**Web前端性能优化:**

内容优化

(1)减少HTTP请求数:这条策略是最重要最有效的，因为一个完整的请求要经过DNS寻址，与服务器建立连接，发送数据，等待服务器响应，接收数据这样一个消耗时间成本和资源成本的复杂的过程。常见方法：合并多个CSS文件和js文件，利用CSS Sprites整合图像，Inline Images(使用 data：URL scheme在实际的页面嵌入图像数据 )，合理设置HTTP缓存等。

(2)减少DNS查找

(3)避免重定向

(4)使用Ajax缓存

(5)延迟加载组件,预加载组件

(6)减少DOM元素数量:页面中存在大量DOM元素,会导致javascript遍历DOM的效率变慢。用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能

缓存DOM节点查找的结果

(7)最小化iframe的数量：iframes 提供了一个简单的方式把一个网站的内容嵌入到另一个网站中。但其创建速度比其他包括JavaScript和CSS的DOM元素的创建慢了1-2个数量级。

(8)避免404：HTTP请求时间消耗是很大的，因此使用HTTP请求来获得一个没有用处的响应（例如404没有找到页面）是完全没有必要的，它只会降低用户体验而不会有一点好处。

服务器优化

(1)使用内容分发网络（CDN）：把网站内容分散到多个、处于不同地域位置的服务器上可以加快下载速度。

(2)GZIP压缩

(3)设置ETag：ETags（Entity tags，实体标签）是web服务器和浏览器用于判断浏览器缓存中的内容和服务器中的原始内容是否匹配的一种机制。

(4)提前刷新缓冲区

(5)对Ajax请求使用GET方法

(6)避免空的图像src

Cookie优化

(1)减小Cookie大小

(2)针对Web组件使用域名无关的Cookie

CSS优化

(1)将CSS代码放在HTML页面的顶部

(2)避免使用CSS表达式

(3)使用<link>来代替@import

(4)避免使用Filters

javascript优化

(1)将JavaScript脚本放在页面的底部。

(2)将JavaScript和CSS作为外部文件来引用：在实际应用中使用外部文件可以提高页面速度，因为JavaScript和CSS文件都能在浏览器中产生缓存。

(3)缩小JavaScript和CSS

(4)删除重复的脚本

(5)最小化DOM的访问：使用JavaScript访问DOM元素比较慢。

(6)开发智能的事件处理程序

(7)javascript代码注意：谨慎使用with,避免使用eval Function函数,减少作用域链查找。

图像优化

(1)优化图片大小

(2)通过CSS Sprites优化图片，初始首屏之外的图片资源按需加载

(3)不要在HTML中使用缩放图片

(4)favicon.ico要小而且可缓存

**谈谈以前端角度出发做好SEO需要考虑什么？**

a. 了解搜索引擎如何抓取网页和如何索引网页

b. meta标签优化

c. 关键词分析

d. 付费给搜索引擎

e. 链接交换和链接广泛度（Link Popularity）

f. 合理的标签使用

****为什么利用多个域名来存储网站资源会更有效？****

CDN 缓存更方便

突破浏览器并发限制

节约 cookie 带宽

节约主域名的连接数，优化页面响应速度

防止不必要的安全问题

**前端页面的性能优化有哪些工作可以做？**

**优化JavaScript:**

避免使用setTimeout或setInterval来进行页面的更新。这些将callback在框架中的某个点调用，可能在最后。我们想要做的是在框架开始时触发视图变化。

将长时间运行的JavaScript计算移动到Web Workers 。

使用微任务在几个帧中引入DOM更改。这是因为任务需要访问DOM，Web Workers无法访问它。这基本上意味着你会打破大任务分解成较小的和内部运行它们requestAnimationFrame ，setTimeout，setInterval这取决于任务的性质。

**优化CSS**

通过添加和删除元素，更改属性等来修改DOM将使浏览器重新计算元素样式，并且在许多情况下，重新计算整个页面的布局或至少部分布局。

要优化渲染，请考虑以下事项：

降低选择器的复杂性。与构建样式本身的其余工作相比，选择器复杂度可能占用元素样式所需时间的50％以上。

减少必须进行样式计算的元素数量。实质上，直接将样式更改为几个元素，而不是使页面整体无效。

**优化布局**

浏览器的布局重新计算可能非常繁重。考虑以下优化：

尽可能减少布局数量。更改样式时，浏览器会检查是否有任何更改需要重新计算布局。对诸如宽度，高度，左侧，顶部以及通常与几何相关的属性等属性的更改需要布局。因此，尽量避免更改它们。

flexbox尽可能使用较旧的布局模型。它的工作速度更快，可以为您的应用创造巨大的性能优势。

避免强制同步布局。要记住的是，当JavaScript运行时，前一帧中的所有旧布局值都是已知的，可供您查询。如果您访问box.offsetHeight它将不是一个问题。但是，如果您在访问框之前更改了框的样式（例如，通过向元素动态添加一些CSS类），则浏览器必须首先应用样式更改然后运行布局。这可能非常耗时且资源密集，因此请尽可能避免使用它。

**优化绘制**

这通常是所有任务中运行时间最长的，因此尽可能避免重绘的操作。这是我们可以做的：

更改transforms或opacity以外的任何属性都会触发重绘。谨慎使用它。

如果触发页面重构，则会触发重绘，因为更改几何体会导致元素的视图更改。

通过图层提升和动画编排减少绘画区域。

**21、请说出三种减低页面加载时间的方法**

1.减少 http 请求次数：CSS Sprites, JS、CSS 源码压缩、图片大小控制合适；网页

Gzip，CDN 托管，data 缓存 ，图片服务器。

2.前端模板 JS+数据，减少由于 HTML 标签导致的带宽浪费，前端用变量保存 AJAX

请求结果，每次操作本地变量，不用请求，减少请求次数

3.用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 javascript 性能。

4.当需要设置的样式很多时设置 className 而不是直接操作 style。

5.少用全局变量、缓存 DOM 节点查找的结果。减少 IO 读取操作。

6.避免使用 CSS Expression（css 表达式)又称 Dynamic properties(动态属性)。

7.图片预加载，将样式表放在顶部，将脚本放在底部。

8.避免在页面的主体布局中使用 table，table 要等其中的内容完全下载之后才会显

示出来，显示比 div+css 布局慢。

**突破浏览器并发限制=》详解：**

一个HTTP 连接请求在同一时间只能被一个线程访问。

浏览器的并发请求数目限制是针对同一域名的。

意即，同一时间针对同一域名下的请求有一定数量限制。超过限制数目的请求会被阻塞，这就是很多网站专门解决这个问题的原因。

有的请求会持续很长时间，如果把 img, css, js… 都放到http://一个域名下面，其他请求就迟迟无法完成，浏览者看来就是『卡住了』。而把图片放到另一个域名之后，css和图片就可以并发请求了。

**浏览器限制并发的原因**

由于 TCP 协议的限制，PC 端只有65536个端口可用以向外部发出连接，而操作系统对半开连接数也有限制以保护操作系统的 TCP\IP 协议栈资源不被迅速耗尽，因此浏览器不好发出太多的 TCP 连接，而是采取用完了之后再重复利用 TCP 连接或者干脆重新建立 TCP 连接的方法。

如果采用阻塞的套接字模型来建立连接，同时发出多个连接会导致浏览器不得不多开几个线程，而线程有时候算不得是轻量级资源，毕竟做一次上下文切换开销不小。

这是浏览器作为一个有良知的客户端在保护服务器。就像以太网的冲突检测机制，客户端在使用公共资源的时候必须要自行决定一个等待期。当超过2个客户端要使用公共资源时，强势的那个邪恶的客户端可能会导致弱势的客户端完全无法访问公共资源。从前迅雷被喷就是因为它不是一个有良知的客户端，它作为 HTTP 协议客户端没有考虑到服务器的压力，作为 BT 客户端没有考虑到自己回馈上传量的义务。