


## 《京程一灯》精英班第四周笔试题 姓名：

请按要求完整作答。

- 1.请说明前端工程化与性能优化的关系，你在项目中有性能优化的经验么？（5分）

- 答：

 把能自动化进行性能优化的地方都交给前端工程化，千万别说没有如果实在没有可以把下面的这些题组合一下写到你的简历里去。

1.这里千万不要瞎说，比如你用Webpack分析打包体积大小，具体是怎么做到的呢。借助了duplicate-package-checker-webpack-plugin，通过执行webpack --profile --json > stats.json然后讲数据导入进去进行分析。

- 2.请说明静态资源文件放入CDN的好处？（5分）

- 答：

 这个题是一定要好好说的，如果一个站点连CDN都没有是一件太忧伤的故事。

1.什么是CDN?CDN是构建在网络之上的内容分发网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率。CDN的关键技术主要有内容存储和分发技术。

CDN的基本原理是广泛采用各种缓存服务器，将这些缓存服务器分布到用户访问相对集中的地区或网络中，在用户访问网站时，利用全局负载技术将用户的访问指向距离最近的工作正常的缓存服务器上，由缓存服务器直接响应用户请求。

2.加快了静态资源的访问速度。

3.更加能够合理的设置静态资源的缓存策略。

4.减少对主域名的请求压力。请求CDN主域名的Cookies不随着来回传递，减少请求体积。

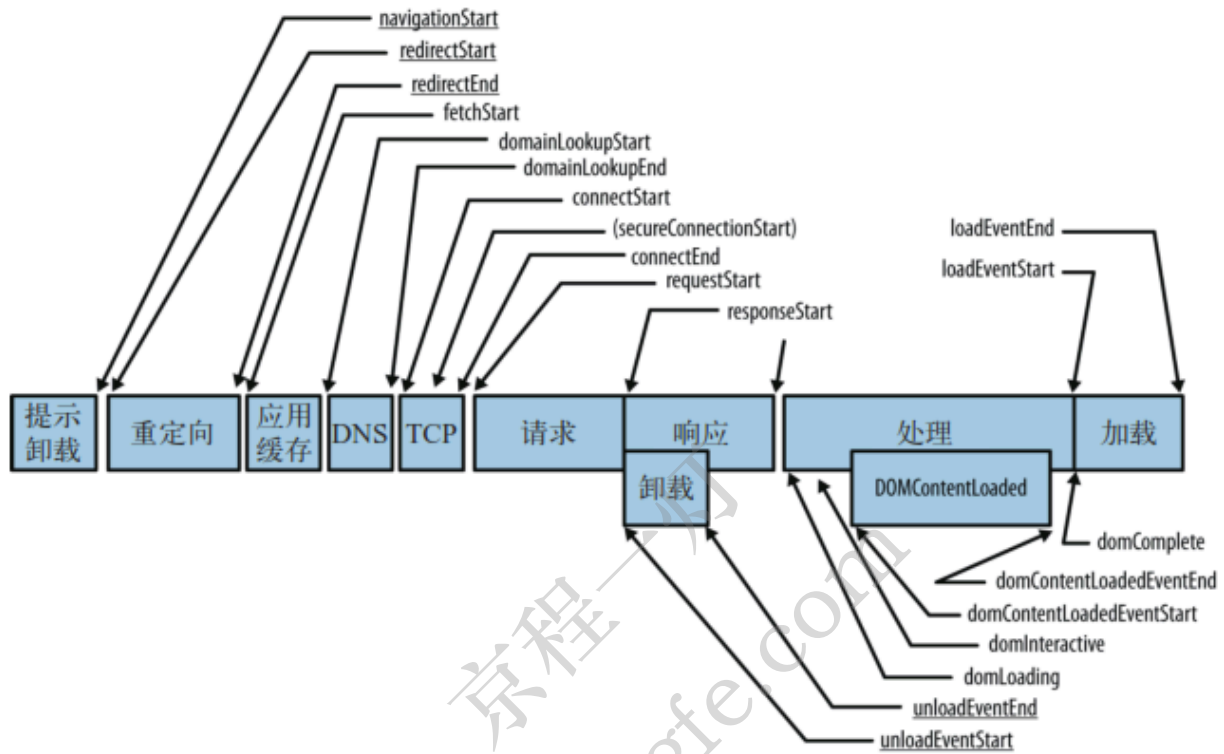
- 3.请你写出你知道的缓存前端静态资源的常用方法。（10分）

 这个题如果你能答出ORM的库势必增分。

1.ORM封装localStorage + basket.js 解决问题

2.使用PWA完成所有缓存的控制（请开始你的表演）

- 4.下图是一张完整的网络请求阶段关键节点，如果让你开发一个性能统计平台，请写出你的思路（10分）



- 答：

🍊 上图我特意用了中文，请大家务必要脑子中清晰的反应出每一阶段。

1.对我们比较有用的页面性能数据大概包括如下几个：

DNS查询耗时、TCP链接耗时、request请求耗时、解析dom树耗时、白屏时间、domready时间、onload时间等，而这些参数是通过上面的performance.timing各个属性的差值组成的，计算方法如下：

DNS查询耗时： $\text{domainLookupEnd} - \text{domainLookupStart}$

TCP链接耗时： $\text{connectEnd} - \text{connectStart}$

request请求耗时： $\text{responseEnd} - \text{responseStart}$

解析dom树耗时： $\text{domComplete} - \text{domInteractive}$

白屏时间： $\text{responseStart} - \text{navigationStart}$

domready时间： $\text{domContentLoadedEventEnd} - \text{navigationStart}$

onload时间： $\text{loadEventEnd} - \text{navigationStart}$

NavigationTiming的目的是用于分析页面整体性能指标。如果要获取个别资源（例如JS、图片）的性能指标，就需要使用Resource Timing API。

2.打点请求数据 要用navigator.sendBeacon(“日志请求地址”)发送，因为它是等主渲染进程不忙的时候进行发送。做埋点最怕的就是影响主进程。

3.下面这张图完整的阐释了各种开发框架的区别，大家仔细研究。

- 5.在移动端特别重视首屏时间，请写出你掌握的降低首屏时间的方法？(10分)

	CSR	预渲染	SSR	同构
优点	<ul style="list-style-type: none"> <li>不依赖数据</li> <li>FP 时间最快</li> <li>客户端用户体验好</li> <li>内存数据共享</li> </ul>	<ul style="list-style-type: none"> <li>不依赖数据</li> <li>FCP 时间比 CSR 快</li> <li>客户端用户体验好</li> <li>内存数据共享</li> </ul>	<ul style="list-style-type: none"> <li>SEO 友好</li> <li>首屏性能高，FMP 比 CSR 和预渲染快</li> </ul>	<ul style="list-style-type: none"> <li>SEO 友好</li> <li>首屏性能高，FMP 比 CSR 和预渲染快</li> <li>客户端用户体验好</li> <li>内存数据共享</li> <li>客户端与服务端代码公用，开发效率高</li> </ul>
缺点	<ul style="list-style-type: none"> <li>SEO 不友好</li> <li>FCP、FMP 慢</li> </ul>	<ul style="list-style-type: none"> <li>SEO 不友好</li> <li>FMP 慢</li> </ul>	<ul style="list-style-type: none"> <li>客户端数据共享成本高</li> <li>模板维护成本高</li> </ul>	<ul style="list-style-type: none"> <li>Node 容易形成性能瓶颈</li> </ul>

- 答：

🍊 这题是个精华。

1.我们采用SSR的方式让首屏直出，SSR有很多方式其实就是后端直接渲染模板。对我们来说最直接的方式就是Node+Swig。

2.首屏的时间是保证了，接下来要保证用户的可操作性我们页面全部采用a。

3.页面可操作性是保证了，可是多页又造成了资源重复加载。所以要采用对所有a进行代理完成SPA。

- 6.我们知道一个网站的并发是有限制的，那即使合并了很多请求还是不能解决静态资源请求过多的情况，你有什么优化手段么。(5分)

🍊 合并能减少网络请求，但是文件变大。拆分文件文件体积变小，但是请求增多。你怎么平衡？

1.首先同一个CDN的请求数量不能超过5个因为很多浏览器有并发限制，其次每一个文件体积已GZIP压缩过后32k最大为一个衡量标准（这个32是一个经验数字，因为原文件就100多很大了）

2.启动类库HTTP强缓存，然后业务JS缓存。同时开启业务JS增量更新方案，也可以控制更细粒度的JS增量更新方案如mtjs。

- 7.性能优化不仅仅限于前台静态资源文件，很多时候还需要优化Node,那你知道哪些优化后台的技术手段么。(20分)

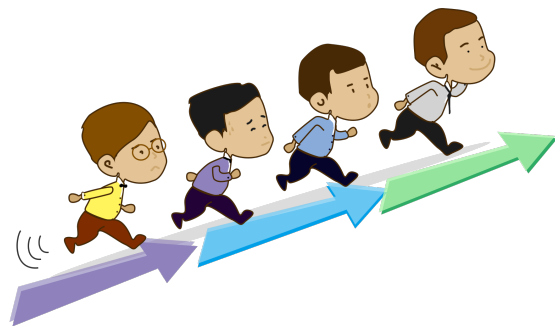
🍊 切记上线前要进行严格的压力测试与内存泄露排查。防范于未然。

- 1.通过jmeter或wrk等压力测试工具，进行qps预演。计算所需服务器数量和配置，同时寻找是否代码存在内存泄露。大对象、消费不及时、闭包都会造成内存泄露，可以通过process.memoryUsage或memwatch+heapdump对内存泄露进行排查。
- 2.配合运维进行负载均衡的配置，包括Nginx或者Docker。
- 3.使用chrome://inspect/#devices 和 node --inspect app.js 对Node应用进行调试。

- 8.请写出你知道的雅虎军规的细则。(10分)


🍊 这个题就是小学的教科书，每一条请务必理解记忆。

- 1.尽量减少HTTP请求数
- 2.减少DNS查找
- 3.避免重定向
- 4.让Ajax可缓存
- 5.延迟加载组件
- 6.预加载组件
- 7.减少DOM元素的数量
- 8.跨域分离组件
- 9.尽量少用iframe
- 10.杜绝404
- 11.避免使用CSS表达式
- 12.选择<link>舍弃@import
- 13.避免使用滤镜
- 14.把样式表放在顶部
- 15.去除重复脚本
- 16.尽量减少DOM访问
- 17.用智能的事件处理器



- 18.把脚本放在底部
- 19.把JavaScript和CSS放到外面
- 20.压缩JavaScript和CSS
- 21.优化图片
- 22.优化CSS Sprite
- 23.不要用HTML缩放图片
- 24.用小的可缓存的favicon.ico ( P.S. 收藏夹图标 )
- 25.给Cookie减肥
- 26.把组件放在不含cookie的域下
- 27.保证所有组件都小于25K
- 28.把组件打包到一个复合文档里
- 29.Gzip组件
- 30.避免图片src属性为空
- 31.配置ETags
- 32.对Ajax用GET请求
- 33.尽早清空缓冲区 ( Bigpipe )
- 34.使用CDN ( 内容分发网络 )
- 35.添上Expires或者Cache-Control HTTP头

- 9.假设现在给你一个接口能够取到用户的网速和机型等关键信息，你能做出什么具体的抉择优化你的H5 WebAPP。 (10分)

 这个题能体现出你对性能优化的细致性。

- 1.可以根据空图和多普勒测速拿到用户的网速之后，就可以对很多图片进行选择。比如用户的网速差不显示高清的图，甚至根据用户的网速我们可以出一个简版的项目，已达到极佳的解决用户体验。
- 2.可以根据用户的机型，来渲染比较适合渲染UI的解决方案（如项目中有需要Canvas渲染的部分和也可选择图片的部分，可以直接出最优方案）。

- 10.请绘制浏览器实现缓存机制。 (10分)

- 答：附：HTTP 缓存资源小结

- 1. expires

expires: Thu, 16 May 2019 03:05:59 GMT

在 http 头中设置一个过期时间，在这个过期时间之前，浏览器的请求都不会发出，而是自动从缓存中读取文件，除非缓存被清空，或者强制刷新。缺陷在于，服务器时间和用户端时间可能存在不一致，所以 HTTP/1.1 加入了 cache-control 头来改进这个问题。

- 2. cache-control

cache-control: max-age=31536000

设置过期的时间长度（秒），在这个时间范围内，浏览器请求都会直接读缓存。当 expires 和 cache-control 都存在时，cache-control 的优先级更高。

- 3. last-modified / if-modified-since 他有一组请求/响应头

响应头：last-modified: Wed, 16 May 2018 02:57:16 GMT

请求头：if-modified-since: Wed, 16 May 2018 05:55:38 GMT

服务器端返回资源时，如果头部带上了 last-modified，那么资源下次请求时就会把值加入到请求头 if-modified-since 中，服务器可以对比这个值，确定资源是否发生变化，如果没有发生变化，则返回 304。

- 4. etag / if-none-match 他也有一组请求/响应头

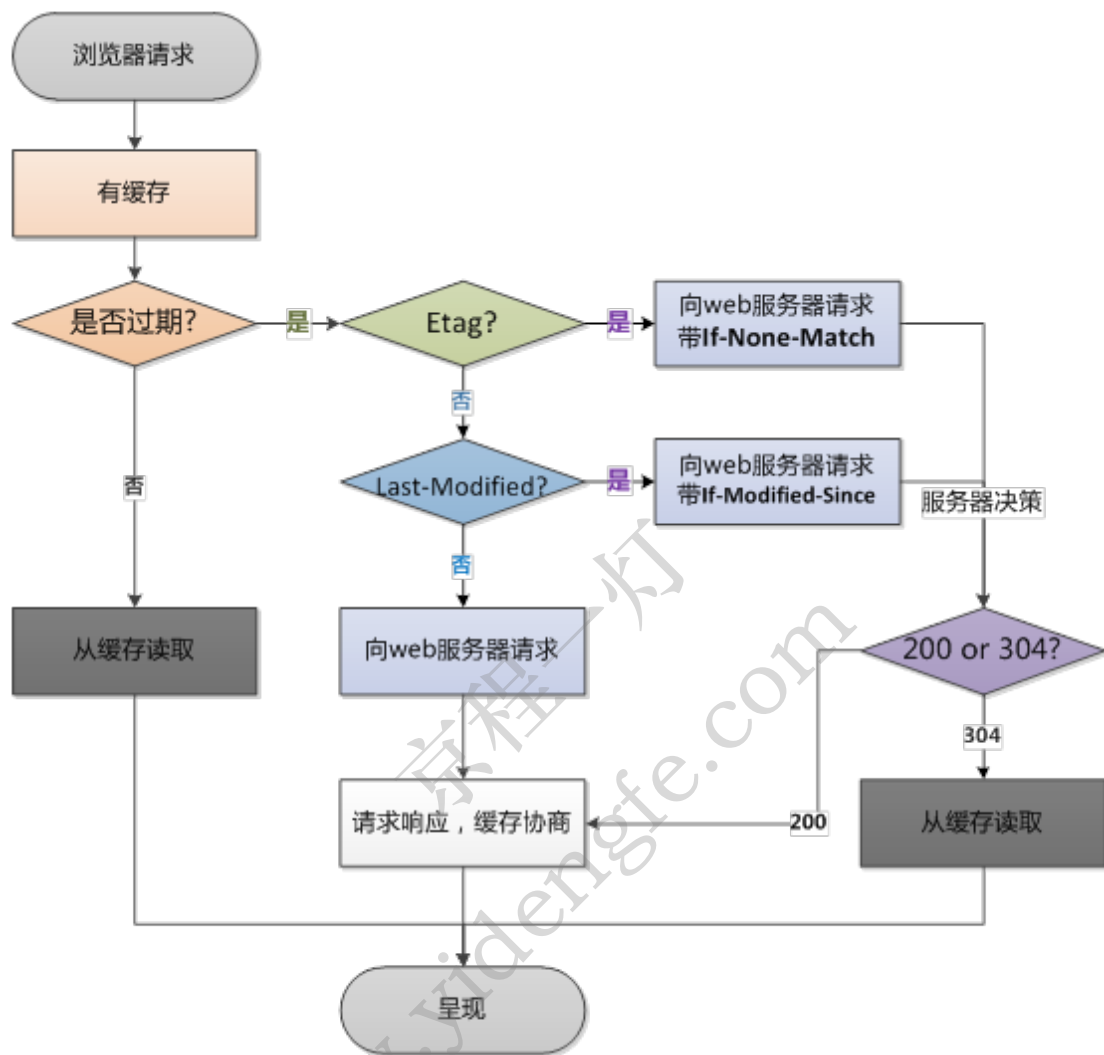
响应头：etag: "D5FC8B85A045FF720547BC36FC872550"

请求头：if-none-match: "D5FC8B85A045FF720547BC36FC872550"

原理类似，服务器端返回资源时，如果头部带上了 etag，那么资源下次请求时就会把值加入到请求头 if-none-match 中，服务器可以对比这个值，确定资源是否发生变化，如果没有发生变化，则返回 304。

-  上面四种缓存的优先级：cache-control > expires > etag > last-modified

- 具体的流程图请看下一页~



- 12.你了解过quicklink么，能大概描述下他的原理么？（5分）

- 答：

Intersection Observer、requestIdleCallback、navigator.connection.effectiveType、<link rel=prefetch> or XHR

前端路漫漫，这是最好的结束，也是全新的开始。京程一灯永远是您的后盾，无论何时需要帮助，我们永远都在，无论你在哪，如果需要任何帮助请随时联系我们，祝好~

