5. 单独拎出来的缓存问题， HTTP 的缓存（这部分包括http缓存头部， ETag ， catch-control 等)

6. 浏览器接收到 HTTP 数据包后的解析流程（解析 html -词法分析然后解析成 dom 树 、解析 css 生成 css 规则树 、合并成 render 树，然后 layout 、 painting 渲染 、复合图层的合成 、 GPU 绘制 、外链资源的处理 、 loaded 和 DOMContentLoaded 等)

7. CSS 的可视化格式模型（元素的渲染规则， 如包含块，控制框， BFC ， IFC 等概念)

8. JS 引擎解析过程（ JS 的解释阶段，预处理阶段，执行阶段生成执行上下文， VO ，作用域链 、回收机制等等)

9. 其它（可以拓展不同的知识模块， 如跨域，web安全， hybrid 模式等等内容)

5 如何进行网站性能优化

- content 方面
  - 减少 HTTP 请求：合并文件 、 CSS 精灵 、 inline Image
  - 减少 DNS 查询： DNS 缓存 、将资源分布到恰当数量的主机名
  - 减少 DOM 元素数量
- Server 方面
  - 使用 CDN
  - 配置 ETag
  - 对组件使用 Gzip 压缩
- Cookie 方面
  - 减小 cookie 大小
- css 方面
  - 将样式表放到页面顶部
  - 不使用 CSS 表达式
  - 使用 <link> 不使用 @import
- Javascript 方面
  - 将脚本放到页面底部
  - 将 javascript 和 css 从外部引入
  - 压缩 javascript 和 css
  - 删除不需要的脚本

- 减少 DOM 访问

- 图片方面
  - 优化图片：根据实际颜色需要选择色深 、压缩
  - 优化 css 精灵
  - 不要在 HTML 中拉伸图片

6 HTTP状态码及其含义

- 1XX ： 信息状态码
  - 100 Continue 继续，一 般在发送 post 请求时， 已发送了 http header 之后服务端将返回此信息，表示确认， 之后发送具体参数信息
- 2XX ： 成功状态码
  - 200 OK 正常返回信息
  - 201 Created 请求成功并且服务器创建了新的资源
  - 202 Accepted 服务器已接受请求，但尚未处理
- 3XX ： 重定向
  - 301 Moved Permanently 请求的网页已永久移动到新位置。
  - 302 Found 临时性重定向。
  - 303 See Other 临时性重定向，且总是使用 GET 请求新的 URI 。
  - 304 Not Modified 自从上次请求后，请求的网页未修改过。
- 4XX ： 客户端错误
  - 400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。
  - 401 Unauthorized 请求未授权。
  - 403 Forbidden 禁止访问。
  - 404 Not Found 找不到如何与 URI 相匹配的资源。
- 5XX: 服务器错误
  - 500 Internal Server Error 最常见的服务器端错误。
  - 503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

7 语义化的理解

- 用正确的标签做正确的事情!
- HTML 语义化就是让页面的内容结构化，便于对浏览器 、搜索引擎解析;
- 在没有样式 CSS 情况下也以一种文档格式显示， 并且是容易阅读的。
- 搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 SEO 。
- 使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解

8 介绍一下你对浏览器内核的理解?

- 主要分成两部分：渲染引擎(layout engineer 或 Rendering Engine)和 JS 引擎
- 渲染引擎：负责取得网页的内容（ HTML 、 XML 、图像等等）、整理讯息（例如加入 CSS 等)， 以及计算网页的显示方式，然后会输出至显示器或打印机 。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同 。所有网页浏览器 、电子邮件客户端以及其它需要编辑 、显示网络内容的应用程序都需要内核
- JS 引擎则：解析和执行 javascript 来实现网页的动态效果
- 最开始渲染引擎和 JS 引擎并没有区分的很明确，后来JS引擎越来越独立， 内核就倾向于只指渲染引擎

9 html5有哪些新特性、移除了那些元素?

- HTML5 现在已经不是 SGML 的子集， 主要是关于图像，位置，存储， 多任务等功能的增加
  - 绘画 canvas
  - 用于媒介回放的 video 和 audio 元素
  - 本地离线存储 localStorage 长期存储数据， 浏览器关闭后数据不丢失
  - sessionStorage 的数据在浏览器关闭后自动删除
  - 语意化更好的内容元素， 比如 article 、 footer 、 header 、 nav 、 section
  - 表单控件， calendar 、 date 、 time 、 email 、 url 、 search
  - 新的技术 webworker 、 websocket 、 Geolocation
- 移除的元素：
  - 纯表现的元素： basefont 、 big 、 center 、 font 、 s 、 strike 、 tt 、 u
  - 对可用性产生负面影响的元素： frame 、 frameset 、 noframes
- 支持 HTML5 新标签：
  - IE8/IE7/IE6 支持通过 document.createElement 方法产生的标签
  - 可以利用这一特性让这些浏览器支持 HTML5 新标签
  - 浏览器支持新标签后， 还需要添加标签默认的样式
- 当然也可以直接使用成熟的框架 、比如 html5shim

10 HTML5 的离线储存怎么使用， 工作原理能不能解释一下?

- 在用户没有与因特网连接时， 可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件
- 原理： HTML5 的离线存储是基于一个新建的 .appcache 文件的缓存机制(不是存储技术)， 通过这个文件上的解析清单离线存储资源， 这些资源就会像 cookie 一样被存储了下来。之后当网络在处于离线状态下时， 浏览器会通过被离线存储的数据进行页面展示
- 如何使用:
  - 页面头部像下面一样加入一个 manifest 的属性;
  - 在 cache.manifest 文件的编写离线存储的资源
  - 在离线状态时， 操作 window.applicationCache 进行需求实现

```json
CACHE MANIFEST
#v0.11
CACHE:
js/app.js
css/style.css
NETWORK:
resourse/logo.png
FALLBACK:
/offline.html
```

11 浏览器是怎么对 HTML5 的离线储存资源进行管理和加载的呢

- 在线的情况下， 浏览器发现 html 头部有 manifest 属性， 它会请求 manifest 文件， 如果是第一次访问 app ，那么浏览器就会根据manifest文件的内容下载相应的资源并且进行离线存储。如果已经访问过 app 并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面，然后浏览器会对比新的 manifest 文件与旧的 manifest 文件， 如果文件没有发生改变，就不做任何操作， 如果文件改变了，那么就会重新下载文件中的资源并进行离线存储。
- 离线的情况下， 浏览器就直接使用离线存储的资源。

12 请描述一下 cookies ， sessionStorage 和 localStorage 的区别?

- cookie 是网站为了标示用户身份而储存在用户本地终端（Client Side)上的数据（通常经过加密)
  - cookie数据始终在同源的http请求中携带（即使不需要)， 记会在浏览器和服务器间来回传递
  - sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存
  - 存储大小：
    - cookie 数据大小不能超过4k
    - sessionStorage 和 localStorage 虽然也有存储大小的限制，但比 cookie 大得多， 可以达到5M或更大
- 有期时间：
  - localStorage 存储持久数据， 浏览器关闭后数据不丢失除非主动删除数据
  - sessionStorage 数据在当前浏览器窗口关闭后自动删除
  - cookie 设置的 cookie 过期时间之前一直有效， 即使窗口或浏览器关闭

13 iframe有那些缺点?

- iframe 会阻塞主页面的 Onload 事件
- 搜索引擎的检索程序无法解读这种页面，不利于 SEO
- iframe 和主页面共享连接池， 而浏览器对相同域的连接有限制，所以会影响页面的并行加载
- 使用 iframe 之前需要考虑这两个缺点。如果需要使用 iframe ， 最好是通过 javascript 动态给 iframe 添加 src 属性值， 这样可以绕开以上两个问题

14 WEB标准以及W3C标准是什么?

- 标签闭合、标签小写、不乱嵌套、使用外链 css 和 js 、结构行为表现的分离

15 xhtml和html有什么区别?

- 一个是功能上的差别
  - 主要是 XHTML 可兼容各大浏览器、手机以及 PDA ， 并且浏览器也能快速正确地编译网页
- 另外是书写习惯的差别

- XHTML 元素必须被正确地嵌套， 闭合， 区分大小写，文档必须拥有根元素

16 Doctype作用? 严格模式与混杂模式如何区分？ 它们有何意义?

- 页面被加载的时， link 会同时被加载， 而 @imort 页面被加载的时， link 会同时被加载， 而 @import 引用的 CSS 会等到页面被加载完再加载 import 只在 IE5 以上才能识别， 而 link 是 XHTML 标签， 无兼容问题 link 方式的样式的权重 高于 @import 的权重
- <!DOCTYPE> 声明位于文档中的最前面， 处于 <html> 标签之前 。告知浏览器的解析器， 用什么文档类型 规范来解析这个文档
- 严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行
- 在混杂模式中， 页面以宽松的向后兼容的方式显示 。模拟老式浏览器的行为以防止站点无法工作 。 DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现

17 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？行内元素和块级元素有什么区别?

- 行内元素有： a b span img input select strong
- 块级元素有： div ul ol li dl dt dd h1 h2 h3 h4… p
- 空元素：
 <hr> <img> <input> <link> <meta>
- 行内元素不可以设置宽高，不独占一行
- 块级元素可以设置宽高， 独占一行

18 HTML全局属性(global attribute)有哪些

- class :为元素设置类标识
- data-* ：为元素增加自定义属性
- draggable ：设置元素是否可拖拽
- id ：元素 id ， 文档内唯一
- lang ：元素内容的的语言
- style ：行内 css 样式
- title ：元素相关的建议信息

19 Canvas和SVG有什么区别?

- svg 绘制出来的每一个图形的元素都是独立的 DOM 节点， 能够方便的绑定事件或用来修改 。 canvas 输出的是一整幅画布

svg 输出的图形是矢量图形，后期可以修改参数来自由放大缩小，不会失真和锯齿 。而 canvas 输出标量画布，就像一张图片一样，放大会失真或者锯齿

20 HTML5 为什么只需要写 <!DOCTYPE HTML>

- HTML5 不基于 SGML ， 因此不需要对 DTD 进行引用，但是需要 doctype 来规范浏览器的行为
- 而 HTML4.01 基于 SGML ,所以需要对 DTD 进行引用， 才能告知浏览器文档所使用的文档类型

21 如何在页面上实现一个圆形的可点击区域?

- svg
- border-radius
- 纯 js 实现 需要求一个点在不在圆上简单算法 、获取鼠标坐标等等

22 网页验证码是干嘛的， 是为了解决什么安全问题

- 区分用户是计算机还是人的公共全自动程序 。可以防止恶意破解密码 、刷票 、论坛灌水
- 有效防止黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试

23 viewport

```js
<meta name="viewport" content="width=device-width,initial-scale=1.0,minimu
   // width     设置viewport宽度，为一个正整数，或字符串‘device-width’
   // device-width  设备宽度
   // height    设置viewport高度，一 般设置了宽度，会自动解析出高度，可以不用设置
   // initial-scale     默认缩放比例（初始缩放比例），  为一个数字，可以带小数
   // minimum-scale     允许用户最小缩放比例，为一个数字，可以带小数
   // maximum-scale     允许用户最大缩放比例，为一个数字，可以带小数
   // user-scalable     是否允许手动缩放
```

- 延伸提问
  - 怎样处理 移动端 1px 被 渲染成 2px 问题

局部处理

- mate 标签中的 viewport 属性， initial-scale 设置为 1
- rem 按照设计稿标准走，外加利用 transfrome 的 scale(0.5) 缩小一倍即可;

全局处理

- mate 标签中的 viewport 属性， initial-scale 设置为 0.5
- rem 按照设计稿标准走即可

24 渲染优化

- 禁止使用 iframe （阻塞父文档 onload 事件）
  - iframe 会阻塞主页面的 Onload 事件
  - 搜索引擎的检索程序无法解读这种页面，不利于SEO
  - iframe 和主页面共享连接池， 而浏览器对相同域的连接有限制，所以会影响页面的并行加载
  - 使用 iframe 之前需要考虑这两个缺点。如果需要使用 iframe ， 最好是通过 javascript
  - 动态给 iframe 添加 src 属性值， 这样可以绕开以上两个问题
- 禁止使用 gif 图片实现 loading 效果（降低 CPU 消耗，提升渲染性能）
- 使用 CSS3 代码代替 JS 动画（尽可能避免重绘重排以及回流）
- 对于一些小图标， 可以使用base64位编码， 以减少网络请求。但不建议大图使用， 比较耗费 CPU
  - 小图标优势在于
    - 减少 HTTP 请求
    - 避免文件跨域
    - 修改及时生效
- 页面头部的 <style></style> <script></script> 会阻塞页面；（因为 Renderer 进程中 JS 线程和渲染线程是互斥的）
- 页面中空的 href 和 src 会阻塞页面其他资源的加载 (阻塞下载进程)
- 网页 gzip ， CDN 托管， data 缓存， 图片服务器
- 前端模板 1S+数据，减少由于 HTML 标签导致的带宽浪费， 前端用变量保存A1AX请求结果，每次操作本地变量，不用请求，减少请求次数
- 用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 javascript 性能
- 当需要设置的样式很多时设置 className 而不是直接操作 style
- 少用全局变量、缓存 DOM 节点查找的结果。减少 IO 读取操作

- 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳
- 对普通的网站有一个统一的思路，就是尽量向前端优化 、减少数据库操作 、减少磁盘 IO

25 meta viewport相关

```html
<!DOCTYPE html>  <!--H5标准声明，使用 HTML5 doctype，不区分大小写-->
<head lang="en">  <!--标准的 lang 属性写法-->
<meta charset='utf-8'>  <!--声明文档使用的字符编码-->
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>  <!--优先使
<meta name="description" content="不超过150个字符"/>  <!--页面描述-->
<meta name="keywords" content="" />  <!-- 页面关键词-->
<meta name="author" content="name, email@gmail.com"/>  <!--网页作者-->
<meta name="robots" content="index,follow"/>  <!--搜索引擎抓取-->
<meta name="viewport" content="initial-scale=1, maximum-scale=3, minimum-sc
<meta name="apple-mobile-web-app-title" content="标题">  <!--iOS 设备 begin--
<meta name="apple-mobile-web-app-capable" content="yes"/>  <!--添加到主屏后的标
是否启用 WebApp 全屏模式，删除苹果默认的工具栏和菜单栏-->
<meta name="apple-itunes-app" content="app-id=myAppStoreID, affiliate-data=
<!--添加智能 App 广告条 Smart App Banner (iOS 6+ Safari) -->
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
<meta name="format-detection" content="telphone=no, email=no" />  <!--设置苹果
<meta name="renderer" content="webkit">  <!-- 启用360浏览器的极速模式(webkit)--
<meta http-equiv="X-UA-Compatible" content="IE=edge">  <!--避免IE使用兼容模
<meta http-equiv="Cache-Control" content="no-siteapp" />  <!--不让百度转码-
<meta name="HandheldFriendly" content="true">  <!--针对手持设备优化，主要是针
<meta name="MobileOptimized" content="320">  <!--微软的老式浏览器-->
<meta name="screen-orientation" content="portrait">  <!--uc强制竖屏-->
<meta name="x5-orientation" content="portrait">  <!--QQ强制竖屏-->
<meta name="full-screen" content="yes">  <!--UC强制全屏-->
<meta name="x5-fullscreen" content="true">  <!--QQ强制全屏-->
<meta name="browsermode" content="application">  <!--UC应用模式-->
<meta name="x5-page-mode" content="app">  <!-- QQ应用模式-->
<meta name="msapplication-tap-highlight" content="no">  <!--windows phone
设置页面不缓存-->
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
```

26 你做的页面在哪些流览器测试过？这些浏览器的内核分别是什么？

- IE ： trident 内核
- Firefox ： gecko 内核

- Safari ：webkit 内核
- Opera :以前是 presto 内核， Opera 现已改用Google - Chrome 的 Blink 内核
- Chrome:Blink (基于 webkit ， Google与Opera Software共同开发)

27 div+css的布局较table布局有什么优点?

- 改版的时候更方便 只要改 css 文件。
- 页面加载速度更快 、结构化清晰 、页面显示简洁。
- 表现与结构相分离。
- 易于优化（ seo ）搜索引擎更友好， 排名更容易靠前。

28 a： img的alt与title有何异同？b：strong与em的异同?

- alt(alt text) :为不能显示图像 、窗体或 applets 的用户代理 （ UA ）， alt 属性用来指定替换文字 。替换文字的语言由 lang 属性指定 。(在IE浏览器下会在没有 title 时把 alt 当成 tool tip 显示)
- title(tool tip) :该属性为设置该属性的元素提供建议性的信息
- strong :粗体强调标签， 强调，表示内容的重要性
- em :斜体强调标签，更强烈强调，表示内容的强调点

29 你能描述一下渐进增强和优雅降级之间的不同吗

- 渐进增强：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果 、交互等改进和追加功能达到更好的用户体验。
- 优雅降级：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

> 区别：优雅降级是从复杂的现状开始， 并试图减少用户体验的供给， 而渐进增强则是从一个非常基础的， 能够起作用的版本开始， 并不断扩充， 以适应未来环境的需要 。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带

30 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

31 简述一下src与href的区别

- src 用于替换当前元素， href用于在当前文档和引用资源之间确立联系。
- src 是 source 的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档内，例如 js 脚本，img 图片和 frame 等元素

> <script src ="js.js"></script> 当浏览器解析到该元素时，会暂停其他资源的下载和处理， 直到将该资源加载、编译、执行完毕， 图片和框架等元素也如此， 类似于将所指向资源嵌入当前标签内。这也是为什么将js脚本放在底部而不是头部

- href 是 Hypertext Reference 的缩写，指向网络资源所在位置， 建立和当前元素（锚点）或当前文档（链接）之间的链接， 如果我们在文档中添加
- <link href="common.css" rel="stylesheet"/> 那么浏览器会识别该文档为 css 文件，就会并行下载资源并且不会停止对当前文档的处理。这也是为什么建议使用 link 方式来加载 css ， 而不是使用 @import 方式

32 知道的网页制作会用到的图片格式有哪些?

- png-8 、 png-24 、 jpeg 、 gif 、 svg

> 但是上面的那些都不是面试官想要的最后答案。面试官希望听到是 Webp ,Apng 。 （是否有关注新技术，新鲜事物）

- Webp： WebP 格式，谷歌（google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有 JPEG 的 2/3 ， 并能节省大量的服务器带宽资源和数据空间。 Facebook Ebay 等知名网站已经开始测试并使用 WebP 格式。
- 在质量相同的情况下， WebP格式图像的体积要比JPEG格式图像小 40% 。
- Apng：全称是 “Animated Portable Network Graphics” ,是PNG的位图动画扩展， 可以实现png格式的动态图片效果。04年诞生，但一直得不到各大浏览器厂商的支持， 直到日前得到 iOS safari 8 的支持，有望代替 GIF 成为下一代动态图标准

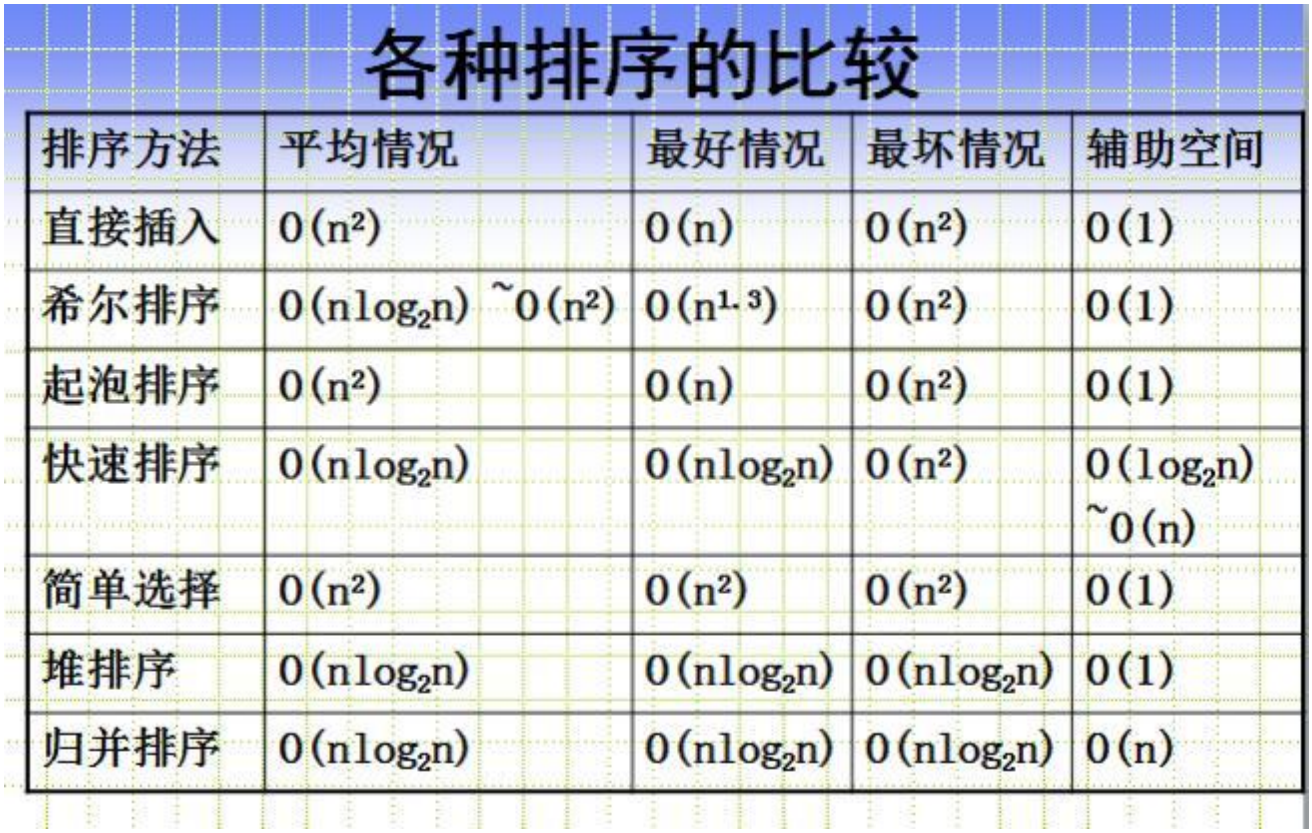
33 在css/js代码上线之后开发人员经常会优化性能，从用户刷新网页开始， 一次js请求一般情况下有哪些地方会有缓存处理?

> dns 缓存， cdn 缓存， 浏览器缓存， 服务器缓存

33 一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。

- 图片懒加载，在页面上的未可视区域可以添加一个滚动事件， 判断图片位置与浏览器顶端的距离与页面的距离， 如果前者小于后者，优先加载。
- 如果为幻灯片 、相册等， 可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。
- 如果图片为css图片， 可以使用 CSSsprite ， SVGsprite ， Iconfont 、 Base64 等技术。
- 如果图片过大， 可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图， 以提高用户体验。
- 如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩， 图片压缩后大小与展示一致。

34 常见排序算法的时间复杂度,空间复杂度



35 web开发中会话跟踪的方法有哪些

- cookie
- session
- url 重写
- 隐藏 input
- ip 地址

36 HTTP request报文结构是怎样的

1. 首行是Request-Line包括：请求方法，请求URI，协议版本， CRLF
2. 首行之后是若干行请求头， 包括general-header， request-header或者entity-header，每个一行以CRLF结束
3. 请求头和消息实体之间有一个CRLF分隔
4. 根据实际请求需要可能包含一个消息实体 一个请求报文例子如下：

```
GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.1
Host: www.w3.org
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
Referer: https://www.google.com.hk/
```

```
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: authorstyle=yes
If-None-Match: "2cc8-3e3073913b100"
If-Modified-Since: Wed, 01 Sep 2004 13:24:52 GMT

name=qiu&age=25
```

37 HTTP response报文结构是怎样的

- 首行是状态行包括：HTTP版本，状态码，状态描述，后面跟一个CRLF
- 首行之后是若干行响应头， 包括：通用头部，响应头部，实体头部
- 响应头部和响应实体之间用一个CRLF空行分隔
- 最后是一个可能的消息实体 响应报文例子如下：

```
HTTP/1.1 200 OK
Date: Tue, 08 Jul 2014 05:28:43 GMT
Server: Apache/2
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
ETag: "40d7-3e3073913b100"
Accept-Ranges: bytes
Content-Length: 16599
Cache-Control: max-age=21600
Expires: Tue, 08 Jul 2014 11:28:43 GMT
P3P: policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Content-Type: text/html; charset=iso-8859-1

{"name": "qiu", "age": 25}
```

二、CSS部分

1 css sprite是什么,有什么优缺点

- 概念：将多个小图片拼接到一个图片中 。通过 `background-position` 和元素尺寸调节需要显示的背景图案。
- 优点：
  - 减少 `HTTP` 请求数，极大地提高页面加载速度
  - 增加图片信息重复度，提高压缩比，减少图片大小

- 更换风格方便， 只需在一张或几张图片上修改颜色或样式即可实现

- 缺点：
  - 图片合并麻烦
  - 维护麻烦，修改一个图片可能需要从新布局整个图片，样式

2 `display: none;` 与 `visibility: hidden;` 的区别

- 联系：它们都能让元素不可见
- 区别：
  - `display:none` ;会让元素完全从渲染树中消失， 渲染的时候不占据任何空间; `visibility: hidden` ;不会让元素从渲染树消失， 渲染师元素继续占据空间， 只是内容不可见
  - `display: none` ;是非继承属性， 子孙节点消失由于元素从渲染树消失造成， 通过修改子孙节点属性无法显示 ; `visibility: hidden;` 是继承属性， 子孙节点消失由于继承了 `hidden` ， 通过设置 `visibility: visible;` 可以让子孙节点显式
  - 修改常规流中元素的 `display` 通常会造成文档重排 。修改 `visibility` 属性只会造成本元素的重绘。
  - 读屏器不会读取 `display: none` ;元素内容; 会读取 `visibility: hidden;` 元素内容

3 `link` 与 `@import` 的区别

1. `link` 是 `HTML` 方式， `@import` 是CSS方式
2. `link` 最大限度支持并行下载， `@import` 过多嵌套导致串行下载， 出现 `FOUC` (文档样式短暂失效)
3. `link` 可以通过 `rel="alternate stylesheet"` 指定候选样式
4. 浏览器对 `link` 支持早于 `@import` ， 可以使用 `@import` 对老浏览器隐藏样式
5. `@import` 必须在样式规则之前， 可以在css文件中引用其他文件
6. 总体来说： `link` 优于 `@import`

4 什么是FOUC?如何避免

- `Flash Of Unstyled Content` ： 用户定义样式表加载之前浏览器使用默认样式显示文档，用户样式加载渲染之后再从新显示文档， 造成页面闪烁。
- 解决方法：把样式表放到文档的 `<head>`

5 如何创建块级格式化上下文(block formatting context),BFC有什么用

- 创建规则：
  - 根元素
  - 浮动元素（ `float` 不取值为 `none` ）
  - 绝对定位元素（ `position` 取值为 `absolute` 或 `fixed` ）
  - `display` 取值为 `inline-block` 、 `table-cell` 、 `table-caption` 、 `flex` 、 `inline-flex` 之一的元素
  - `overflow` 不取值为 `visible` 的元素
- 作用：
  - 可以包含浮动元素
  - 不被浮动元素覆盖
  - 阻止父子元素的 `margin` 折叠

6 display 、float 、position的关系

- 如果 `display` 取值为 `none` ，那么 `position` 和 `float` 都不起作用， 这种情况下元素不产生框
- 否则， 如果 `position` 取值为 `absolute` 或者 `fixed` ，框就是绝对定位的， `float` 的计算值为 `none` ， `display` 根据下面的表格进行调整。
- 否则， 如果 `float` 不是 `none` ，框是浮动的， `display` 根据下表进行调整
- 否则， 如果元素是根元素， `display` 根据下表进行调整
- 其他情况下 `display` 的值为指定值
- 总结起来：绝对定位、浮动、根元素都需要调整 **`display`**

7 清除浮动的几种方式，各自的优缺点

- 父级 `div` 定义 `height`
- 结尾处加空 `div` 标签 `clear:both`
- 父级 `div` 定义伪类 `:after` 和 `zoom`
- 父级 `div` 定义 `overflow:hidden`
- 父级 `div` 也浮动， 需要定义宽度
- 结尾处加 `br` 标签 `clear:both`

- 比较好的是第3种方式， 好多网站都这么用

8 为什么要初始化CSS样式?

- 因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的， 如果没对 CSS 初始化往往会出现浏览器之间的页面显示差异。
- 当然，初始化样式会对 SEO 有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化

9 css3有哪些新特性

- 新增各种 css 选择器
- 圆角 border-radius
- 多列布局
- 阴影和反射
- 文字特效 text-shadow
- 线性渐变
- 旋转 transform

CSS3新增伪类有那些?

- p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
- p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
- p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
- p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。
- p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。
- :after 在元素之前添加内容,也可以用来做清除浮动。
- :before 在元素之后添加内容。
- :enabled 已启用的表单元素。
- :disabled 已禁用的表单元素。
- :checked 单选框或复选框被选中。

10 display有哪些值？说明他们的作用

- block 转换成块状元素。
- inline 转换成行内元素。
- none 设置元素不可见。
- inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。
- list-item 象块类型元素一样显示， 并添加样式列表标记。

- `table` 此元素会作为块级表格来显示
- `inherit` 规定应该从父元素继承 `display` 属性的值

11 介绍一下标准的CSS的盒子模型？低版本IE的盒子模型有什么不同的?

- 有两种， `IE` 盒子模型 、 `W3C` 盒子模型;
- 盒模型： 内容(content)、填充(`padding`)、边界(`margin`)、 边框(`border`);
- 区 别： `IE` 的c `ontent` 部分把 `border` 和 `padding` 计算了进去;

12 CSS优先级算法如何计算?

- 优先级就近原则， 同权重情况下样式定义最近者为准
- 载入样式以最后载入的定位为准
- 优先级为: `!important > id > class > tag` ； `!important` 比 内联优先级高

13 对BFC规范的理解?

- 它决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用

14 谈谈浮动和清除浮动

- 浮动的框可以向左或向右移动， 直到他的外边缘碰到包含框或另一个浮动框的边框为止。

  由于浮动框不在文档的普通流中，所以文档的普通流的块框表现得就像浮动框不存在一样 。浮动的块框会漂浮在文档普通流的块框上

15 position的值， relative和absolute定位原点是

- `absolute` ： 生成绝对定位的元素，相对于 `static` 定位以外的第一个父元素进行定位
- `fixed` ： 生成绝对定位的元素，相对于浏览器窗口进行定位
- `relative` ： 生成相对定位的元素，相对于其正常位置进行定位
- `static` 默认值 。没有定位，元素出现在正常的流中
- `inherit` 规定从父元素继承 `position` 属性的值

16 display:inline-block 什么时候不会显示间隙？ (携程)

- 移除空格
- 使用 `margin` 负值