# FE-learning

FE

FE

结合个人经历总结的前端入门方法，总结从零基础到具备前端基本技能的道路、学习方法、资料。由于能力有限，不能保证面面俱到，只是作为入门参考，面向初学者，让初学者少走弯路。

互联网的快速发展和激烈竞争，用户体验成为一个重要的关注点，导致专业前端工程师成为热门职业，各大公司对前端工程师的需求量都很大，要求也越来越高，优秀的前端工程师更是稀缺。个人感觉前端入门相对容易，但是也需要系统地认真学习，在打好基础后坚持学习，成为优秀前端工程师也只是时间问题。

学习任何知识最重要的都是**兴趣**，如果经过一段时间的学习感觉不喜欢，那可能强迫自己学习是很痛苦的，效果也不会好，毕竟这很可能就是以后很多年生存的技能。不过随着互联网行业的发展，前端必然是Web开发人员需要学习的知识，有时候是没有专业前端工程师一起合作的，所以即使不做专门的前端工程师，掌握基本的前端技能为工作带来方便。

后期邀请了一些同学分享学习经历。如果有同学愿意分享,欢迎push

## 必备基础技能

[前端技能汇总](https://github.com/JacksonTian/fks)这个项目详细记录 了前端工程师牵涉到的各方面知识。在具备基本技能之后可以在里面找到学习 的方向，完善技能和知识面。

[frontend-dev-bookmarks](https://github.com/dypsilon/frontend-dev-bookmarks)是老外总结的前端开发资源。覆盖面非常广。包括各种知识点、工具、技术，非常全面。

以下是个人觉得入门阶段应该熟练掌握的基础技能：

* [HTML4](http://www.w3.org/TR/html401/cover.html#minitoc)，[HTML5](http://www.w3.org/TR/html5/#contents)语法、标签、语义
* [CSS2.1](http://www.w3.org/TR/CSS2/#minitoc)，[CSS3](http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/#table)规范，与HTML结合实现各种布局、效果
* [Ecma-262](http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf)定义的javascript的语言核心，原生[客户端javascript](https://developer.mozilla.org/en-US/docs/Web/API)，[DOM操作](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)，[HTML5新增功能](https://developer.mozilla.org/en/docs/web/Guide/HTML/HTML5)
* 一个成熟的客户端javascript库，推荐[jquery](http://jquery.com/)
* 一门服务器端语言：如果有服务器端开发经验，使用已经会的语言即可，如果没有服务器端开发经验，熟悉Java可以选择Servlet，不熟悉的可以选PHP，能实现简单登陆注册功能就足够支持前端开发了，后续可能需要继续学习，最基本要求是实现简单的功能模拟，
* [HTTP](http://www.w3.org/Protocols/rfc2616/rfc2616.html)

在掌握以上基础技能之后，工作中遇到需要的技术也能快速学习。

## 基本开发工具

恰当的工具能有效提高学习效率，将重点放在知识本身，在出现问题时能快速定位并 解决问题，以下是个人觉得必备的前端开发工具：

* **文本编辑器**：推荐[Sublime Text](http://www.sublimetext.com/)，支持各种插件、主题、设置，使用方便
* **浏览器**：推荐[Google Chrome](http://www.google.cn/chrome/?hl=zh-CN&standalone=1)，更新快，对前端各种标准提供了非常好的支持
* **调试工具**：推荐Chrome自带的[Chrome develop tools](https://developer.chrome.com/devtools)，可以轻松查看DOM结构、样式，通过控制台输出调试信息，调试javascript，查看网络等
* **辅助工具**：PhotoShop编辑图片、取色，fireworks量尺寸，AlloyDesigner对比尺寸，以及前面的到的Chrome develop tools，
* **翻墙工具**：lantern, 壁虎漫步

## 学习方法和学习目标

方法：

1. 入门阶段反复阅读**经典书籍的中文版**，书籍中的每一个例子都动手实现并在浏览器中查看效果
2. 在具备一定基础之后可以上网搜各种教程、demo，了解各种功能的实际用法和常见功能的实现方法
3. 阅读HTML，CSS，Javascript标准全面完善知识点
4. 阅读前端牛人的博客、文章提升对知识的理解
5. 善用搜索引擎

目标：

1. 熟记前面知识点部分的重要概念，结合学习经历得到自己的理解
2. 熟悉常见功能的实现方法，如常见CSS布局，Tab控件等。

## 入门之路

以下是入门阶段不错的书籍和资料

1. HTML先看[《HTML & CSS: Design and Build Websites》](http://www.amazon.cn/gp/product/B00BMK4GKW/ref=s9_simh_gw_p14_d0_i2?pf_rd_m=A1AJ19PSB66TGU&pf_rd_s=center-2&pf_rd_r=1AH2NF64STS19GY8GR54&pf_rd_t=101&pf_rd_p=108773272&pf_rd_i=899254051)1-9章，然后[《HTML5: The Missing Manual》](http://www.amazon.cn/HTML5%E7%A7%98%E7%B1%8D-Matthew-MacDonald/dp/B009DFCZAQ/ref=sr_1_1?ie=UTF8&qid=1414740812&sr=8-1&keywords=html5+the+missing+manual)1-4章。
2. CSS先看[《CSS: The Missing Manual》](http://www.amazon.cn/CSS-The-Missing-Manual-Mcfarland-David-Sawyer/dp/0596802447/ref=sr_1_1?ie=UTF8&qid=1414742710&sr=8-1&keywords=css+the+missing+manual+2)，然后[《CSS权威指南》](http://www.amazon.cn/CSS%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97-%E8%BF%88%E8%80%B6/dp/B0011F5SIC/ref=sr_1_1?ie=UTF8&qid=1414744248&sr=8-1&keywords=css+%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97)
3. javascript先看[《javascript高级程序设计》](http://www.amazon.cn/JavaScript%E9%AB%98%E7%BA%A7%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1-%E6%B3%BD%E5%8D%A1%E6%96%AF/dp/B007OQQVMY/ref=sr_1_1?s=books&ie=UTF8&qid=1414744358&sr=1-1&keywords=javascript+%E9%AB%98%E7%BA%A7%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1)，然后[《javascript权威指南》](http://www.amazon.cn/O-Reilly%E7%B2%BE%E5%93%81%E5%9B%BE%E4%B9%A6%E7%B3%BB%E5%88%97-JavaScript%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97-%E5%BC%97%E5%85%B0%E7%BA%B3%E6%A0%B9/dp/B007VISQ1Y/ref=sr_1_1?s=books&ie=UTF8&qid=1414744401&sr=1-1&keywords=javascript+%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97)
4. HTTP看[HTTP权威指南](http://www.amazon.cn/HTTP%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97-%E5%90%89%E5%B0%94%E5%88%A9/dp/B008XFDQ14/ref=sr_1_1?s=books&ie=UTF8&qid=1414744440&sr=1-1&keywords=HTTP+%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97)
5. 在整个学习过程中HTML CSS JavaScript会有很多地方需要互相结合，实际工作中也是这样，一个简单的功能模块都需要三者结合才能实现。
6. 动手是学习的重要组成部分，书籍重点讲解知识点，例子可能不是很充足，这就需要利用搜索引擎寻找一些简单教程，照着教程实现功能。以下是一些比较好的教程网址
   * 可以搜索各大公司前端校招笔试面试题作为练习题或者他人总结的[前端面试题](https://github.com/darcyclarke/Front-end-Developer-Interview-Questions/tree/master/Translations/Chinese)还有[个人总结的面试题](https://github.com/qiu-deqing/FE-interview)（带参考答案）
   * <http://code.tutsplus.com>有各种各样的教程
   * [MDN](https://developer.mozilla.org/en-US/docs/Web)也有很多教程，更重要的是里面有详细的文档，需要查找某个功能时在Google搜索：xxx site:https://developer.mozilla.org
   * <http://www.html5rocks.com/zh/>也有很多优质教程
   * <http://www.sitepoint.com/>
   * <http://alistapart.com/>
7. 原生javascript是需要重点掌握的技能，在掌握原生javascript的基础上推荐熟练掌握jQuery，在实际工作中用处很大，这方面的书籍有[《Learning jQuery》](http://www.amazon.com/Learning-jQuery-Fourth-Jonathan-Chaffer/dp/178216314X/ref=sr_1_1?s=books&ie=UTF8&qid=1410099243&sr=1-1&keywords=learning+jquery)或者去[jQuery官网](http://learn.jquery.com/)
8. 建一个<https://github.com/>账号，保存平时学习中的各种代码和项目。
9. 有了一定基础之后可以搭建一个个人博客，记录学习过程中遇到的问题和解决方法，方便自己查阅也为其他人提供了帮助。也可以去<http://www.cnblogs.com/>或者<http://www.csdn.net/>这样的网站注册账号，方便实用
10. 经常实用Google搜索英文资料应该经常找到来自<http://stackoverflow.com/>的高质量答案，与到问题可以直接在这里搜索，如果有精力，注册一个账号为别人解答问题也能极大提高个人能力。
11. 经典书籍熟读之后，可以打开前面必备基础技能部分的链接。认真读对应标准，全面掌握知识

## 继续提高

有了前面的基础之后，前端基本算是入门了，这时候可能每个人心中都有了一些学习方向，如果还是没有。 可以参考前面必备技能部分提到的那两个项目，从里面选一些进行发展学习。以下是一些不错的方面：

* [Grunt](http://gruntjs.com/)：前端自动化工具，提高工作效率
* [less css](http://lesscss.org/)：优秀的CSS预处理器
* [bootstrap](http://getbootstrap.com/)：优秀的CSS框架，对没有设计师的团队很不错，与less结合使用效果完美
* [requirejs](http://requirejs.org/)：AMD规范的模块加载器，前端模块化趋势的必备工具
* [Node.js](http://nodejs.org/)：JavaScript也可以做后台，前端工程师地位更上一步
* [AngularJS](https://angularjs.org/)：做Single Page Application的好工具
* [移动端web开发](https://developer.mozilla.org/en-US/docs/Web/Guide/Mobile)：智能手机的普及让移动端的流量正在逐步赶超PC端
* [Javascript内存管理](https://developer.chrome.com/devtools/docs/javascript-memory-profiling?hl=figoogle)：SPA长期运行需要注意内存泄露的问题
* [High Performance JavaScript(Build Faster Web Application Interfaces)](http://www.amazon.com/Performance-JavaScript-Faster-Application-Interfaces/dp/059680279X/ref=sr_1_1?s=books&ie=UTF8&qid=undefined&sr=1-1&keywords=high+performance+javascript)
* [Best Practices for Speeding Up Your Web Site](https://developer.yahoo.com/performance/rules.html)：重要技能

## 一些个人经历

### [LingyuCoder](https://github.com/LingyuCoder)的学习经历

上面的大神都总结得差不多了，我这里就胡扯一些吧

#### 工具

* chrome dev tools：前端开发调试利器，着重注意几个功能：
  + console（废话）
  + elements:元素样式调整，很常用
  + sources：代码中添加断点，单步调试，以及单步调试过程中查看内存中的对象
    - watch expression：通过表达式查看当前内存中的值
    - call stack：查看调用栈，开启async，可以看异步调用栈（这个非常有用，尤其是ajax调试的时候）
    - scope variables：作用域链上的变量，非常有用
  + network：抓包查看每个请求，非常重要，前后端联调必备
  + timeline：分析渲染、js执行等等各个阶段，性能优化利器
  + emulation：模拟移动端环境，mobile页面开发必备
  + 一些插件:
    - liveload: 修改页面后自动刷新，不用按F5
    - dimensions：直接在页面上测量的利器
    - livestyle：css样式修改后自动起效果，不需要刷新，elements修改后也能同步到代码中
    - image tool：测量，取色
    - UC二维码：移动端调试扫码必备
    - pagespeed，YSlow：页面性能分析和优化插件
    - 马克飞象：优秀的在线markdown编辑器，快速写周报，做记录
* sublime text2：编码方便，插件多，速度快，性能好
  + emmet：提升html编码速度必备
  + sublimelinter + 各种语言的lint和hint：代码纠错
  + 一些snippets：自动补全，提升开发效率
* Intellij IDEA和WebStorm：集成开发环境，集成了各种功能，开发比sublime要方便，但会比较吃性能
* Mark Men：测量、取色、标注利器，拿到视觉稿之后第一个打开的软件
* GFW Fucker：我用红杏，可以的话买个虚拟服务器当梯子
* iHosts：非常优秀的hosts管理软件，轻松修改hosts，开发调试必备
* Charles：Mac 平台最好用的抓包分析工具
* Rythem：AlloyTeam出品的代理抓包软件，非常轻量，安装简单，移动端（真机）开发调试很好用
* Wunderlist：一个非常不错的Todo List，任务、需求多的时候管理起来很方便

#### 技能

前端的技能其实除了JavaScript（包括NodeJS）、HTML、CSS以外，还有很多。其实前端的技能树很大，这里只能列一些我开发中见到的说一说 #####语言基础 JavaScript： \* 作用域链、闭包、运行时上下文、this \* 原型链、继承 \* NodeJS基础和常用API

CSS： \* 选择器 \* 浏览器兼容性及常见的hack处理 \* CSS布局的方式和原理（盒子模型、BFC、IFC等等） \* CSS 3，如animation、gradient、等等

HTML： \* 语义化标签

##### 进阶

JavaScript: \* 异步控制（Promise、ES6 generator、Async） \* 模块化的开发方式（AMD、CMD、KMD等等） \* JavaScript解释器的一些相关知识 - 异步IO实现 - 垃圾回收 - 事件队列 \* 常用框架使用及其原理 - jQuery：基于选择器的框架，但个人认为不能叫框架，应该算工具库，因为不具备模块加载机制，其中源码很适合阅读钻研 - AngularJS/Avalon等MVVM框架：着重理解MVVM模式本身的理念和双向绑定的实现，如何解耦 - underscore：优秀的工具库，方便的理解常用工具代码片段的实现 - polymer/React: 组件化开发，面向未来，理解组件化开发的原理

CSS和HTML：主要是CSS3的特性和HTML5的特性，以及浏览器处理的流程和绘制原理 \* DOM树、CSSOM树、渲染树的构建流程及页面渲染的过程 \* 解析HTML、CSS、JavaScript时造成的阻塞 \* HTML5相关 - SVG及矢量图原理 - Canvas开发及动画原理（帧动画） - Video和Audio \* flex box布局方式 \* icon fonts的使用

常用NodeJs的package： \* koa \* express \* underscore \* async \* gulp \* grunt \* connect \* request

一些理念： \* 响应式Web \* 优雅降级、渐进增强 \* don`t make me think \* 网页可用性、可访问性、其中的意义 \* SEO搜索引擎优化，了解搜索引擎的原理 \* SPA的好处和问题

性能优化： \* 减少请求数量（sprite、combo） \* 善用缓存（application cache、http缓存、CDN、localstorage、sessionstorage，备忘录模式） \* 减少选择器消耗（从右到左），减少DOM操作（DOM和JavaScript解释器的分离） \* CSS的回流与重绘

##### 项目

* 版本管理：首推Git，用过Git都不会想用SVN了
  + Git：本地版本管理的机制
  + SVN：远程中心的版本管理机制
* 自动化构建：主要就是less、模板、coffee等的预处理以及对代码压缩和合并
  + Gulp：基于流构建，速度快、模块质量好
  + Grunt：独立任务构建，速度慢，配置蛋疼，灵活性高
* 预处理和模板引擎
  + less：语法简单，但功能有限
  + jade、ejs、velocity等模板引擎，各有各的长处
  + coffee：python工程师最爱，我没用过
* 环境搭建：主要是将线上代码映射到本地，并在本地启动一个demo服务器，至于模拟数据的mock，见仁见智了
  + 本地代理：ihosts
* 自动化测试：在业务较为稳定的情况下，可以通过自动化测试来减少测试的事件，但需求较多的时候，维护测试用例的成本会很高，可能用自动化测试会起到反效果
  + jasmine
  + mocha
* 生态系统
  + npm
  + bower
  + spm
* 搭建一个属于自己的博客
  + git pages
  + hexo
  + jekyll

##### 未来

* Web Componets：面向未来的组件化开发方式
  + HTML模板
  + Shadow DOM
  + Custom Elements
  + HTML Import
* 移动端Native开发：这也是需要了解的，以后前端工程师会经常地和webview打交道，也要了解native开发

##### 其他

有些东西不是考敲码就能弄好的，我参与实习的时候感受到了很多，这些是我遇到的也是我感觉自己做的不好的地方 \* **对于业务的思考**：我个人这方面非常欠缺，所以放在最前面，在敲码前要多思考业务 \* 交流和沟通能力：这个非常重要，前端同时需要与项目经理、产品、交互、后台打交道，沟通不善会导致很多无用功，延缓项目 \* 知识管理、时间管理：input和output的平衡，output是最好的input。如何做好分享，参与社区，做好交流，作好记录 \* 对新技术的渴望，以及敢于尝试

#### 入门书

入门可以通过啃书，但书本上的东西很多都已经过时了，在啃书的同时，也要持续关注技术的新动态。这里推几本我觉着不错的书：

* 《JavaScript高级编程》：可以作为入门书籍，但同时也是高级书籍，可以快速吸收基础，等到提升再回来重新看
* 《JavaScript权威指南》：不太适合入门，但是必备，不理解的地方就去查阅一下，很有帮助
* 《编写可维护的JavaScript》和：
* 《Node.js开发指南》：不错的Nodejs入门书籍
* 《深入浅出Node.js》：Nodejs进阶书籍，必备
* 《JavaScript异步编程》：理解JS异步的编程理念
* 《JavaScript模式》和《JavaScript设计模式》：JavaScript的代码模式和设计模式，将开发思维转变到JavaScript，非常好的书
* 《JavaScript框架设计》：在用轮子同时，应当知道轮子是怎么转起来的，讲解很详细，从源码级别讲解框架的各个部分的实现，配合一个现有框架阅读，可以学到很多东西
* 《Don`t make me think》：网页设计的理念，了解用户行为，非常不错
* 《CSS禅意花园》：经久不衰的一部著作，同样传递了网页设计中的理念以及设计中需要注意的问题
* 《高性能JavaScript》和《高性能HTML5》：强调性能的书，其中不只是性能优化，还有很多原理层面的东西值得学习
* 《HTML5 Canvas核心技术》：我正在读的一本书，对于canvas的使用，动画的实现，以及动画框架的开发都非常有帮助
* 《HTTP权威指南》：HTTP协议相关必备，前端开发调试的时候也会经常涉及到其中的知识
* 《响应式Web设计》：技术本身不难，重要的是响应式网页的设计理念，以及移动先行的思想
* 《JavaScript语言精粹》：老道的书，也是普及JavaScript的开发思维的一本好书，非常适合入门

#### 一些不错的网站

* [github](https://github.com)：没啥好说的，多阅读别人的源码，多上传自己的源码，向世界各地的大牛学习
* [codepen](http://codepen.io/)：感受前端之美的必选之地，里面有很多酷炫的效果和优秀的插件
* [echojs](http://www.echojs.com/)：快速了解js新资讯的网站
* [stackoverflow](http://stackoverflow.com/)和[segmentfault](segmentfault.com)：基本上各种问题都能在上面获得解答
* [google web fundamentals](https://developers.google.com/web/fundamentals/)：每篇文章都适合仔细阅读
* [static files](http://www.staticfile.org/)：开放的CDN，很好用
* [iconfont](http://www.iconfont.cn/)：阿里的矢量图标库，非常不错，支持CDN而且支持项目
* [html5 rocks](http://www.html5rocks.com/): 一个不错的网站，很多浏览器的新特性以及前沿的技术，都能在这上面找到文章
* [css tricks](http://css-tricks.com/)：如何活用CSS，以及了解CSS新特性，这里可以满足你
* [JavaScript 秘密花园](http://bonsaiden.github.io/JavaScript-Garden/zh/#object.general) JavaScript初学必看，非常不错
* [w3cplus](http://www.w3cplus.com/)：一个前端学习的网站，里面的文章质量都挺不错的
* [node school](http://nodeschool.io/)：一个不错的node学习网站
* [learn git branch](http://pcottle.github.io/learnGitBranching/?demo)：一个git学习网站，交互很棒
* [前端乱炖](http://www.html-js.com/)：一个前端文章分享的社区，有很多优秀文章
* [正则表达式](http://deerchao.net/tutorials/regex/regex.htm)：一个正则表达式入门教程，非常值得一看
* [阮一峰的博客](http://www.ruanyifeng.com/blog/)和[张鑫旭的博客](http://www.zhangxinxu.com/wordpress/)：快速了解某些知识的捷径，但是如果需要深挖，还需要其他的资源
* 各路大牛的博客：这个太多了，就不贴了，知乎上有很全的
* 各种规范的官方网站，不懂得时候读规范

#### 历程

以前是做Java SSH的，半路出家做的前端，所以水平比较弱，遇到问题也比较多。基本上入门靠看书和[W3C School](http://www.w3school.com.cn/)上的教程，以及一些前端博客，如[汤姆大叔的博客](http://www.cnblogs.com/TomXu/)。以前也只是使用jQuery，原生js也没有太多的钻研，后来逐渐看了很多本动物书，比如老道的语言精粹等等。从这些书中学到了很多语言层面的知识。但这显然是不够的，所以我经常会去社区上看看大家在谈论什么，然后去看看相关的资料，感兴趣就会多找些资料看看，或者写一写demo。学CSS主要就是通过这种方式。后来开始更多的关注各路大牛的博客和一些比较深的书籍，以及关注一些新的知识和框架，并且不断地练手提交代码到github，这样也学到了很多知识。在实习的过程中，切身参与到实际项目开发之中，能学到很多在学校学不到的理念和思维，这点也有很大的帮助。不说了，我要去搬砖求offer了...

### [MrRaindrop](https://github.com/MrRaindrop)的学习经历

应[qiu神](https://github.com/qiu-deqing)的邀请分享一下前端学习经验，这里对前端知识体系架构就不做总结了，各位大神们的总结已经相当到位了，我就贡献几个个人认为还比较有用的链接大家研究研究就好，然后主要分享一下我在前端学习过程中遇到的问题和总结的经验教训吧，如果能帮到想要入门的FE初学者（我就姑且假定为本文的读者受众类型了），让他们少走点弯路，每走一步都知道自己下一步的方向，这是最好了。各位大神的总结和分享详见qiu神整理的[FE-learning](https://github.com/qiu-deqing/FE-learning)。

先说下，前端这个东西每个人都可以有适合自己的学习方法，这篇仅作参考，写的有点乱，各位凑合看。

#### 缘起

我是属于误打误撞进了前端，之前一直往做游戏的方向去来着，搞过游戏网站，玩过游戏引擎，比如unity，unreal这种商业引擎，捣鼓了几个游戏原型，不过自打研一进了实验室，直接就被导师派去写了js，导师给了我半个月时间让我写个基于百度地图api的数据展示页面，虽然这个时间还是相当宽裕的，不过之前没怎么写过js，也不会用地图api，于是我就一边啃着[《Javascript权威指南》](http://book.douban.com/subject/2228378/)（犀牛书）一边参考实验室前人留下的“代码”，总算是把功能都写出来了。那个页面算我的js入门作了，也是我前端学习路线的开始。

现在想来，虽然指派了去做前端，但是一直做下去并做好还是得靠兴趣维持，当然前端是一个趣味性十足的技术领域，而且社区每天都很“热闹”。

#### 项目，下一个项目

我个人认为前端的学习，初学阶段你可以完全脱离开书本，以**项目驱动**。虽然我个人是从犀牛书开始啃的，不过如果你没有充足的时间，或者觉得啃大部头乏而无味的话，还是别像我这样。当然了如果决定啃书最好是把书里的例子都跟着敲一遍的。我上研之前没接触过js，4月份还没开学呢就被直接被导师甩了个百度地图api的项目到脸上，接着就是各种ERP，地图数据展示，虽然换着花样来一点不重样，不过基本上都是前端的活，SSH和android开发也打过酱油，整个实验室就我一个人写前端敢信？富客户端SPA时代的后端就是一个restful接口，代码量基本都在前端啊，写的我一个人怎一个爽字了得...期间跟着导师感受了一把创业，每天从7点搞到晚上10点，也算是经历了一段快速成长期。

掌握一门技术先掌握它的大体框架，想一个能实现的点子，做一个能跑就行的demo，再去完善它的细节，等到demo完成了，对这门技术有了一个感性的认识，再去啃书，收获会大很多。我从开始原生js写到jquery，再到extjs，再到angularjs，从导师指定技术，到自己做技术选型，一个项目接着一个项目的练，就跟打怪升级似的。当然没有项目就去自己创造项目，动手实现自己的想法是件有乐趣和成就感的事。

#### 收集癖和知识管理

前端学习有个特点，很多东西都很零碎，分散，需要你自己去整理、归纳和总结。在微博、知乎上follow了众多的大神，你不仅仅是为了听八卦，大神们的只言片语有时候留下的是无尽的余味，很有可能一个不经意提到的一个词就成为你下一个学习的目标。**收集这些信息，善用google，提问，思考。**就像游戏里的收集要素，前端学习也是充满搜集要素的一个“游戏”，只不过你需要一个知识管理工具来充当物品栏和仓库，我所知道的大牛们无一不是知识管理工具的重度使用者。以前用的oneNote，那时候还没绑定到云存储，现在基本上用evernote，笔记已经累计到1200+篇。书签一直打算用delicious，因为它是基于tag管理的，但一直没用起来。当然重点不在于这些工具，但是趁手的工具可以提高你的学习效率。最关键当然是随时**保持旺盛的学习欲望**，你的目标是了解有关前端的一切（当然不是所有都要掌握，因为毕竟你的精力有限，而且现实的说这也不太可能）。

#### 跟对神

这个可控性貌似不大...跟对老大这个就不多说了，一定程度要看造化。不过话说回来，多跟身边的高手交流是王道，这个高手不一定要多高，但是一定要对技术有热情。研一的时候热情高涨，每天7点进实验室门，然后发现有个家伙居然比我还早到。后来发现这家伙上午就走了，下午又来了，而且导师对此习以为常，原来这家伙晚上不睡觉通宵写代码，上午才跑回去睡。后来经常和这位神讨论问题，每次感觉经验值蹭蹭蹭的往上涨。然后实验室还有一位神，被前面这位通宵神形容为“只能望其项背，一直在追赶，从来没赶上”，两位神的特点都是什么都了解一点，所以什么都能跟你讨论得起来，我有段时间做了个读书计划，从c/c++到vc/mfc再到unix网络编程，最后一路看到java核心技术和MSDN上的C#编程指南，和神们也能扯得很high了。

总之就是这两位神把我拉进了坑，或者说从一个坑跳进另一坑，虽然两位神都不是搞前端的，不过技术之间总有相通之处。

#### 读书

读书，多读书，读好书。在[刘未鹏的博客](http://mindhacks.cn/topics/learning-method/)里看到过一个公式，**你第一个月的工资等于之前买过（读过）的技术书价格总和**（这里说的技术书指那些经典的公认的好书）。讨论这个公式的正确性似乎没什么意义，然而它的合理性是毋庸置疑的，那就是多读经典技术书。最极端的一个例子，google的徐宥在[我的大学](http://blog.youxu.info/2009/08/06/my-undergraduate-year/)里面说他扫荡了图书馆的整个TP312书架...对于前端的经典书籍，后面列了一个我收集的[前端书列](http://book.douban.com/doulist/13701898/)（如果有遗漏的前端经典好书，还请留言告诉我），有条件可以尝试刷一遍这些书，我也是在找完整的时间去啃完它们。之前说的，前端知识点松散，收集零散的知识点，从博客里快速学习等，这些只是前端学习的一个方面，如果你要想深入理解一个知识体系，了解它的来龙去脉，对它建立系统认识，读经典书还是必不可少的。

我从最开始啃完犀牛书，然后接着去看了其他一些和前端干系不大的经典技术书，再后来通过实验室的项目和自己弄的一些小项目逐渐对前端领域比较上路以后，又看了《Javascript模式》、《Javascript设计模式》、《编写可维护的Javascript》，后来了解到node并开始用node搞点小玩意儿，又看了本《NodeJS up and run》和《Mongodb权威指南》，不过感觉前者略坑。那会儿朴灵那本深入浅出（晒书么么哒）还没出，后来出了就去图书馆借来看完，这么看下来感觉还不错，不过感觉看的还是偏少了，还需要继续刷（参照上面的书列）。

#### 前端的定位

前端的定位关乎到你需要吸收什么样的知识和技能，决定在技术世界里你对什么需要格外敏感。如果你认为前端仅仅停留在切页面，实现交互和视觉的要求，那你对前端的认识还停留在初级阶段。阿里终面的时候我问了考官这么个问题：前端技术日新月异，范围越扩越宽，标准越来越丰富，似乎任何一个触角都能伸出很远。怎么给前端一个合适的定位？考官给我分析了半天，然后总结成一句话，就是用户和网站的联结者，用户体验的创造者（原话不是这样，但大体是这个意思）。也就是说前端的终极目标其实就是创造用户体验，提升用户体验，**以用户体验为中心**。不管你是从交互设计上下手，还是从性能优化出发，或者改进工作流提升工作流效率，最终都是为了创造和提升用户体验，最终都要体现到用户体验这一点上来。我认为这个总结非常有道理（当然“用户体验”这个词太宽泛了，并且不仅仅是前端工程师的范畴，比如开发后台的时候对一个数据处理过程进行优化，提升了整体性能，这也是对用户体验的一个提升）。

现在的前端工程师做到一定阶段不可避免会接触到很多比切页面、实现视觉要求、实现交互等更深入的问题，比如前端自动化、图像编程、性能优化等等，再往后推一点就是PHP/JSP/ASP/nodeJs，过去后端模板一般属于后端的范畴，现在随着[前端架构的演进](https://github.com/lifesinger/lifesinger.github.com/issues/184)，可能会让你去写后端模板的代码，需要用到后端语言（PHP/Java/C#等），这就是所谓**大前端**（然而这与前端的定位并不是相背离的，大前端处理的依然是与用户接触的部分，仍然是对用户体验的优化）。可能最常见或者被谈论最多的就是node，其实这几种技术选型都可以，bat三家据说百度用PHP比较多，阿里用node比较多。

玉伯在[他的博客](https://github.com/lifesinger/lifesinger.github.com/labels/blog)里提过所谓全端是横向的，全栈是纵向的。**全端**即所有的终端说白了都是前端，因为都关乎到用户体验，直接和用户接触。适应多终端的开发，要求你在web前端的基础上，可能还要去扩展android开发和ios开发的知识，好在由于hybrid开发方式的流行，对使用native语言开发的技能会要求的不那么深入。

**全栈**可以说是最适合初创公司的一种发展类型，广义上认为是从前端干到后端，从开发干到运维，这种就不说了，一般人应该不会想要去往这个方向发展，想要成为这种意义上的full-stack dev的，可能用不着来看我这篇文章了；而狭义上的全栈特指使用js语言从前端写到架设在nodeJs上的后端，前后端统一语言，统一编程模型，甚至公用同一套代码。更多了解全栈开发可以看看玉伯这篇[说说全栈工程师](https://github.com/lifesinger/lifesinger.github.com/issues/185)。

以上是我对前端以及衍生出来的技术路线的一些浅薄理解，学习一个领域掌握它的整体上的走向和趋势还是挺重要的。另外如果想要对前端学习方向、职业成长路径有一个整体的认识，推荐看看拔赤总结的这篇[前端开发十日谈](https://github.com/jayli/jayli.github.com/issues/1)。

#### 最后

贡献几个对前端学习、面试有帮助的链接：

* [前端面试问题合集（Front-end-Developer-Interview-Questions）](https://github.com/darcyclarke/Front-end-Developer-Interview-Questions)
* [前端技能汇总（JacksonTian）](https://github.com/JacksonTian/fks)
* [另一张前端技能汇总图](http://www.f2er.info/)
* [前端那点事儿（书列）](http://book.douban.com/doulist/13701898/)

### byr论坛yiyizym的建议

与grunt相比，学习gulp会比较简单

做SPA的话，推荐backbone.js和 backbone.marionette.js

翻墙不用折腾，花十块钱买一个月的 红杏。

把基础打扎实了再学这些都没问题。

html 没什么好说的，有空学学html5。

css 尽量看[文档](http://www.w3.org/TR/CSS21/cover.html#minitoc) ，因为很多中文资料都各执一辞，看多了反而会糊涂。

有个[网站](http://caniuse.com/)可以查找html/css标签、属性在各个浏览器中的支持情况，挺好用的。

javascript 就看 javascript高级程序设计 。不过这么厚的书看过就会忘。对javascript核心概念的讲解：对象/原型链/ 构造函数/执行上下文/作用域链/闭包/this，[这里](http://weizhifeng.net/javascript-the-core.html)有篇不错的文章。

有闲情可以看看 ecmascript 6，计划明年6月就发布啦。[阮一峰的网站](http://es6.ruanyifeng.com/)有入门资料。

jquery 有很多 API,[这个网站](http://www.css88.com/jqapi-1.9/)可以方便查到。有时间弄清楚jquery deferred 的用法。

多给 sublimetext 装插件，比如说检查代码错误的，新建目录文件的，整理代码的。