阿里社招两年前端面经

面试流程：简历筛 -> 一面技术面 -> 二面技术面 -> 三面技术面 -> 四面 HR 面 -> 电话沟通薪资 -> 体检 -> 背调 -> offer （其中任何一个流程挂掉都没有下一步）

注：以下文章内容涉及项目细节的已打码或者未记录。

「一面 (2h)」

» 1.1 手写深拷贝（30min手撕代码）

> 题目是一段代码，乍一看是深拷贝，再咋一看其实是复杂一点的深拷贝，拷贝元素囊括对象、数组、日期、正则、DOM 树。

```js
// 编写一个深度克隆函数，满足以下需求（此题考察面较广，注意细节）
function deepClone(obj) {}

// deepClone 函数测试效果
const objA = {
  name: 'jack',
  birthday: new Date(),
  pattern: /jack/g,
  body: document.body,
  others: [123,'coding', new Date(), /abc/gim,]
};

const objB = deepClone(objA);
console.log(objA === objB); // 打印 false
console.log(objA, objB); // 对象内容一样
```

优化后的代码：

```js
// 深拷贝：对对象内部进行深拷贝，支持 Array、Date、RegExp、DOM
const deepCopy = (sourceObj) => {
  // 如果不是对象则退出（可停止递归）
  if(typeof sourceObj !== 'object') return;

  // 深拷贝初始值：对象/数组
```

```
  let newObj = (sourceObj instanceof Array) ? [] : {};

  // 使用 for-in 循环对象属性（包括原型链上的属性）
  for (let key in sourceObj) {
    // 只访问对象自身属性
    if (sourceObj.hasOwnProperty(key)) {
      // 当前属性还未存在于新对象中时
      if(!(key in newObj)){
        if (sourceObj[key] instanceof Date) {
          // 判断日期类型
          newObj[key] = new Date(sourceObj[key].getTime());
        } else if (sourceObj[key] instanceof RegExp) {
          // 判断正则类型
          newObj[key] = new RegExp(sourceObj[key]);
        } else if ((typeof sourceObj[key] === 'object') && sourceObj[key].nodeType === 1 ) {
          // 判断 DOM 元素节点
          let domEle = document.getElementsByTagName(sourceObj[key].nodeName)[0];
          newObj[key] = domEle.cloneNode(true);
        } else {
          // 当元素属于对象（排除 Date、RegExp、DOM）类型时递归拷贝
          newObj[key] = (typeof sourceObj[key] === 'object') ? deepCopy(sourceObj[key]) : sourceObj[
        }
      }
    }
  }


  return newObj;
}

// deepCopy 函数测试效果
const objA = {
  name: 'jack',
  birthday: new Date(),
  pattern: /jack/g,
  body: document.body,
  others: [123,'coding', new Date(), /abc/gim,]
};

const objB = deepCopy(objA);
console.log(objA === objB); // false
console.log(objA.others === objB.others); // false
console.log(objA, objB); // 对象内容一样
```

» 1.2 自我介绍

面试官你好，我叫XXX，XXXX年毕业于XXXX大学XXXXXXXX专业，（在校实习经历），（目前就职于），（当前岗位职责）。

XXXXXXXXXX...（围绕当前就职公司项目展开，可以初步介绍简历上的项目经验）

XXXXXXXXXX...（围绕自己在当前就职公司做的贡献，体现你的个人价值）

XXXXXXXXXX...（介绍一下在当前就职公司的绩效情况）

XXXXXXXXXX...（简单介绍一下跳槽的原因）

» 1.3 简历问答

1.3.1 工作经历

> 基于自我介绍提及，扩展回答。

1.3.2 绩效情况

> 基于自我介绍提及，扩展回答。

1.3.3 为何在 Vue 和 Angular 和 React 中选了 Vue?

刚入职的时候XXXX有用 Vue 的有用 Angular 的，以前坐我隔壁的XXXX就是用 Angular。

在2020年初的时候我们团队决定统一前端技术栈，当时我也跟领导就是有交涉交流过一些想法。目前市面上三大前端框架 Vue、React、Angular，三个框架本质上没有优劣之分，但技术选型主要考虑合适的场景。选 Vue 第一个是：

1. Vue 好上手，对新手友好，人员变动情况下能够更快地承接项目。
2. Vue 技术栈在市场占有率比较高，在招聘上能够选择的人员更多。
3. Vue 技术社区更新迭代快，社区活跃度高，遇到问题方便查。

1.3.4 JavaScript 和 TypeScript 有什么区别？在技术选型的时候什么因素使你考虑用?

TypeScript 是 JavaScript 的一个超集，它本质上其实是在 JavaScript 上添加了**可选的静态类型**和**基于类的面向对象编程**。

TypeScript 的特点：

- 解决大型项目的代码复杂性
- 可以在编译期间发现并纠正错误
- 支持强类型、接口、模块、范型

最开始技术选型的时候，主要考虑是XXX这个项目它不算特别大但是也不算小，而且行内系统可以起到一个试验性参考性的示范。它又是包含了桌面端和移动端，后面有团队想在自己的团队内推广这样的技术选型，都是参考XXX的项目设计。

在实际的使用中，最大的好处还是：第一个是强类型，规范大型工程中变量声明，可控可预知，减少不同开发人员引入的隐性 bug。第二个是接口，在XXX里面的接口，其实主要是用于定义数据结构，也是规范数据结构的作用。第三个是继承，避免重复实现一些功能，`protected`、`public`、`private`等关键字也可以实现方法的隔离。

1.3.5 关于桌面端 UI 框架选型的思考：为何在 Element 和 iView 中选了 Element?

前端技术框架在选定 Vue 的情况下，当时我在跟领导敲定室内桌面端框架的选择有两个：一个是 iView、一个是 Element。最初我是比较青睐 iView 的，因为整个 UI 设计看起来比 Element 更加清爽，然后在最初XXX重构的时候直接 iView 用上去。

最初使用的时候就发现有个地方其实两个框架有很大区别，Element 的 Table 倾向于元素展示在 HTML 结构上，iView 的 Table 倾向于 template 模式，只有一个 table 标签，所有的数据和内部行列结构都放在 data 里面。（虽然 iView 现在也支持非 template 结构了）

我个人是比较倾向于 Element 的 Table 展示形式，一目了然能看清整个数据结构，而无需把结构类相关的东西绑定到数据里面，使得数据是数据，展示是展示。

有趣的是，当我使用 iView 开发了一个 Table 的时候，发现 IE 浏览器展示不出表格，查了一下发现它不支持 IE，所以我就用了一个周末两天的时间把 iView 整个替换成 Element，这是一个惨痛的教训哈哈。

说回正题，最后敲定 Element 而不选用 iView 的原因：

1. 从 GitHub 上看，Element 50.7k Star，iView 23.9k Star。Element 的用户群更大。
2. Element 是饿了么开发开源 UI 组件框架，背后靠着阿里；iView 已经进行商业化了，有一些付费定制化的组件出现。如若是没有采购预算的情况下，Element 的全开源性质和文档清爽度会更有优势一些。
3. iView 的向下兼容性会差一点。

1.3.6 关于移动端 UI 框架选型的思考：为何在 Vant 和 Cube 中选了 Vant?

前端技术框架在选定 Vue 的情况下，在移动端框架选型上也走了一些弯路。在 2019 年刚进XX的时候做完新员工项目之后开始做票据移动端，当时选用的框架是 cube-ui。后面在做XXX移动端的时候基本已经敲定室内统一使用 Vant 了。

选用 Vant 而不选用 cube-ui 的原因：

1. 从 GitHub 上看，Vant 18k Star，cube-ui 8.8k Star。Vant 的用户群更大。
2. Vant 属于有赞自己开发的开源 UI 框架，cube-ui 属于滴滴自己开发的开源 UI 框架，cube-ui 官网经常打不开，更新活跃度在逐渐下降。Vant 的社区更加活跃一些。
3. Vant 样式设计风格更加通用，cube 设计风格统一黄灰色比较明显的滴滴风。

1.3.7 互联网公司和金融科技公司在开发流程上的区别

- 互联网公司：`产品经理+交互设计+UI设计` -> `前端开发+后端开发` -> `测试`
- 金融科技公司：`业务` -> `开发` -> `测试`

» 1.4 工作方式：当需求方在开发周期内临时改动需求时怎么办?

- 首先是对改动点的评估，前端人员初步进行评估，假设是"改动按钮颜色"之类的小改动，可以直接更改但是要做好需求登记。
- 其次当改动点较大时，举个例子假如是"新增加一个金融推广活动页面"，前端后端测试共同评估是否对开发周期造成影响，若是超过开发周期太大的时间可以跟需求方提出将改动放到下个迭代；若是需求改动要求比较急，在请示上级领导评估之后，可以加急改动。

» 1.5 vue2 双向绑定原理

> 双向绑定是什么?

首先明确一下双向绑定和响应式的概念，双向绑定是双向的，表示**数据改变驱动视图改变，视图反过来也可以改变数据**。响应式是单向的，只代表**数据改变驱动视图改变**，响应式的主要原理是**数据劫持**和**观察者模式**，是 Vue 最核心的模块。

> Vue 双向绑定和 React 单向绑定

其中 Vue 和 React 的区别之一就是：**Vue 是双向绑定；React 是单向绑定，因为 React 视图的改变需要手动执行 `this.$setState()` 来改变数据**。

» 1.6 Vue2 数据劫持的原理

数据劫持核心是 `defineReactive` 函数，里面主要使用 `Object.defineProperty` 来对对象访问器 `getter` 和 `setter` 进行**劫持**。数据变更时 `set` 函数里面可以通知视图更新。

在使用 `Object.defineProperty` 进行数据劫持的时候，对象和数组是分开处理的：**对象是遍历对象属性之后进行递归劫持；数组是重写数组的原型方法比如 splice**。这个我看了一些源码和资料。`Object.defineProperty` 本身是可以监控到数组下标的变化的，但尤大在 github issue 回复过从性能/体验的性价比考虑弃用了这种对数组的劫持方案。举例子就是对象属性通常比较少对每一个属性劫持不会消耗太多性能，但数组可能有成千上万个元素，如果每一个元素都劫持，无疑消耗过多性能。

» 1.7 Vue2 数据劫持的缺陷

第一个缺陷是由于 Vue2 数据劫持底层是用 ES5 的 `Object.defineProperty` 实现的，所以不兼容 IE8 以下。

第二个缺陷是 Vue2 数据劫持无法检测数组和对象的变化，只会劫持一开始存在 data 选项里面的数据，这就是官网建议我们把可能要使用的数据一开始声明在 data 里面并提供初始值。对象新增属性可以通过 `Vue.$set()` 进行数据劫持，数组新增元素也可以通过 `Vue.$set()`，或者因为数组原型方法已经被重写了可以用 splice、push、unshift 等方法新增元素。

» 1.8 Vue3 数据劫持的优势

Vue3 数据劫持底层主要是使用 ES6 的 Proxy 实现。

`Proxy` 的优势如下:

- Proxy 可以直接监听对象（`const proxy = new Proxy(target, handler)`）；defineProperty 需要遍历对象属性进行监听。
- Proxy 可以直接监听对象新增的属性；defineProperty 只能劫持一开始就存在的属性，新增属性需要手动 Observer。
- Proxy 可以直接监听数组的变化；defineProperty 无法监听数组的变化。
- Proxy 有多达 13 种拦截方法：不限于 `get`、`set`、`has`、`deleteProperty`、`apply`、`ownKeys`、`construct` 等等；除开 `get` 和 `set` 其他都是 `defineProperty` 不具备的。
- Proxy 返回的是一个新对象，我们可以只操作新的对象达到目的；defineProperty 只能遍历对象属性直接修改；

`Proxy` 的劣势如下:

- ES6 的 Proxy 的存在浏览器兼容性问题。

> Proxy 和 Reflect 结合实现 Vue3 底层数据劫持原理。Reflect 设计的目的是为了优化 Object 的一些操作方法以及合理的返回 Object 操作返回的结果，对于一些命令式的 Object 行为，Reflect 对象可以将其变为函数式的行为。比如 `('name' in obj)` = `Reflect.has(obj, 'name')`

» 1.9 Vue3 有什么新特性

Vue2.x 的组织代码形式，叫 `Options API`，而 Vue3 最大的特点是 **Composition API** 中文名是**合成函数**：**以函数为载体，将业务相关的逻辑代码抽取到一起，整体打包对外提供相应能力**。可以理解它是我们组织代码，解决逻辑复用的一种方案。

其中 `setup` 是 `Composition API` 的入口函数，是在 `beforeCreate` 声明周期函数之前执行的。还提供了 `ref` 函数定义一个响应式的数据，`reactive` 函数定义多个数据的响应式等等。

» 1.10 面试官建议

1. 简历上项目的深度分析，回答的时候要有逻辑条理和准备。
2. 工作上的沉淀和积累可以在掘金平台以文章的形式输出。

「二面 (45min)」

» 2.1 假设团队内从 Vue 转为 React，你认为要做什么准备?

1. 从团队角度上考虑：主要是做好技术培训、技术分享、建立可参考性的模版工程。
2. 从个人角度上考虑：从技术细节、底层原理上对 vue 和 react 进行对比。然后在现有的工程上面结合技术和业务进行学习，快速成长。

» 2.2 微前端解决方案 qiankun？JS沙箱的原理？样式隔离的原理？

1. qiankun 是基于 single-spa 封装的，提供了更加开箱即用的 API，使得微应用的接入像使用 iframe 一样简单，实现把应用改造的工作量降到最低。并且主应用和微应用都是技术栈无关的，解决了开发中的两个问题：第一是空间上不同团队的协同开发不必统一技术栈，第二是时间上不同版本技术栈的升级维护无需统一。因此**技术栈无关**是微前端的核心价值。

2. JS 沙箱：

- **快照沙箱(snapshotSandbox)**：在应用沙箱挂载和卸载的时候记录快照，在应用切换的时候依据快照恢复环境。qiankun 的快照沙箱是基于 `diff` 来实现的，主要用于不支持 `window.Proxy` 的低版本浏览器，而且也只适合单个实例的子应用用，且会污染全局 `window` 。
- **代理沙箱(proxySandbox)**：qiankun 基于 es6 的 `Proxy` 实现了两种应用场景不同的沙箱，一种是 `legacySandbox` (单例)，一种是 `proxySandbox` (多例)。因为都是基于 Proxy 实现的，所以都称为代理沙箱。
  - **单例沙箱(legacySandbox)**：同样会对 `window` 造成污染，但是性能比快照沙箱好，不用遍历 `window` 对象。
  - **多例沙箱(proxySandbox)**：不会污染全局 `window` 并支持多个子应用同时加载。

3. 样式隔离：微前端框架里面可能会遇到的样式冲突有两种，一种是主子应用样式冲突，另一种是子应用之间的应用冲突。

- **动态样式表(Dynamic Stylesheet)**：qiankun 自动实现子应用切换时子应用样式动态切换，能够保证你在单应用模式下（就是同时只能有一个应用活跃的情况下）保证子应用和子应用之间的样式不会冲突。
- **工程化手段(css module)**：可以通过手动的方式确保主应用与微应用之间的样式隔离，比如给主应用的所有样式添加一个前缀（或者使用库）；也可以通过配置 `{ sandbox : { experimentalStyleIsolation: true } }` 的方式开启运行时的 scoped css 功能，从而解决应用间的样式隔离问题。
- **严格样式隔离(Shadow DOM)**：默认情况下沙箱可以确保单实例场景子应用之间的样式隔离，但是无法确保主应用跟子应用、或者多实例场景的子应用样式隔离。当配置为 `{ strictStyleIsolation: true }` 时表示开启严格的样式隔离模式。这种模式下 qiankun 会为每个微应用的容器包裹上一个 `shadow dom` 节点，从而确保微应用的样式不会对全局造成影响。（但是使用时还是要适配和考虑特殊情况）

» 2.3 长列表渲染底层原理？业内有哪些解决方案？

- 百万PV商城实践系列 - 前端长列表渲染优化实战

「三面 (20min)」

> 自我介绍，技术选型，项目细节。全长 20 分钟，跟一面二面问题有些重复，团队大领导面试。主要是工作方式和技术思考方面的考察。

「四面 HR 面 (30min)」

» 4.1 你觉得前端在整个产品里面扮演什么样的角色?

1. 在技术流程上，前端担任一个桥梁的作用。在拥有比较清晰岗位职责的互联网公司，通常都是 `产品经理` 、 `交互设计师` 、 `UI 设计师` 给出 `需求文档` 、 `原型图` 、 `设计图` 给前端工程师，前端工程师再跟后端工程师进行开发联调。在实际工作中，前端工程师作为抽象产品和技术实现之间的桥梁，可以担任更多的职责，为产品实现把好最后一道关，从技术实现和真正成品的角度反馈意见给产品经理。而我目前工作中前面只有业务人员，所以角色任务更加重一些，需要帮助业务人员去落地产品的概念，出UI图原型图，核实业务人员真正的需求。

2. 从产品角度上，随着时代的发展，越来越多的产品看重前端，因为产品的客户体验具有强大的竞争力。市场上优秀的互联网产品很多，当我们的产品是对公且涉及金钱交易的尤其需要前端工程师，因为客户体验是第一位的，技术承载了业务，前端承载了客户体验，具有举足轻重的地位。

» 4.2 一个好的前端工程师具备什么样的品质?

由我上面的回答可以略知：

1. 好的前端工程师需要拥有较强的沟通能力。除开技术能力强之外，前端工程师仍然需要很强的沟通能力。因为他担任着产品需求和技术实现之前的桥梁，有些时候甚至需要担任产品和后端之间的沟通桥梁。前端工程师经常会遇到 `产品经理` 、 `交互设计师` 、 `UI 设计师` 更改需求的时候，这个时候就会需要前端工程师具有较强的沟通能力，互相沟通协调将产品真正地落地。（可以补充自己工作中的经历：比如XXX的业务沟通）

2. 好的前端工程师需要以客户体验为优先。比如当产品工程师提的需求不符合以客户体验优先的前提，前端工程师过需求的时候发现了，其实可以去跟产品沟通一个更好的实现方案。明确我们整个开发条线的目标是落地一个好产品，并不是说产品说什么就是什么，前端需要及时给产品一个反馈。（可以补充自己工作中的经历：比如XXX的业务沟通）

» 4.3 女性工程师数量比较少，作为女性，在这一方面有什么感悟吗?

一路走来确实有很多感悟，从校招到社招确实没有看到过女技术面试官，我是希望着去更大更好的地方出现可以仰望的女程序员，那我会在这条路上得到更多鼓舞。20多年时间里确实是付出了很多努力甚至更多的努力追上男工程师，也得到很多人的鼓励和帮助，我一直告诉自己，不要害怕失败，坦然接受失败，然后进步改进。未来，我希望能走的更远，在年轻的时候多见见世面多感受和多体验，珍惜当下的努力。以后回首往事的时候，一路走来每一步脚印都是勋章。

» 4.4 假设能进入阿里，你想在这里收获什么？有什么方向规划吗？

希望能收获更多成长吧，不管是技术上还是工程上，阿里提供的平台比我目前的情况要好的多。前端的话其实有三个大致的方向：管理方向、研发方向、业务方向，但是这三个方向在日常工作中其实不互斥是穿插着的。我的规划是希望能在业务和技术上精进，有一定全局的认知，在广度上深度上都有一定的了解，再考虑走上管理的岗位。

- 管理方向：作为主管带一个前端小组、项目小组或前端团队，兼职部分管理职能如招聘、打绩效等。
- 研发方向：作为技术骨干做偏技术方向的探索和实践，做各种工具链建设提效降本，如搭建平台研发。
- 业务方向：做业务型的项目为主，与业务团队一起完成公司目标，交付 toB/toC/toG 的各种项目。

这些方向并不互斥，往往是并行的，跟一个同学的工作年限、公司的业务和团队现状都有很强的关联性，比如我在阿里时候的一个主管，工作头两三年做业务开发中后台网页，后面两三年偏研发沉淀框架和组件化，再往后几年在兼顾业务的前提下，同时兼顾研发及带一个小团队，再往后一两年又重新做纯技术研发，最近又开始带一个更大的团队做业务。

» 4.5 你目前薪资总包是多少？

「总结」

社招面试整体感觉跟两年前校招面试相比，校招容易问八股文或者算法基础，社招在技术基础之上更重视项目经验和技术思考。

我其实也面试了其他的公司，总的来说社招跟不到十位面试官视频对线过，他们面对社招员工更加看重**价值思考能力**、**主动解决能力**、**管理带队能力**。很多码农（包括我）都觉得纯粹做技术写代码是最单纯开心的，但是只要人在职场，势必是要面对很多问题然后需要你出力去解决，这才是公司需要的“普通员工”。

关于非技术能力的理解和锻炼可以看看这篇：和导师去摸鱼的时候我们都聊了什么

还有就是这一次社招这波面试让我意识到了**表达沟通能力**很重要，有些问题也好知识点也罢，怎么将这些知识用口语表达给面试官确实很讲技巧。在准备面试之前可以找个人（你家的猫）跟你玩 cosplay，猫问你 Vue2 双向绑定原理，那你就思考如果用口语表达清楚通顺。

最后可以分享一下心态调整方面，很多人觉得裸辞找工作会焦虑会不安，我这次社招情况比较特殊，我下定决心要离开这里的时候就提前两个月跟领导打招呼了（得看领导人好不好不可随意效仿），当时我就抱着两个月走不掉就裸辞的心态出来面试的，因为确实抛开工作内容之外，面试有很多东西要准备，如果不提前打招呼平时加班啊就没有时间和精力去准备面试。

反正我认真复习认真总结，认真对待每一场面试，把每一场面试都当成一次学习的机会，你要换个思路不要把面试当成一场非赢即输的考试，而是把它当成一次学习交流的机会，面试终究是双向选择的，不成功即成仁。

其次，要认可自己，足够的爱自己，不要否定自己，失败了就再来，有什么大不了呢？**人生漫漫长路，没有人是赢家。快乐至上，珍惜当下，冲就完事了。**