---

link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=28 sentences=124, words=480

1.大厂

2.职级

3.晋升

- P5 对应工作 0～2 年，大部分人会在参加工作后 1～4 年内升到下一级别
- P6 对应工作 2～5 年，大部分人会在参加工作后 3～8 年内升到下一级别
- P7 对应工作 5～8 年
- P8 对应工作 8～12 年

4.能力模型

4.1编程能力

- 创造高质量合规代码(具备低耦合，高扩展性，高性能，安全性等特征）及相关工具的能力。**4.2工程能力#**
- 使用各种（开发、调式、编译、测试、部署等）工具，提升项目开发过程中编码体验和交付质量的能力。**4.3架构能力#**
- 灵活运用各种编码、工程、网络等技术，从技术选型、模块划分、开发部署、产品交付、系统监控等多个方面，从无到有地设计、统筹大型前端项目的能力

5.级别

** 5.1 初级(一般外包开发公司,月薪8000左右) #**

5.1.1 编程能力

- 可以使用 HTML 和 CSS 还原简单的页面
- 可以使用 HTML 和 CSS 还原较复杂的页面，并可以使用 JavaScript 做一些简单的动效
- 熟练掌握 HTML/CSS 和 JavaScript ，能和其他岗位配合，做相对简单的网站或系统开发
- 进一步夯实语言基础（如面向对象、正则表达式等），熟悉常见的算法，掌握一种流行的开发框架**5.1.2 工程能力#**
- 了解常用前端开发工具，并对代码进行测试、调试定位问题
- 了解常见的工程化工具（Webpack，Babel 等），熟悉网络抓包工具（Fiddler，Whistle等），熟悉版本管理工具（Git等）**5.1.3 架构能力#**
- 熟悉基本的网络的知识（例如：HTTP、TCP/ IP）

** 5.2 中级(创业公司、一线互联网企业月薪 20000 左右) #**** 5.2.1 编程能力 #**

- 熟悉多种流行框架，深入理解其中一种的设计原理；熟悉常见的设计模式；了解 Node.js 的一些基本概念
- 可以提炼模块，输出高质量的代码，并且有基本的性能考量；可以比较快速地熟悉小程序开发、TypeScript 等。
- 熟悉前端技术标准，熟悉浏览器兼容性问题，熟练掌握和运用常见的设计模式；开始熟悉 Node.js 常见 API 。
- 能编写符合规范、高性能、易维护的代码，并注重代码的复用性、维护性和扩展性。**5.2.2 工程能力#**
- 更深入的认识工程化工具（Webpack，Babel）、版本管理工具（Git）等各种工具，了解其设计原理。
- 掌握自动化测试工具来保障代码质量，编写文档来保障代码的可读和可维护。
- 能给各种工具编写插件，可以串联掌握各种工具，初步实现开发、测试、部署的自动化。
- 能根据项目需要编写（常见或不常见的）工具，提升代码质量和提升开发效率（例如：NodeJS扩展，浏览器插件，Electron工具）**5.2.3 架构能力#**

- 了解浏览器/小程序等常见运行端的技术原理及其架构；开始认识到网络安全的重要性
- 深入理解HTTP协议，及其常用用法。综合网络和端（浏览器、小程序等）的设计原理，熟悉网站的各种性能指标。
- 熟悉业务开发中常见的前后台通讯协议，（例如：Protobuffer，JCE，HTTP最佳实践）；熟悉服务器开发的基本过程
- 有日志、监控的意识，注意生产环境的质量；有系统层面定位问题的能力（协议不合理、网络瓶颈、浏览器限制等），并给出优化方案

** 5.3 高级("一线互联网企业、BAT月薪 30000 以上") #**** 5.3.1 编程能力 #**

- 综合 T1-T8 的编码能力，有能力从整个项目的角度提炼和输出组件和框架，同时有能力指导他人写出更好的代码。**5.3.2 工程能力#**
- 从团队和业务角度，建设工程化体系，创造生产力引擎/生产线，系统化的提升效能，并用于组织大规模生产。**5.3.3 架构能力#**
- 综合编码能力和工程能力，同时结合对客户端、网络、服务器等节点的理解，独立完成中大型项目的设计。

6.简历

- 大厂简历筛选有一套机制，有大厂经历或学历好或经验匹配的会比较容易通过筛选，缺少光环的需要有其他东西来证明，比如优秀的项目经历，参与过好的开源项目等
- 简历上描述的技术/内容/项目确保自己是真的熟悉/掌握，看看每个技能是不是自己真的掌握了，能说出个1.2.3；每个项目是否自己能说清楚，一些细节是否了解，有哪些复盘点，是否有改进空间
- 简历上描述的应该是与目标岗位直接/间接相关的，其他的比较优秀的点可以一笔带过，不需要花大篇幅介绍这些与目标岗位不符的能力
- 面试官简历评估时也会看跳槽频率，像1年1跳这种会被评为不稳定，这时除非学历/经历特别出色的，其他基本就不通过了

7.面试

- 面试除了技能/项目知识外，状态也很重要；接到面试电话说明简历评估通过了，时间可以你自己定，如果没准备好，可以把时间拉长些，给自己一些准备时间；要求当场面试的可以礼貌拒绝然后定一个合适的时间
- 对不同工作年限的同学会有不同的要求，校招主要看潜力，所以基础（计算机、网络）和算法会考得比较多；1~3年除了潜力外还看经验是否与业务匹配，项目经验；3~5年看是否有独挡一面的能力，需要在技术上有较好的深度，在做事情方面有自己的一套；大于5年的除了深度外对广度也有要求，且需要有跨端和架构设计的能力，对于管理岗位也会看带团队的能力
- 面试时遇到不会的不用慌，每个人的知识面不一样，碰到不会的很正常，但可以积极思考，首先坦诚表示没有了解过相关知识，然后以现有的知识体系思考下这个问题，说明思路，合理猜测结果
- 有时会有面试官会刻意施加压力，这时不在于问题回答的是否正确，而在于是否能在这些压力下仍然能够理性思考，面对面试官的每个问题，可以尝试想下面试官问这个问题的背后目的是什么

8.知识点

** 8.1 CSS #**

- 盒子模型
- CSS选择器
- BFC
- position
- flex布局
- css优先级
- 双飞翼/圣杯布局
- CSS3新特性
- CSS样式隔离
- CSS性能优化
- 层叠上下文
- div居中
- 浮动**8.2.JavaScript #**
- 原型链
- 继承
- 作用域
- 闭包
- 变量提升
- this的指向
- 立即执行函数
- instanceof原理
- bind的实现
- apply和call
- 柯里化
- v8垃圾回收机制
- 浮点数精度
- new操作符
- 事件循环机制
- promise原理
- generator原理**8.3.HTML&浏览器#**
- 行内元素、块级元素
- 跨标签页通信
- history和hash两种路由
- DOM树
- 事件模型
- 缓存策略
- 浏览器架构
- 浏览器工作原理
- 内存泄露**8.4.性能优化#**
- 前端性能优化指标
- 前端性能优化手段
- 重排和重绘
- 白屏
- 大量图片加载优化
- 描述下浏览器从输入网址到页面展现的整个过程
- 动画性能
- 渲染合成层**8.5.工程化#**
- 模块化机制
- tree shaking
- uglify原理
- babel原理
- webpack工作流程
- webpack插件机制
- webpack loader机制
- 前端微服务**8.6.React #**
- 合成事件
- virtual dom
- setState过程
- fiber
- 高阶组件
- 错误处理
- 性能优化
- redux核心原则
- redux核心逻辑**8.7.Vue #**
- 数据绑定原理
- computed和watch
- slot
- next tick原理
- keep alive **8.8.算法#**
- 斐波那契数列
- 合并二维有序数组成一维有序数组
- 链表：反转链表

- 链表：链表有环
- 堆栈队列：判断括号字符串是否有效
- 返回数组中第k个最大元素
- 找出数组中和为sum的n个数
- 贪心：具有给定数值的最小字符串
- 二叉树：最大深度
- 二叉树：层次遍历
- 剪枝：判断数独是否有效
- 二分查找：求解平方根
- 字典树：实现一个字典树
- 爬楼梯问题
- 最短距离
- LRU缓存
- 翻转二叉树**8.9.编程#**
- 实现一个trim方法
- 实现一个deepClone方法
- 实现 add(1)(2)(3)
- 大数相加
- 拍平数组
- 实现防抖函数
- 实现节流函数
- 实现字符串翻转
- 数组去重
- 实现千位分隔符
- 判断是否是回文数
- 实现一个模板引擎
- 判断一个数是否是素数
- 获取n以内所有的素数**8.10.操作系统#**
- 进程和线程
- 进程通信
- 进程调度策略
- 死锁
- IO多路复用**8.11.网络#**
- 七层网络模型
- http
- https
- http2.0
- http3.0
- websocket
- tcp
- udp **8.12.Node #**
- 模块机制
- require原理
- 事件循环
- cluster原理
- 流机制
- pipe原理
- 守护进程
- 进程通信
- 异常处理**8.13.架构与模式#**
- 常用设计模式
- 重构
- MVVM
- MVC
- MVP

姓名:

密码: