【初/中级前端面经】中小型公司面试时都会问些什么?

现在百度，阿里，腾讯，字节跳动等大厂的面试真题、面经随处可见。甚至还有多家教育机构专门针对这一部分设计了课程，但是中小厂的面经倒是很稀缺。

但其实中小厂面经的需求还是有的，很多的开发者可能由于地理，能力，兴趣，志向的不同，短期或长期并没有进入大厂的打算，即便可以通过网上的各种渠道获取到所谓的“题海”。然后疯狂的刷面试题，进行题海战术，但还是对面试感到迷茫。

我本人正是由于这样那样的原因（当然还是实力不行），短期内并没有进入大厂的打算，再加上最近准备跳槽，所以亲身在大连选择了近十家中小企业进行了面试，采用录音的形式将面试题记录下来，并整理出本篇坐标大连的中小厂面经。

为了避免产生不必要的麻烦，公司名采用 `A`， `B`， `C` 进行处理。

本文的主旨是 **面经**，而非 **面试题** 的整合，文中会对部分面试题进行讲解，也会放出很多我认为很优秀的文章链接，也会有部分我在面试中的经历和我当时是怎么回答这个问题的，只是给大家提供个 **思路**，绝非 **标准答案**。

A公司

整体总结

`A` 公司上来会问几道基础题，问题之间没有什么逻辑。然后就会根据简历的内容和回答的情况进行追问，我的简历基本都是 `vue` 技术栈的，由于 `A` 公司是朋友内推的，可能是他们公司不用 `vue`，所以不关心，也就没有问到 `vue` 技术栈的任何问题，也没有问到有关项目经验的问题。

面试题一览

- 谈一谈 `css` 盒模型
- 多种方式实现上面 `100px` 下面自适应的布局

- `display` 都有哪些属性
- 块元素和行内元素、行内块元素的区别
- `js` 原型和原型链
- `Person.prototype.constructor` 是什么
- 函数有没有 `__proto__` 属性
- 谈一谈 `js` 数据类型
- 如何判断数据类型的多种方式，有什么区别，适用场景
- `Promise` 如何一次进行多个异步请求
- `Promise.all` 的返回机制是什么
- 如果想要其中一个请求出错了但是不返回结果怎么办
- `webpack` 打包优化知道多少
- 大前端了解吗
- `koa` 如何启动一个服务器
- `new koa` 都做了什么
- `koa` 洋葱圈模型原理
- `koa` 洋葱圈和 `express` 中间件有什么区别
- 长列表优化，一万条数据不用分页和懒加载，如何提升性能
- 数据请求从发起到接收数据之间发生了什么
- 前端安全了解吗
- `csrf` 和 `xss` 是什么，如何避免
- 前端怎样对用户的数据进行加密传输
- 基于 `md5` 提升安全性

谈一谈 `css` 盒模型

👉👉 【面试题解】CSS盒子模型与margin负值

多种方式实现上面 `100px` 下面自适应的布局

- `flex` 布局
- `gird` 布局
- `margin-top + calc`
- `定位 + calc`

display 都有哪些属性

值	描述
none	此元素不会被显示。
block	此元素将显示为块级元素，此元素前后会带有换行符。
inline	默认。此元素会被显示为内联元素，元素前后没有换行符。
inline-block	行内块元素。
table	此元素会作为块级表格来显示，表格前后带有换行符。
inherit	规定应该从父元素继承 display 属性的值。
flex	弹性盒模型。
grid	网格布局。

块元素和行内元素、行内块元素的区别

一.块级元素

（1）常见的块元素有哪些?

常见的块元素有 `<h1>~<h6>`、`<p>`、`<div>`、`<ul>`、`<ol>`、`<li>` 等，其中 `<div>` 标签是最典型的块元素。

（2）块级元素有什么特点?

- 自己独占一行
- 高度，宽度、外边距以及内边距都可以控制。
- 宽度默认是容器（父级宽度）的 `100%`
- 是一个容器及盒子，里面可以放行内或者块级元素

（3）注意问题：

只有文字才能组成段落，因此 `p` 标签里面不能放块级元素，特别是 `p` 标签不能放 `div` 。同理还有这些标签 `h1,h2,h3,h4,h5,h6,dt` ，他们都是文字类块级标签，里面不能放其他块级元素。

二.行内元素

（1）常见行内元素有哪些?

常见的行内元素有 `<a>、<strong>、<b>、<em>、<i>、<del>、<s>、<ins>、<u>、<span>` 等，其中 `<span>` 标签最典型的行内元素，也称内联元素。

（2）行内元素有哪些特点?

- 相邻行内元素在一行上，一行可以显示多个
- 高、宽直接设置是无效的
- 只可以设置水平方向的外边距
- 默认宽度就是它本身内容的宽度
- 行内元素只能容纳文本或则其他行内元素

（3）注意问题：

链接里面不能再放链接。

特殊情况 `a` 里面可以放块级元素，但是给 `a` 转换一下块级模式最安全。

三.行内块元素

（1）常见行内块儿元素有哪些?

在行内元素中有几个特殊的标签 `<img />、<input />、<td>` ，可以对它们设置宽高和对齐属性，有些资料可能会称它们为行内块元素。

（2）行内块元素有什么特点?

- 和相邻行内元素（行内块）在一行上,但是之间会有空白缝隙，一行可以显示多个。
- 默认宽度就是它本身内容的宽度。
- 高度，行高、外边距以及内边距都可以控制。

四. 块级元素、行内元素和行内块元素的区别

元素模式	元素排列	设置样式	默认宽度	包含
块级元素	一行只能放一个块级元素	可以设置宽度高度	容器的100%	容器级可以包含任何标签
行内元素	一行可以放多个行内元素	不可以直接设置宽度高度	它本身内容的宽度	容纳文本或则其他行内元素
行内块元素	一行放多个行内块元素	可以设置宽度和高度	它本身内容的宽度	

五. 块级元素、行内元素和行内块元素互转

- 块转行内： `display:inline`；
- 行内转块： `display:block`；
- 块、行内元素转换为行内块： `display: inline-block`；

js 原型和原型链

👉👉 深入JavaScript系列（六）：原型与原型链

Person.prototype.constructor 是什么

```js
Person.prototype.constructor === Person // true
```

函数有没有 __proto__ 属性

```js
let fn = function() {}
// 函数（包括原生构造函数）的原型对象为Function.prototype
fn.__proto__ === Function.prototype // true
```

函数都是由 `Function` 原生构造函数创建的，所以函数的 `__proto__` 属性指向 `Function` 的 `prototype` 属性。

谈一谈 js 数据类型

👉👉 【面试题解】JavaScript数据类型相关的六个面试题

如何判断数据类型的多种方式，有什么区别，适用场景

👉👉 【面试题解】谈一谈JavaScript数据类型判断

Promise 如何一次进行多个异步请求

答：利用 `Promise.all` 。

Promise.all 的返回机制是什么

除了 Promise.all ，还有其他几个原型方法也要知道。

👉👉 看了就会，手写Promise原理，最通俗易懂的版本！！！

如果想要其中一个请求出错了但是不返回结果怎么办

答：使用 Promise.allSettled 。

webpack 打包优化知道多少

👉👉 玩转 webpack，使你的打包速度提升 90%

koa 如何启动一个服务器

```js
const Koa = require('koa');
const app = new Koa();

app.use(async ctx => {
  ctx.body = 'Hello World';
});

app.listen(3000);
```

new koa 都做了什么

- 构建上下文 ctx
- 构建洋葱圈模型。

koa 洋葱圈模型原理

👉👉 浅析koa的洋葱模型实现

koa 洋葱圈和 express 中间件有什么区别

👉👉 Koa2 和 Express 中间件对比

长列表优化，一万条数据不用分页和懒加载，如何提升性能

👉👉 后端一次给你10万条数据，如何优雅展示，到底考察我什么?

数据请求从发起到接收数据之间发生了什么

这个我不会，也没有找到什么有效的资料，希望有大佬可以指点一下。

`csrf` 和 `xss` 是什么，如何避免

👉👉 浅说 XSS 和 CSRF

前端怎样对用户的数据进行加密传输

答： `md5` ，我其实不太了解，只是用 `md5` 做过登录注册的密码加密，也不会别的了。

基于 `md5` 提升安全性

答： `md5（md5）` ，我开玩笑的，有没有大佬可以指点一下。

B公司

整体总结

`B` 公司上来就是个咔咔咔一顿问，没有什么逻辑，问完一个就换下一个，像一个没有感情的提问机器，没有问到项目经验相关的问题，刷过面试题的基本都能应对自如，整体来说非常简单，这种基本上就是招 `coder` ，能干活就行，薪资不会开到太高。

面试题一览

- let,const,var 有什么区别
- 遍历数组的 n 种方法
- 说一下 vue 有哪些优点和特点
- 什么是虚拟 dom
- vue 组件间传值的 n 种方式
- vue 有哪些内置指令
- v-show 和 v-if 有什么区别
- 如何让 CSS 只在当前组件中起作用
- 如何解决 vue 初始化页面闪动问题
- 什么是 SPA，有什么优点和缺点
- vue 首屏渲染优化有哪些
- vue 生命周期函数有哪些
- 第一次页面加载会触发哪几个钩子
- 在哪个生命周期中发起数据请求
- vue-router 有几种模式
- hash 和 history 有什么区别
- vuex 有哪些属性
- git 常用命令了解哪些
- 搭一个新项目的框架,需要考虑哪些问题
- 如何做权限认证
- 如何做 mock 数据

let,const,var 有什么区别

（1）块级作用域： 块作用域由 { } 包括， let 和 const 具有块级作用域， var 不存在块级作用域。块级作用域解决了 ES5 中的两个问题：

- 内层变量可能覆盖外层变量
- 用来计数的循环变量泄露为全局变量

（2）变量提升： var 存在变量提升， let 和 const 不存在变量提升，即在变量只能在声明之后使用，否在会报错。

（3）给全局添加属性： 浏览器的全局对象是 window， Node 的全局对象是 global 。 var 声明的变量为全局变量，并且会将该变量添加为全局对象的属性，但是 let 和 const 不会。

（4）重复声明： `var` 声明变量时，可以重复声明变量，后声明的同名变量会覆盖之前声明的遍历。 `const` 和 `let` 不允许重复声明变量。

（5）暂时性死区： 在使用 `let` 、 `const` 命令声明变量之前，该变量都是不可用的。这在语法上，称为**暂时性死区**。使用 `var` 声明的变量不存在暂时性死区。

（6）初始值设置： 在变量声明时， `var` 和 `let` 可以不用设置初始值。而 `const` 声明变量必须设置初始值。

（7）指针指向： `let` 和 `const` 都是 `ES6` 新增的用于创建变量的语法。 `let` 创建的变量是可以更改指针指向（可以重新赋值）。但 `const` 声明的变量是不允许改变指针的指向。

区别	var	let	const
是否有块级作用域	×	✔	✔
是否存在变量提升	✔	×	×
是否添加全局属性	✔	×	×
能否重复声明变量	✔	×	×
是否存在暂时性死区	×	✔	✔
是否必须设置初始值	×	×	✔
能否改变指针指向	✔	✔	×

遍历数组的 n 种方法

👉👉 【面试题解】你了解JavaScript常用的的十个高阶函数么?

说一下 vue 有哪些优点和特点

- **渐进式框架**：可以在任何项目中轻易的引入；
- **轻量级框架**：只关注视图层，是一个构建数据的视图集合，大小只有几十 `kb` ；
- **简单易学**：国人开发，中文文档，不存在语言障碍 ，易于理解和学习；
- **双向数据绑定**：在数据操作方面更为简单；
- **组件化**：很大程度上实现了逻辑的封装和重用，在构建单页面应用方面有着独特的优势；

- **视图，数据，结构分离**：使数据的更改更为简单，不需要进行逻辑代码的修改，只需要操作数据就能完成相关操作；

什么是虚拟 dom

Virtual DOM 是 DOM 节点在 JavaScript 中的一种抽象数据结构，之所以需要虚拟 DOM，是因为浏览器中操作 DOM 的代价比较昂贵，频繁操作 DOM 会产生性能问题。

虚拟 DOM 的作用是在每一次响应式数据发生变化引起页面重渲染时， Vue 对比更新前后的虚拟 DOM，匹配找出尽可能少的需要更新的真实 DOM，从而达到提升性能的目的。

虚拟 DOM 的实现原理主要包括以下 **3** 部分：

- 用 JavaScript 对象模拟真实 DOM 树，对真实 DOM 进行抽象；
- diff 算法 — 比较两棵虚拟 DOM 树的差异；
- pach 算法 — 将两个虚拟 DOM 对象的差异应用到真正的 DOM 树。

vue 组件间传值的 n 种方式

(1) props / $emit 适用 父子组件通信

(2) ref 适用 父子组件通信

- ref：如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件实例

(3) $parent / $children / $root：访问父 / 子实例 / 根实例

(4) EventBus （$emit / $on） 适用于 父子、隔代、兄弟组件通信

这种方法通过一个空的 Vue 实例作为中央事件总线（事件中心），用它来触发事件和监听事件，从而实现任何组件间的通信，包括父子、隔代、兄弟组件。

(5) $attrs / $listeners 适用于 隔代组件通信

- $attrs：包含了父作用域中不被 prop 所识别 (且获取) 的特性绑定（ class 和 style 除外）。当一个组件没有声明任何 prop 时，这里会包含所有父作用域的绑定（ class 和 style 除外），并且可以通过 v-bind="$attrs" 传入内部组件。通常配合 inheritAttrs 选项一起使用。

- `$listeners`：包含了父作用域中的 (不含 `.native` 修饰器的) `v-on` 事件监听器。它可以通过 `v-on="$listeners"` 传入内部组件

(6) `provide / inject` 适用于 隔代组件通信

祖先组件中通过 `provide` 来提供变量，然后在子孙组件中通过 `inject` 来注入变量。 `provide / inject API` 主要解决了跨级组件间的通信问题，不过它的使用场景，主要是子组件获取上级组件的状态，跨级组件间建立了一种主动提供与依赖注入的关系。

(7) `Vuex` 适用于 父子、隔代、兄弟组件通信

`Vuex` 是一个专为 `Vue.js` 应用程序开发的状态管理模式。每一个 `Vuex` 应用的核心就是 `store`（仓库）。 `store` 基本上就是一个容器，它包含着你的应用中大部分的状态（ `state` ）。

- `Vuex` 的状态存储是响应式的。当 `Vue` 组件从 `store` 中读取状态的时候，若 `store` 中的状态发生变化，那么相应的组件也会相应地得到高效更新。
- 改变 `store` 中的状态的唯一途径就是显式地提交 `(commit) mutation` 。这样使得我们可以方便地跟踪每一个状态的变化。

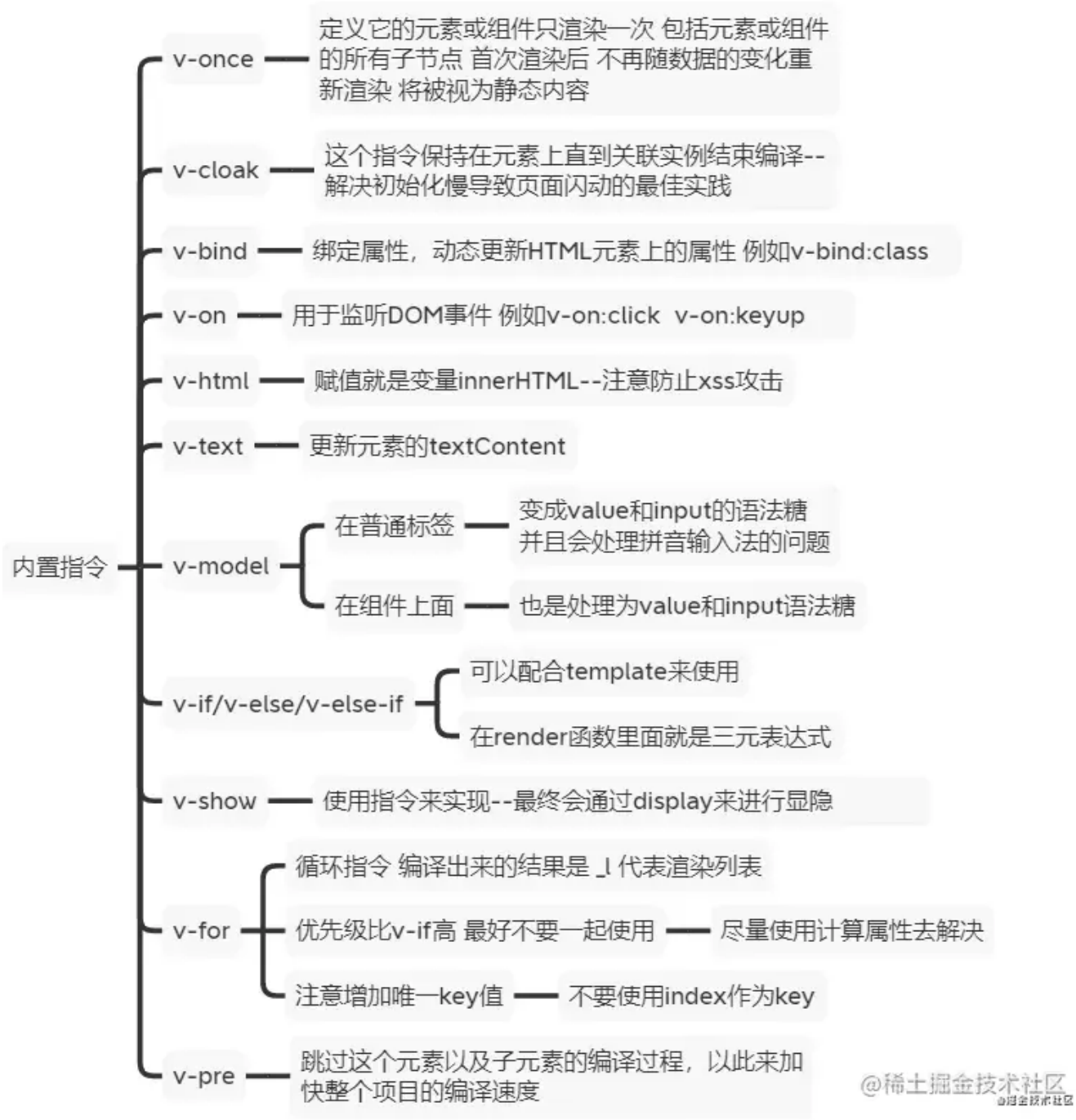
(8) 插槽

`Vue3` 可以通过 `usesolt` 获取插槽数据。

(9) `mitt.js` 适用于任意组件通信

`Vue3` 中移除了 `$on` ， `$off` 等方法，所以 `EventBus` 不再使用，相应的替换方案就是 `mitt.js`

vue 有哪些内置指令



v-show 和 v-if 有什么区别

- **手段**： v-if 是动态的向 DOM 树内添加或者删除 DOM 元素； v-show 是通过设置 DOM 元素的 display 样式属性控制显隐；
- **编译过程**： v-if 切换有一个局部编译/卸载的过程，切换过程中合适地销毁和重建内部的事件监听和子组件； v-show 只是简单的基于 css 切换；
- **编译条件**： v-if 是惰性的，如果初始条件为假，则什么也不做；只有在条件第一次变为真时才开始局部编译；v-show 是在任何条件下，无论首次条件是否为真，都被编译，然后被缓存，而且 DOM 元素保留；

- **性能消耗**： v-if 有更高的切换消耗； v-show 有更高的初始渲染消耗；
- **使用场景**： v-if 适合运营条件不大可能改变； v-show 适合频繁切换。

如何让 CSS 只在当前组件中起作用

在组件中的 style 标签中加上 scoped

如何解决 vue 初始化页面闪动问题

使用 vue 开发时，在 vue 初始化之前，由于 div 是不归 vue 管的，所以我们写的代码在还没有解析的情况下会容易出现花屏现象，看到类似于 {{message}} 的字样，虽然一般情况下这个时间很短暂，但是我们还是有必要让解决这个问题的。

首先：在 css 里加上 [v-cloak] { display: none; } 。如果没有彻底解决问题，则在根元素加上 style="display: none;" :style="{display: block }"

什么是 SPA，有什么优点和缺点

SPA 仅在 Web 页面初始化时加载相应的 HTML 、 JavaScript 和 CSS 。一旦页面加载完成， SPA 不会因为用户的操作而进行页面的重新加载或跳转；取而代之的是利用路由机制实现 HTML 内容的变换， UI 与用户的交互，避免页面的重新加载。

优点：

- 用户体验好、快，内容的改变不需要重新加载整个页面，避免了不必要的跳转和重复渲染；
- 有利于前后端职责分离，架构清晰，前端进行交互逻辑，后端负责数据处理；

缺点：

- 初次加载耗时多：为实现单页 Web 应用功能及显示效果，需要在加载页面的时候将 JavaScript 、 CSS 统一加载，部分页面按需加载；
- 不利于 SEO ：由于所有的内容都在一个页面中动态替换显示，所以在 SEO 上其有着天然的弱势。

vue 首屏渲染优化有哪些

- 图片压缩/懒加载
- 禁止生成 `.map` 文件
- 路由懒加载
- `cdn` 引入公共库
- 开启 `GZIP` 压缩

vue 生命周期函数有哪些

`Vue` 的生命周期钩子核心实现是利用发布订阅模式先把用户传入的的生命周期钩子订阅好（内部采用数组的方式存储）然后在创建组件实例的过程中会一次执行对应的钩子方法（发布）。

- `beforeCreate`：是 `new Vue()` 之后触发的第一个钩子，在当前阶段 `data`、`methods`、`computed` 以及 `watch` 上的数据和方法都不能被访问。
- `created`：在实例创建完成后发生，当前阶段已经完成了数据观测，也就是可以使用数据，更改数据，在这里更改数据不会触发 `updated` 函数。可以做一些初始数据的获取，在当前阶段无法与 `Dom` 进行交互，如果非要想，可以通过 `vm.$nextTick` 来访问 `Dom`。
- `beforeMount`：发生在挂载之前，在这之前 `template` 模板已导入渲染函数编译。而当前阶段虚拟 `Dom` 已经创建完成，即将开始渲染。在此时也可以对数据进行更改，不会触发 `updated`。
- `mounted`：在挂载完成后发生，在当前阶段，真实的 `Dom` 挂载完毕，数据完成双向绑定，可以访问到 `Dom` 节点，使用 `$refs` 属性对 `Dom` 进行操作。
- `beforeUpdate`：发生在更新之前，也就是响应式数据发生更新，虚拟 `dom` 重新渲染之前被触发，你可以在当前阶段进行更改数据，不会造成重渲染。
- `updated`：发生在更新完成之后，当前阶段组件 `Dom` 已完成更新。要注意的是避免在此期间更改数据，因为这可能会导致无限循环的更新。
- `beforeDestroy`：发生在实例销毁之前，在当前阶段实例完全可以被使用，我们可以在这时进行善后收尾工作，比如清除计时器。
- `destroyed`：发生在实例销毁之后，这个时候只剩下了 `dom` 空壳。组件已被拆解，数据绑定被卸除，监听被移出，子实例也统统被销毁。

第一次页面加载会触发哪几个钩子

`beforeCreate`， `created`， `beforeMount`， `mounted`

在哪个生命周期中发起数据请求

可以在钩子函数 created 、 beforeMount 、 mounted 中进行调用，因为在这三个钩子函数中， data 已经创建，可以将服务端端返回的数据进行赋值。

推荐在 created 钩子函数中调用异步请求，有以下优点：

- 能更快获取到服务端数据，减少页面 loading 时间；
- ssr 不支持 beforeMount 、 mounted 钩子函数，所以放在 created 中有助于一致性；

vue-router 有几种模式

vue-router 有 3 种路由模式： hash 、 history 、 abstract ：

- **hash**: 使用 URL hash 值来作路由。支持所有浏览器，包括不支持 HTML5 History Api 的浏览器；
- **history** : 依赖 HTML5 History API 和服务器配置。
- **abstract** : 支持所有 JavaScript 运行环境，如 Node.js 服务器端。如果发现没有浏览器的 API ，路由会自动强制进入这个模式.

hash 和 history 有什么区别

(1) hash 模式的实现原理

早期的前端路由的实现就是基于 location.hash 来实现的。其实现原理很简单， location.hash 的值就是 URL 中 # 后面的内容。比如下面这个网站，它的 location.hash 的值为 #search ：

```
https://www.word.com#search
```

hash 路由模式的实现主要是基于下面几个特性：

- URL 中 hash 值只是客户端的一种状态，也就是说当向服务器端发出请求时， hash 部分不会被发送；
- hash 值的改变，都会在浏览器的访问历史中增加一个记录。因此我们能通过浏览器的回退、前进按钮控制 hash 的切换；
- 可以通过 a 标签，并设置 href 属性，当用户点击这个标签后， URL 的 hash 值会发生改变；或者使用 JavaScript 来对 loaction.hash 进行赋值，改变 URL 的 hash 值；
- 我们可以使用 hashchange 事件来监听 hash 值的变化，从而对页面进行跳转（渲染）。

(2) history 模式的实现原理

HTML5 提供了 History API 来实现 URL 的变化。其中做最主要的 API 有以下两个：history.pushState() 和 history.repalceState()。这两个 API 可以在不进行刷新的情况下，操作浏览器的历史纪录。唯一不同的是，前者是新增一个历史记录，后者是直接替换当前的历史记录，如下所示：

```js
window.history.pushState(null, null, path);
window.history.replaceState(null, null, path);
```

history 路由模式的实现主要基于存在下面几个特性：

- pushState 和 repalceState 两个 API 来操作实现 URL 的变化；
- 我们可以使用 popstate 事件来监听 url 的变化，从而对页面进行跳转（渲染）；
- history.pushState() 或 history.replaceState() 不会触发 popstate 事件，这时我们需要手动触发页面跳转（渲染）。

vuex 有哪些属性

有五种，分别

- **State**：定义了应用状态的数据结构，可以在这里设置默认的初始状态。
- **Getter**：允许组件从 Store 中获取数据， mapGetters 辅助函数仅仅是将 store 中的 getter 映射到局部计算属性。
- **Mutation**：是唯一更改 store 中状态的方法，且必须是同步函数。
- **Action**：用于提交 mutation，而不是直接变更状态，可以包含任意异步操作。
- **Module**：允许将单一的 Store 拆分为多个 store 且同时保存在单一的状态树中。

git 常用命令了解哪些

👉👉 Git命令速查表

搭一个新项目的框架,需要考虑哪些问题

👉👉 结合代码实践，全面学习前端工程化

如何做权限认证

答：在路由守卫中根据 `url` 地址结合 `token` 做权限认证。

如何做 `mock` 数据

答：可以使用 `mock.js`

C公司

整体总结

`C` 公司首先要在线上申请面试，然后会填一个很长很长的 **报名表**，填完以后又会填一个很长很长的 **问卷**，问卷的内容就好像是公众号心理测评一样，跟技术没有丁点关系。然后就让我去公司面试了，面试形式是 `上机` ，题目已经整理在下面了，整体来说不算难，手写表格比较恶心一点。然后笔试完了，又给了我一套小学 **奥数题**，不能用方程式解，这个属实有点坑，这个过了应该还有个面试环节，但是我在笔试环节被毙掉了，也不知道后面是什么情况。

面试题一览

- `ajax` 是什么?有什么优缺点
- 同步和异步的区别
- 如何解决跨域问题
- `WEB` 应用从服务器主动推送 `Data` 到客户端有哪些方式
- 经常遇到的浏览器的兼容性问题有哪些?原因是什么?如何解决
- 有哪些常用的 `hack` 技巧
- 前端开发性能优化,你有什么经验
- 谈谈你对 `webpack` 的看法
- 主流的前端框架的优缺点是什么
- 你最喜欢哪个框架，为什么
- 如何消除一个数组里面重复的元素
- `css` 如何实现一个幻灯片效果
- 手写表格

ajax 是什么?有什么优缺点

ajax 是一种创建交互网页应用的一门技术。

优点：

- 实现局部更新(无刷新状态下)，
- 减轻了服务器端的压力

缺点：

- 破坏了浏览器前进和后退机制(因为 ajax 自动更新机制)
- ajax 请求多了，也会出现页面加载慢的情况。
- 搜索引擎的支持程度比较低。
- ajax 的安全性问题不太好(可以用数据加密解决)。

同步和异步的区别

同步： 同步的思想是：所有的操作都做完，才返回给用户。这样用户在线等待的时间太长，给用户一种卡死了的感觉（就是系统迁移中，点击了迁移，界面就不动了，但是程序还在执行，卡死了的感觉）。这种情况下，用户不能关闭界面，如果关闭了，即迁移程序就中断了。

异步： 将用户请求放入消息队列，并反馈给用户，系统迁移程序已经启动，你可以关闭浏览器了。然后程序再慢慢地去写入数据库去。这就是异步。但是用户没有卡死的感觉，会告诉你，你的请求系统已经响应了。你可以关闭界面了。

如何解决跨域问题

👉👉 九种跨域方式实现原理（完整版）

WEB 应用从服务器主动推送 Data 到客户端有哪些方式

- **轮询**
- **WebSocket**

经常遇到的浏览器的兼容性问题有哪些?原因是什么?如何解决

由于我的公司是一个致力于培养用户习惯的公司，遇到兼容问题都是**请下载最新版本谷歌浏览器**😝，所以我这个其实没有什么实际的经验，只知道 `babel` 和 `postcss` 。

有哪些常用的 `hack` 技巧

说实话，没听说过。

前端开发性能优化,你有什么经验

👉👉 前端性能优化 24 条建议（2020）

谈谈你对 `webpack` 的看法

👉👉 当面试官问Webpack的时候他想知道什么

主流的前端框架的优缺点是什么

只用过 `vue` ，优点上面讲过了，缺点嘛，可能就是不支持 `IE8` 。

你最喜欢哪个框架，为什么

同上一个问题。

如何消除一个数组里面重复的元素

- `set`
- `reduce`
- `for循环`
- 能实现方式有很多，原理都是对比两个数组，没有就放进去。

`css` 如何实现一个幻灯片效果

👉👉 使用CSS实现一个简单的幻灯片效果

D公司

整体总结

这个面试过程就很随和，面试官是个架构师，`level` 挺高的，不会问八股文，他给我出了几个现实中的场景，然后转换成代码的逻辑，去让我实现他，一共问了没几个问题，不需要背面试题，会由一个问题一直追着问我，然后还会不断的引导我，提示我，直到我啥也答不上来。

面试题一览

- 一般请求后端接口，你都怎么弄?
- 你的后端要给你什么样的信息，你才能请求成功呢?
- 请求参数有什么格式?
- 如何给后端传递一个文件?
- 你如何理解前端工程化?
- 要买个电脑，找 `A` 借 `1k`，找 `B` 借 `2K`，找 `C` 借 `3K`，拿着六千块钱买电脑，抽象成前端的逻辑就是请求不同的接口获取数据，拿到所有的数据之后进行展示，这个怎么实现?
- 不使用 `promise.all`，`async/await` 怎么实现?
- `promise.all` 和 `async/await` 有什么区别?
- `promise.all` 是为了解决什么问题?
- 有一批不定数量的人，第一个人去超市买一个东西，第一个人买回来以后第二个人再去买，第二个回来以后第三个再去买，抽象成前端的逻辑如何实现? （其实他想听的答案就是递归，结果我把洋葱圈原理讲了一遍）
- 如何删除事件监听，一个元素绑定了多个事件，你怎么确认删除的是哪个?
- 你都如何调试代码? （这个阶段就是打开谷歌控制台，一个个讲各种功能都能干什么事）
- 谷歌调试工具你都会用什么功能?
- 怎么进行断点调试?
- 控制台都能干什么事?
- 不熟悉的项目，如何找到接口所在的代码?
- 如果接口地址是动态的呢，是其他接口返回的?
- `localStorage，session，cookie` 的区别是什么?
- 然后问了我自己的几个开源项目

- 因为我带了电脑，所以还看了看我的代码

一般请求后端接口，你都怎么弄?

这个问题其实没有搞懂面试官想问什么，参照接口文档发起请求就行了呗，顶多就是再二次封装一个 `axios` 。

你的后端要给你什么样的信息，你才能请求成功呢?

- 请求方式
- 请求参数

请求参数有什么格式?

👉👉 HTTP 请求参数之三种格式

- `Query String Parameters`
- `Form Data`
- `Request Payload`

如何给后端传递一个文件?

👉👉 前后端文件上传过程以及方法

你如何理解前端工程化?

👉👉 结合代码实践，全面学习前端工程化

上面提到的买电脑这个例子怎么实现?

答：使用 `promise.all` 或者 `async/await` 。

不使用 `promise.all` , `async/await` 怎么实现?

不知道了😭😭。

promise.all 和 async/await 有什么区别?

答： Async Await 是基于 promise 实现，是改良版的 promise ，使代码看起来更加简洁，异步代码执行像同步代码一样。

promise.all 是为了解决什么问题?

答：汇总大量的异步操作结果。

如何删除事件监听，一个元素绑定了多个事件，你怎么确认删除的是哪个?

```js
element.removeEventListener(type, handler, false/true)
```

- **type**:事件类型
- **handler**:事件执行触发的函数
- **false/true**: false 为冒泡， true 为捕获，参数是 true ，表示在捕获阶段调用事件处理程序； 如果是false ，表示在冒泡阶段调用事件处理程序。

需要注意的是，通过匿名函数是无法消除监听事件，只有通过实名函数才能。

不熟悉的项目，如何找到接口所在的代码?

答：可以搜接口的地址。

localStorage，session，cookie 的区别是什么?

👉👉 理解cookie、session、localStorage、sessionStorage之不同

E公司

整体总结

这个完全没必要去，就当是避雷吧，首先他这个公司的楼非常破，公司内部非常破，像个传销窝点，然后就是他面试官不是个搞技术的，不知道是个什么负责人，问的问题也跟技术没啥关系，因为看到办公环境就不想继续了，所以没录音，也没问什么问题，很快就结束了。

E 公司其实算不上什么面经，只是提醒大家在找工作之前一定要先看一下目标公司的环境，规模等，从根源上避免像我一样踩雷。

F公司

整体总结

F 公司两个面试官轮流提问，基本上都不会问八股文的问题，项目经验问的比较多，只要是真实的工作经验，基本都能应对，然后问了我自己的开源项目一些问题，这个是根据简历来的，如果没写是不会问的。第一个面试官感觉是一直在自言自语，我一直在嗯嗯嗯嗯嗯，第二个面试官会在问项目的时候结合我的回答情况，追问一些基础知识，涉及到的都整理在下面了。

面试题一览

面试官一：

- 我看你掘金了，简单描述一下 `call` ， `apply` ， `bind` 有什么区别和应用场景。（说的好像是我不写他就不问了）
- 说一说盒子模型？
- 公司项目负责哪部分功能？
- 表单和表格封装过么？
- 表单是怎么封装的？
- 大数据的表单怎么处理， `select` 选项过多的时候？（优化问题）
- 学完 `vue3` 有什么想法么？ `nuxt` 什么感想？
- 自己做过脚手架么？
- 然后就是聊项目……

面试官二：

- 在公司都做哪些事？项目是干什么的？
- 然后就是聊项目……
- `vue2` 和 `vue3` 有什么区别？
- 有没有遇到性能优化问题？
- 从输入 `url` 到页面渲染完成之间发生了什么？
- 浏览器原理了解过么？
- `http` 状态码都有哪些？
- 前端如何处理这些状态码？
- `localStorage` ， `session` ， `cookie` 的区别是什么？
- 前端安全问题， `CSRF` ， `XSS`
- 如何解决跨域问题？
- 跨域问题实际上改的是 `http` 里面哪个参数？

`call` ， `apply` ， `bind` 有什么区别和应用场景

👉👉 【面试题解】你了解call，apply，bind吗？那你可以手写一个吗？

说一说盒子模型

👉👉 【面试题解】CSS盒子模型与margin负值

学完 `vue3` 有什么想法么？ `nuxt` 什么感想

答： `vue3` 在代码编写层面，可以更好的完成结构和逻辑的复用。 `nuxt` 是一个约定大于配置的框架，很多东西人家都给封装好了，按照约定去写就可以了，有利于团队协作。

`vue2` 和 `vue3` 有什么区别

- 响应式原理
- 生命周期钩子名称
- 自定义指令钩子名称
- 新的内置组件
- `diff` 算法
- `Composition API`

有没有遇到性能优化问题

👉👉 前端性能优化 24 条建议（2020）

从输入 `url` 到页面渲染完成之间发生了什么

👉👉 史上最详细的经典面试题 从输入URL到看到页面发生了什么?

浏览器原理了解过么

👉👉 深入理解浏览器工作原理

`http` 状态码都有哪些

状态码第一位数字决定了不同的响应状态，如下：

- 1 表示消息
- 2 表示成功
- 3 表示重定向
- 4 表示请求错误
- 5 表示服务器错误

1xx

代表请求已被接受，需要继续处理，这类响应是临时响应，只包含状态行和某些可选的响应信息，并一空行结束

常见的有：

- `100` （客户继续发送请求，这是临时响应） 这个临时响应是用来通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端印当据需发送请求的剩余部分，或者如果请求已经完成，忽略这个响应，服务器必须在请求完成后向客户端发送一个最终响应
- `101` 服务器根据客户端的请求切换协议，主要用于 `websocket` 或 `HTTP2` 升级

2xx

代表请求已成功被服务器接收，处理，并接受

- `200` （成功） 请求已成功，请求所希望的响应头或数据体将随此响应返回

- 201 （已创建）请求成功并且服务器创建了新的资源
- 202 （已创建）服务器已经接受请求，但尚未处理
- 203 （非授权信息）服务器已成功处理请求，但返回的信息可能来自另一来源
- 204 （无内容）服务器成功处理请求，但没有返回任何内容
- 205 （重置内容）服务器成功处理请求，但没有返回任何内容
- 206 （部分内容）服务器成功处理了部分请求

3xx

表示要完成请求，需要进一步操作，通常这些状态代码用来重定向

- 300 （多种选择）针对请求，服务器可执行多种操作。
- 301 （永久移动）请求的网页已永久移动到新位置。
- 302 （临时移动）服务器目前从不同位置的网页响应请求，但请求者应该继续使用原有位置来进行以后的请求
- 303 （查看其它位置）请求者应当对不同位置使用单独的 GET 请求来检索响应时，服务器返回此代码
- 305 （使用代理）请求者只能使用代理访问请求的网页。
- 307 （临时重定向）服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求

4xx

代表了客户端看起来可能发生了错误，妨碍了服务器的处理

- 400 （错误请求）服务器不理解请求的语法
- 401 （未授权）请求要求身份验证。
- 403 （禁止）服务器拒绝请求
- 404 （未找到）服务器找不到请求的网页
- 405 （方法禁用）禁用请求中指定的方法
- 406 （不接受）无法使用请求的内容特性响应请求的网页
- 407 （需要代理授权）此状态代码与 401 （未授权）类似，但指定请求者应当授权使用代理
- 408 （请求超时）服务器等候请求时发生超时

5xx

表示服务器无法完成明显有效的请求。这类状态代码代表了服务器在处理请求的过程中有错误或异常状态发生

- 500 （服务器内部错误）服务器遇到错误，无法完成请求
- 501 （尚未实施）服务器服务器不具备完成请求的功能
- 502 （错误网关）服务器作为网关或代理，从上游服务器收到无效响应
- 503 （服务不可用）服务器目前无法使用，（由于超载或停机维护）
- 504 （网关超时）服务器作为网关或代理，但是没有及时从上游服务器收到请求
- 505 （ HTTP 版本不受支持）服务器不支持请求中所用的 HTTP 协议版本

前端如何处理这些状态码

答：在 axios 的请求拦截当中根据不同的状态码进行不同的操作。

localStorage， session， cookie 的区别是什么

👉👉 理解cookie、session、localStorage、sessionStorage之不同

前端安全问题， CSRF， XSS

👉👉 浅说 XSS 和 CSRF

如何解决跨域问题

👉👉 九种跨域方式实现原理（完整版）

跨域问题实际上改的是 http 里面哪个参数

答： Access-Control-Allow-Origin ？ 这个不确定，有大佬可以指点一下。

G公司

整体总结

G 公司一上来给我展示了一个设计图，让我现场进行布局，很简单，就是一个 header 和一个 sider，还有一个内容区，内容区一个两栏布局，挺简单的。然后就是根据我的简历上写的项目，问几个公司的，问几个我自己开源的，期间会穿插着问一些相关知识点，涉及到的已经整理下面了。

面试题一览

- 给看了设计图，让现场布局
- localStorage， session， cookie 的区别是什么
- vuex 的 mutation 和 action
- vuex 模块化
- 数组深拷贝
- css 盒子模型
- css 实现斑马线的效果
- 跨域问题

localStorage， session， cookie 的区别是什么

👉👉 理解cookie、session、localStorage、sessionStorage之不同

vuex 的 mutation 和 action

👉👉 【初学者笔记】一文学会使用Vuex

vuex 模块化

👉👉 【初学者笔记】一文学会使用Vuex

数组深拷贝

👉👉 【面试题解】JavaScript的深浅拷贝，如何手写深拷贝？

css 盒子模型

👉👉 【面试题解】CSS盒子模型与margin负值

css 实现斑马线的效果

答：可以通过伪元素，父级元素使用绝对定位，伪元素使用相对定位，大小和父元素一样，位置重合。 再利用 nth-child 选择器选择奇数行，只给奇数行设置伪元素即可实现。

跨域问题

👉👉 九种跨域方式实现原理（完整版）

H公司

整体总结

H 公司好像并不是很诚心招人的样子，面试官象征性的问了几个问题就让我回家等消息了，问的都很基础，参考意义不大，等真想招人的时候肯定不是这个样子的。

咱也不清楚是不是年终了在刷绩效，还是说在我去之前已经招够人了，但我已经约面试了也不好意思毁约，有没有类似经历的?

面试题一览

- es6 有哪些新特性
- promise 都有哪些方法
- 遍历数组的 n 种方法
- vue 生命周期
- watch 和 computed 区别和使用场景
- vue3 和 vue2 的区别
- 虚拟 dom 和真实 dom 的区别
- 组件传值的 n 种方式
- ts 和 js 的优缺点

es6 有哪些新特性

👉👉 阮一峰ES6 入门教程

promise 都有哪些方法

👉👉 看了就会，手写Promise原理，最通俗易懂的版本！！！

遍历数组的 n 种方法

👉👉 【面试题解】你了解JavaScript常用的的十个高阶函数么?

vue 生命周期

B 公司的面试题中解答过了。

watch 和 computed 区别和使用场景

对于Computed:

- 它支持缓存，只有依赖的数据发生了变化，才会重新计算
- 不支持异步，当 Computed 中有异步操作时，无法监听数据的变化
- 如果一个属性是由其他属性计算而来的，这个属性依赖其他的属性，一般会使用 computed
- 如果 computed 属性的属性值是函数，那么默认使用 get 方法，函数的返回值就是属性的属性值；在 computed 中，属性有一个 get 方法和一个 set 方法，当数据发生变化时，会调用 set 方法。

对于Watch:

- 它不支持缓存，当一个属性发生变化时，它就会触发相应的操作
- 支持异步监听
- 监听的函数接收两个参数，第一个参数是最新的值，第二个是变化之前的值
- 监听数据必须是 data 中声明的或者父组件传递过来的 props 中的数据，当发生变化时，会触发其他操作
- 函数有两个的参数:
    - **immediate**：组件加载立即触发回调函数

- **deep**：深度监听，发现数据内部的变化，在复杂数据类型中使用，例如数组中的对象发生变化。

vue3 和 vue2 的区别

F 公司的面试题中解答过了。

虚拟 dom 和真实 dom 的区别

- 虚拟 DOM 不会进行排版与重绘操作
- 虚拟 DOM 就是把真实 DOM 转换为 Javascript 代码
- 虚拟 DOM 进行频繁修改，然后一次性比较并修改真实 DOM 中需要改的部分，最后并在真实 DOM 中进行排版与重绘，减少过多 DOM 节点排版与重绘损耗

组件传值的 n 种方式

B 公司的面试题中解答过了。

ts 和 js 的优缺点

- ts 是 js 的超集，即你可以在 ts 中使用原生 js 语法。
- ts 需要静态编译，它提供了强类型与更多面向对象的内容。
- ts 最终仍要编译为弱类型，基于对象的原生的 js ，再运行。

I公司

整体总结

说明:这个挺中规中矩的吧，大多数也都是网上的面试题八股文，但是他会根据你的回答继续深入的追问，而不是无脑问问题的那种，没有问项目。但是他和上面的文华财经差不多，要先填一个非常长非常长的资料，再做一套跟代码无关的逻辑思维题，然后才能面试，比较麻烦。

面试题一览

- 盒模型
- 元素水平垂直居中的方法
- `flex` 和 `grid` 有什么区别
- `flex：1` 是什么意思
- 一个父容器，三个子容器，两边的子容器宽度固定，中间自适应，如何实现?
- 说一下闭包和函数柯里化
- 解释一下事件循环，微任务和宏任务都有哪些?
- 解释一下原型链
- 所有的对象都有原型吗?
- `vue` 的生命周期
- 跨域，解决跨域问题的方案
- 说一下 `sourcemap` 都有哪些配置，开发环境和生产环境如何选择?
- 浏览器从输入 `url` 到页面渲染之间做了哪些事情?

盒模型

👉👉 【面试题解】CSS盒子模型与margin负值

元素水平垂直居中的方法

👉👉 【面试题解】宽高固定的12种和宽高不固定的29种CSS居中方案。

`flex` 和 `grid` 有什么区别

👉👉 「一劳永逸」48张小图带你领略flex布局之美

👉👉 最强大的 CSS 布局 —— Grid 布局

`flex：1` 是什么意思

👉👉 「一劳永逸」48张小图带你领略flex布局之美

一个父容器，三个子容器，两边的子容器宽度固定，中间自适应，如何实现?

- `flex` 布局
- `grid` 布局
- `定位 + calc`

说一下闭包和函数柯里化

👉👉 【面试题解】初识 JavaScript 闭包

👉👉 「前端进阶」彻底弄懂函数柯里化

解释一下事件循环，微任务和宏任务都有哪些?

👉👉 做一些动图，学习一下EventLoop

解释一下原型链

👉👉 深入JavaScript系列（六）：原型与原型链

所有的对象都有原型吗?

答：不是的, 用 `Object.create(null)` 创建的对象没有原型。

`vue` 的生命周期

`B` 公司的面试题中解答过了。

跨域，解决跨域问题的方案

👉👉 九种跨域方式实现原理（完整版）

说一下 `sourcemap` 都有哪些配置，开发环境和生产环境如何选择?

👉👉 当面试官问Webpack的时候他想知道什么

浏览器从输入 `url` 到页面渲染之间做了哪些事情?

👉👉 史上最详细的经典面试题 从输入URL到看到页面发生了什么?

J公司

整体总结

J 公司有三个面试官，一个应该是 HR ，问一些技术无关的问题，两个问技术的，看样子 level 是很高的。

上来第一句话就是，我看你简历熟悉的不少啊。吓得我赶紧说了解而已。

然后就开始问问题，有基础的，大多数都是难得，也怪我简历上写了很多"熟悉"，几个问题以后我说，我对"熟悉" 有了新的理解。

这个因为面试官进来的比较突然，没来得及录音，再加上当时确实是被问懵住了，没记住几个问题，但实际不止下面这些个，而且大多都是源码，原理什么的。。。没有问项目

面试题一览

- 我看你熟悉 vue ，那你讲讲 vue 的模板编译原理吧
- vue 为什么要用 template 啊
- 讲一讲 vuex 的挂载过程
- 讲一讲 vue-router 的几种模式和守卫吧
- nuxt 怎样配置路由，如何自定义路由，自定义的和约定路由哪个优先级高
- promise 你都用过哪些方法
- express 和 koa 有什么区别
- ts 跟 js 有什么区别，优点和缺点

我看你熟悉 vue ，那你讲讲 vue 的模板编译原理吧

👉👉 Vue模板编译原理

vue 为什么要用 template 啊

答：我说的是书写起来更像原生的 `html` 。

讲一讲 `vuex` 的挂载过程

👉👉 vuex的store是如何挂载到每个组件中

讲一讲 `vue-router` 的几种模式和守卫吧

模式前面讲过了

👉👉 【面试题解】vue-router有几种钩子函数？具体是什么及执行流程是怎样的？

`nuxt` 怎样配置路由，如何自定义路由，自定义的和约定路由哪个优先级高

答：约定路由， `nuxt` 内部会根据文件路径自动生成路由，自定义路由不会了。

`promise` 你都用过哪些方法

👉👉 看了就会，手写Promise原理，最通俗易懂的版本！！！

`express` 和 `koa` 有什么区别

1. 语法区别

- `experss` 异步使用 回调
- `koa1` 异步使用 `generator + yeild`
- `koa2` 异步使用 `await/async`

2. 中间件区别

- `koa` 采用洋葱模型，进行顺序执行，出去反向执行，支持 `context` 传递数据
- `express` 本身无洋葱模型，需要引入插件，不支持 `context`

3. 集成度区别

- `express` 内置了很多中间件，集成度高，使用省心

- koa 轻量简洁，容易定制

ts 跟 js 有什么区别，优点和缺点

- ts 是 js 的超集，即你可以在 ts 中使用原生 js 语法。
- ts 需要静态编译，它提供了强类型与更多面向对象的内容。
- ts 最终仍要编译为弱类型，基于对象的原生的 js ，再运行。

高频问题

其实高频算不上，因为只面试了十家公司，样本容量太小了，但是这么几家都出现多次的题目，可见也是挺热们的，结合实际的面试情况，整理出相对高频的面试题，可以着重看一下，即便是看不懂，不感兴趣，也要强迫自己背一下。

- CSS 盒模型
- 一些常用的页面布局要了解
- es6 新特性，遍历数组的 n 种方法
- JS 原型和原型链
- ts 跟 js 有什么区别，特点，优点和缺点
- promise 的使用及其原型方法
- 浏览器从输入 url 到页面渲染之间做了哪些事情
- 什么是跨域问题，如何解决
- 前端安全问题
- 如果你写了自己会 node ，可能会问 express 和 koa 的相关问题
- webpack 优化

还有就是自己的项目一定要了解，虽然可能确实是自己做的，但是时间长了会忘，建议面试之前再把自己的项目好好看一下，大多数公司还是会占用一定时间去问一下你之前负责的项目的，这个答不上来大概率会让面试官觉得你项目经历是编的，或者不是自己写的，从而扣分。

必备问题

最终总结了几个面试官常问的非技术类的问题，可以提前准备一下，以免当场卡住，语无伦次导致给面试官留下不好的印象。

- 做个简单的自我介绍
- 为什么从上一家公司离职
- 你觉得上家公司做的怎么样
- 为什么要干前端
- 你平时是怎么学习的
- 看过什么书
- 你觉得你的优势/不足有什么
- 举一个例子，证明自己是乐于学习的
- 你对自己的职业规划
- 你还有什么想要问我的么

做个简单的自我介绍

主要介绍一下自己的工作内容，技术栈，稍微带点兴趣爱好什么的也可以。

为什么从上一家公司离职

我回答的主要两个原因，一个是想学习更多东西，另一个是老生常谈的薪资问题。

你觉得上家公司做的怎么样

先大概的夸一夸上家公司，然后再谈一谈让你不想待的原因，千万不要开吐槽大会，抱怨这个，抱怨那个的。比如说我觉得上家公司各种方面做的都不错，无论是公司环境，管理制度，团队氛围都很好，但是技术栈更新比较缓慢，调薪制度不理想。

为什么要干前端

觉得前端比较有意思，可以看到界面，可以做一下好玩的东西，给父母孩子女朋友显摆。

你平时是怎么学习的

看纸质书，电子书，视频，博客论坛。

看过什么书

看过什么就说什么，千万不要瞎编。我听说过有的面试官会问你某本书的封面是什么颜色的，目录是什么等奇葩问题。

你觉得你的优势/不足有什么

优势在于主动学习，乐于分享。不足是不感兴趣的内容学习不下去，比如数据库。

举一个例子，证明自己是乐于学习的

掘金社区优秀作者

公司的年度进步奖

你对自己的职业规划

四个字，逼格直接上来了，一专多精。

你还有什么想要问我的么

我一般会问技术团队的规模

- 技术团队有几个人，
- 几个前端，
- 几个后端，
- 高级，中级，初级的分别是多少人 然后就是技术栈
- 目前在使用什么技术栈
- 将来打算使用什么技术栈
- 自己是否可以决定未来技术栈的走向 然后是自己在团队中的角色
- 负责什么工作
- 对于项目的决策度

最后祝大家在新的一年里，都能找到满意的工作，升职加薪，赚的盆满钵满！