新添加的属性是你买书看不到的。   
 5、被鄙视。各种鄙视前端。。你们懂得。。。

## **47.[前端开发基础扎实的标准是什么？](https://www.zhihu.com/question/38922374" \t "https://www.zhihu.com/topic/19749040/_blank)**

既会肉眼量度像素，又能随性处理请求。

白手切图要准确，手中无尺心中有。

工程思维要具备，打包构建样样会。

javascript玩得欢，css写得溜。

跟得上时代的脚步，学得了多样的框架。

看得懂原型，实现得需求。

聊得过设计，劝得了产品。

心之所向，无所不能。

## **[48.前端开发的行业状况是怎样的？对于自学web前端的人从业有何建议？](https://www.zhihu.com/question/59813534" \t "https://www.zhihu.com/topic/19749040/_blank)**

应届生阔以试一试，先花一个月入门html和css,(PS：css3阔以先不看)。然后花3个月时间，好好的学习js原生语法，先不要管这框架，那框架，也不要在意什么webpack,gulp打包工具。感觉又要抛书单，但是不得不的说，《javascript高级程序员设计》和《javascript …

## **49.[计算机基础知识不牢的前端都是瞎扯淡吗？为什么？对于前端来说哪些计算机基础知识很重要？](https://www.zhihu.com/question/35703265" \t "https://www.zhihu.com/topic/19749040/_blank)**

**、**我的计算机基础知识可不牢靠了，大学里上就正经上过一个 CS 入门课。当然也不是说我基础知识啥都不会，七七八八自学了一些，但是跟科班出身的，尤其是我那帮 MIT 毕业的同事比还是比较自卑的... 买了 CLRS 和 SICP 从来没看完...  
  
回到主题，我是觉得『前端』这个词现在涵盖的内容也越来越广了，尤其是应用化了以后，俨然前端自己有一个迷你技术栈：

1. 纯表现层。用户体验、布局、特效、研究 CSS 各种奇技淫巧（例子：CodePen 上各种无比酷炫但基本靠 hard-code 搞出来的特效）；对于很多设计/前端兼修的人来说，技术层面就到此为止了。他们很多可能一辈子都不会写翻转二叉树，但他们也能搞出一些科班出生的人一辈子也搞不出的用户体验。当然不排除一些走 creative coding / 数据可视化路线的人需要对物理、数学、甚至计算机图形方面的知识进行针对性的强化。
2. 应用实现层。可能是大公司初级工程师主要干的活：拿着别人设计好的框架、工具去实现具体的应用逻辑。实话实说这个层面对计算机基础的要求确实不高，只要对 JS、CSS 这些领域专门的东西基础扎实 + 学习能力 ok 就可以了。但是这个层面其实需求巨大，而且有一个独特的需求：开发效率。要提升效率就得对手上的工具了解得非常细致，比如 XX 框架的 N 种优化小窍门之类的... 而这种东西只能靠实战经验去积累，基础再扎实影响也有限。
3. 应用架构层。技术选型、开发底层框架、制定开发规范、设计应用结构... 这些东西就涉及到知识的广度和深度了，对业务需求的理解很重要，而且碰到具体的纯技术问题的可能性也大得多。编译原理、算法、数据结构在这里都会派上实际的用处。
4. 基础设施层。自动化构建、部署、测试、加载方案、性能优化、代码质量管理等等... 这一层更加技术化了，而且涉及很多所有软件工程共通的东西，并不局限于前端。
5. 理念层。通过借鉴整个计算机体系中其他领域的思想，从根本上改进前端的开发范式。Facebook 的人现在做的就是这种事情。事实上能做到这一层的人基本不以前端自居了。

这个问题里现有答案的一些分歧可能就在于『是否人人都应该做大前端』。我的看法是：不看具体业务需求就谈一个前端是不是靠谱都是耍流氓。一个人称不称职，看的是他的技能树和他当前的职责是否匹配。有没有必要学习特定的知识，这是一个要看情况讨论的问题。假如一个人科班出身，基础扎实来做前端，那他可能很快就可以开始涉及架构层和基础设施层的工作，这是比较理想的情况。但如果之前并没有基础，那就看你有没有野心向其他层去发展了，如果你真的有兴趣，在研究过程中自然会发现自己的短板，并且有足够的动力去弥补它。  
  
当然，有野心拓展自己的技能树固然好，但我不会因为一个负责搞交互的人不会翻转二叉树就觉得 ta 不靠谱。你让一个科班出身但是一直搞理论研究的人去做 angular 应用，未必效果就比一个半路出家但是写了 3 年 angular 的人来得好。从这个角度讲，玉伯的看法是有一定道理的（可能表达得确实比较模糊）。  
  
补充一下：上面的这些层次并不是一个发展路线，不是说是个前端就一定要冲着最高的层次去，这不现实，因为每个层次都可以深入钻研，对于公司来说，尤其是大公司，往往更需要在一个层次深入专精的人而不是每个层次都半桶水的人（对于多层次专精的人的需求也是有的，但是这种一般都是 senior 职位了，不会太多）

****50.Javascript 的复杂度不逊色于 Perl 和 Python！****

****如何学习前端知识？****

　　我们生活在一个充满规则的宇宙里面。社会秩序按照规则运行，计算机语言几乎全部是规则的集合。计算机前辈们定义规则，规则约束我们，我们用规则控制数据。****大部分时候，对数据的合理控制，来自于你对规则的掌握。****

　　学习 HTML，CSS 应该先跟着书仔细、扎实的学一遍。然后就需要做大量的练习，做各种常规的、奇怪的、大量的布局练习来捆固、理解自己的知识。

　　而学习 Javascript 首先要知道这门语言可以做什么，不能做什么，擅长做什么，不擅长做什么！

　　如果你只想当一个普通的前端程序员，你只需要记住大部分 Javascript 函数，做一些练习就可以了。

　　如果你想当深入了解Javascript，你需要了解 Javascript 的原理，机制。需要知道他们的本源，需要深刻了解 Javascript 基于对象的本质。

　　还需要深刻了解浏览器宿主 下 的 Javascript 的行为、特性。

　　因为历史原因，Javascript一直不被重视，有点像被收养的一般！ 所以他有很多缺点，各个宿主环境下的行为不统一、内存溢出问题、执行效率低下等问题。

　　作为一个优秀的前端工程师还需要深入了解、以及学会处理 Javascript 的这些缺陷。