

2021大厂前端核心面试题详解 三

- 面试题分析
- 如何写出漂亮的简历
- 自由提问

面试题


这里是部分字节跳动、百度、美团的面试题

编程题实战


1. 实现 `lodash` 中的 `get` 函数. 难度: 

```
// get(data, 'a[3].b')



const get = (data, path, defaultValue = void 0) => {
  // todo
}
```

2. 实现 `add(1)(2)(3)` 难度: 

```
const add = (a, b, c) => a + b + c;

// curry 
const curry = (fn) => // todo

// homework: 不定参数实现
```



3. 简单实现 `async` 难度:  

```
function spawn(fn) {
  // todo
}
```

工程化问题实战

1. 如何优化 `node` 镜像制作 难度: 

- DOCKER_BUILDKIT 查看 dockerfile instruction 耗时
- FROM YOUR_OLD_DOCK 基于历史最新的业务镜像构建
- COPY 等指令, 充分利用 cache
- 优化 OS 大小, alpine
- npm i --only=production 移除 devDependencies
- 抽出来放 CDN
- ...

2. `webpack` 热更新原理 难度:  

```
/**
```

```

*      内存文件系统
*      |
*      读写
*      |
*      webpack compile      - watch -      代码
*      |                                |
*      -----change
*      |
*      server(websocket) --> manifest(hash.hot-update.json / hash.hot-
update.js) | hash & chunk
*      |
*      |
*      Browser: hotDownloadManifest(拉 manifest)
*      |
*      | get hash chunkid
*      |
*      hotDownloadUpdateChunk(拉 chunkjs 文件)
*      |
*      |
*      hotAddUpdateChunk(update the chunk)
*      |
*      |
*      hotUpdateDownloaded
*/

// homework: 思考如何让传统的 webpack hmr 更快?

```

开放性问题实战

1. `obj.a.b.c` 和 `obj['a']['b']['c']` 哪一个性能更好?
2. 如何突破 `localStorage` 的大小限制?

算法题实战

1. 最短编辑距离算法问题 难度: 

```

// 给出两个单词word1和word2，计算出将word1 转换为word2的最少操作次数。

// 你总共三种操作方法：

// 插入一个字符
// 删除一个字符
// 替换一个字符

// 解答 

const levenshtein = (s1, s2) => {
  // todo

}

// homework: 思考 `Levenshtein Distance` 算法和 `React` 千丝万缕的联系。

```

如何写好个人简历

首先，基于经验的一些 tips：

1. 简历的标题一定要带上名字！！(比如：张三-本-4年-高级前端开发工程师.pdf)
2. 强烈建议不要搞些花哨的网页版简历
3. 简历最好不要超过 1 页，至多不超过 2 页
4. 简历三大原则：清晰，简短，必要
5. 简历不是一成不变，可针对 JD 定制，指哪儿打哪儿

对开发人员来说，简历的核心重点是突出你的能力

对开发人员来说，简历的核心重点是突出你的能力

对开发人员来说，简历的核心重点是突出你的能力

个人介绍部分

这一 part 没啥好说的，写清楚**联系方式**，**意向工作地**，**岗位诉求**，**学校学历** 就行，排版简单点，很多公司都有 OCR 提取关键信息，不要难为它们就行。

个人技术部分

一般来说，我不介意大家写一个「个人技术」的列表，大部分都会给自己挖坑，当然确实是精通的同学当我没说。

如果非要写，切记以下内容不建议写上去：

1. 其实你不是很精深的知识领域，比如只是配置过 `webpack`
2. 常识性的知识技能不要写了，比如你写上去熟练掌握 `git` 操作，wtf?
3. 过时已久的也不建议写了，比如你写一个熟练使用 `backbone`

注意以下措辞的区别：

1. 了解：表示你听说过这个概念，甚至了解与此概念有关的基本原理
2. 熟悉：表示你通过 Demo 的形式实践过某个技术，或做过一两个与该技术有关的项目，但缺乏沉淀
3. 熟练掌握：表示你在工业级环境下，通过数个项目的实践已经掌握了某种技术的核心原理，并能够灵活地应用在开发中
4. 精通：表示你通过很多次的项目实践和潜心研究，已经对某种技术的原理和应用掌握到近乎尽善尽美的程度

加分项：

1. 社区博客文章，高转发高点赞那种，加分
2. `Github` 一片绿油油的，主导/参与维护一些社区知名项目，`PR` 内容都是逻辑代码级别的，不是文档
3. 除前端常用技术栈，还掌握了一些硬核技能，比如 `rust` 开发 `wasm`，`c++`，...

个人项目部分

对于大多数公司来说，招聘的开发都是做业务的。所以这部分是**很重要很重要很重要的！**

严格说，这一 part 基本占你简历评分的 60% 以上

基本按照 `STAR` 原则写就行。

bad case:

我在该项目中完成了 XXX, YYY 需求, 运用了 a, b, c 技术。

good case:

XXX 项目出现 XXX 问题, 我作为 XXX, 负责其中的 XXX 部分, 我通过 XXX 方式 (或技术方案) 成功解决了该问题, 使 XXX 提高了 XXX, XXX 增长了 XXX

这当然基于事实的基础上也是有一些技巧的 (阴险脸)

提问时间
