新鲜出炉的8月前端面试题

Belinda 脚本之家 2018-08-16

脚本之家

你与百万开发者在一起

关注

> 作者：Belinda
> https://segmentfault.com/a/1190000015916686

前言

最近参加了几场面试，积累了一些高频面试题，我把面试题分为两类，一种是基础试题: 主要考察前端技基础是否扎实，是否能够将前端知识体系串联。一种是开放式问题: 考察业务积累，是否有自己的思考，思考问题的方式，这类问题没有标准答案。

基础题

> 题目的答案提供了一个思考的方向，答案不一定正确全面，有错误的地方欢迎大家请在评论中指出，共同进步。

怎么去设计一个组件封装

1. 组件封装的目的是为了重用，提高开发效率和代码质量
2. 低耦合，单一职责，可复用性，可维护性
3. 前端组件化设计思路

js 异步加载的方式

1. 渲染引擎遇到 script 标签会停下来，等到执行完脚本，继续向下渲染
2. defer 是“渲染完再执行”，async 是“下载完就执行”，defer 如果有多个脚本，会按照在页面中出现的顺序加载，多个async 脚本不能保证加载顺序
3. 加载 es6模块的时候设置 type=module，异步加载不会造成阻塞浏览器，页面渲染完再执行，可以同时加上async属性，异步执行脚本（利用顶层的this等于undefined这个语法点，可以侦测当前代码是否在 ES6 模块之中）

css 动画和 js 动画的差异

1. 代码复杂度，js 动画代码相对复杂一些
2. 动画运行时，对动画的控制程度上，js 能够让动画，暂停，取消，终止，css动画不能添加事件
3. 动画性能看，js 动画多了一个js 解析的过程，性能不如 css 动画好

XSS 与 CSRF 两种跨站攻击

1. xss 跨站脚本攻击，主要是前端层面的，用户在输入层面插入攻击脚本，改变页面的显示，或则窃取网站 cookie，预防方法：不相信用户的所有操作，对用户输入进行一个转义，不允许 js 对 cookie 的读写
2. csrf 跨站请求伪造，以你的名义，发送恶意请求，通过 cookie 加参数等形式过滤
3. 我们没法彻底杜绝攻击，只能提高攻击门槛

事件委托，目的，功能，写法

1. 把一个或者一组元素的事件委托到它的父层或者更外层元素上
2. 优点，减少内存消耗，动态绑定事件
3. target 是触发事件的最具体的元素，currenttarget是绑定事件的元素(在函数中一般等于this)
4. JavaScript 事件委托详解

线程，进程

1. 线程是最小的执行单元，进程是最小的资源管理单元
2. 一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程

负载均衡

1. 当系统面临大量用户访问，负载过高的时候，通常会使用增加服务器数量来进行横向扩展，使用集群和负载均衡提高整个系统的处理能力
2. 服务器集群负载均衡原理?

什么是CDN缓存

1. CDN 是一种部署策略，根据不同的地区部署类似nginx 这种服务服务，会缓存静态资源。前端在项目优化的时候，习惯在讲台资源上加上一个 hash 值，每次更新的时候去改变这个

hash，hash 值变化的时候，服务会去重新取资源

2. (CDN)是一个经策略性部署的整体系统，包括分布式存储、负载均衡、网络请求的重定向和内容管理4个要件
3. CDN_百度百科

闭包的写法，闭包的作用，闭包的缺点

1. 使用闭包的目的——隐藏变量，间接访问一个变量,在定义函数的词法作用域外，调用函数
2. 闭包的内存泄露，是IE的一个 bug，闭包使用完成之后，收回不了闭包的引用，导致内存泄露
3. 「每日一题」JS 中的闭包是什么?
4. 闭包造成内存泄露的实验

跨域问题，谁限制的跨域，怎么解决

1. 浏览器的同源策略导致了跨域
2. 用于隔离潜在恶意文件的重要安全机制
3. [jsonp ， 允许 script 加载第三方资源]https://segmentfault.com/a/11...
4. nginx 反向代理（nginx 服务内部配置 Access-Control-Allow-Origin *）
5. cors 前后端协作设置请求头部，Access-Control-Allow-Origin 等头部信息
6. iframe 嵌套通讯，postmessage

javascript 中常见的内存泄露陷阱

1. 内存泄露会导致一系列问题，比如：运行缓慢，崩溃，高延迟
2. 内存泄露是指你用不到（访问不到）的变量，依然占居着内存空间，不能被再次利用起来
3. 意外的全局变量，这些都是不会被回收的变量（除非设置 null 或者被重新赋值），特别是那些用来临时存储大量信息的变量
4. 周期函数一直在运行，处理函数并不会被回收，jq 在移除节点前都会，将事件监听移除
5. js 代码中有对 DOM 节点的引用，dom 节点被移除的时候，引用还维持
6. JavaScript 中 4 种常见的内存泄露陷阱

babel把ES6转成ES5或者ES3之类的原理是什么

1. 它就是个编译器，输入语言是ES6+，编译目标语言是ES5
2. babel 官方工作原理
3. 解析：将代码字符串解析成抽象语法树

4. 变换：对抽象语法树进行变换操作
5. 再建：根据变换后的抽象语法树再生成代码字符串

Promise 模拟终止

1. 当新对象保持“pending”状态时，原Promise链将会中止执行。
2. return new Promise(()=>{}); // 返回“pending”状态的Promise对象
3. 从如何停掉 Promise 链说起(promise内存泄漏问题)

promise 放在try catch里面有什么结果

1. Promise 对象的错误具有冒泡性质，会一直向后传递，直到被捕获为止，也即是说，错误总会被下一个catch语句捕获
2. 当Promise链中抛出一个错误时，错误信息沿着链路向后传递，直至被捕获

网站性能优化

1. http 请求方面，减少请求数量，请求体积，对应的做法是，对项目资源进行压缩，控制项目资源的 dns 解析在2到4个域名，提取公告的样式，公共的组件，雪碧图，缓存资源，
2. 压缩资源，提取公共资源压缩，提取 css ，js 公共方法
3. 不要缩放图片，使用雪碧图，使用字体图表（阿里矢量图库）
4. 使用 CDN，抛开无用的 cookie
5. 减少重绘重排，CSS属性读写分离，最好不要用js 修改样式，dom 离线更新，渲染前指定图片的大小
6. js 代码层面的优化，减少对字符串的计算，合理使用闭包，首屏的js 资源加载放在最底部

js 自定义事件实现

1. 原生提供了3个方法实现自定义事件
2. createEvent，设置事件类型，是 html 事件还是 鼠标事件
3. initEvent 初始化事件，事件名称，是否允许冒泡，是否阻止自定义事件
4. dispatchEvent 触发事件

angular 双向数据绑定与vue数据的双向数据绑定

1. 二者都是 MVVM 模式开发的典型代表
2. angular 是通过脏检测实现，angular 会将 UI 事件，请求事件，settimeout 这类延迟，的对象放入到事件监测的脏队列，当数据变化的时候，触发 $diget 方法进行数据的更新，视

图的渲染
3. vue 通过数据属性的数据劫持和发布订阅的模式实现，大致可以理解成由3个模块组成，observer 完成对数据的劫持，compile 完成对模板片段的渲染，watcher 作为桥梁连接二者，订阅数据变化及更新视图

get与post 通讯的区别

1. Get 请求能缓存，Post 不能
2. Post 相对 Get 安全一点点，因为Get 请求都包含在 URL 里，且会被浏览器保存历史纪录，Post 不会，但是在抓包的情况下都是一样的。
3. Post 可以通过 request body来传输比 Get 更多的数据，Get 没有这个技术
4. URL有长度限制，会影响 Get 请求，但是这个长度限制是浏览器规定的，不是 RFC 规定的
5. Post 支持更多的编码类型且不对数据类型限制

有没有去研究webpack的一些原理和机制，怎么实现的

1. 解析webpack配置参数，合并从shell传入和webpack.config.js文件里配置的参数，生产最后的配置结果。
2. 注册所有配置的插件，好让插件监听webpack构建生命周期的事件节点，以做出对应的反应。
3. 从配置的entry入口文件开始解析文件构建AST语法树，找出每个文件所依赖的文件，递归下去。
4. 在解析文件递归的过程中根据文件类型和loader配置找出合适的loader用来对文件进行转换。
5. 递归完后得到每个文件的最终结果，根据entry配置生成代码块chunk。
6. 输出所有chunk到文件系统。

ES6模块与CommonJS模块的差异

1. CommonJs 模块输出的是一个值的拷贝，ES6模块输出的是一个值的引用
2. CommonJS 模块是运行时加载，ES6模块是编译时输出接口
3. ES6输入的模块变量，只是一个符号链接，所以这个变量是只读的，对它进行重新赋值就会报错

模块加载AMD，CMD，CommonJS Modules/2.0 规范

1. 这些规范的目的都是为了 JavaScript 的模块化开发，特别是在浏览器端的

2. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行
3. CMD 推崇依赖就近，AMD 推崇依赖前置

Node 事件循环，js 事件循环差异

1. Node.js 的事件循环分为6个阶段
2. 浏览器和Node 环境下，microtask 任务队列的执行时机不同
    - Node.js中，microtask 在事件循环的各个阶段之间执行
    - 浏览器端，microtask 在事件循环的 macrotask 执行完之后执行
3. 递归的调用process.nextTick()会导致I/O starving，官方推荐使用setImmediate()

浅拷贝和深拷贝的问题

1. 深拷贝和浅拷贝是只针对Object和Array这样的复杂类型的
2. 也就是说a和b指向了同一块内存，所以修改其中任意的值，另一个值都会随之变化，这就是浅拷贝
3. 浅拷贝， ”Object.assign() 方法用于将所有可枚举的属性的值从一个或多个源对象复制到目标对象。它将返回目标对象
4. 深拷贝，JSON.parse()和JSON.stringify()给了我们一个基本的解决办法。但是函数不能被正确处理

开放性问题

> 开放性问题主要是考察候选人业务积累，是否有自己的思考，思考问题的方式，没有标准答案。不过有些问题挺刁的，哈哈哈哈，比如：" 你见过的最好的代码是什么？ "总之提前准备下没错。

1. 先自我介绍一下，说一下项目的技术栈，以及项目中遇到的一些问题
2. 从整体中，看你对项目的认识，框架的认识和自己思考
3. 项目中有没有遇到什么难点，怎么解决
4. 如果你在创业公司你怎么从0开始做（选择什么框架，选择什么构建工具）
5. 说一下你项目中用到的技术栈，以及觉得得意和出色的点，以及让你头疼的点，怎么解决的
6. 一个业务场景，面对产品不断迭代，以及需求的变动该怎么应对，具体技术方案实现
7. 你的学习来源是什么
8. 你觉得哪个框架比较好，好在哪里
9. 你觉得最难得技术难点是什么
10. 你见过的最好的代码是什么

顺便奉上作者的github地址：

https://github.com/bailinlin

精彩回顾 点击链接即可

- ♡ 饿了么CEO王磊送餐 体验工作还是作秀?
- ♡ 各类学习视频、编程资源都在这儿，欢迎收藏~
- ♡ ：**脚本之家粉丝福利，请查看！**
- ♡ 代码我只服雷布斯！分享雷军22年前写的代码
- ♡ 你家里如果有一个程序员，请一定心疼他！
- ♡ 雷军：十年编程路，我想给程序员几点建议！
- ♡ 程序员都在看的这 9 篇python学习文章！
- ♡ 周鸿祎谈程序员创业，条条都是中肯建议
- ♡ 99%的程序员都会收藏的书单，你读过几本?

长按下方图片
识别二维码 关注脚本之家

♡ 版权声明：转载文章和图片均来自公开网络，版权归作者本人所有，推送文章除非无法确认，我们都会注明作者和来源。如果出处有误或侵犯到原作者权益与我们联系删除或授权事宜。

阅读原文