# IT行业都有哪些职位

目的：了解不同职位的主要工作，如果有问题或者需要沟通需要找哪个同事。（对于web前端自己多了解）

1.产品2.UI3.CSS4.JS5.后端（Java/php/python）6.DBA(mysql/oracle)7.运维（OP）8.测试（QA）9.算法（分类/聚类/关系抽取/实体识别）10.搜索（Lucene/Solr/elasticSearch）11.大数据工程师（Hadoop）12.Android13.IOS14.运营

\***薪水都以北京为例**\*运营和市场都有更细致的划分

## 产品

1工作内容：了解用户需求，做竞品调研，画产品原型，写产品文档，讲解产品需求，测试产品Bug，收集用户反馈。

2需要技能：PPT，Word,Axure,XP,MVP，行业知识，沟通

PPT是最容易被忽视的技能，很多时候很多人都把产品经理=画原型，如果你这么想，再做五年的产品经理也是一个画原型的。产品经理更多的是理解一个行业，熟悉这个行业的用户，知道这个行业有哪些没有满足的需求，寻找解决方案，对比行业，找到自己的切入点，最终落地成：哪些功能要做，哪些功能不要做。

这些都是要在PPT里写清楚的。所以，PPT并不是简简单单的一个工具，而且是用这个工具把你对这个行业的了解和分析和解决方案展示出来。

XP是敏捷开发，不会敏捷开发的产品经理很难说会是一个好的产品经理。敏捷开发对于产品经理来讲，是逼着产品经理去用Story的方式去思考，去考虑自己每一个Story的价值是什么，所有的Story优先级是什么。

MVP是最小化验证产品的缩写，这在精益创业里是很重要的概念。也就是说，你臆想了一个产品，他有了什么样的功能，但是！请注意，我们认为所有没有经过用户验证的功能都是在扯淡。所以，你其实是设计了一个扯淡的功能然后交给一堆人去开发，再交给人去运营，直到一个月或者是三个月以后才能拿到反馈结果。我们等不了，所以你必须在程序员开始启动之前，先想办法验证自己要解决的问题是存在的，要解决的需要是用户确实需要的---你有无数的办法可以这么做，真的。好比是，并没有先直接做线上的网站，而是直接去贴吧发贴，用QQ群，在论坛直接做，去观察这些弟子的行为，然后去修正自己的产品设计，我发现了他们需要日报，需要点评，需要有班级的概念，需要有学号，需要有首席大弟子等等等等这些。然后我会把这些需要分析整理之后加入到网站的设计里去，这就是MVP的价值--当年穷疯了每个决策都慎重的时候，就会特别喜欢这种方式，土豪除外，只会浪费时间而已。

Word是产品需求文档，我自己不太喜欢Word文档，更喜欢XP的Story，不过会一点也没有关系。

行业知识是产品经理最重要最重要的技能，没有之一，你可以不懂PPT，不会Axure，不会Word，也不会什么XP和MVP，但是绝对不能不懂一个行业。某种程度上来讲，产品经理拥有的其实是快速了解一个行业的能力。现在的互联网已经和以前的不一样了。以前的产品经理懂点儿交互，知道什么是表单，知道点IT术语就能混日子了，现在必须要懂一个行业，地产，金融，汽车，教育，餐饮，医疗等等各种行业，你都需要了解这个行业的模式，如果你不懂，你就是一个画原型的人。

沟通也是产品经理的必备技能，出门不带沟通技能，一定会被程序员砍。一方面是要跟用户沟通，一方面是要跟程序员沟通，一方面要跟领导沟通。

3发展前景

不用说，产品经理绝对是一个炙手可热的职业，而且，TNND的几乎是零门槛，这表示，是个人都可以说，我想成为一个产品经理。然而，好的产品经理很多么？并不多，所以，好像围棋一样的，入门容易，学精难。

而且，似乎并没有一个系统的教程（这真是一个很难让人相信的事情啊，这么一种职业，没有大学的知识培养，甚至都很少有标准化的职业化培训，全靠业界手把手培养和自发的交流，就能够拿着无数DS程序员根本就想像不到的薪水--产品经理真的跟DS有点儿不搭边的，当然，我大修真还是在努力做产品人员的培养的）。

产品经理的发展前景是很赞的，只是越来越体系化了，而且很容易出人头地。

三个月到半年左右就可以算是迈入这个行业啦，如果遇到P2P这种对于专业属性要求比较高的，随便在一家公司混点经验，跳到另外一家新的公司就是一个产品总监：因为并没有其他人又了解金融又了解互联网。

产品助理-产品经理-高级产品经理-产品总监-CEO

5.哪些行业适合做产品经理

IT界：程序员，UI，QA，运营，市场其他界：无限制

PS：想了想，还真是没见过身边有做运维的转产品经理的。

6.职业限制

很多产品经理都很怕跟程序员沟通。而且经常会苦恼，觉得自己设计的功能，程序员做不了，做不出来，做的太慢，做的太差。惟独不会觉得自己设计的太烂。

所以，很多产品经理都会认为不同技术是自己的一个弱项，我曾经整理过一个产品经理需要了解的技术术语。s

## UI

首先写软文，又要写成干货，又要带广告真心不是一件容易的事儿。我在之前如何在一个月之内从零基础成为初级CSS前端开发工程师？-xdyl的回答的贴子中，写了整整一个星期才写出来。我写小说都是分分钟好么。

可是没办法。一个是确实需要跟大家介绍一下每个职业的比较和特点，一个是也确实需要告诉更多的人，写代码真的不是看视频看会的。

好了我接着说。

1工作内容：收到产品原型，给原型上色，偶尔会自作主张调整下原型的位置，出不同的风格给老板和客户选，然后听他们的意见给出一个自己极不喜欢的风格，最好给Android，IOS或者是CSS做好标注，还有的需要直接帮他们切好图，最后要练出来象素眼，看看这些不靠谱的程序员们有没有上错色或者是有偏差。

2需要技能：PS，Illustrator，Sketch，耐性，找素材

PS是最基础的技能。各种图层通道蒙版神马的细节的东西就不用说了，很多时候很多公司都愿意出所谓的高保真图。什么叫高保真图，简单说就是不折磨产品折磨UI的破玩意儿。就是把所有的图都画出来然后对着UI图来改，偶尔还会要求做出来网页能够点，对于这点我不想再吐槽了，总之，PS是少不了的技能，基本上很多时候其他人都会以为你只需要用PS就能搞定一切。

然而PS始终只是一个工具而已，所以在这里，PS代表的更多的是一种方案的选择。比如什么行业用什么配色，怎么样对齐，怎么突出重点，PC下应该是多宽，手机应该是多宽，等等等等。

ILLustrator最近似乎是用的少了，我不太确定，但是做矢量图以前是就是靠他的。什么是位图什么是矢量图，自行百度就好。以及还有CoreDraw，不知道现在是否常用。然而，慢慢的似乎Sketch流行起来了。

SKetch和Illustrator和CoreDraw之间的差别我说不上来，我记得我的UI小姑娘跟我讲这个工具很好用。

耐性必须是UI必备的技能，因为有的时候，老板跟你较真起来，一个星期不停的反复修改一个首页是很正常的事儿。可能颜色要从红换成黑再换成绿。所以你必须要有耐性，除非你做的设计没人管，你说了算。而且，UI这东西，是个人都有自己的审美观。更不用说之前朋友圈的测试，人和人看到的颜色真心不一样。所以你必须要有耐性，这是很重要的。

找素材的能力也是必不可少的，你手上必须要备几个常用的素材网站，大部分的人都是在别人的素材基础上改来改去。

说到这里，其实忍不住想加上一句。UI设计师最重要的能力并非是以上几种，而在于是理解产品经理的需求--前提条件是产品经理很明确的说出来了而且讲清楚了需求。

比如说，一个页面的重要是什么？最想引人注目的内容是什么？应该配什么样的图？加上什么样的元素能让网站的风格更醒目一些呢？

这里大部分其实都是原型设计师要做的事儿，我一直有一个观点，就是UI其实最应该做原型，只有这样，才能让产品经理着重去考虑功能点，让设计完美传达产品的功能点。然而原型设计师这个角色慢慢的被消融了，已经变成了产品经理了。所以大部分产品经理都以为自己是画原型的，然后大部分UI都会自作主张改掉原型，因为没有美感。所以通常你会发现美感说有了，也有那么一点儿，可是总觉得会有一些不对。

我对他们的要求就是，先别管美不美，你先告诉我这张页面每个模块想要重点表达的内容是什么，能不能让人分清主次。做到了这一点，再去考虑美感。

3发展前景

UI的发展前景是很赞的，在两到三年前，甚至都有人推动让UI来代替产品经理来驱动。某种程度上来说，UI其实最好的发展方向就是UE，也就是交互，然后就是所谓的用户体验优化师--坦白的说我并未觉得有多少公司会真正的设置这个职位，因为太泛了。但是对于现在火热的O2O来说，无论是线上还是线下，都希望提供一个极致的用户体验，所以我说不好。但是大部分的UI都应该去朝着交互的方向发展，而不是去切页面。。。

UI薪水其实跟产品经理在前期差不太多，中期差点。后期相差比较远，一般来说：

0~6个月：4K~8K6~12个月：6K~12K1年~3年：10K~20K3年~5年：16K~30K5年~：24K~

UI的成长时间大概是在两年左右，基本上PC，IOS，Android，H5全部都做过一些项目，设计过几个Logo，做过一些专题和Banner图，才算是比较完美报。

然而想要做的更好一些就有点困难了，除非你的遇到了一家公司，对用户体验要求极致，然而那个时候，并不仅仅是UI和UE相关的内容了。

4.入门门槛

UI对入门门槛要求还算是比较高的。首先你得有一台Mac。。大部分的情况下你都需要Mac。然后你要懂之前提到的常用软件，UI的交互规范，配色等等等。

6.职业限制

UI这个职业的限制在某种程度上就是不太清楚自己应该朝着UE发展。所以时间长了工作就会变得单调，两年或者是三年之后原地踏步走没怎么有进步的人很多。像Metro（扁平化）或者是materialdesign这样的风格一不熟悉二不了解三不想尝试的人还是挺多的。

很多人做了很久其实还是分不清产品经理想要在一个页面上表达什么--这有时候真的并不是UI的错。

## CSS

1工作内容：产品设计好原型，UI做出来了效果图，剩下的就是CSS工程师用代码把静态文件写出来的。

所以CSS工程师和OP一样，都是可以在不用了解业务需求的情况下就能完成工作的唯一两个特殊职业。一般来说，CSS工程师并不存在，写CSS的人多少要写一些JS代码，然而确实也会有一些妹子真的就是学不会JS，有一些汉子真的就是写不出来JS，所以我还是把他们两个拆开了。

CSS工程师要考虑更多的就是兼容性，简单说，IE内核和非IE内核（WEBKit），IE10以下和其他。我是强烈建议新人不要去学IE10以下的Hack的写法的，让IE10以下的，不支持Html5的都去死吧，表再来跟我说用户还在用IE678了。

当页面切完以后，另一个工作就是去帮助菜鸟后端和某些JSER把页面又重新套错了，什么是套页面 http://www.zhihu.com/question/22689579/answer/69899644。

**2需要技能：环境【IDE（WEBStorm，Sublime，EditPlus），源码管理（SVN/Git），WEB服务器(nginx)】基础【PS，域名，Html，Html5，CSS，CSS3】扩展【自适应，响应式，Bootstrap，Less，Flex】**

无论是哪一种程序员，学会搭好的自己的环境都是第一步。选好IDE，选好源码管理工具，学会每天把自己做好的Html文件发布到服务器上可以直接访问。这都是最基础的技能，记着这些东西并不难，你只是需要不断的去练习去熟悉它而已。所以从第一开始就去适应它。

会用PS是因为一是要切图，二是要制作雪碧图。这儿简单说一下，很多人都会说，切图这个活倒底分给UI还是分给前端。我一直都觉得这是CSS必须要做的，因为只有CSSER才会知道怎么切合适。然而大部分Android和IOS人员都不具备切图的能力，这也是我一直很无语的工作方式，反正，我要求我的Team,无论是Android还是IOS都必须要会自己切图。

另外再解释一下所谓切图和切页面的概念。切图指的是把需要用到的小图标了神马的切出来。切页面一般指的是就是把整个效果图做成Html网页。

自适应和响应式是前端人员必须要懂的概念，也是必须要能做到的。毕竟现在是各种跨屏时代，以及各种分辨率都层出不穷，H5又是眼下最火红的职业和技能。

曾经有过工作半年或者是一年都不太懂什么是自适应和响应式的，这并不好。

Bootstrap，Less和Flex也是CSS工程师必须要会的。

学会Bootstrap更重要的还是要去学习它的思想，这是很重要的一点，它帮助你开阔视野，你才会明白，靠，原来可以这么写。

Less也是一样，在推荐less和Sass之前我犹豫了很久，然而看到Sass要Ruby环境就有点不喜欢了，虽然Bootstrap已经转向Sass，但是在一般的项目中，Less也够用了。所以还是果断的推荐Less了。

Flex是一个好东西，我了解的还不多，在真实的项目中还未有使用过，不过挺看好的。哦哦，想起来了，CSS的知识比较零散，要死记的东西其实是比较多的。

3发展前景

CSS的发展前景最近几年好的不行。在Android和IOS大行其道的时候，一大部分人都转去做了这两个行业，然而突然之间，因为微信的缘故加上JS的魅力，H5突然又火爆起来，所以呢，CSS整体来说还是一个挺靠谱的职业。最关键的是，CSS入门最快的职业啊，几乎零门槛，只是转JS还是挺不容易的。

CSS薪水如果纯粹只是做CSS的话，12K到15K基本上就是封顶了

我的要求比较高，所以所谓的独立做项目就一定是独立做项目，不用怀疑这一点。包括自适应，响应式和Bootstrap。所以。想要拿高薪，CSS都必须要学JS，没有其他的途径。

成长路径：

CSS工程师-初级JS工程师-JS工程师-前端架构师-前端Leader（IOS，Android和JS通吃）

不要相信NodeJS，那只是玩玩而已。不要想从前端转后端，太难了，后端和前端是两个不同的概念，特别是对基础知识要求的特别高。

CSS和JS最好的方式就是IOS和Android通吃，精通一种，然后其他的能做项目，并不需要考虑其他的选择，推荐再学点其他脚本语言，首推Python，永远不要去学PHP

6.职业限制

CSS的职业限制有三个，一个就是不去学自适应和响应式，一个是不去学框架，另一个就是学不会JS。这三点都是很重要的突破。然后这个职业呢，怎么讲，在一家公司里可替代的程度比较高，并不算特别重要，做不了核心岗，也不存在什么代码交接。你就算代码写的再烂，也最多就是花一个月时间重写。。。

所以如果你不想做JS，你就真的真的发展受限制了，一定要记着，CSS只是你的一个跳板。另外，因为零门槛，所以你其实还是有很多计算机相关的知识要补的，这是一定的，没什么可商量的。

只是做前端还好，如果想做后端，只是不想花那么长的时间才进入这个行业，就必须要从CSS做起，跟着用自己的其他时间恶补计算机基础了。

## 前端JS

JS说起来必须是一个神器，这个当年10天内被开发出来的神器，以一种谁也想象不到的速度快速发展，它击败了JavaApplet，逼死Flash，当Android和IOS看似一统全球的时候，JS慢条斯理的和Html5一起蚕食App市场，甚至还出来了NodeJS这种连后端都不放过的残暴东西。

原生和WEB的两种方式一直在争论不休，而我自己在三年前就是绝对的WEB支持者，windows对于JS支持的更彻底，只是Winphone一直不给力。

好吧，对于拿NODEJS做后端工程项目我还是有点不情愿，然而提供各种辅助工具什么的还是挺不错的。JQuery，Ajax和RequireJS和Angularjs和PhoneGap都是一些标志性的转变，还包括ReactJS。JS简直了，简直了，简直什么都不放过。

所以JS简单么？我带着你们了解一下JS工程师是什么样子的。

1工作内容：JS工程师其实分成两类，在之前讲CSS的时候已经提到过，一个是套页面的，一个是前后端分离的。对这两个概念还是分不太清的，可以回过头去看CSS的部分。

对于套页面来说，JS工程师的职责非常简单，一个是做一些简单的动画和交互和验证，轮播图，弹框，验证用户名是否为空等等。另一个就是通过Ajax取一些后台数据，然后在页面上展示出来。

那个时候的JS工程师并不被太多人重视，很多时候一些后端的人把前端的工作也做了--除了写CSS，大部分的后端工程师写点JS还是不成问题的。

可是前后端分离的出现，彻彻底底的改变了前端的世界。前端居然有架构了，前端居然也有“编译”的概念了（你能想象么，JS的项目居然不能直接打开源码直接使用了，必须要经过编译才可以）。前端工程师和后端工程师只通过Json数据交互（感谢Nginx提供了一个完美的跨域解决方案，再也不用什么Jsonp了），然后服务器端的同学发现：好简单啊，我对Android，对IOS，对Html都只需要提供一套API就够了~~

前端同学更开心，再也不用去搭建各种悲剧的开发环境了！前端变的很轻，而且控制欲很强（AngualarJS好赞），页面跳转神马的SoEasy，而且也有了Bower这样的包管理工具，还有类似于JSP自定义Tag的指令，MVC，还有了Service的概念

所以我们现在说到的，基本上就是前后端分离的工程师，他们的工作职责就是和IOS&Android的工程师一样，跟后端定接口，确定框架和架构，分模块，项目结束之后修复线上Bug，不停的跟用户说：在我这里是正常的，你换个浏览器试试？不行就清一下Cookie，再不然就换台电脑。

2需要技能：环境【IDE（WEBStorm，Sublime，EditPlus），源码管理（SVN/Git），WEB服务器(nginx)】基础【Http，REST，跨域，语法，组件，F12，Json，Websocket】框架【JQuery，AngularJS，Bower，RequireJS，GruntJS，ReactJS，PhoneGap】业务【金融，教育，医疗，汽车，房产等等等等各种行业】第三方【微信，QQ等各种第三方登录，支付，IM，地图，语音，视频，图片】

环境不说了，参考CSS的。基础跟CSS差别就好大了，要有一些网络协议的基本概念，要了解什么是Http，什么是Https，什么是WebSocket，什么是Rest，各种JS的语法（太复杂的完全不用管，别去管什么闭包什么Prototype什么This的指向），Json是必不可少的东西。

框架就多了去了。直接拿JQuery来学习就好，表去太在意什么原生语法，大部分场景你都不用担心引入一个JQuery的库导致系统变慢，AngularJS真是一个好东西，表太纠结一些细节，Bower，RequireJS和GruntJS这些会有一些前端大牛帮你搭好框架你直接用就好了。想做点App就用PhoneGap，想尝试一下ReactJS就去试吧（我很想试一下然而看到有Render就看不下去了）。

必须提醒一次，如果你是从CSS转到JS的，这里强调的就是，你必须要懂业务逻辑，这跟CSS完全不需要懂业务逻辑简直是天壤之别，很多时候你并不是因为技术而无法学会JS，而是缺少对于业务逻辑的理解。

当然啊，这并没有完。还有无数的第三方库等着你，做微信呢你必须要用WX的SDK，要用百度地图呢就必须要用百度的SDK，还有各种支付啊神马的。

所以。JS就是一个真正的程序员，一个真正的工程师，不但要能把代码实现，还需要理解需求，需要懂得编码规范，需要知道如何控制项目流程和进度风险，需要修复线下Bug，需要调优，需要做版本管理，需要制订接口，需要编写技术文档，还需要做新技术的调研或者是带新人。

3发展前景

我说了那么多，你们对JS的期望也很高了吧。坦白的说，有点高的离谱！JS是唯一一个可以在两年之内薪水到20K的职业。虽然工作五年或者是七年的JS在薪水上差别并不太大，但是也已经让很多人难以置信了好么。

好多人在别的行业，工作十年都拿不到10K啊亲。

0~6个月：6K~10K6~12个月：8K~14K一年~两年：12K~20K两年~五年：18K~30K五年以上：24K~

JS的入门相对来说比CSS难的多，但是他的好处是学会一点是一点，并不像Java一样，绝对是前期Gank型英雄，到了6级就能四处杀人。Java是后期，要到16级以后才能出山才能够震撼全场。

所以JS相对来说比较平滑，而且现在学习的曲线也比较清晰，慢慢在业界都形成了相对完善的框架。JS可以选择前进的路有很多种。

成长路径：

JS菜鸟-JS初级工程师-JS中级工程师-JS架构师-UED-前端Leader-前端大牛

大概这个样子啦，JS工程师在后面有一点后续无力，一般来说，我还是强烈推荐JS，IOS，Android三者通吃，在我的带的Team中也一直要求他们这么做（虽然并未能完全实施成功）。

这三者有太多相似之处了，到最后你会发现，语言只是一种工具而已。要做的事情，其实是差不太多的，并不需要对所有的语言都特别熟悉，有一种打底的，其他的都可以随意去玩了。

5.哪些行业适合做JS

IT界：CSS，然后是所有的都可以其他界：所有的都可以

好蛋疼的结论然而事实上就是如此，这波JS的红利受益于微信，以及移动端的普及。我不确定在三年或者是五年之后会不会还是这种状态，然而三年和五年已经足够可以改变你的人生了，所以，你在等毛线？

6.职业限制

JS的职业限制也挺多的，主要就在于学习的方向上很容易误入歧途。比如说一直坚持喜欢用原生，各种喜欢写一些奇奇怪怪语法（我把我看不懂的语法都归结为奇怪的语法），各种回调等等等等。

学到一些好的框架其实很赞的，对于大多数人来讲，就够了。只是缺少一个清晰的指引。有的时候你会发现，你对Http了解的比较少，会是一个瓶颈。你对数组啊，循环啊不了解，是一个瓶颈。你对MVC不了解，也是另一个瓶颈。对Json不懂，不知道怎么定接口，也是个问题。

这些都学会了，也会遇到一些问题。就是JS的工程师三年，五年，七年的差别，大多数人都不算特别的大。所以总会有人想学后端--但是我绝对不赞成学Java，NodeJS也可以不去学，不如去学习Python，更不要提PHP啦（我真心不喜欢PHP，不知道为嘛）

或者说，后端并不是语法，而是牵涉到算法，架构，性能，业务，选型等等跟前端几乎没有任何关系的知识，所以学习一门脚本语言，能做点简单的东西就好了。

还是需要把注意力转到IOS，Android上去。

## 后端（Java/python/go）

说到后端，其实我主要想说的是Java。C和PHP这两种语言我都不喜欢。

我之前在贴吧跟Java吧的吧主一直在撕逼（原因很简单，我说教大家学Java，一个月收400块钱，他说我是骗子封我贴。然后我说好吧，我不说教大家学Java了，我来给大家解决在学习过程中遇到的困惑，他说贴吧不能发问答贴。我说行，那么我就写一些新人学Java必须要学数据库，数据结构和计算机网络，于是一群吧主过来喷我说，我没学过这些我也照样学会Java了啊，什么多线程什么继承等等，我无语了解释说Java语法不重要，重要的是要学会后端的架构，要懂算法，要懂业务，要懂系统的扩展性，要会调试程序，于是吧主们就把我封了，我很不爽，就新开贴子跟他们撕逼对骂--我从来不是一个重身份的人，我骂人会很脏，所以如果看到这个贴子觉得我很厉害那么你瞎眼了，我就是一个不喜欢就说，谁喷我一脸我喷谁一身的性格，现在的结果就是Java吧的吧主每隔10天就来封我一次-哈哈哈哈哈比闹钟都要准，所以三个月过去了，我带出来很多CSS和JS的学员，然而Java的学员并没有多少。所以如果有人去Java吧替我骂一下那些XX吧主并且截图给我看，我会很开心很开心很开心，说不定就会给你们开小灶哈哈哈哈。）

之所以说这些，一方面是400多的赞让我觉得有点羞愧，另一方面也是想强调一个概念，学后端，学会语法只是开始而已，最后一个就是我是一个小人，谁欺负我我就想欺负回去。

我想想该怎么描述后端的工作。后端跟前端是截然不同的，之前讲过。前端是Gank，后端是大后期，要等到16级以后才能V5起来，而且我非常不推荐前端去学后端（所谓的全栈工程师完全是扯，我有时间会写一下，为什么不要去做一个全栈工程师）。后端要积累到足够多的项目经验，才能够成为一个靠谱的后端工程师。我觉得。我来举一个跟着我线下半年的小培宇的例子就能简单说明一下后端的工作。

小培宇是第一个来到我大修院面试（嗯，最初我是给他们发工资然后带他们学习的）的人，跟我讲他是考研失败，差了几分，然后也做过点项目，我随便问了几句就知道了他的状态：人挺聪明的，但是在学校肯定玩的疯，所以问点排序算法还是能够答的出来，数据结构也懂一点儿，LinkedList和ArrayList删除数据谁更快也能答的挺靠谱的，但是绝对绝对没写过一行工程代码。

他打动我的那句话就是：不在乎工资多少就想多学点东西。很好，我默默的点个赞，因为我本身就想把自己这几年积累的经验和知识和走过的坑整理出来，告诉互联网的新人，所以也不抵触带新人,坦白说，愿意像我这样带新人的公司，几乎没有，带新人真不是一般的累。幸好我之前在各种公司中都带过各种新人，好的坏的都带过，所以还算是熟悉。

于是我给培宇精心设计了他的学习曲线：

1.先搭建基础环境（Maven，SVN，Eclipse，jetty,SecureCRT,Linux，Mysql）

2.做简单的CRUD（DAO-自己封装的数据层，junit,Log4j,Rest,Spring,SpringMVC,Json,JsonTaglib）

3.做一个相对复杂的系统DB设计，接口设计，项目部署，错误提示，Bug查找，怎么打系统日志。

4.做了一个微信相关的项目，了解微信的API，交互方式，Cookie，拦截器，AOP，登录系统的设计，命名规范等。

5.拆分Service，将Home和Service分开，使用RMI调用，实现各个层次之间都可以完成分布式的部署，使用Tuscany（真心喜欢Tuscany）完成SCA。

6.使用MongoDB完成地理位置的搜索，短信，图片上传，云存储，使用Tiles来配置页面模板。

这些内容他花了将近三个月的时间。对他来说已经是学会了很多东西了，这三个月是几乎没日没夜的学出来的，要知道他之前压根就不知道什么是Spring，生成Json和套JSP的区别我骂了他好几次他才弄明白，经常会遇到一些Maven或者是Tuscany的报错不知道该怎么解决，数据库字段的规范和接口规范常常被我黑的体无完肤。很多东西都只是知道个皮毛而已，你们自己说说，学会Java语法算什么？

这还是有我来带，有我来教，有我给他定制合适的项目教给他去做，如果没有这些，你们自己学，学会我说的这些东西要多久？别的不说，学会怎么打日志，怎么根据线上的报错去找错就不是一个月两个月能解决的。做为一个工程师，应该明白，很多时候要学会正确的路怎么走，还必须要知道错误的路是走不通的。正确的路大概就那么几条，错误的路呢？

为什么很多时候我看到错误日志就会明白是什么地方报错了，是因为我之前花了无数的心血和心力在查找这些错误上，所谓的经验就是这样，看的多了，一眼就知道大概什么地方报错了，然后随便百度下，就能找到解决方案，跟着去尝试倒底行或者是不行。这些是看书，或者是看视频能教会你的么？

到现在为止培宇已经跟了我快半年了，还是被我骂的狗血喷头，别的不说，就是接口的Wiki文档和代码保持一致，他都会经常犯错---这跟Java语法有什么关系，然而不经过一个好的训练，想做的很好，很不容易。

其实他接下来要学的东西更多。Memcache或者是Redis，ActiveMQ或者是RabbitMQ或者是QPid，Mybatis或者是SpringJDBC，Struts或者是SpringMVC，我告诉他的只是一个我们在项目中经过实践的，认为最合适的架构体系，然而他并不知道是怎么选择的。他必须要把这些相关的选择都有所了解，然后才能成为一个架构师。这个时间，如果一直跟着我，我觉得应该是在一年到两年左右。

这是一个横向扩展的内容，在这个时候我还没有要求他去看一些深层的东西。只是需要他停留在会用的状态就可以。在会用这些技术之后，再去了解一些自己喜欢的技术的细节，不成为一个只会使用工具的码农，所以培宇问我还需要多久才能达到我的水平的时候，我其实并不想打击他。我也是很刻苦努力的人啊，曾经无数个日夜也是默默的去一行一行代码去用最笨的方法调错，并没有人告诉我怎么样是正确的只有靠一个又一个的项目总结出来的经验。

而且我还会一些Drools，CRM，Lucene等等一些和架构师关系不大的事儿偏算法一些的东西，毕竟当年也学过点数据挖掘机器学习之类的内容。

就算是学会这些了，对于一个后端工程师来说就够了么？不不不，还需要学习JVM优化，监控，部署流程，发布流程，项目进度管理，代码重构等等等等。

所以，你们自己算算，这些东西如果都学会，一个Java工程师要多久才能成为架构师？

然而我还是对带培宇很有信心，首先他相信我，他愿意学，跟我当年一样，不怕苦不怕累，人也够聪明，做事也有责任心，其次我知道他应该怎么走这条路，先做什么，再做什么，哪些该花时间和精力，哪些不该花。

我希望他能够在一年之内就成为一个架构师。就如他在三个月和六个月之间独立做项目已经不成问题了一样（记着，我说的是独立做项目，自己设计DB设计接口设计架构完成需要的功能，从设计到实现完全自己来）。

我也希望我能够帮助很多和培宇一样，有实力有能力只是没有遇到我的那些人，这也是我为什么在知乎发贴的原因，IT技术的培训，哪些培训机构能做到这一点？

这个真实的小故事，就是想跟大家提前说清楚，我对后端的要求有多高，这也是后端特别好玩的地方。你必须要会很多种框架，有足够宽广的视野，还需要有足够多的项目经验（做金融和做地产是两个完全不同的概念），还需要懂项目开发流程以及快速定位线上问题的能力。

这些，就是我说的后端的主要工作内容了，这也是为嘛我说到后端的时候，大部分就是在指Java，而我说Java的时候，基本上是只指后端，根本就不是指Java的语法。更不是说是Android。

现在明白为什么后端是大后期了么，为什么不建议前端学后端了么。后端要懂的东西，太多了。言归正传，我来讲一下后端工程师的相关内容。

1工作内容：

大部分的后端工程师都停留在功能实现的层面上。这是现在国内二流或者是三流的公司的现状，甚至是在某些一流的公司。很多时候都是架构师出了架构设计，更多的外包公司根本就是有DBA来做设计，然后后端程序员从JS到CSS到Java全写，完全就是一个通道，所有的复杂逻辑全部交给DB来做，这也是几年前DBA很受重视的原因。

所以你能看到成千上万行的存储过程（存储过程，视图，事务，外键这些东西我真心希望永远不要在Mysql里出现），这就是外包公司中最常见的架构体系。来个SSH，Over。

好一点的会个WebService，用过ActiveMQ，也用过Redis，甚至还会用过Dubbo。然而大多数情况也根本不了解为什么这么用。

很多人写了两年或者三年代码都没做过独立的DB设计，不知道什么是REST，不懂怎么做接口设计，也不知道怎么去定位问题。

所以对于他们来说，拿到产品经理的需要，会有一个项目经理或者是Leader分配任务，跟着按步就班的把代码写完，跟前端调试完，QA测试不通过，加班改回来重新改，改完QA又没通过，再加班再改，QA终于通过了然后上线了突然发现另一个好的功能不能用了，跟着再接着改，在线上发布一次又一次。。眼睛熬的通红最终真的受不了了，休息几天换另一家公司涨个40%左右的薪水继续这样的日子。

不不不。我带出来的后端程序员并不要这么做。所以，我带的后端程序员的工作方式是这样的。

拿到产品需求-》后端程序员做接口设计，架构设计，DB设计-》拿出方案来做技术方案评审-》评审通过，开始预估时间-》每日更新自己的Task-》接口完成自测一百遍，每日部署到开发环境，随时集成-》CodeReview-》重构代码-》性能测试-》Demo通过-》发布到测试环境-》修正Bug-》重新发布-》发布到线上环境。

这中间需要理解需求，需要拿出多个方案，需要跟前端配合，需要跟QA配合，需要跟运维配合。需要跟产品沟通，有时候还需要找UI。后端几乎是一个核心节点，而这个核心节点接起来了所有的人。

我不知道我讲清楚没，很多时候我都发现我可能太久没做一个IT新人了，都忘记了新人们关心的问题或者是困惑是什么。这就是我知道的，两种后端程序员的工作内容。你选哪一种？

2需要技能：

环境【IDE（Idea/Eclipse，Maven，jenkins,Nexus,Jetty,Shell，Host），源码管理（SVN/Git），WEB服务器(nginx,tomcat,Resin)】基础【Http，REST，跨域，语法，Websocket,数据库，计算机网络，操作系统，算法，数据结构】框架【Spring，AOP，Quartz，JsonTagLib,tiles,activeMQ,memcache,redis,mybatis,log4j,junit等等等等等】业务【金融，教育，医疗，汽车，房产等等等等各种行业】第三方【微信，QQ等各种第三方登录，支付，IM，地图，语音，视频，图片】

环境不说了，搭环境永远是后端人员比较头疼的事儿，所以才会有很多人想用简单方便的的语言来解决这些问题，比如说Python之类的。我还是喜欢Java，大概很多人觉得重，然而我喜欢，我觉得不是“重”，而是“正”。好像剑一样，王者之剑，路子很正。

基础知识太多了，正是我一直强调的，做后端，这些基础知识了解多少，其实就是决定了你以后能走多远。这些科班出身的计算机ER，会了这些，才有了一个平台，才可以站在这个平台之上去搭建更高层的建筑，如果根基不稳，你觉得你会对上层的知识理解透彻么？

框架是Java最有资格说自己是架构师的原因。无数的开源框架，选型，筛选，对比，填坑，优化，维护，寻找最适合的业务场景，很多时候很多公司的架构简直了（我不吐了，很多技术都在用然而每一种用法几乎都是错误用例的典范）。所以你想想，你大概要有多少框架要学要用？很多时候，你必须要想清楚，哪些是需要认真了解的，哪些是需要一笔带过的。

业务对于后端人员来讲无比重要，不懂业务，就没有架构。这是我经常说的一句话，这个世界上不存在不懂业务的架构师（我不怕被打脸），一个架构师必须要深入了解业务体系，知道哪些是会变的，哪些是不会变的，哪些是重要的，哪些是不重要的，然后才能做出来适合某个应用场景的架构来。比如说，同样的表，几千万的量和几亿的量差别非常大，频繁读和频繁写的设计也完全不同。会有一些通用的架构思想和理念在里面，但是都是需要跟业务结合落地的。

PS：很多金融证券行业的程序员，就是靠业务知识混饭吃的。对他们来说，对业务体系的了解要比在技术上的追求重要的多。

第三方的东西和JS的内容相似，我不想多说了，而且JAVA的第三方的东西更是多的离谱，坦白的说Drools这东西我就没彻底研究明白，虽然很喜欢。而像这种类似的东西，太多太多了。

3发展前景

对于后端人员的发展前景，我有两点想说的。

A.无论是B/S还是C/S,无论是WEB还是原生，或者是智能硬件，后端都会屹立不倒。B.随着后端架构体系的稳定和成熟，后端人员在性能上需要担心的问题不多（再加上大部分应用场景其实并不需要那么多的性能），所以更多的应该会关注于一个稳定的扩展性好的架构，以及快速实现能够复用的业务逻辑模块实现上。

最近后端人员在价格上，其实有点偏低于前端人员的，就向我之前所说。两年的JS可能拿到20K。两年的Java想拿到这个，非常难。然而，五年的Java或者是七年的Java，拿到30~40K，不难。

更高的，也不稀奇。

0~12个月：4K~10K一年~三年：8K~20K三年~五年：18K~30K五年以上：30K~

成长路径：

Java初级工程师-Java中级工程师-架构师-技术经理-技术总监-CTO-CEO

后端的爆发力并不差，只要你给他时间，只要你愿意前进，后端的路线很深，深到你有时候会觉得自己还没来得及全部了解，就已经有无数的新人涌进来要替换你的位置了。

4.入门门槛

计算机网络，数据结构，数据库，操作系统，Java基础语法。Java是入门门槛最高的一个，没有之一。（好吧，我虽然说的是后端，然而一直把Java等同于后端）

当然，如果你的志向并不是一个架构师，只是像NodeJS和Python或者是PHP一样随便做点小项目，那么也可以说的得上是没有门槛，但是我说过我有偏见，所以可以直接把我无视掉。如果你觉得我说的哪点不对，你过来揍我啊。

要跟我学Java，就必须把这些基础知识学好，我只带想成为架构师的人。

5.哪些行业适合做后端工程师

IT界：无其他界：无科班生：计算机专业的中等水平能力以上

所以如果有各种培训学校告诉你零基础4个月20000块钱把你教出来做Java后端，然后你月薪上万，你就直接一锅盖盖他脸上吧。

那么零基础的人想做后端，怎么样才能入门呢，我比较推荐的是先做前端，然后把自己的基础知识各种补，补回来，再去转后端--但是实际上，你做了前端，再想转后端，就太难了，因为你到时候就不太想放弃自己一年或两年就可以轻松拿到的高薪，去做一个苦逼的后端了。

6.职业限制

后端的职业限制有很多，第一个职业限制就是不去做独立的项目，不做DB设计，不做接口设计。第二个职业限制就是视野不开阔，不知道有什么样的开源软件可以用。第三个职业限制就是不重视线上环境，不知道如何写日报，也不知道如何快速定位。我不得不说我带过的兄弟，有一次解决线上问题的时候快把我气疯了，他们在那里猜测问题出现的原因，跟玩福尔摩斯一样，不打日志不看日志，根据现象倒推结果，直接盲改代码再扔到线上看看有没有解决问题---那是最后逼不得已的办法好么，在此之前能不能安静的把日志打出来，确认一下到底是哪里出错了？第四个职业限制就是不懂版本管理，不懂Bug修复流程，不懂开发流程。这些其实都是一整套的流程体系（等我心情好了，有人把Java贴吧吧主骂的狗血喷头了，我大概也会写出来）

大部分后端的人员都会抱怨自己不会写前端代码，不会写Android或者是IOS，不能自己独立完成项目，所以他们才倾向于自己做一个全栈工程师，做一个自己喜欢做的东西。这也是后端人员会经常觉得不爽的地方，自己写的东西完全感受不到，而且一旦出问题很多时候都是大问题，解决起来很麻烦，经常不敢改代码，因为看不懂前人的东西。

有时候后端人员会比较木，虽然很各种职业都交流，但是多数都会觉得自己很NB其他人都很SB。这也是后端人员比较大的问题，往上走的话也容易遇到各种瓶颈，做技术的，做到CTO，再去做CEO，其实很难的。

而且，等你走到足够高的高度，你会发现，一个七年工作经验的正常发展的后端工程师，一定会有一个七年工作经验的产品或者是运营，在薪水和职业上秒杀他。这也是做技术的最大的悲剧。

不过大部分的后端工程师都比大部分的产品和运营人员薪水高，这也是这个行业的特征之一，所谓高不成低不就，小富即安，就是这样的。

如果你是一个有理想的后端工程师，我建议你多关注一些敏捷开发，多关注一些项目管理，学会带着自己的兄弟们一起做事儿。再不然，就是在技术这条路上一直走到黑。

7.去哪里学，怎么成长

基础知识不要来找我，我不教。随便去其他的学校补基础也可以。想要提高或者是想要成为架构师，就来找我。

我在说培宇的时候其实已经讲清楚了我是怎么带人的，但是在线上做这些，确实有难度，而我又不想教一些虚头巴脑的东西害人害已，所以我还在想办法解决怎么样在线上把后端工程师带好的问题。

如果看到这里觉得我说的有道理，一定要称赞我，催促我，还有帮我宣扬大修真院的思想和做事风格，让我觉得这个事情有意义，值得做。

想做好，真的挺不容易的。所以，能转就转，能收藏就收藏，能推荐就推荐。我想我应该能想到一个好的办法，来帮助后端工程师，把成长的时间从五年到七年，缩短到一年到三年，至少在线下，我是绝对有信心的，这种信心来自于我之前这么做过。以及我正在做。

你们可以随时找培宇聊聊，看看他这些日子倒底是怎么被我骂的狗血喷头然后开心快乐的活着的。

## DBA

首先说，我对DBA的了解并不专业，也不够多，而且对这个职业也有偏见。所以，我只能把我感受到的，我会的讲出来，然后如果说你们觉得我说的不对，要么自己开贴回答来打我的脸，我虚心学习，要么就直接笑笑走开，表在评论里说三道四，最烦这个。

七年或者八年或者很早之前，DBA是非常吃香的职业。讲这个，大概要从系统的性能瓶颈说起。很早之前，互联网刚开始的时候，算是蛮荒时代。那时候大家写代码还没有规范，能把功能做出来就不错了，大家拼的是什么呢，Sql的性能。基本上就是没有中间层，也不会分什么服务层和Web层，很多时候SQL都写到页面上。

然后Sql呢，又属于那种外键，视图，存储过程的天下。这就导致了出现一个问题。大部分的功能都是通过DB来实现的，也就是说，什么计算啊，分组啊，排序啊，筛选啊，全是靠DB来做。小功能还没问题，功能一多，问题就出来了，一个Sql语句执行了半个小时没做完，然后整个系统崩溃掉了。

那么，怎么解决呢，解决的方案就是。。。。我其实很难理解这种思考方式。。。。就是找一些人，对DB特别熟悉，他的职责就是审核所有程序员的Sql语句，去找出来这些Sql哪些用到索引了，哪些没用，能不能执行，怎么优化，以及监控线上的慢Sql。一个公司能养得起DBA的，很NB了。很贵的！

所以这是那个时候的DBA，但是，很快大家发现有不同的方式了，这种方式就是，我靠，原来我可以用分库分表，我可以做读写分离，我能做主从。于是对于DBA的依赖又重了一些，再加上数据的安全和备份，所以DBA的作用已经有点偏移，然而最关键的还是系统架构的发展变化了。分布式的概念慢慢的起来了，大家明白了一件事儿：机器不够，并不是说把服务器升级成小型机就能搞定了，而是应该用更多的机器来做，因为便宜，而且更简单。所以后台的系统架构慢慢的演化出来很多不同的层。WEB层，服务层，缓存层，DB层。对于缓存的使用越来越重要，由此而变化的观点就是数据分成了缓存和持久两种结果，DB慢慢的变成了持久层-也就是说，只是要把数据持久化，并不希望它去承载用户的压力，缓存主要用来扛并发，不需要做持久。这是一个很关键的点，也是决定DBA命运的转折点。

当然现在还看不出来（像MongoDB，Cassandra，这些，又是另外一种不同的技术走向，包括Mysql也在不断的想要提升自己的性能），所以这些东西我们先抛到一边不谈。只说这中间发生了一个变化，对于后端人员来说，对数据库访问的变的严格起来了。尽量单表操作，不允许复杂查询，设计架构的时候必须考虑缓存，甚至我们在白社会的时候还设计了一套通用的DB访问机制--虽然是七年前的设计然而现在一直都觉得很赞，只是再也没有如我在搜狐的时候那群人做这些事了-反正我见识少，一直在小公司混，也不怕你们嘲笑我见识少。

这样就导致DBA的一个很重要的工作职责，失去意义：就是查找慢Sql，因为我们在系统架构层已经决定了不再这么使用DB。这样使得Oracle什么的也慢慢的失去了价值-我知道我说的每一句话都有可能会引起争论，所以我不得不再次强调一次，纯属个人的脑残关点，不喜欢的话，要么认真的回复来打我的脸教我做人我认真学习，要么就是滚远点表理我。包括建表，去除外键，去除事务，去掉视图等等等，一瞬间，DB的使用简单多了。

那么，DBA还能做什么呢？对于我现在的理解来说，DBA的职责慢慢变成了数据备份和安全策略--然而这部分又跟运维的工作有了冲突，所以在某种程度上来讲，我都会在五十人左右的公司把DBA安排到运维部分，跟运维的兄弟们做基友。可是现在云服务器也变的越来越好用了，这里也推荐一下好友的金山云和Ucloud。阿里云跟我并没有神马认识的人，所以不推。

DBA除了之前提到的主从，读写，数据备份，权限控制，分库等等，还应该再扩展视野，把MongoDB，Redis，memcache，elasitcSearch，hadoop等等这些数据全部管起来。我觉得，更像是一个运维的分支了。

这就是我目前认可的DBA的价值和意义，已经从之前的性能优化部分转移到了数据备份和安全。毕竟，性能，架构，和优化这些东西，是离不开业务系统的。

那么，接下来，和之前一样，继续介绍一下DBA的工作内容。

1工作内容：

如果你做了一个DBA，基本上会遇到两种情况。一种是你的后端工程师懂架构，知道怎么合便使用DB，知道如何防止穿透DB，那么恭喜你，你只是需要当一个DB技术兜底的顾问就好，基本上没什么活可以做，做个监控，写个统计就好了。你可以花时间在MongoDB了，Hadoop了这些，随便玩玩儿。再按照我之前说的，做好数据备份。如果需求变动比较大，往往会牵涉到一些线上数据的更改，那么就在发布的时候安静的等着，等着他们出问题。。。。如果不出问题就可以回家睡觉了。

另一种情况就是我刚刚提到的，大部分程序还是靠SQl，然后有时候DBA还需要写几万行的存储过程，那么你的主要职责还是优化Sql，优化Sql，永远不停的优化SQL。

嗯。就这样。

还有就是多花点时间把MongoDB和hadoop这些都维护起来，或者简单说，只要跟数据安全，备份相关的东西，都维护起来。

2需要技能：

环境【Linux，Mysql，Oracle，MongoDB，Hadoop】工具【各种DB的版本，工具，备份，日志等】

这个说是环境已经有点勉强了，毕竟是一些吃饭的家伙。就是各种DB，各种维护什么的。工具也是相关的内容，再强调一下对版本的熟悉程度。

马丹我感觉我没什么可说的了。因为刚刚又被我就静静的发图～\_java吧这个贴子恶心到了。我被Java吧封了，然后他们还@我。我要赶紧写完然后想办法去跟他们对骂去。

3发展前景

DBA的发展前景我说不好。一些简单的工作。运维也是慢慢学会了。包括薪水，这个是我比较没把握的。之前的薪水都是有迹可寻的，DBA的薪水我接触的比较少，实在是没什么底气。

1年~5年：8K~25K5年以上：20K~40K（我好心虚。。我只给一个DBA开过工资）

成长路径：

也不知道有啥成长路径，感觉这个职业的物种越来越稀少了。

4.入门门槛

DBA的入门门槛也是比较高的，而且，很少于有刚工始就是做DBA的，大部分都是工程师转的，所以呢，至少要两到三年左右的时间才有可能做DBA，才能负责一些相对负责DB的事情。

5.哪些行业适合做DBA

IT界：后端工程师，运维工程师其他界：无

其他行业的想转DBA，刚刚也说过了，不合适，只能先写代码，再慢慢的转。

6.职业限制

这个职业最大的限制大概就是。。很容易无事可做，前面有后台架构师蚕食，后面有运维工程师侵入，中小公司都不太会设置这个岗位，所以有的时候会比较尴尬，大概还有一些外包公司，或者是传统的IT企业，会是由DBA去设计表，去理清业务还有一些岗位，其他的都不太好。

所以对于其他的各种持久化数据的备份和优化，特别是对一些正在使用的框架，又不够成熟的东西，更容易找到自己的位置。如果你要做DBA的话，就记着，跟持久层相关的优化，数据安全，备份都要去了解--顺便再学点运维的东西

7.去哪里学，怎么成长

基本上，DBA可以由运维工程师和后台架构师去转，也会有一些数据库方面的认证之类的的过程。这个岗位啊，跟其他的岗位真心有点不一样，很少有刚毕业或者刚入行的人就要去做DBA的--你也做不了亲，跟数据相关的东西，没个三年五年的经验，真的很难做，而且有些技术细节，是必须要了解的比较多的。

PS：刚刚想到了，云服务器里是DBA的一个好去处，其实这里跟大数据也有一些相关的。这么说的话，DBA的职业需求度跟云服务器公司的发展是有比较密切的关系。但是总之，不太可能变成一个标配了。

## 运维

不然还真的是不想再写下去了。以及，其他的答案也让我很无语，特别是居然并不知道CSS和JS可以是两个职位，实在忍不下去了，我会在后续讲敏捷开发的时候再重新提一下软件开发的全过程，来告诉你们为什么CSS和JS要分开，那个才是我执行敏捷开发落地七年的干货，当然，时间不定，只是赞越多，分享越多，我觉得自己的答案越有价值，心情就会好，然后知道修真院的人也会越多。

我可能会出一个系列，这个只是第一个，来讲一下互联网的各个职业发展，第二个，讲敏捷开发，第三个，讲应该怎么样对IT技术人员的团队管理和绩效考核，第四个，讲如何做自己的职业规划以及如何找到一份更好的工作或者是怎么判断一个公司靠不靠谱，第五个，大概会聊一下创业公司怎么做，以及如何用MVP的方式去创业。

这些如果你们都看到了，我讲出来了，也差不多是我这八年中经历和感受到的所有了，至少会对互联网的各个方面都有了一个全面的认识。

这里面不可避免的会有对修真院的介绍，但是我能确保的是，每一篇都是干货，只不过不保证客观，也不保证有用。所以，让赞和分享更多一些好么。

言归正传，讲一下运维，这是一个很神奇的行业，一个在渐渐蚕食其他各种职业的职业，一个本来不是编程但是现在却编程越来越多的职业，你要问为什么?

一个字：云

说到运维，就不得不提起来服务器，以前呢，服务器很贵，带宽也很贵，机房也贵，电源也贵，反正啥都贵，而且还不稳定，那个时候运维工程师的大部分技能还是在硬件上，或者是说硬件是必不可缺少的技能，所以守着机房也是运维工程师比较常见的事儿。

而且有的时候还会兼着网管，因此，大部分的运维工作就是装个系统，搭些内部服务软件，如邮件，Wiki，Jenkins，VPN等等等。很多的时候还要会布线，会路由器，会交换机，会做资产盘查，把网管的工作也做了。

当然线上的正常发布也少不了，规范的点的公司，会分成三套或者是四套环境（我是不太能接受四套环境），就是Dev，Test，Online（三个环境的叫法会有不同，我现在越来越喜欢这种叫法），开发环境就是给开发人员自己瞎折腾用，运维只需要把机器安装好，IP分配好就行了，开发过程中你们自己瞎折腾，每天部署多少次都可以--细节请等我讲敏捷开发的时候再说。测试环境就会严谨一些，正常情况下每天只发布一次，太频繁QA会杀人的。线上环境更严格，一般来说不允许频繁上线，所以只能在周二或者是周四发布，一周两次，对于WEB来说，其实是足够的（但是很搞乱的是，很多公司是周一定需求，周二开发，周三测试，周四上线，周五改Bug，这些都会在讲敏捷开发的时候提到）。

所以测试环境和线上的发布都是运维来做，因为有一个最简单的真理：只要你把权限扔给工程师了，他就会偷偷重启服务以及偷偷改数据以及一不小心删掉某个文件，或者是清空所有数据。

而且一定会被你抓到，不要问我为什么，我要是有线上服务器的器权限，我也经常干这种事儿。。。。这比偷情的感觉要好多了，一个人偷偷摸的瞒着所有人把Bug改掉。。

这个时候运维还是运维工程师，直到虚拟化技术疯起来，直接生成了阿里云，金山云（修真院的重要合作伙伴），Ucloud等这些改变了运维工程师命运走向的公司。

云时代来临了。。。视频云，IM云，图片云，APP云等等各种跟业务相关的云服务也做起来了，确切的说短信，邮件也是各种云服务。那么运维工程师的变成两极分化。一个是，云服务商的运维工程师越来越重要了，一些大的公司也在搭建很NB的运维自动化系统。另一个是，非云服务商或者是非大公司的运维工作变简单了，再也不用去机房换硬盘了。。反而是要熟悉和了解各种云服务器的特性。

所以，当我们现在说到运维工程师的时候，其实是分成两个阶段的，一个是传统的运维工程师做的东西，一个是现在云时代运维工程师做的内容，如果你现在想做运维，你恰恰是处在这个变革中间，我说不好未来的走向，但是这是一脉相传的，所以并不用担心运维中技术体系的中断。

这就是对运维的一个简单的介绍，哦对了。运维是一个公司里非常重要的岗位，刚刚还忘了说到了数据备份和灾备和监控。如果你们公司有值班手机，一定是运维同学手里拿着。24小时不准关机。

无论你洗澡，上厕所，还是吃饭，或者是逛街，还是跟妹子恩爱（天地良心，我没见过一个妹子做运维，听都没听过），只要你收到报警，你就必须第一时间远程连接到服务器，然后通知相应的工程师，立刻解决问题，或者是直接滚蛋。

通宵熬夜发版本更是正常的事儿。你要问为什么会通宵，呵呵。发了一次版本，上线，发现不正常，于是回滚，或者是直接在线上Bug。于是运维同学就等着，老婆催也没用。然后过了两个小时，大后期后端同学说好了，可以发布了，于是执行脚本，然后又发现出了问题，接着回滚或者是在线上改。。。

你要问不是有测试环境么，不是有QA么，我就呵呵一笑。我不知道我说的关于运维的介绍是多了还是少了。反正，这个答案只是一个初稿，如果点赞过5000（感觉是不太现实的），我就再整理一版。

1工作内容：

运维的工作大概分成几个部分，我对于修真院学习运维的少年们都这么说，大概是：

A。基础环境的搭建和常用软件的安装和配置（兼网管的还有各种程控机）,常用软件指的是SVN，Git，邮箱这种，更细节的内容请参考修真院对于运维职业的介绍。

B。日常的发布和维护，如刚刚讲到的一样，测试环境和线上环境的发布和记录，原则上，对线上所有的变更都应该有记录。

C。数据的备份和服务的监控&安全配置。各种数据，都要做好备份和回滚的手段，提前准备好各种紧急预案，服务的监制要做好。安全始终都是不怎么被重点考虑的问题，因为这个东西无底洞，你永远不知道做到什么程度算是比较安全了，所以大多数都是看着情况来。

D。运维工具的编写。这一点在大的云服务器商里格外常见，大公司也是一样的。

E。Hadoop相关的大数据体系架构的运维，确实有公司在用几百台机器做Hadoop，所以虽然不常见，我还是列出来吧。

这些就是运维的基本工作内容了，然后如我之前所说，运维一般来说，大点的有两个，小点的就一个，再小的就是工程师兼着。而且大部分都在使用云服务器了。所以普通的运维工程师的工作都可以标准化和简单了。这里并没有提到性能优化相关的部分，实际上这也是运维工作中比较常见的内容。特别是在出现问题的时候，运维永远是一个在不出问题可能就被忽视，一旦出问题就会被痛骂的。。。职业。

之前有句话说的好，线上不出故障，是不现实的。现实的做法是，出了故障之后，多久能快速恢复。我对这个很认可，原话应该是出自一家云服务器公司还是。。我不记得了。

但是这里不得不推荐一下【猎聘网的技术总监刘中兵】，我的老同事和好基友，前段时间刚刚在一个神马神马会（我靠，我只知道是一个很厉害的会）上分享了猎聘在自动化运维上技术积累，如果PPT能够公开的话，回头我会把它也放到运维的体系里。

当然最好的结识这些技术大牛的方式就是加入修真院，显示自己的实力之后，让修真院来帮你牵针引线。

2需要技能：

环境【Linux，Mysql，Oracle，MongoDB，Hadoop，nginx,apache,F5,lvs,vpn,iptable，svn,git,memcache,redis】工具【linux常用工具，Mysql常用工具，Jenkins，zabbix,nagios】自动化运维【openstack,docker,ansible】语言【shell,python】

运维的技术体系也很多，基本上是所有的工具都想要你会用，我会慢慢的整理出来一套运维体系的技术架构，当然不只包括运维，包括我说的这些所有偏技术的职业。所以，如果觉得我写的还不错，还算靠谱，就关注我，关注群，关注微信公众号。如果觉得我是在扯淡，直接评论里骂我就好了。

这里说到的其实并不算全，确切的说，做为一个运维工程师，严谨的性格是绝对少不了的，因为线上的操作全部由你来执行，你必须要心细，要严谨，如果做不到，就别考虑了。真出了事，压力又把你压死。想想最近几年出现的运维事故吧。。。

还有就是必须对Linux系统的底层了解的深入一些。如果你想做一些高级运维的话。Python和Shell脚本都是必不可少的，最常见的运维必备语言。

3发展前景

写的有点不耐烦了，马丹如果不是为了宣传大修真院，我才不要做这种丹藤的事儿。我去勾搭小姑娘多好。运维和前端其实有点想像，都是属于柳暗花明又一村的。

本来运维的工作要被后端工程师蚕食了（后端工程师真是一个可怕的职业，可以这么说，没有后端工程师做不了的。。。包括产品，市场和运营）

然而随着云的发展，运维工程师简直是有了新生。

1年~3年：6K~20K3年以上：28K~40K

成长路径：

网管-初级运维工程师-中级运维工程师-高级运维工程师

运维做到这个级别，已经到头了。很少有运维的同事能做到技术总监的级别的。所以，对于资质一般的你来说，选择了做运维，就代表着天花板很近，但是对于大多数人来说，已经可以了。

4.入门门槛

OP的门槛其实并没有太高，经常玩玩Linux会好一些，但是要学的东西，确实挺多的。处理线上故障，也是一个经验积累的过程，所以说呢，基本上算是0门槛，谁都可以考虑做，只是要看你喜欢，或者是不喜欢，进度快和慢的问题而已。

5.哪些行业适合做运维

IT界：后端工程师，网管其他界：All

刚刚介绍过了，门槛不太高，所以比较容易一些，而且，可以考虑做一段时间网管。半年或一年之后转运维，网管的薪水比较低。3000或者是5000已经到顶了。

但是转到运维，就不一样了，简直是新生。

6.职业限制

运维的大部分事情其实都是跟业务无关的。这在之前介绍CSS的时候说到的，两个不需要懂业务逻辑的职业。曾经有过运维工程师觉得工程师写的Sql不对帮他改了。。。然后被我狠狠的骂了一顿。这表示着：运维工程师必须按工程师所写的步骤去执行，不允许有任何的差错，如果出错了，如果没在测试环境跑通，或者发现有问题可以跟后端工程师沟通，但是，不可以改他们的东西。

这样责任和权利分开。所以运维工程师能够相对有更多的时间去研究技术，然而坏处就是对一个行业了解的会比较少。

所以这个职业比较大的问题就是，想要再转行成其他的职业，已经很难了，只能一条路走到黑。嗯。比较难。

天花板和CSS一样，能看得到，但是！已经很适合了，薪水毕竟不算低呢。

7.去哪里学，怎么成长

当然是修真院了，第一个。然后还有一些其他的运维培训，我不太了解，运维这个其实是几种职业培训里相对来说比较简单的。因为大部分都是跟业务关系不大，是可以培养出来的。

自己找资料的话，只能根据技术的关键字去找了，还好大部分都不算特别难，等遇到难题的时候，你也一定是工作了不少，然后会遇到一些技术上大牛的运维。

## QA

“QA最好的出路就是产品经理”。这是我对QA最好的认知。我知道这里有无数反对的声音，我说的每一句话大概都会有认同和不认同的人，所以如果看到这句话不喜欢，就表再继续看了。

QA的这个职位，大致分成两个流派，一个是功能测试，一个是性能测试。功能测试就是指的是各种点点点点。然后看看功能和数据有没有问题。什么浏览器，什么版本，什么情况，能否复现。

性能测试就是指：用Jemter，LoadRunner等做压测，测跨后端人员的信心。

基本上就是这两种，很少有专门针对安全的层次去做测试的。那么问题就来了，工程师和QA之间，倒底谁该做功能测试，谁该做性能测试？

我姐告诉我说：工程师才应该做性能测试，因为只有他们才最应该知道自己开发的系统性能瓶颈在什么地方。然后也是因为这句话，我举一反三的瞬间懂了什么才是一个工程师，并且在不断的践行者这个理念：如果你是一个后端工程师，你必须对服务器的线上数据了如指掌，有多少张表，在哪台机器上，分了几个库，一个表里有多少条数据，数据的分布是什么样的，起了多少台Service，占用多大的内存，高峰期的TPS有多少，CPU的负载怎么样，页面总响应速度有多少，一个请求调用了几个方法，每个方法执行时间是多少，是否加载了缓存，从缓存里加载的数据是多少个，大概多少MS，访问一个数据库需要多久。

你做不到这一点，你就没办法做好性能优化。所以，QA很难做到这些，也完全没有必要做到这些-实际上我们的团队是没有QA的。工程师负责所有的问题。

前端也是一样的，做了一个App，耗电量，带宽，内存，兼容性，都是前端应该关心的问题。

所以我在这里给出的一个明确答案就是：性能测试，开发人员做，功能测试，QA做。

但是这里的功能测试，并不是指的是普通的功能测试。实际上，开发人员应该自测一百遍再提交到测试环境（我在讲到敏捷开发的话，会再次提到整个开发流程是怎么样的，以我在几家公司实施的经验来看，这套敏捷开发流程，对于很多公司梳理内部结构都会有帮助。）

QA做的应该是自动化的回归测试，边界条件测试，极限条件测试等种种很难测试得到的问题。对了，再解释一下为什么要有QA，以及为什么Bug总是测不完。

因为在我根本记不清的一篇文章里说到，哪怕是最简单的几个功能组合，想无穷穷举测完都很困难--马丹，原话我记不住了。

所以这就是给无数擅长性能测试的QA判了死刑，你之所以能做性能测试，是因为你们的开发模式和流程不对--别看我，我就是喜欢这么说，一点都不委婉。爱听听，不听拉倒。

但是QA真的不是一个很简单的活儿，可以这么说吧。QA几乎是所有职业中，最熟悉系统的人-甚至包括设计它出来，开发它出来的产品经理和各种工程师！

很多时候QA做的事情都是非常单调的，但是又必须有责任心的，拿我之前的话说，QA就是最后一道关口。

所以，如果你想成为一个PM，你可以考虑先做QA，当然，前提是你必须遇到一个愿意给你机会做产品的好老大-比如说我。

So，接下来就开始来闲扯一下QA了。

1工作内容：

QA需要了解需求，很多公司会要求QA写测试用例，我觉得是扯淡。完全是在浪费时间。通常开发三周，QA测试的时间只有一周到一周半。还有关于提前写测试用例的，都不靠谱。

但是总之，了解需求，就算是不写也要自己知道自己测，这是QA的必备职责。

跟着就是等开发人员开发，这个时候也会有一些奇葩公司，要求QA提前界入的，理由是加快上线周期。之前还有说过完成一个Story就要测试完整的，我也不想吐槽了。总之，如果你真的遇到了这样的公司，你的工作就是测试，然后跟开发人员说不对。然后开发人员说改好了，然后再测。然后你大喊一声，好毛线啊。然后开发人员说，稍等，我先梳个头。然后再告诉你好了。然后你再说，好毛线啊。然后开发人说。。。这次真的好了，于是你测了一下，真的好了，开发人员很Nice的说：完美！你也很开心，但是没过几秒钟，你就会喊：什么鬼，之前的功能是好的，为什么又改回去了？

如果你呢。遇到好点的流程。会在上QA之前，打版本（我始终无法理解不打版本是什么流程），会Demo，那么你可以有一个安心的测试环境了。

这个时候稍微正常点的工作，就是测试，然后把Bug录入到Jira，禅道，或者Bugzilla等各种专用的Bug管理工具。

跟着就是追踪bug(如果你们公司有Bug处理流程的话)，如果有晨会就在晨会上说明，如果有周会，就在周会上统计。大部分的QA都比较羞涩，并不太敢提Bug，并不太敢说哪个程序员的代码写的有问题。毕竟，这是唯一一个必须要当面揭短的职业，哈哈哈哈。

基本上到这里QA的任务就没了。等着发布上线就好了。实际情况上发布上线的时候QA要等着程序员改代码，验证。。。

2需要技能：

流程【Bug修复流程，版本发布流程】工具【禅道，BugZilla，Jira,Excel表格来统计Bug数，自动化测试】性格【严谨，耐心】

QA里经常会嘲笑自己是技术Team里最没技术的一个。如果说你是一个标准的QA的话，真的不用去特地研究一些压测工具的。

不过也确实可以了解一下，Jmeter怎么用，TPS是什么概念，90%线是什么意思，PostGet什么的。

自动化测试工具是我一直都强烈推荐的，无论是神马办法，只要你能做到哪怕只有一部分自动化的测试，你做回归测试就很容易了。相信我，回归测试在每一个版本发布都是需要的。只不过有的时候，只是需要跑一下脚本就好了。有问题再详细测一下。

严谨和耐性是非常非常难得的，也是QA最重要的能力。当然也包括需求的理解能力啦。

3发展前景

QA的发展前景。。真的不算好，跟网管一样，如果说并不能转产品的话，20K基本上就封顶了，就算是有公司，愿意出30K让你们做性能测试，也是到天花板了。

1年~2年：4K~15K2年以上：12K~20K

成长路径：

测试工程师-测试组Leader-PM

如果能转到PM，真的就很赞了，我推荐的时间应该是在半年到一年左右就开始转。

4.入门门槛

QA并没有什么门槛，一般来说，是妹子比较多。汉子比较少，是汉子的，也多数是做点性能测试相关的。但是请相信我，这绝对不是一个好的天赋加点方式，就算是性能测试做的再6又怎么样？

所以稍微懂一点Bug修复流程就好啦。勇敢的去做QA吧。。。

5.哪些行业适合做运维

IT界：all其他界：all

虽然是IT界的All都可以转QA，但是说实话，我没有看到过一个从其他职业转到QA的。。也许除了运维。。

6.职业限制

职业限制也说过啦。总有重复的使用技能的厌倦感，毕竟这些技术水平都不需要太多。理解能力好一些，严谨一些就能做。

说的直接点，从事五年的QA和从事一年的QA，本质上并没有什么差别啊。但是好处就是，如果你有心，你会对系统特别特别的了解，这对于你转行做PM，是一个非常非常大的优势。再强调一遍，并不推荐QA转走技术路线。

7.去哪里学，怎么成长

修真院并不提供QA的培训，暂时并没有，以后大概会讲一下Bug处理流程和敏捷开发流程。所以，我也不知道哪里有QA的成长机会，一般都是在公司。

## 算法

“真正NB的程序员都是写算法的”-----我自己瞎编的。算法工程师其实是一个很虚的概念，但是在大数据忽悠了人这么多年以后，慢慢的也开始吃香起来了。很多经典的算法都是十年前，二十年前的，在算法本质上，诸多数学家或者是IT学者没有做过太多的改进，大家的统一的结论就是：先拿到最应用，然后改进一点是一点。

可是即便是算法很多时候不完善，也已经暴发出了强大的战斗力。

那么什么是算法呢，什么是算法工程师呢？简单来说，算法就是瞎扯淡想的，从各种自然现象映射到计算机上去的。什么爬山算法，模拟退火。。种种种种，反正就是解决问题啦。算法工程师就是吃饱了没事干，调个参数，训练个模型玩玩的闲人。

一般来说。说到底，还是在机器学习和数据挖掘中最常见的几种东西，首先就是分类。分类算法是最常见的。分类一般会用来匹配，比如说，上周刚刚有朋友来问我怎么做穿衣搭配种的一套方案可以快速推给其他同类用户的设计。那么首先就要收集用户的信息，设定特征值，然后训练分类模型等等。最常见的就是新闻的分类，如我之前做过的财经新闻分类，把2000多家上市公司相关的新闻自动分到所属行业。

分类完之后就是聚类，聚类用的其实不多啦，算是用的比较少的。我了解到的使用聚类算法的并不太多。

实体识别是一个核心的技术。在新闻的领域用的是比较多的，确切的说，在文本领域用的都比较多。人名，地名，股票，时间等等都是实体识别的一部分。

关系抽取是一个相对来讲比较靠上的应用。事件识也是一样，我们曾经做过上市公司高管变动的东西。也看不出有太大的价值。

协同推荐是在三四年前，还是五六年前的时候比较靠谱的，那个时候个性化推荐刚出来，各种相关商品，书籍，还有个性化阅读，音乐等等等等。然而也并没有爆发出什么好的应用来。

豆瓣和亚马逊据说推荐的还不错，我没仔细对比过。

简来说，常用的算法，指的就是这些。而算法工程师呢？并不是去研究算法。真的。算法工程师只是拿别人的算法：有的是自己实现，有的是直接拿开源软件来用，跑语料，找特征，调参数，优化修改而已。

在我还年少无知的时候，对算法这个很东西痴迷，后来就渐渐明白了，算法是那些博士要做的。博士只负责来做算法。在公司呢。大部分都是要接近业务的，而且说实话，算法之间的差别，并不大。很多时候都是减轻人工负担，提供决策支持，更简单的更有效的可能就是关键词过滤。

嗯。还有很多算法工程师，做到最后，只能跑个Demo，连个像样的工程代码都写不出来。经常是一个月调出一个模型，然后再优化一两个月，然后再优化一两个月，准确率可能都提高不了一个点。

所以这其实是一个跟业务结合特别密切的岗位，很多时候的先难知识都比较有用。而且也是个很难见到效果的岗位，需要太多语料的配合了。

好的吧。大概就是这么多。

1工作内容：

算法工程师的工作内容，大部分时间都是在调优。就是调各种参数和语料，寻找特征，验证结果，排除噪音。也会和Hadoop神马的打一些交道，mahout神马的，我那个时候还在用JavaML。现在并不知道有没有什么更好用的工具了。

有的时候还要自己去标注语料---当然大部分人都不爱做这个事儿，会找漂亮的小编辑去做。

2需要技能：

基础【机器学习，数据挖掘】工具【Mahout，JavaML等其他的算法工具集】

这是一个技术含量很高很高的职业。。。。特别是对不懂这个行业的人来说。虽然有时候大部分做的东西也就那么回事儿，但是对于最基本的SVM算法，隐马模型，最大熵等还是有比较高的要求的。毕竟，懂还是不懂，差别挺大的。

3发展前景

我对算法工程师的薪资并不太了解，只是知道不会比较架构师的薪水低。这个职业的人数比较少，最少也是需要硕士以上（硕士需要学的东西和写的论文真不是盖的，并不排除有很多都是抄的），所以很难给出一个精准的数据。

1年~3年：10K~25K3年以上：20K~40K

4.入门门槛

嗯。门槛最高的一个。最低也是要硕士，基本上都是博士在做。大部分都是有一个博士震楼，然后一帮硕士帮着写具体的代码。

所以入门的门槛极高。

5.哪些行业适合做算法工程师

IT界：计算机算法相关的硕士或博士其他界：无

6.职业限制

怎么说呢。算法工程师很少会写工程代码，所以写代码的能力基本上等于零。这个是一个瓶颈。另外，很少有机会能让你研究算法，只能去不断的跟业务数据去死磕。

7.去哪里学，怎么成长

说不上来怎么成长，学这东西只能是跟着学校走。自学可能性太低了。成长的路径也就是去一些大数据的公司--放心，只要是大数据的公司，基本上都是会考虑算法的。

## 搜索

这儿说的搜索工程师是偏Java的。并不是百度，Google这种大型的搜索，而是指一般的网站常用的搜索工程师。

搜索工程师的主要工作流程就是抓取，解析，去重，处理，索引，查询。每个环节都有比较强的算法要求。我自己比较喜欢不区分搜索和应用组，一直觉得这两个组的人技能是相通的，能做架构的，算法也不问题。只能写模块代码的，也可以写搜索的项目。

但是实际上搜索还是值得拿出一个单独的领域的。每个领域都有比较深的知识点，因此单独拿出一个搜索工程师的概念也并不算错。特别是NoSQLDB兴起之后，我还是比较起Search+NoSqlDB的架构。

搜索工程师算是属于偏高级别的了，一般来说也是要求硕士以上。最早的时候能做搜索的都挺厉害的，后来有一段时间之后，开源框架越来越成熟，搜索工程师也变成了一个门槛低的行业，直到NoSqlDB的兴起，搜索并不是局限于是来做前端模糊搜索了，反而成了系统架构的一种方式，同样是可以在做分布式，做分层，分负载均衡，性能比单纯用DB好太多，也解析了Cache的问题。特别是Solr和ElasticSearch的出现。

或者这么说吧。以前的搜索有很大一部分都是为了前端的索引查询，只是满足于搜索一个功能的需要。后来慢慢演变出来了很多不同的需求，其中一个比较关键的词就叫做Facet。

某种程度上来讲,Facet才给搜索带来了新生，慢慢的才发现，原来并不是我在搜索框中才叫搜索。原来电商的分类搜索也是搜索，原来很多列表也可以用搜索来做。

这个又刚刚解决了Mysql的问题，各种NoSqlDB做这种索引的东西实在又有点有心无力。所以搜索才能慢慢演变成一种架构方式。

嗯。至少在我这里一直是要不断的去尝试一下。

1工作内容：

所以搜索现在其实分成两种。

一种是传统的搜索。包括：A。抓取B。解析C。去重D。处理E。索引F。查询

另一种是做为架构的搜索。并不包括之前的抓取解析去重，只有索引和查询。A。索引B。查询从传统的搜索来讲。先说A抓取。

抓取最需要解决的问题就是抓取的优先级，理论上来讲，对一个网站上的访问，不能过超过2秒一次的频率，这个并不会给别人网站带来压力，被称为是友好的访问--表问我怎么知道这些的，我也不记得了，反正并不严谨。

因此你需要有一个很好的机制来给Url打分，判断哪些抓过了。哪些没抓。这就是所谓的蜘蛛，也叫爬虫。但是实际上我被这个名字误导好久，真心是丹藤。只是发一个网络请求而已，根本没有到对方网站去，马丹刚开始想出这个名字的人抽象的能力简直弱爆了。我一开始就在幻像各种程序写成的小虫子会在对方网络上跑。打野的最后才明白原来这些虫子都不出门，只是在家吐口水而已。

B解析解析并不是一件容易的事儿。解析要做的就是把网页的文本内容和其他没用的信息去掉。知道什么是标题，什么是作者，什么是时间，或者是其他特定的需要。一般来说，都是有两种方式。一种是配模板，不同的网站我有不同的模板。对方网站改版我也改。这种方式最精准。但是很愁人。所以另一种方式，自动解析，用算法或者是规则等自动解析内容的东西层出不穷。我到现在为止都觉得一切都是扯淡。寻找最大正文本块再反向解析是最科学的。

之前还听到过说，模板的自动生成。抓下来一个网站的数据，然后从上千个网页中自动解析网页解析。这个我挺感兴趣的，但是一直没试过。

总之，如果只是抓一个网站上的数据（比如说抓取百度的搜索结果哈哈哈哈）。用Python的Scrapy这种神器，几分钟搞定还是没问题的。

C。去重。去重是一个很好玩的事儿。不去重不行。因为国内的风气就是这样。你抄我，我抄你。而且。为了怕被大家发现是抄袭，还会手动的增加一段，删除一段。做为一个智能的去重系统，你就必须把这些问题都解决掉。

而且，随着时间的积累，你甚至有可能去做全量的去重（虽然我一直理解不了，为什么要做全量）。六七年前我知道的最好的算法就是SpotSig。嗯。一个很机智的算法。

D。处理。这里的处理多了去了。所有的跟NLP相关的东西都可以放到这里。包括分词，词性识别，实体识别，关系抽取，事件识别，分类，聚类，热点等等等等等。

而且对于实时性的要求比较高。一篇新闻从别的网站发布，到我们的搜索系统，最快的可能就30秒或者是1两分钟--全部都是30秒不可能，因为抓取受限，只能说，最快的能看到这种新的数据。

E。索引。索引是比较简单的。主要牵涉到的就是怎么做实时索引。大概是在内存里做两个索引，然后文件有一个索引，跟着来回倒腾。。具体算法我不记得了，也是一个很机智的策略，在Bob中使用的。如果有Facet的话，也要考虑一下Facet的设置。

F。查询。查询里主要是分词，并不牵涉到句法分析。还有一个高亮算法，以及Facet的查询。相对简单一些。不过如果是分布式的搜索就会麻烦一些，又会出现如何分片的问题。是垂直分，还是按时间分？不同的业务有不同的选择。

2需要技能：

环境【Linux】框架【Luence,Slor,ElasticSearch,Cassandra,MongoDB】算法【倒排索引，权重计算公式，去重算法，Facet搜索的原理，高亮算法，实时索引】

嗯。算法偏多。跟NLP的算法又有很大的不一样。比那些分类聚类的难度要低一些，又比做项目和工程应用要再高一些。有点儿不上不下的感觉。

3发展前景

现在来看的话，搜索工程师的作用越来越大，某种程度上和后台工程师是重合了，之前说到过拿搜的Facet来做架构。

1年~3年：10K~20K3年以上：15K~35K

成长路径：

搜索工程师-搜索组leader

在我印象中，很少有做搜索的能够做CTO或者是架构师的，这跟搜索工程师以前的工作职责是有很大关系的，拿搜索工程师跟Application工程师做对比就能看得出来。

一个偏算法，偏调优，一个是偏项目，偏工程。偏工程的更需要视角全面，在某一个领域上没有搜索工程师研究的深入，但是整体的大局观会好一些。搜索工程师需要的则是深度。

4.入门门槛

有过两到三年Java工程师经验的人，或者是硕士毕业的，或者是数学比较好的，或者做过NLP的.这就是搜索工程师的门槛.如果你对数据结构,对算法不了解,做个搜索工程师挺没意思的.

5.哪些行业适合做搜索工程师

IT界：后端工程师中有两年经验,或者是硕士,或者是NLP其他界：无

6.职业限制

搜索的职业限制就是他很少去做工程化的代码,对于业务逻辑的理解跟工程化的工程师还是差一些.可以选用的框架也比较少.所以到了搜索组的leader之后,再往上走有点难.会遇到大数据的部门,跟他们比,搜索工程师又不太懂文本分类等算法,跟项目的工程师比,在代码重构,项目规范上又会差一些.

这个就是搜索工程师的职业限制.不过,这三者之间也在不断的融合.有些工程出身的,不太了解搜索算法,只是拿来做架构体系来用,也已经够了.

7.去哪里学，怎么成长

并不知道去哪里学,等修真院有了搜索工程师的职业以后,修真院一定是一个好的学习搜索的地方.

## 大数据

其实一直都不太喜欢大数据工程师这个说法，总觉得有种多此一举的感觉。不过实际上呢，确实也会有这么一种岗位，一般来说，公司里大概会有个6~7人的小团队，或者是10个，都无所谓了。

在五六年前的时候，Hadoop的家族还不太多，大数据工程师其实并没有太多事情可以干，基础环境搭起来，然后能够部署，能够跑起来就可以了。Hadoop是要在上百台PC上跑才有显示出来优势，然而很多公司只有几台机器也是一样喜欢玩。

后来大数据的家族就突然间多了起来。最主要的就是Spark，Storm，Mahout，Pig，Hive这些框架，以及配套的日志系统Kafa和消息队列，搜索，NoSqlDB和ZooKeeper等。Spark，Storm，Mahout，Pig，Hive，Hadoop都算是大数据专用的框架（包括NLP了），其他的都是别的系统架构中也会使用的。

嗯。那么，什么叫大数据呢？马丹我忘记了这是给IT新人介绍的。简单来讲，大数据就是指数据非常多非常多非常多，多到一台机器放不下，要用上百台机器存储。而我们又常常需要从这些非常多的数据中找出来一些有价值的内容，所以就需要对运营的指标有要求，这也是MapReduce产生的原因，当解决了离线数据的时候，很多人又想问我能否做一些在线数据的实时处理。这就有了Storm。再加上其他的NLP模块，大数据就变成了这么一坨怪东西。

记着，大数据只在几百台机器的时候才有用处，其他的都是小打小闹，也是Hadoop一直被称为笨重的原因。你要说，几十台，甚至是十几台能不能用，也可以。只是并没有把他的价值发挥的更好。

1工作内容：

工作内容在前期会比较多一些，基础搭建还是一个挺讲究的事儿。系统搭建好之后呢，大概是两种，一种是向大数据部门提交任务，跑一圈给你。一种是持续的文本信息处理中增加新的处理模块，像我之前说的增加个分类啦，实体识别神马的。

好吧第一种其实我也不记得是从哪得来的印象了，我是没有见到过的。架构稳定了之后，大数据部门的工作并不太多，常常会和算法工程师混到一起来。

其他的应该就是大数据周边产品的开发工作了。再去解决一些Bug什么的。

2需要技能：

环境【Linux】框架【Hadoo,spark,storm,pig,hive,mahout,zookeeper】算法【mapreduce,hdfs,zookeeper】

嗯。这些框架的原理和实现都要了解的比较清楚。这才算是一个合格的大数据工程师。只会使用这些东西的人，终究还是菜鸟而已。

3发展前景

不知道。看不清楚。从某种程度来说，并不觉得对这些东西的人很NB，但是总会有一些公司人为的把这些东西加上去。所以应该说现在还处于比较混乱的地位。一般人要学的话，也确实是有一些门槛，所以，也确实是属于偏高薪的职业。

1年~3年：10K~20K 3年以上：15K~35K

成长路径：

说不上来，大数据这个部门以后还会不会存在都不一定。

4.入门门槛

入门的门槛算是比较高的吧。比不上NLP，跟搜索比较类似，主要就是五六个框架的原理问题。全熟悉下来，也就是两三个月或者是半年左右的样子。

5.哪些行业适合做大数据

IT界：后端工程师中有两年经验,或者是硕士,或者是NLP其他界：无

6.职业限制

想想该怎么说，熟悉了这些架构之后，也并没有什么东西可以做。跟NLP相比呢，在算法的深度上比不过。跟原来的搜索工程师比较相似，但是现在搜索和NoSqlDB也成为了现在系统架构的主要场景，所以在工程方面也会偏差很多。

所以有点不伦不类，高不成低不就的样子。要说以后会不会发展的很好，或者说会不会是这个职业的空缺很大，我并不知道。

7.去哪里学，怎么成长

不知道。有一些培训在做这个，但是，也只能说讲讲算法的东西。这个还是需要使用场景的。

## Android工程师

Android工程师在两年前是最疯狂的时候。一个五年的Java工程师的薪水，比不上一个三年的Android.一方面是苹果这种高端的装逼工程师成了很多人不想去学IOS的门槛，一方面是Android的应用使用人数实在是太多了。

在三年前和四年前，个人开发者还很多的时候，经常盛传一个人开发了几个Android小应用。一年收成过三百万或者是一千万的事儿。这事是真的还是假的我不知道，我身边还没有遇到过这种人，至少都是听别人讲的，也许是有人挣了钱并不会告诉我们。

反正，现在啊。这种可能性越来越少了。所以Android工程师整体来讲呢。属于高潮过后的喘息，慢慢平稳。不算高也不算少。另外，也是受H5的影响，现在有两种应用会使用H5，一种是做的特别简陋。低制烂造的，一种是懒得频繁发布版本，经常需要动态替换内容的。

前者是特殊产物，慢慢会被终结掉。后者呢，是正常的需求，以后的发展空间会越来越大。所以，Android也是会受到JS的侵蚀。记着我之前说的。前端工程师一定要Android，IOS，JS全通，然后主精一种。

还有，Android的兼容性是最烦的了。碎片化很严重，版本，机型都是有问题的。特别是哪家公司的老板用的什么机型，这款机型的适配就必须做的很好。否则。。。

以及Android是用Java开发，资料神马的都是比较全的，上手也比较容易。不过。Android做的事儿，和Java后端做的事儿，几乎没有任何关联。所以，当说Android。就是Android。当说Java。就是指后台架构。

1工作内容：

Android工程师的日常就是听产品经理讲需求，跟后端定接口，听QA反馈哪款机器不兼容，闹着申请各种测试机，以及悲催的用Android做IOS的控件。

马丹我其实并没什么东西可以了，一瞬间不知道该对这个职业说些神马。嗯。想起来了。WEB端几乎都是要学会自己切图的，Android往往会要求UI切图，原因是啥我不知道，大概是Android程序员都认为自己是程序员？而CSS工程师往往不会被当成人看？

还有还有，想起来了。各种闪退的修复，以及各种性能的调优，这些都是Android工程师的日志。崩溃啊，慢啊。开了会就没电啊，如果是要做到跟视频音频相关的东西更悲剧。Android和IOS之间还需要做一些转码。

别的似乎也没了。嗯。我又想起来了。Android其实是对地图，支付，IM，第三方登录，短信，统计，音视频，文件，照片等等要求比较多的。业务上其实一般并不会有太多的内容。也很少会要求一些动画效果。再加一些手势密码。

还有各种奇葩的问题。比如说Android的65536的问题。曾经折磨了一家公司半年多。

2需要技能：

环境【AndroidStudio，Maven,Gradle】基础【数据结构，Java，计算机网络】组件【IM，地图，支付，拍照，视频，音频，统计，分享，手势密码】

做的好的会对控件做一些自适应的改动。会把网络请求也封装一下。处理好本地缓存，处理好内存的占用。另外，接口的制订是比较重要的，而且，果断的使用Rest吧。表再使用一些过时又无聊的接口风格了。

3发展前景

单纯纯Android的话，我觉得发展前景不算大。如我之前所说的一样，差不多两到三年就已经够用了。五年的Android并不见得比三年的Android强太多，或者站在公司的角度上来说，两到三年的Android基本上就够用了。

0年~1年：6K~10K1年~3年：8K~30K3年~：20K~

成长路径：

Android工程师->AndroidLeader-》前端Leader-》

很少有Android工程师能够做到技术总监的。所以还是早做准备，Android，IOS和JS通吃吧。

4.入门门槛

门槛比较低了。比JS稍稍高一点儿，但也差别不大。主要是因为Android是纯客户端，要做就必须什么都懂。但是JS就不太一样了，有的时候要求不高也能做事儿。

这就是Android比JS的门槛高一点的原因。简单来讲的话，最好是懂Java语法，然后是懂Http请求的含义，然后懂点儿MVC的概念，就可以了。

5.哪些行业适合做Android

IT界：都可以。哪怕你是产品，运维或者是QA其他界：都可以，只要你有点儿基础。

6.职业限制

偏客户端的职业怎么说呢，做的时间长了，公司对你的要求其实并没有那么高了。这个是Android工程师有点儿不太好的地方。靠。现在已经快两点了。我有点神智不清了。

7.去哪里学，怎么成长

做Android培训的挺多的，基本上是四个月1万6的价格。所以，为嘛不考虑修真院的线上免费教学呢？

我对于看视频一直都不看好啊。

## IOS工程师

什么是IOS工程师？培训4个月，薪水10K的工程师，这就是IOS工程师。虽然有一点夸张，但是也确实存在，只是那个666的人不太可能会是你而已。

IOS之前是用ObjectC开发，现在又推出来了Swift。至少从目前来看，Swift并未成为主流和开发IOS语言。所以现在要学的IOS的话，还是会以OC为主。

从难度上来讲，IOS比Android要简单一些，Android要兼容N多机型，系统底层的API也没有封装的很好，IOS的编译器偶尔会出点问题，但是总体来讲还是比较简单的。

只是IOS对UI的要求会比较高一些。做出的效果也会炫很多。

1工作内容：

IOS工程师的工作内容真的挺简单的，听需求，定接口。做个适配，抛弃一下iphone4。还有啥。。

马丹，以我为数不多的IOS知识来讲，真的不知道还有啥了。我知道的比较复杂的系统也是各种背景高斯模糊，各种渐变，各种图片滤镜处理，其他并没有什么。

支付，地图，统计这些东西。嗯。

2需要技能：

环境【Xcode】基础【数据结构，Object，计算机网络】组件【IM，地图，支付，拍照，视频，音频，统计，分享，手势密码】

有崩的地方注意一下就好。

3发展前景

IOS是属于高端的项目必备的，因为IOS的使用者比Android的还要多一些。很多人都在解释为什么自己不用IOS，最被认可的答案就是：穷。

所以IOS的发展前景怎么样？我不知道。但是在五年之内，IOS的发展不会有问题。

0年~1年：6K~10K1年~3年：8K~30K3年~：20K~

成长路径：

IOS工程师->IOSLeader-》前端Leader-》

4.入门门槛

IOS的门槛比Android稍稍低点儿。毕竟不用做兼容真的是一件很Happy的事儿。

5.哪些行业适合做Android

IT界：都可以。哪怕你是产品，运维或者是QA其他界：都可以，只要你有点儿基础。

6.职业限制

偏客户端的职业怎么说呢，做的时间长了，公司对你的要求其实并没有那么高了。这个是IOS工程师有点儿不太好的地方。靠。现在已经快两点了。我有点神智不清了。

7.去哪里学，怎么成长

做IOS培训的挺多的，基本上是四个月1万6的价格。所以，为嘛不考虑修真院的线上免费教学呢？

我对于看视频一直都不看好啊。

## 运营

运营，产品，技术是创业Team的三大主心骨，这从侧面也显示出来了运营同学的重要性。而且，在收入上，运营也是产品和技术远远无法比拟的。你想想。技术人员做了七年，积累的是神马？一堆破代码，一身颈椎病。产品人员做了七年，积累的是神马？一堆破原型，一身铁布衫。而运营人员呢？做了七年之后，产品再烂，技术再烂，身边的渠道啊，人脉啊，行业的内幕啊这才是最有价值的。毕竟，什么事情能离开了人？

不过运营也不好做，因为很多脑残的公司都有这样的逻辑，通过一个脑残的想法拉来一个活不过三集的替死鬼产品经理，折腾了老实巴交的技术人员半年多弄出了一个不伦不类的破系统，一切就绪，就差一个NB的运营了。

如果运营做的不好，就是运营的错，如果运营做的很好，但是用户还是不喜欢，那就是产品的错，如果用户喜欢了，但是用不了系统总是闪退和504，那就是技术的错。总之，产品技术运营都有错，唯一没错的就是老板自己。呃。不对，老板的错就是太信任手下这些人了。

运营这个概念呢。在互联网上怎么兴起来的，说不上来，很多传统互联网公司在转行的时候，经常会不清这些概念，我遇到过好几家公司，都以为做个网站，找个CTO就行了。CTO下面管着产品，管着技术，管着运营---不对，你太天真了，他们会以为产品运营都是技术做的。

不过他们对如何能保障公司正常运转还是有了解的。所以对运营还是有一些模糊的概念，但是，马丹，就算是我也对运营只有一些不清楚的理解而已，他们能理解多深刻呢？简单说，产品决定做什么，技术来把它做出来，市场来把它推出去，运营来把它用起来。

嗯。我本来计划写一点市场相关的东西，后来想想，自己也实在分不清市场和运营究竟该怎么分。什么活动，策划，执行，渠道，品牌，内容，数据分析。并不能向技术一样分的这么清楚。

所以就把市场和运营混在一起了。对于你们这些新入行的人来说，基本上是老板让做什么，就做什么呗。只是不管怎么说，运营是一个非常非常有前景的，在之前的介绍里，我想你们都看到了，有很多职业不适合做CTO，而后端是做CTO的最佳人选。然而，也大多数要止步于此，很多时候，我们要是奔着CEO的话，你会发现，排名第一的是产品，排名第二是的是运营，排名第三的。。没有了。

而且，很多时候产品和运营是分不开的，也是要经常勾搭在一起的。当然，产品和技术也分不开。技术面向产品，产品和运营沟通，运营收集用户反馈，交由产品设计，然后技术开发。嗯。

1工作内容：

我之前给我们家超乖简单的介绍过，运营可以从几个大方面来分一下。

1.推广运营：这里包括各种论坛，知乎，贴吧，自媒体，大号，微博，SEO，导流等等所有的手段，简单说就是用很多种方式来扩大自己网站或者是WEB的知名度，也有可能是策划一场活动，去人民大会堂办个结业典礼，也有可能是找某个漂亮明星做阵，来做一出秀，也有可能是找很多行业资源一起来打造个什么百家蛋糕店对修真院弟子免单之类的东西。

2.内容运营：推广的话，肯定要有什么东西才会推，刚刚提到的很多内容，有很多都是要靠内容的。以前是软文，现在的方式比较多了。什么易企秀，秀米之类的H5制作，但是绝大多数都是以H5为主，大部分都是线上传播，形成热点，靠成口碑效应，一般来说，三天五天之内是传播的最佳时间，过了三天或者是五天，热度就会直线下降。

这也是内容团队需要不断的生产内容的原因。并没有一招鲜吃遍天的说法。这篇文章同样，在最初的一个星期之内，瞬间挤进来1000人，现在呢，每天稳定在4~5个人左右。

3.系统运营：系统运营并不仅仅是包括给出一个后台。开个账户，找回个密码，查一下用户的数据是否异常等。有的时候还包括你的系统的核心设置。比如说，要不要有VIP，VIP应该有什么特权，多长时间合适，能谈到多少合作伙伴，应该分几个级别。这些跟产品要做的事情比较接近了，只是这些都是要根据用户的表现不断的反馈来调整的。别的不说。现在大修真院就有好几个QQ群。这些QQ群怎么设计，怎么管理，本身也是运营工作中的一种。

4.数据分析：这个其实是运营中最有技术含量的一种，也是让运营这个岗位比产品接地气的原因。运营人员是最接近用户的，也是最了解用户行为的，在大数据的时代，对用户行为的分析真的不是一丁半点的重要-虽然在某种程度上，你一点不分析也可以，反正都没有人用。

数据分析包括很多，用户的留存率，日活，月活，召回率，转换率，跳出率，PV,UV，男女，职业，地域，年龄，浏览时长，页面热点，浏览路径，高峰低谷。这些数据运营人员要头脑清晰，把这些数学全部整合在一起，结合自己对业务的了解，然后制订定出后续的计划，看看是需要提高召回率，还是需要提高转换率，看看系统的主要用户属性，借此去寻找更多的用户。

嗯。很有意思。所以运营的工作，有的时候是需要你去申请一个百度认证，提交各种资料，有的时候是需要你去找一些合作伙伴，商量换个广告位，有的时候是需要你去谈几个合作商家，签几个战略协议，有的时候是需要你去策划一场活动，挨个去跟活动参与者去谈，有的时候是要去刷脸，请朋友来帮你站台，有的时候是需要你坐下来，写一篇色情幽默的文章，更多的时候要收集用户的反馈，帮助他们解答为什么你们的网站注册不了，短信验证码收不到等一系列问题，后台加个V，换个广告，最重要的是每周或者是每月出个报表，来给大家汇报一下当前的用户情况，给领导层做决策使用--虽然我一直都觉得完全是在扯淡，再好的数据都比不上真实的了解这个行业，数据只能是辅以支持，二手数据的价值并不是没有，只是很容易让人跑偏，就好像我今天刚好看到的说Facebook几年没改版，是因为在小范围内改版发现用户的活跃度下降--真心觉得蛋疼。FaceBook最终的价值并不应该仅仅是在页面上，这些会有影响，但是绝对不应该是一个决策的关键性因素。

然而做运营最重要的一点就是，不要去询问用户，而是去观察用户。对于这点我是非常认可的。

2需要技能：

行业知识【各种行业知识，必须要懂行业，像推广，渠道，活动，SEO这些都需要了解一些】人脉【各种刷人脸，凑热闹，换资源】文笔【文笔太差就找个小妹妹，风趣幽默点，否则没人看】数据分析【看数据，写各种各样的PPT，然后做决策】

还有就是，跟产品和技术搞好关系。运营的职位其实是天生比产品和技术高一层，也正是因为这样，才会经常招产品和技术骂，哈哈哈哈哈哈哈。特别是当一部分需要从运营这边传递到产品里的时候。

如果你绕过了产品，直接用所谓的运营驱动去做开发，那更完蛋了。

3发展前景

绝对好。无论技术发展到什么程度，无论产品变换多少次风格，运营的核心理念都不会变，积累的人脉不会变。运营，可以做100年。

0年~1年：6K~10K1年~3年：8K~20K3年~：20K~（百万年薪级别的运营多的是。）

成长路径：

运营专员-》部门经理-》运营总监-》CEO

4.入门门槛

这个说要门槛，有点谈不上。毕竟有深有浅，情商要高是必须的，其他的只能说有了更好。好的吧，就算是情商低也可以去做编辑，每天只是对着后台系统一阵改。

5.哪些行业适合做运营

IT界：都可以。哪怕你是产品，运维或者是QA其他界：都可以，只要你有点儿基础。

6.职业限制

我想想，运营人员常常会受限于资源，也常常会迷失在数据里，很多运营的人，其实是做不到那么高的职位的，对于他们来说，自己的行业积累和人脉积累没什么用处，因为只有你本身有价值，你才能跟别人互相交换。所以有的时候会高不成低不就，退回去做产品和技术，又缺少专业技能，也是一场人间悲剧。这大概也是运营这个岗位的职业限制吧，缺少干货，心里容易没底气。

# 前端怎样入门？

1、前端工具箱（dreamwear/sublime/Photoshop/SVN等）

2、零基础入门（html，CSS）

页面基本结构、文档声明、编码声明、css语法、style属性、link和style标签、id属性、基本样式、Border、Background、Font、盒模型、文本设置...

常用标签集合

header、article、aside、section、footer、nav、h1-h6、p、ul、ol、li、img、dl、dt、dd...绝对路径、相对路径、标签语义化、标签嵌套规范、SEO...

常用选择器&标签类型划分

d、class、类型选择、包含选择、群组选择、通配符、选择器优先级、标签样式初始化订制方案、超链接及伪类划分、标签类型划分及特性、inline、inline-block、block..

浮动进阶

浮动的作用、浮动的特性、文档流、浮动的各种问题、clear、BFC(块级格式化上下文)、触发BFC的条件、Haslayout、Haslayout的触发条件...

定位

relative相对定位、Absolute绝对定位、Absolute绝对定位、Fixed固定定位、inherit继承、static静态定位、默认值、zIndex层级问题、margin负值、透明度...

表格和表单

表格标签、表格样式重置、单元格合并、表单元素、表单相关的属性操作、表单默认样式初始...

### 兼容性问题处理

兼容性问题总结、浮动在IE6，7下的各种问题、表单在低版本IE的问题、处理低版本IE对新增标签的支持、CssHack、条件注释语句、PNG问题、透明度的问题、固定定位在IE低版本的处理方式...

整站进阶

样式规划、favicon、CssSprite、DataURI、隐藏元素、测试工具使用、滑动门、等高布局、三列布局、未知宽高图片在容器内水平垂直居中、文本水平垂直居中、多行文本水平垂直居中...

css3入门

transition、属性选择器、nth-of-type、nth-child、backgroundSize、box-sizing、圆角，盒模型阴影、文字阴影、rgba、表单高级、H5表单新增属性、E:not(s)、E:target、E::selection、background-clip...

移动端布局

测试环境Emulation、viewport、window.devicePixelRatio、物理分辨率、MediaQueries、rem、window.screen、移动端布局相关问题、window.deviceorientationevent、横竖屏判断...

Animation和Transform

浏览器前缀、keyFrames、Animation调用、播放次数设置、动画偶数次调用顺序、Animation的问题、无缝滚动、动画播放|暂停、rotate旋转、deg、skew斜切、scale缩放、translate位移、transform-origin、transform的执行顺序问题...

Bootstrap前端开发框架

Html语法规范、CSS语法规范、Less和Sass中的嵌套、class命名、选择器、Normalize.css、栅格系统、排版、代码、响应式工具...

3.项目实战

PC端的企业网站布局（例：sony官网）  
PC端的电商类网站布局（例：京东商城）  
移动端常见页面布局（例：微信活动页/商城）

推荐图书：

学习前端书籍自然不可少，这里给大家推荐《html5与css3权威指南》

第二阶段——CSS的学习

CSS，层叠样式表——（CascadingStyleSheets），是能够真正做到网页表现与内容分离的一种样式设计语言。相对于传统HTML的表现而言，CSS能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力，并能够进行初步交互设计，是目前基于文本展示最优秀的表现设计语言。CSS能够根据不同使用者的理解能力，简化或者优化写法，针对各类人群，有较强的易读性。

CSS看似比较繁杂，其实只要掌握了CSS中的盒子模型、定位、以及页面布局，就基本上掌握了大半啦~这时我们就以及能够对网页中各个元素进行精准的排版，做出符合我们意愿的网页啦！

关于CSS的各种属性，我们还是可以参考学习HTML那样。可以说CSS的属性几乎完全是语义化的。我们需要改变边框，那就是“border”，那我们需要右侧边框做一些改变，那就是“border-right”。很明显，接下来按照我们的需求还有“右边框的宽度——border-right-with”，”右边框颜色——border-right-color”等等等，诸如此类~

第三阶段——JavaScript

JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。

简单来说，javascript是一个可以运行在浏览器上面的语言（当然现在基于一些库/框架已经远不只是在浏览器上了），它可以操控浏览器，让浏览器听从我们的命令。这有点像一个电视剧和遥控器，而javascript就是我们的遥控器。浏览器根据javascript的指令，做出相应的反馈。比如操作DOM（也可以理解成为操作HTML），可以操作CSS。有了javascript就可以让我更加灵活&动态的操控HTML,CSS.

严格来说、HTML和CSS还算不上真正的“语言”，而javascript却是如假包换的脚本型语言，既然是语言，那就会有字符串，数组，对象等等。而我们的javascript可以处理这些与数据有关的工作，比如给数组排个序，去个重等等。

而我们依靠javascript可以做些什么呢？网页上常见的轮播图，网站的注册功能，提交我们的留言，刷新获取新闻等等。

然而因为历史原因，比如我们灰常“尊敬”的IE浏览器~出了一套自己的javascript标准，没有和主流javascript标准兼容啊！！作为一个前端er是不是应该把更多精力用在有意义的地方呢，而不是解决兼容。。。

第四个阶段——jQuery

jQuery是一个兼容多浏览器的javascript库，核心理念是writeless,domore(写得更少,做得更多)。jQuery在2006年1月由美国人JohnResig在纽约的barcamp发布，吸引了来自世界各地的众多JavaScript高手加入，由DaveMethvin率领团队进行开发。如今，jQuery已经成为最流行的javascript库，在世界前10000个访问最多的网站中，有超过55%在使用jQuery。

如果继续用盖房子做例子，jQuery更像是一个装修队。我们只需要关注我们的房子设计本身，把我的想法，理念告知这个装修队，让它来给我们处理那些杂七杂八的事情。比如会不会吵到邻居？怎么协调物业？干活总得有人手的，对吧...

在写css前，其实html的结构要是比较合理的，这样写css也会比较顺手。在写一个网页之前，建议先琢磨几分钟，不要上来就写。多去看看别人是怎么写的，然后结合自己的项目，心中有一个大致规划。可以先把最外层轮廓写好，先不着急去写某一个具体的部分。

这里给大家分享一些小技巧：

1、使用reset.css

火狐和IE这两种不同的浏览器，在绘制CSS样式方法上截然不同。这种情况下，使用reset.css重置所有的基本样式会让你得到一个全新的空样式表。（可以去网上找一下，很多的）

2、CSS缩写

CSS缩写简化了你的CSS代码，更重要的是，它让你的代码更加整洁易懂。

不是像这样创建CSS

.header{

background-color:#fff;

background-image:url(image.gif);

background-repeat:no-repeat;

background-position:topleft;

}

而是像这样创建CSS

.header{

background:#fffurl(image.gif)no-repeattopleft

}

4、实用的<li>

<li>也叫链接列表，在与<ol>或<ul>正确搭配的时候非常好用，尤其是用在导航菜单样式上。

5、少用<table>多用<div>

CSS最大的优势之一是使用<div>达到样式上的灵活多变。不同于<table>，<div>里的内容不会被锁在单元格<td>中。可以说几乎所有的表格布局都可以在<div>和样式的正确使用下完成。当然，有大量表格内容时，还是用<table>吧。

6、CSS调试工具

在设计CSS时，能够得到页面布局的预览对于优化CSS样式和纠错是很有帮助的。这里有一些免费的CSS调试工具推荐给你，你可以把它装在浏览器上：比如FireFoxWebDeveloper、DOMInspector、Firebug等

7、!Important

所有被!important标记的样式，即使它后来被重写，浏览器也只会采用被标记的那条。

!important用来强制使一个样式避免在之后的编码中被修改，遗憾的是IE不支持。

# 常见名词

首先要知道网站访问大概是什么个过程：  
假设你在浏览器地址栏输入这个问题的地址

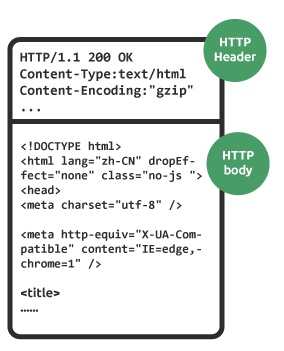
http://www.zhihu.com/question/22689579

**HTML5与XHTML**  
像人的语言一样，网络上的网页里的HTML代码也不一定是标准的，好比有时候你发音不太标准，别人会去猜测你说的到底是什么一样，有些时候前端程序猿不小心写错了HTML，浏览器也会试图猜测他们原来想写的是什么，做对应的渲染，而猜是要有一个常识做依据的。

加上有些浏览器支持一些标签，有一些又不支持，以及其他混乱的情况，为了防止大家鸡同鸭讲，我们需要对HTML代码里能有什么标签，标签怎么写，标签可以有什么属性这些东西有一定的共识，建立一个通行的标准，HTML5就是其中一个比较新的标准。

这个标准新加了很多可以用的标签和属性，然后各大浏览器也吭哧吭哧按这个标准去实现了很多这些新加的标签和属性，本来前端程序员要写一堆代码去实现的效果，现在浏览器都给你实现好了，只要写两三行，调用一下浏览器给你实现的部分就能搞定，简单愉快，所以很多人都在热情地推广这个标准~（当然新标准也不可能是完美的，总会有一些问题，怕跑题这里按下不表~）  
  
至于XHTML，就是HTML的近亲XML和HTML自己的杂交品种，对语法要求比较严格，并且为了兼容XML，在语法上与HTML有一些不同~  
  
**JavaScript与浏览器脚本**  
有了表示内容和语义的HTML，规定样式的CSS，得到的是一个静态的页面，没什么动画（其实用CSS还是可以有一些动画的，不过这个跑题了），按F5才会刷新数据，于是我们有了Javascript（JS）来给页面添加一些动态的效果，比如知乎问题的标签，鼠标移上去会弹出一个小窗口，这个就是JS实现的效果啦。  
  
  
浏览器都会帮你实现一些JS可以用的工具（函数，对象什么的），你只要写一些JS的代码，保存在xxx.js里，在html文件中用<script>关联进来就可以用了，像上图这个效果应该就包括了

1. 鼠标悬停到标签上时创建一个新的<div>小窗口
2. 用JS向知乎服务器发送一个请求，得到这个小窗口应该显示的数据，放在这个小窗口里（这就是所谓的AJAX，不用刷新就能与服务器进行交互，更新页面的一小部分~）

浏览器拿到这样的代码，就会解析并实现出相应的效果，楼主问的浏览器脚本，指的就是这样子的代码。其实用来写浏览器脚本的，也不是非得JavaScript不可，当年还为到底用什么语言写这种代码有过一段战火纷飞的时期，但现在尘埃落定，各大浏览器都默认了：请用JS写这些动态效果的代码给我解析~  
  
  
**WebServer和WebServices**  
浏览器给服务器发一个请求，服务器不是一看就知道怎么响应的。首先这些请求和响应要有一个通用的写法，也就是要有一个协议，常用的是HTTP协议。  
像最前面的图，服务器的响应写了一个状态码200OK，是HTTP协议里约定俗成的一个东西，服务器写200OK在响应里，表示“你请求的这个东西我有”，如果是404NotFound，就是“你请求的这个东西我这里没有”。  
HTTP响应里还包括很多东西，比如Content-type表示服务器发过来的文件类型是什么（文本？动画？图片？音频？），这样发过去了人家浏览器好知道怎么展示给用户看。人家服务器怎么知道按协议要写什么东西进去呢，这就是WebServer干活的时候了。  
  
形象化一下HTTP响应，大概就长这样：  
  
再上个锤子，浏览器和服务器之间请求响应的过程大致是长这样的，右下角的那些东西就是由WebServer生成的（服务器脚本可以做一些改动，但这些一般是WebServer的份内活）：  
  
  
再比如说很多时候你访问一个网站，浏览器里输的地址并没有写明你请求的文件，比如这个问题的地址是：

http://www.zhihu.com/question/22689579

但知乎的服务器其实返回了一个html给你，服务器怎么知道这个地址对应要返回什么样的html代码给你的？也是WebServer干的活。  
  
除了浏览器输地址敲回车这种赤裸裸的访问，客户端与服务器的交互还有很多种，比如：

* 前面提到的用JS完成的AJAX，有点像浏览器和服务器之间的悄悄话~

还有其他应用软件与服务器的交互，比如：

* 微信、QQ与腾讯的服务器的交互
* 网游客户端与网游公司服务器的交互
* 搜索引擎用来搜集网页信息的程序（爬虫）与各种各样的网站服务器的交互
* 只要你知道用什么地址访问、怎样访问人家的服务器，并且有相应权限，你也可以自己写一些程序去和他们的服务器交互（比如用微博API-新浪微博API获取微博，开发第三方应用或者做数据分析）。

从这些栗子里可以看出，客户端与服务器的交互的主体、客体、载体是五花八门的：

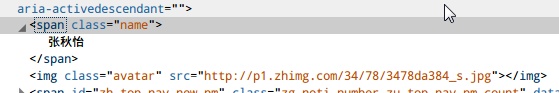
* 服务器可以是大型机也可以是个人电脑，只要能跑相应的程序就行
* 客户端像前面举的栗子里一样，可以是各种软件，而且这些软件不一定运行在个人电脑上，也可以是手机、平板、智能穿戴设备等等
* 有时候不是传生成好的HTML或者其他服务器上已经有的文件，而是传输经过一定逻辑处理后生成的字符串或者其他各种封装好的数据

像前面提到的HTML需要有一定标准一样，为了防止混乱和鸡同鸭讲，我们又需要先对这些机器需要怎么交互达成一定共识，再让它们进行交流。人与人之间通信，需要先有一种大家都认识的写法（比如简体字/繁体字）和一种彼此都懂的语言（比如普通话/广东话）。要让这些形形色色的机器能够通过网络进行交互，我们就需要指明一种协议（比如HTTP/HTTPS）和一种数据封装格式（比如HTML/XML），WebServer提供的WebService，指的就是这种协议+格式的交流体系。不过WebService的生态系统和HTML的标准不一样，用户可以选择的协议和数据封装格式更多，普通的网站访问用的HTTP+HTML只是其中一种，一些封闭系统内的交流还可以自己定义一个协议和格式来用（比如QQ）。  
  
WebService传输的数据再经由本地客户端（浏览器、QQ/微信，网游客户端等）的分析渲染，就能够以普通人能够理解的形式展现出来。此外还有一些WebService并不是为普通用户设计的，像前面提到的微博API，是用来给程序猿进行二次开发的~  
  
除了提供WebService，WebServer还会兼顾很多功能，包括提供缓存，平衡负载，这样在访问量比较大的时候能有有条不紊地接客。常见的现成的WebServer有开源的Apache、Nginx和微软的IIS，你也可以用一些工具（比如Node.js）自己定制一个。因为WebServer需要比较好的性能，所以投产时用的WebServer通常是C/C++/Java写的，但是其实很多语言都可以写，而且配合上语言底层的优化和好的模型，其他语言写的WebServer也可以有不错的表现。  
  
**PHP，服务器脚本，WebFramework**  
开头那张图里服务器接到请求之后可以给访客发送对应的文件，但21世纪的服务器怎么可能只会“接请求-发文件”这么弱智的一招呢，人家还可以处理你上传来的文件的！还可以接受你发过来的各种请求，去操作服务器本地的文件or数据库的！要干这些事，自然服务器那边也少不了要有代码了，这些代码就是服务器脚本。前面说的WebService传输的数据，主要也是由服务器脚本生成，再交由WebServer，按照某种协议套好整个响应的格式，返回给客户端的。  
  
同一个网址，每个人看到的页面不一定是一样的，比如知乎首页的网址都是

http://www.zhihu.com/

但是没登陆和登陆之后看到的东西不一样，登陆之后每个人看到的导航栏的用户信息，关注的动态，都不一样。服务器脚本可以对这些不同的状态，生成不同的页面，交给WebServer返回给浏览器。

知乎的主页给大家看到的html整体来说是差不多的，都有导航栏，左边是关注的动态，右边是广告和边栏，每一块的整体构造大同小异，只是一些地方内容有所区别。服务器脚本就是利用已知的数据，在这些因人而异的地方填入相应的内容，生成给每个人看的页面。

比如我的主页，导航栏右边的头像和名字跟别人看到的不一样，就是因为这块地方有一个放图片的<img>标签和一个写名字的<span>标签，服务器脚本在查询本地的数据之后给我返回的页面里<img>的标签填了我头像的图片链接，<span>标签里填了我的名字，给别人的页面就填其他链接、其他名字，这样每个人看到的页面就不一样了。  
  
  
PHP就是一种常见的用来写服务器脚本的语言，其实只要是能拿来写大家传输数据的通用接口（CGI）的语言都可以用来写服务器脚本（也就是说几乎所有编程语言都可以写==b），只是因为现成工具的丰富程度和专攻程度不一样，所以有一些语言在写服务器端脚本的时候会比较热门。  
  
为了方便，我们在写服务器脚本的时候，通常还会用个同语言写的WebFramework来处理各种细节，防御一些常见的攻击，提供跨站认证（比如用已有的微博账号注册其他网站）的接口，利用cookie处理登陆状态和用户设置，生成网页模版之类的。如果你用C#或者VisualBasic写服务器脚本，就可以用http://ASP.NET这个框架实现这些功能，帮你省点麻烦。不过现在不少人是反过来为了一个好用的WebFramework去选择它对应的服务器脚本语言的。

### 网站访问的过程

简单概括一下，对于我们普通的网站访问，涉及到的技术就是：

1. 用户操作浏览器访问，浏览器向服务器发出一个HTTP请求；
2. 服务器接收到HTTP请求，WebServer进行相应的初步处理，使用服务器脚本生成页面；
3. 服务器脚本（利用WebFramework）调用本地和客户端传来的数据，生成页面；
4. WebServer将生成的页面作为HTTP响应的body，根据不同的处理结果生成HTTPheader，发回给客户端；
5. 客户端（浏览器）接收到HTTP响应，通常第一个请求得到的HTTP响应的body里是HTML代码，于是对HTML代码开始解析；
6. 解析过程中遇到引用的服务器上的资源（额外的CSS、JS代码，图片、音视频，附件等），再向WebServer发送请求，WebServer找到对应的文件，发送回来；
7. 浏览器解析HTML包含的内容，用得到的CSS代码进行外观上的进一步渲染，JS代码也可能会对外观进行一定的处理；
8. 用户与页面交互（点击，悬停等等）时，JS代码对此作出一定的反应，添加特效与动画；
9. 交互的过程中可能需要向服务器索取或提交额外的数据（局部的刷新，类似微博的新消息通知），一般不是跳转就是通过JS代码（响应某个动作或者定时）向WebServer发送请求，WebServer再用服务器脚本进行处理（生成资源or写入数据之类的），把资源返回给客户端，客户端用得到的资源来实现动态效果或其他改变。

注意这只是小网站里比较常见的模型，大网站为了解决规模问题还会有很多处理，每个环节都会有一些细微的差异，中间还会使用各种各样的工具减轻服务器的压力，提高效率，方便日常维护~  
  
**延伸阅读**——**那些看花眼的名词**  
为了方便调试，很多WebFramework会自带一个简单的WebServer，或者有些WebServer会自带一个简单的WebFramework，实际部署到服务器上开放使用的时候为了性能或者安全等多方面的考虑，可以把内置的WebServer换成其他的，比如Apache或者Nginx（举个栗子，知乎用的是Tornado做Framework，Server换成了Nginx，见知乎使用了哪些框架和开源库？）。如果是开源的东西，还可以在遵守开源协议的前提下自己改一下再用~  
  
因为后端不像前端已经有HTML+CSS+JS这样的既定事实标准，服务器脚本与WebFramework的选择很多，所以新手会听到很多眼花缭乱的技术名词的地方多在这里~举一些栗子，早年常见的服务器端语言有：

* 开源的PHP
* Sun公司的JSP中使用的Java
* 微软的ASP中使用的VBScript

现在在这方面的应用热起来的语言有

* Python，对应常见的Framework包括知乎和Quora有用到的Tornado（其实是自带Framework的WebServer），社区很成熟的Django（用户包括Instagram、Pinterest）等
* Ruby，一般都用Rails这个Framework，用户包括Github、早期的Twitter等
* 逆天的JavaScript，有了Node.js这个平台，WebServer、服务器脚本和浏览器脚本全都可以用JavaScript来写……Node.js上最常用的Framework是Express
* 微软家的则跟着http://ASP.NET转移到了C#或者VisualBasic
* Erlang，擅长大规模的并发，不少游戏公司拿来写服务器，靠几十个工程师支撑几亿用户的WhatsApp也是用的这个~

几种常见的架构包括：

* LAMP=Linux+Apache+MySQL+PHP（P还可能是Python或Perl。有时候L会改成W=Windows。），也就是服务器上的操作系统是Linux，WebServer用Apache，数据库用MySQL，服务器脚本用PHP，这些都是开源技术，网站起步时用起来的成本会比较低，所以是普通网站里非常常见的架构（虽然对于发展得很大的网站会遇到很多瓶颈），Facebook就是这种，淘宝也曾经是。
* J2EE，Java世界的架构，通常是企业用的（银行、大型公司,.etc），比较常见地还会搭配一种UNIX做操作系统，Apache做WebServer，Tomcat转换JSP到Java给服务器程序用（其实它也自带WebServer），Oracle数据库等等。不一定拿来建站，常常用来提供企业里的各种需要用到网络的业务。我们学校教务系统就是用J2EE做的=。=淘宝现在也是从LAMP转型到了这个。
* http://ASP.NET，微软家的架构，通常会搭配WindowsServer操作系统，SQLServer数据库，IIS做WebServer。StackOverflow和京东（曾经）就是这个架构。
* 神奇的MEAN架构，MongoDB做数据库，Express做WebFramework，Angular做前端的JavaScript框架，Node.js用于编写WebServer。神奇之处在于这几个东西的语言都是JavaScript（MongoDB的实现不是，但与外界沟通用的语言是）。因为是比较新的架构，还有待时间的考验，不过被很多人（尤其是靠JavaScript吃饭的前端程序猿们）热切关注。
* 一般来说重点不在技术而且在乎成本的新网站比较喜欢用LAMP，重视安全稳定和速度的企业和机构喜欢J2EE，想省事的网站喜欢http://ASP.NET，比较Geek的网站和创业公司喜欢折腾各种Python、Ruby、Node.js世界的东西，Google这样现成的技术都解决不了需求的超大型网站就自己折腾解决方案。

虽然可以用的语言和所属体系五花八门，其实服务器端程序要做的事情本质上都差不多的，就好比自然世界中要表达“吃过了没”这句话的意思，你可以用各种各样的语言在各种各样的场景里表达出来~

# Web前端怎样入门？

对于绝大部分零基础转行的同学来说，你要做的不是想着怎么快速入门前端，而是应该补回计算机基础知识，拥有最基本的专业素养。可以直接拉到答案的最后了……

统一更新下前端现状：

文本编辑器：VSCode+一系列插件

包管理工具：npm

JS：紧跟每年更新的ES标准，强烈推荐上TypeScript

CSS：多了个PostCSS，以及CSSModules等一系列模块化方案……

框架：React全家桶，Vue全家桶，本身就是全家桶的Angular（至少擅长其一）

构建工具：webpack

流程工具：npmscript足够了，实在不够再用gulp

额外值得强烈推荐的：函数式编程，响应式编程（RxJS）

前端是干什么的我就不赘述了，推荐看这个答案：Web建站技术中，HTML、HTML5、XHTML、CSS、SQL、JavaScript、PHP、ASP.NET、WebServices是什么？-张秋怡的回答直接进入正题。前端入门是一个螺旋上升的过程，既要反复看书，也必须抓紧时间实践。

IFE是一个绝佳的练习平台，题目都是开源的。15年的学习资料比较多，而16年的题目题量较大，难度也稍大，新手很容易在task1-6丧失信心，那个任务实在是太麻烦了。所以我个人建议先刷15年的task1和task2，都是很基础的题目了，不会做也可以参考别人的代码。

* IFE2015：ife/2015\_spring/taskatmaster·baidu-ife/ife·GitHub
* IFE2016：百度前端技术学院

我把这三个月大致分为三个阶段。

**1.HTML+CSS**前端的入门门槛极低，体现在HTML和CSS上。运行环境就是浏览器，推荐Chrome。你需要的只是一个文本编辑器，推荐SublimeText3，有不少好插件比如Emmet，谷歌搜一下很容易了解到的。当然你非要用记事本的话，也不是不行的。刚入门查阅资料可以用w3school或者MDN。

HTML和CSS不是编程语言，前者只是结构标签，后者则是样式配置，入门是非常简单的。网上资料也有很多，推荐慕课网HTML+CSS基础课程。我当时是看了一本书*HeadFirstHTMLandCSS*，讲得浅显易懂，不过价格比较感人，也是只翻一遍的书，没有必要买了。迅速刷一遍慕课网，对HTML和CSS有个大致印象就好。想巩固HTML标签可以去看看16年的task1-1。HTML5的API可以先放一放，回头再看。

接下来就是深入学习CSS了。推荐：

* 《CSS权威指南（第3版）》。很枯燥的一本书，但我确实不知道哪本书更适合了。花两三天硬啃下来就好了。属性细节不必记忆，以后用到肯定要再查的。着重点放在大局上，比如盒模型，浮动和定位这些，抓住重点快速过一遍。
* 《CSS3专业网页开发指南》。CSS3也是需要掌握的内容。但这里还是以了解为主，知道CSS3有什么内容就好，记忆属性是枯燥且毫无意义的。

以上内容用时5天左右，下面是实践。学了几天HTML和CSS了，应该也有点成果了。打开IFE2015task1，写个静态页面吧。

我X，完全写不出来。

这是正常的。去看下别人的代码吧，看一小部分就开窍了。忘掉的属性就查书或者w3c，多尝试，不断踩坑才有进步。

重写吧，是的。重写之前先看一份代码规范GitHub-ecomfe/spec:Thisrepositorycontainsthespecifications.。当然代码规范不是唯一的，我最早看的是这一份，所以代码风格也一直维持到现在。再去看一下别人提交的代码，多看几份。当然自己也要判断，不能听风就是雨啊，人家写得不好你再去重写一次，等于你也有责任对不对。开始重写了，会发现功力大增，写代码速度也快了很多的。写到第三张页面的时候，应该比较熟练了。如果看到布局就大概知道应该怎么写了，那就可以进入JavaScript的学习了。以上内容用时10天左右。

进阶部分可以回头再看：

* 掌握预处理工具Sass，自动化工具Gulp。
* 阅读Bootstrap源码。
* 《CSS揭秘》，极其惊艳的一本书，涵盖了CSS3的很多奇技淫巧，虽说有些地方不太实用，但让人眼前一亮，很值得看。

**2.JavaScript**

这是至关重要的阶段。

* 强烈推荐《JavaScript高级程序设计（第3版）》，俗称红宝书。前七章是重中之重，必须反复阅读，直至完全理解，期间可配合其他书一起读。DOM，事件流，表单，JSON，Ajax与最后几章也相当重要。其余章节可以略读或跳过（比如浏览器嗅探，XML以及那些列举大量API的章节，完全可以用到再查）
* 推荐《JavaScript语言精粹》，俗称蝴蝶书。超薄的一本，半天就可以看完。JavaScript是一门有很多坑的语言，我个人是喜欢把这些坑点全部搞清楚的，但这本书却避而不谈了，剩下的也就是所谓的“精粹”了。但清晰地过一遍知识点总是好的。
* 强烈推荐《你不知道的JS》。精彩至极的一本书，将JavaScript的坑一网打尽。之前搞不懂的问题，比如闭包，this之类的都可以在这里找到答案。
* ES6也是必学的内容，推荐阮一峰老师的《ES6标准入门》。但这本书以API居多，所以还是留个大概印象，以后写到类似的地方，查一查有没有ES6更简洁的写法就好，不必死记硬背。以及需要学会Webpack的使用，Babel和模块化就靠Webpack了。
* 不太推荐《JavaScript权威指南》，也就是犀牛书。那就是一本字典……

进阶：

* 《JavaScript设计模式与开发实践》，设计模式是必须了解的内容，这本是写得不错的。
* 《高性能JavaScript》，红宝书作者的另一力作，讲了一些优化技巧与性能瓶颈问题，值得一读。

以上内容用时1个月左右，中途可穿插IFE2015task2的题，比2016年的要简单。这些书全部刷完的话，应该可以跟人谈笑风生了。开始实践，IFE2016阶段二的题，想怎么刷就怎么刷吧。如果有编程经验的话，应该没什么压力了。

**3.JavaScript框架**这部分就比较自由了，每个人点的技能树都不一样的。前端的发展是爆炸式的，换工具比翻书还快，所以还是以看文档为主了。目前主流框架经常被提及的是React，Angular，Vue。知乎搜一搜就有相当多的优秀答案了。不过这个答案也有时效性，说不定过两年这些框架全都被淘汰了呢【逃学习至少一种框架，把IFE2016刷通关吧。耗时1个半月左右。

进阶：看各种源代码。这也是我最近打算做的事情，但是好像期末考要到了TAT

最后补充一下，计算机基础知识是很重要的。由于本人有OI的经验所以稍微占点优势。再推荐几本书，抽空还是要看看的：

* 《深入理解计算机系统》，CSAPP，也是我们专业这学期的课程（但是我的专业明明是EE啊）。
* 《计算机网络自顶向下方法》，看名字就知道必读了吧。
* 操作系统好书挺多的，推荐一本*OperatingSystems:ThreeEasyPieces*，英文不够好就《现代操作系统》吧。
* 算法和数据结构，推荐两本：《算法导论》《数据结构与算法分析》。似乎算法和数据结构与前端关系不大，但作为一个码农，不要求你写红黑树，至少快速排序和二分查找这种要会写的吧。