

HTML、HTTP和Web综合

4 从浏览器地址栏输入URL到显示页面的步骤

基础版本：

- 浏览器根据请求的URL交给DNS进行域名解析，找到真实的IP，然后向服务器发起请求
- 服务器交给后台处理完成后返回数据，浏览器接收文件
- 浏览器对加载到的资源进行语法解析，建立相应的内部数据结构
- 载入解析到的资源文件，渲染页面
- 首先浏览器主进程接管，开了一个下载线程。
- 然后进行HTTP请求（DNS查询、IP寻址等等），中间会有三次握手，等待响应，开始下载响应报文。
- 将下载完的内容转交给Renderer进程管理。
- Renderer进程开始解析css rule tree和dom tree，这两个过程是并行的，所以一般会把link标签放在页面顶部。
- 解析绘制过程中，当浏览器遇到link标签或者script、img等标签，浏览器会去下载这些内容，遇到时候缓存的使用缓存，不适用缓存的重新下载资源。
- css rule tree和dom tree生成完了之后，开始合成render tree，这个时候浏览器会进行layout，开始计算每一个节点的位置，然后进行绘制。
- 绘制结束后，关闭TCP连接，过程有四次挥手

详细简版：

1. 从浏览器接受URL，浏览器主进程接管，到开启网络请求线程
2. 开启网络线程到发出一个完整的HTTP请求
3. 从服务器接收到请求到后台接收到请求
4. 前后台HTTP交互
5. 单拎出来的缓存问题，HTTP的缓存
6. 浏览器接收到HTTP的数据包后的解析流程
7. CSS可视化格式模型
8. JS引擎解析过程
9. 其他（跨域，Web安全，hybrid模式）

6.HTTP状态码及其含义

- 1XX：信息状态码
- 2XX：成功状态码

- 3XX: 重定向
- 4XX: 客户端错误
- 5XX: 服务器错误

8.介绍一下你对浏览器内核的理解

- 浏览器内核主要分成两部分：渲染引擎和JS引擎
 - 渲染引擎：负责取得网页的内容（HTML，图像等等）和整理讯息（加入CSS），计算网页的显示方式，并输出至显示器和打印机
 - JS引擎：负责解析和执行JS从而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有哪些新的特性，移除了哪些元素？

- html5现在已经不是SGML的子集，主要是关于图像，位置，存储和多任务等功能的增加
 - 新增选择器：document.querySelector和document.querySelectorAll
 - 媒体播放：video和audio
 - 绘画: canvas
 - 本地存储: localStorage和sessionStorage
 - 离线应用: manifest
 - 语义化标签: article header footer nav section
 - 多任务: webworker
 - 全双工通信协议： websocket
 - 历史管理： history
 - 跨域资源共享(CORS): Access-Control-Allow-Origin
- 移除的元素：frame frameset noframes center big font tt
- 如何区分html和html5： DOCTYPE声明，新增的结构元素和功能元素

10.html5的离线存储怎么使用，工作原理能不能解释一下

- 当用户未连接因特网时，可以正常访问站点和应用；当用户连接因特网后，更新用户机器上的缓存文件
- **原理**：html5的离线存储时基于一个新建的.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，这些资源会像cookie一样被存储下来。之后当网络处于离线状态时，浏览器会通过离线存储的数据进行页面展示。
- **如何使用**：
 - 页面头部会加入一个manifest属性
 - 在cache.manifest文件中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

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

11.浏览器是怎么对HTML5的离线存储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**是网站为了表示用户身份而存储在用户本地终端上的数据，通常经过加密。并且cookie始终在同源的http请求中携带（即使不需要），即会在浏览器和服务端间来回传递。
- **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不会超过4k，而后者虽然也有限制，但可以达到5M或更大
- **过期时间**：
 - cookie在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭
 - sessionStorage数据在当前浏览器窗口关闭后自动删除
 - 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

13.iframe有哪些缺点？

- 会阻塞页面的onload事件
- 搜索引擎的检索程序无法解读这种页面，不利于SEO
- 与主页面共享连接池，会影响页面的加载
- 最好使用JS动态给iframe的src赋值，这样可以避开以上两个问题

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的区别

1. 功能上的差别：XHTML可以兼容各大浏览器、手机和PDA
2. 书写习惯的差别：XHTML必须被正确地嵌套、闭合、区分大小写，文档必须有根元素

16.DOCTYPE的作用是什么？严格模式与混合模式如何区分？它们有何意义？

- `<!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 不存在或者格式不正确会导致文档以混杂模式呈现

17.行内元素，块级元素分别有哪些？二者有什么区别？void元素有哪些？

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

21.如何在页面上实现一个圆形的可点击区域？

- SVG
- border-radius
- 纯JS

```
document.onclick(e){
    var r= 50;
    var x = y = 100;
    var x2=e.clientX,y2=e.clientY;
    ///半径50，圆心坐标(100,100)
    let distance = Math.abs(Math.sqrt(Math.pow(x2-x,2)+Math.pow(y2-y,2)));
    if(distance<50){
        console.log('in')
    }else{
        console.log('out')
    }
}
```

23.viewport

```
<meta name="viewport" content="width=device-width,initial-scale=1.0,minimum-scale=1.0,maximum-sc
// 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的问题

- 方案1: 针对ios, 使用 border:0.5px solid
 - 方案2: 使用边框图片, border-image:url(...) 2 repeat
 - 方案3: border-shadow:0 1px 1px 1px color
 - 方案4: 伪元素:

```
.setOnePx{
  position:relative;
  &::after{
    content:" ";
    position:absolute;
    display: block;
    top:0;
    left:0;
    width:100%;
    height:1px;
    transform:scale(1,0.5)
  }
}
```

- 局部处理:

- meta标签中设置viewport属性, initial-scale:1
 - rem按照设计稿标准走, 外加利用transform的scale(0.5)缩小一倍即可

- 全局处理:

- meta标签中设置viewport属性, initial-scale:0.5
 - rem按照设计稿标准走

24.渲染优化

- 禁止使用iframe

- iframe会阻塞父文档onload事件
 - 搜索引擎无法检索此种页面, 不利于SEO

- iframe与主页面共享连接池，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
- 如要使用，可以使用JS动态给src添加属性值，这样可以绕开以上两个问题。
- 使用CSS3代码替换JS动画：**尽可能的避免重排重绘和回流**
- 对于一些小图标，可以使用base64位编码，以减少网络请求
 - 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 <script><style> 标签会阻塞页面，因为Renderer进程中JS线程和渲染线程是互斥的
- 空的**href**和**src**会阻塞页面其他资源的加载。
- 使用innerHTML代替DOM操作，减少DOM操作次数，优化JS性能
- 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，脚本放底部加上时间戳，样式表放在顶部

27.div+css的布局较于table布局有什么优点？

- 改进的时候方便，只需要改css文件
- 页面加载速度快，结构化清晰，页面显示简洁
- 表现与结构分离
- 易于优化(SEO)，搜索引擎更友好

31.简述一下src和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**:

`<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

link与@import的区别：

- 1) link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。
- 2) link可以加载CSS，Javascript；@import只能加载CSS。
- 3) link加载的内容是与页面同时加载；@import需要页面网页完全载入以后加载。

4) 避免FOUC: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重新显示文档, 造成页面闪烁。----把样式表放到文档的head中

32.网页制作会用到的图片格式有哪些?

- 需要关注的是Webp和Apng两种新格式
- **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

CDN简介

简单的说, CDN是Content Delivery Network的简称, 即“内容分发网络”的意思。用户在浏览网站的时候, CDN会选择一个离用户最近的CDN边缘节点来响应用户的请求, 一般我们所说的CDN加速, 一般是指网站加速或者用户下载资源加速。

优势:

- 1.CDN节点解决了跨运营商跨地域访问的问题, 访问延时降低
- 2.起到了分流的作用, 减轻了源站的负载

不足:

当服务器资源更新, CDN节点的缓存未更新, 用户访问的就是过期资源。因此开发者需要手动刷新相关资源。

服务接入了CDN后, 数据会经过客户端 (浏览器) 缓存和CDN边缘节点缓存两个阶段

浏览器缓存

ETag,Cache-Control,Expires,Last-Modified

浏览器缓存的不足: 当服务器返回中存在Expires或者Cache-Control设置了max-age响应头的时候, 浏览器不会向服务器发起校验请求, 而是直接复用本地缓存。因此如果此时服务器的资源进行了更新, 那么用户就无法获取最新的资源。浏览器刷新Ctrl+F5

34.一个页面上有大量的图片, 加载很慢, 有哪些方法可以优化, 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。

35.常见排序算法的时间复杂度和空间复杂度

各种排序的比较				
排序方法	平均情况	最好情况	最坏情况	辅助空间
直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$
希尔排序	$O(n\log_2n) \sim O(n^2)$	$O(n^{1.3})$	$O(n^2)$	$O(1)$
起泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$
快速排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n^2)$	$O(\log_2n)$ $\sim O(n)$
简单选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
堆排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n\log_2n)$	$O(1)$
归并排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n\log_2n)$	$O(n)$

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URL，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,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35
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

```
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}
```

39.谈谈Cookie的弊端

- 每个特定的域名下最多生成20个cookie
- 不用浏览器有不同数量cookie的限制
- 如果cookie被拦截，就可以取得所有的session信息