浏览器面试题手册

在地址栏输入一个URL，到这个页面呈现出来，中间会发生什么？

完整回答：

1. 输入 url 后，首先需要找到这个 url 域名的服务器 ip，为了寻找到这个 ip，浏览器首先会寻找缓存，查看缓存中是否有记录，缓存的查找记录为：浏览器缓存 -> 系统缓存 -> 路由器缓存->IPS服务器缓存->根域名服务器缓存->顶级域名服务器缓存->主域名服务器缓存，缓存中没有则查找系统的 hosts 文件中是否有记录。
2. 如果没有则查询 DNS 服务器，DNS解析是一个递归的过程，首先在本地的域名服务器中查找，没找到会去根域名服务器查找，如果没有查找到的话回去 com 顶级域名服务器查找，如此类推下去，直到找到 IP 地址
3. 得到服务器的 ip 地址后，浏览器根据这个 ip 以及相应的端口号，构造一个 http 请求，这个请求报文会包括这次请求的信息，主要是请求方法，请求说明和请求附带的数据
4. 浏览器会将这个 http 请求封装在一个 tcp 包中，这个 tcp 包会依次经过传输层、网络层、数据链路层、物理层到达服务器，服务器解析这个请求来作出响应，返回对应的 html 给浏览器
5. 因为 html 是一个树形结构，浏览器根据这个 html 来构建 DOM 树，在 DOM 树的构建过程中如果遇到了 JS 脚本和外部 JS 链接，则会停止构建DOM树来执行和下载响应的代码，这会造成阻塞，这就是为什么推荐JS代码应该放在 html 代码后面。
6. 根据外部样式、内部样式、内联样式构建一个CSS对象模型树 CSSOM 树，构建完成后和 DOM 树合并为渲染树，这里主要做的是排除非视觉节点，比如 script、meta 标签和排除 display 为 none 的节点，之后进行布局，布局主要是确定各个元素的位置和尺寸，之后是渲染页面，因为 html 文件中会含有图片，视频，音频等资源，在解析 DOM 的过程中，遇到这些都会进行并行下载，浏览器对每个域的并行下载数量有一定的限制，一般是 4-6 个，当然在这些所有的请求中我们还需要关注的就是缓存，缓存一般通过 Cache-Crotrol、Last-Modify、Expires 等头部字段控制。Cache-Control 和 Expires 的区别在于 Cache-Control 使用相对时时间，Expires 使用的是基于服务器端的绝对时间，因为存在时间差问题，如果没有过期则直接使用本地缓存，过期则请求并在服务器校验文件是否修改，如果上一次响应设置了ETag值会在这次请求的时候作为 If-None-Match 的值交给服务器校验，如果一致，继续校验 Last-Modified，没有设置 Etag 则直接验证 Last-Modified，再决定是否返回304

简单回答：

- DNS解析
- TCP连接
- 发送HTTP请求

- 服务器处理请求并发送 HTTP 报文
- 浏览器解析渲染页面
- 连接结束

说一下跨域？怎样实现跨域？

跨域，是指浏览器不能执行其他网站的脚本，他是由浏览器的同源策略造成的，是浏览器对JavaScript实施的安全限制，那么只要协议、域名、端口有任何一个不同，都被当做是不同的域。跨域原理，即是通过各种方式，避开浏览器的安全限制

实现跨域：

- JSONP：通过动态创建script，再请求一个带参网址实现跨域通信
  - JSONP 只支持 get，因为 script 标签只能使用 get 请求
  - JSONP 需要后端配合返回指定格式的数据
- document.domain + iframe 跨域：两个页面都通过 js 强制设置，document.domain 为基础主域，就实现了同域
- location.hash + iframe 跨域：a 欲与 b 跨域相互通信，通过中间页 c 来实现。三个页面，不同域之间利用 iframe 的 location.hash 传值，相同域之间直接 js 访问来通信
- window.name + iframe 跨域：通过 iframe 的 src 属性由外域转向本地域，跨域数据即由 iframe 的 window.name 从外域传递到本地域
- postMessage 跨域：可以跨域操作的 window 属性之一
- CORS：服务端设置 Access-Control-Allow-Origin 既可，前端无须设置，若要带 cookie 请求，前后端都需要设置
- 代理跨域：启动一个代理服务器，实现数据的转发

参考 [前端常见跨域解决方案（全）]

Cookie、sessionStorage、localStorage 的差异?

第一种回答：

共同点：都是保存在浏览器端，并且是同源的

不同点：

- cookie 数据始终在同源的 http 请求中携带（即使不需要），即 cookie 在浏览器和服务器间来回传递。而 sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存。
- cookie 数据还有路径（path）的概念，可以限制 cookie 只属于某个路径下，存储的大小很小，只有 4k 左右。（注：可以在浏览器和服务器端来回传递，存储容量小，只有大约4k左右）
- sessionStorage 仅在当前浏览器窗口关闭有效，自然也就不可能持久保持；localStorage 始终有

效，窗口或浏览器关闭也会一直保存，因此用作持久数据； cookie 只在设置的 cookie 过期时间之前一直有效，即使窗口或浏览器关闭。（注：本身就是一个会话过程，关闭浏览器后消失，session 为一个会话，当页面不同即使是同一页面打开两次，也被视为同一次会话）

- localStorage 在所有同源窗口中都是共享的；cookie 也是在所有同源窗口中都是共享的。（注：同源窗口都会共享，并且不会失效，不管窗口或者浏览器关闭与否都会始终生效）

第二种回答：

- 生命周期：
  - cookie 可设置失效时间，否则默认为关闭浏览器后失效
  - localStorage 除非手动清除，否则永久保存
  - sessionStorage 仅在当前页面会话下有效，关闭页面或浏览器就会被清楚
- 存放数据：
  - Cookie 的大小为4K左右
  - localStorage 和 sessionStorage 可以保存 5M的信息
- http 请求：
  - Cookie 每次都会携带在 http 请求头中，如果使用 cookie 保存过多数据会带来性能问题
  - localStorage 和 sessionStorage 仅在客户端即浏览器中保存，不参与和服务器的通信
- 易用性：
  - Cookie 需要程序员自己封装，原生的cookie接口不友好
  - localStorage 和 sessionStorage 采用原生接口即可，也可以再次封装
- 应用场景：
  - 从安全性来说，因为每次 http 请求都会携带 cookie 信息，这样子浪费了带宽，所以 cookie 应该尽可能地少用，此外 cookie 还需要指定作用域，不可以跨域调用，限制很多，但是用户识别用户登录来说，cookie 还是比 storage 好用，其他情况下可以用 storage，localStorage 可以用来在页面传递参数，sessionStorage 可以用来保存一些临时的数据，防止用户刷新页面丢失一些参数

cookie 和 session 的区别?

回答一：

cookie 和 session 都可以用来存储用户信息，cookie 存放于客户端，session 存放于服务器端，因为 cookie 存放于客户端有可能被窃取，所以 cookie 一般用来存放不敏感信息，比如用户设置的网站主题。敏感信息用 session 存储，比如用户的登录信息，session 可以存放于文件、数据库、内存中都可以，cookie 可以在服务器端响应的时候设置，也可以客户端通过 JS 设置，cookie 会在请求时在 http 头部发送给客户端，cookie 一般都在客户端有大小限制，一般为 4K

回答二：

- cookie 数据存放在客户的浏览器上，session 数据放在服务器上
- cookie 不是很安全，别人可以分析存放在本地的 cookie 并进行 cookie 欺骗，考虑到安全应当使用 session
- session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用 cookie
- 单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存20个cookie
- 将登陆信息等重要信息存放为 session，其他信息如果需要保留，可以放在cookie中

Cookie 有什么作用?

- 保存用户登录状态：例如将用户id存储于一个cookie内，这样当用户下次访问该页面时就不需要重新登录了，现在很多论坛和社区都提供这样的功能。cookie 还可以设置过期时间，当超过时间期限后，cookie 就会自动消失。因此，系统往往可以提示用户保持登录状态的时间：常见选项有一个月、三个月、一年等
- 跟踪用户行为：例如一个天气预报网站，能够根据用户选择的地区显示当地的天气情况。如果每次都需要选择所在地是繁琐的，当利用了 cookie 后就会显得很人性化了，系统能够记住上一次访问的地区，当下次再打开该页面时，他就会自动显示上次用户所在地区的天气情况。因为一切都是在后台完成，所以这样的页面就像为某个用户所定制的一样，使用起来非常方便定制页面。如果网站提供了换肤会更改布局的功能，那么可以使用 cookie 来记录用户的选项，例如：背景色、分辨率等。当用户下次访问时，仍然可以保存上一次访问的界面风格

请你谈谈Cookie的弊端，有什么优缺点?

- 每个特定的域名夏最多只能生成20个cookie
  - IE6或更低版本最多20个cookie
  - IE7和之后的版本可以有50个cookie
  - Firefox 最多50个cookie
  - Chrome 或 Safari 没有硬性限制
  - IE 和 Opera 会清理近期最少使用的cookie，Firefox 会随机清理cookie
  - cookie 的最大值大约为 4096 字节，为了兼容性，一般不能超过 4095 字节
  - IE 提供了一种存储可以持久化用户数据，叫做 userdata，从 IE5 就开始支持。每个数据最多128K，每个域名下最多1M。这个持久化数据放在缓存中，如果缓存没有清理，那么会一直存在
- 优点：极高的扩展性和可用性
  - 通过良好的编程，控制保存在 cookie 的 session 对象的大小
  - 通过加密和安全传输技术（SSL），减少 cookie 被破解的可能性

    - 只在 cookie 中存放不敏感数据，即使被盗也不会有重大损失
    - 控制 cookie 的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的 cookie
- 缺点：
    - Cookie 数量和长度的限制。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉
    - 安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了
    - .有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用

跨域时如何处理cookie

浏览器的性能监控你是怎么做的?

[如何进行 web 性能监控]

浏览器是怎么对页面进行渲染，是怎么渲染的？DOM树和CSSOM树是怎么构建的?

- 构建DOM树，将浏览器无法直接理解和使用的HTML，转换为浏览器能够理解的结构——DOM树
- 构建CSSOM树，把 CSS 转换为浏览器能理解的结构（styleSheets），并转换样式表中的属性，使其标准化，计算出 DOM 树中每个节点的具体样式（根据继承规则和层叠规则）
- 创建 Layout 布局树，确定DOM 元素的几何位置信息，遍历 DOM 树中的所有可见节点，加入到布局树（display：none 不包含），并计算布局树节点的坐标位置
- 构建图层树，如果页面有复杂的效果，如常见的页面滚动，或者使用z轴排序等，为了更加方便地实现这些效果，渲染引擎还需要为特定的节点生成专用的图层，并生成一棵对应的图层树（Layer Tree）
- Paint 图层绘制，把一个图层的绘制拆分成很多小的绘制指令，然后再把这些指令按照顺序组成一个待绘制列表
- tiles：将图层转换成图块
- 光栅化：通过进程实现图块转换成位图
- display：浏览器进程拿到 DrawQuad 信息生成页面显示

前端需要注意哪些SEO

- 合理的title、description、keywords：搜索对着三项的权重逐个减小，title 值强调重点即可，重要

关键词出现不要超过2次，而且要靠前，不同页面title要有所不同；description 把页面内容高度概括，长度合适，不可过分堆砌关键词，不同页面description有所不同；keywords 列举初重要关键词即可

- 语义化的HTML代码，符合W3C规范：语义化代码让搜索引擎容易理解网页
- 重要内容 HTML 代码放在最前：搜索引擎抓取 HTML 顺序是从上而下，有的搜索引擎对抓取长度有限制，保证重要内容一定会被抓取
- 重要内容不要用js输出：爬虫不会执行js获取内容
- 少用iframe（搜索引擎不会抓取iframe中的内容）
- 非装饰性图片必须加alt
- 提高网站速度（网站速度是搜索引擎排序的一个重要指标）

XML 和 JSON 的区别?

- 数据体积方面：JSON相对于XML来讲，数据的体积小，传递的速度更快些
- 数据交互方面：JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互
- 数据描述方面：JSON 对数据的描述性比XML较差
- 传输速度方面：JSON的速度要远远快于XML