从输入 url 到页面加载完成发生了什么？一一前端角度

这是一道经典的面试题，这道面试题不光前端面试会问到，后端面试也会被问到。这道题没有一个标准的答案，它涉及很多的知识点，面试官会通过这道题了解你对哪一方面的知识比较擅长，然后继续追问看看你的掌握程度。当然我写的这些也只是我的一些简单的理解，从前端的角度出发，我觉得首先回答必须包括几个基本的点，然后在根据你的理解深入回答。

1、浏览器的地址栏输入 URL 并按下回车。

2、浏览器查找当前 URL 是否存在缓存，并比较缓存是否过期。

3、DNS 解析 URL 对应的 IP。

4、根据 IP 建立 TCP 连接（三次握手）。

5、HTTP 发起请求。

6、服务器处理请求，浏览器接收 HTTP 响应。

7、渲染页面，构建 DOM 树。

8、关闭 TCP 连接（四次挥手）。

说完整个过程的几个关键点后我们再来展开的说一下。

一、URL

我们常见的 RUL 是这样的:http://www.baidu.com,这个域名由三部分组成：协议名、域名、端口号，这里端口是默认所以隐藏。除此之外 URL 还会包含一些路径、查询和其他片段，例如：http://www.tuicool.com/search?kw=%E4%。我们最常见的的协议是 HTTP 协议，除此之外还有加密的 HTTPS 协议、FTP 协议、FILe 协议等等。URL 的中间部分为域名或者是 IP，之后就是端口号了。通常端口号不常见是因为大部分的都是使用默认端口，如 HTTP 默认端口 80，HTTPS 默认端口 443。说到这里可能有的面试官会问你同源策略，以及更深层次的跨域的问题，我今天就不在这里展开了。

二、缓存

说完 URL 我们说说**浏览器缓存**,HTTP 缓存有多种规则，根据是否需要重新向服务器发起请求来分类，我将其分为强制缓存，对比缓存。

强制缓存判断 HTTP 首部字段：cache-control，Expires。

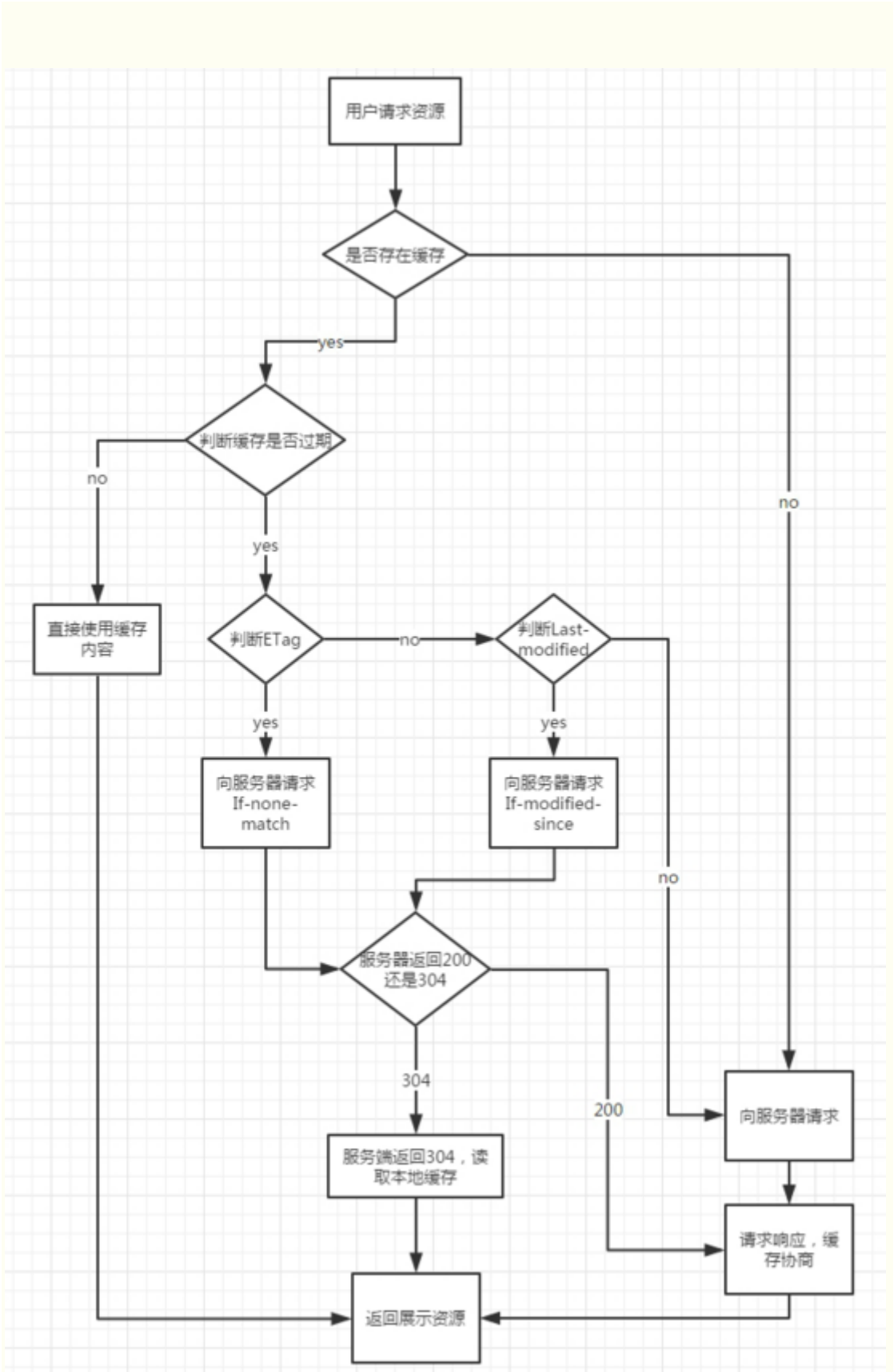
Expires 是一个绝对时间，即服务器时间。浏览器检查当前时间，如果还没到失效时间就直接使用缓存文件。但是该方法存在一个问题：服务器时间与客户端时间可能不一致。因此该字段已经很少使用。

cache-control 中的 max-age 保存一个相对时间。例如 Cache-Control: max-age = 484200，表示浏览器收到文件后，缓存在 484200s 内均有效。 如果同时存在 cache-control 和 Expires，浏览器总是优先使用 cache-control。

对比缓存通过 HTTP 的 last-modified，Etag 字段进行判断。

last-modified 是第一次请求资源时，服务器返回的字段，表示最后一次更新的时间。下一次浏览器请求资源时就发送 if-modified-since 字段。服务器用本地 Last-modified 时间与 if-modified-since 时间比较，如果不一致则认为缓存已过期并返回新资源给浏览器；如果时间一致则发送 304 状态码，让浏览器继续使用缓存。

Etag：资源的实体标识（哈希字符串），当资源内容更新时，Etag 会改变。服务器会判断 Etag 是否发生变化，如果变化则返回新资源，否则返回 304。


用户请求资源
是否存在缓存
yes
no
判断缓存是否过期
no
yes
直接使用缓存内容
判断ETag
no
判断Last-modified
yes
yes
向服务器请求If-none-match
向服务器请求If-modified-since
no
服务器返回200还是304
304
200
服务端返回304，读取本地缓存
向服务器请求
请求响应，缓存协商
返回展示资源

三、DNS 域名解析

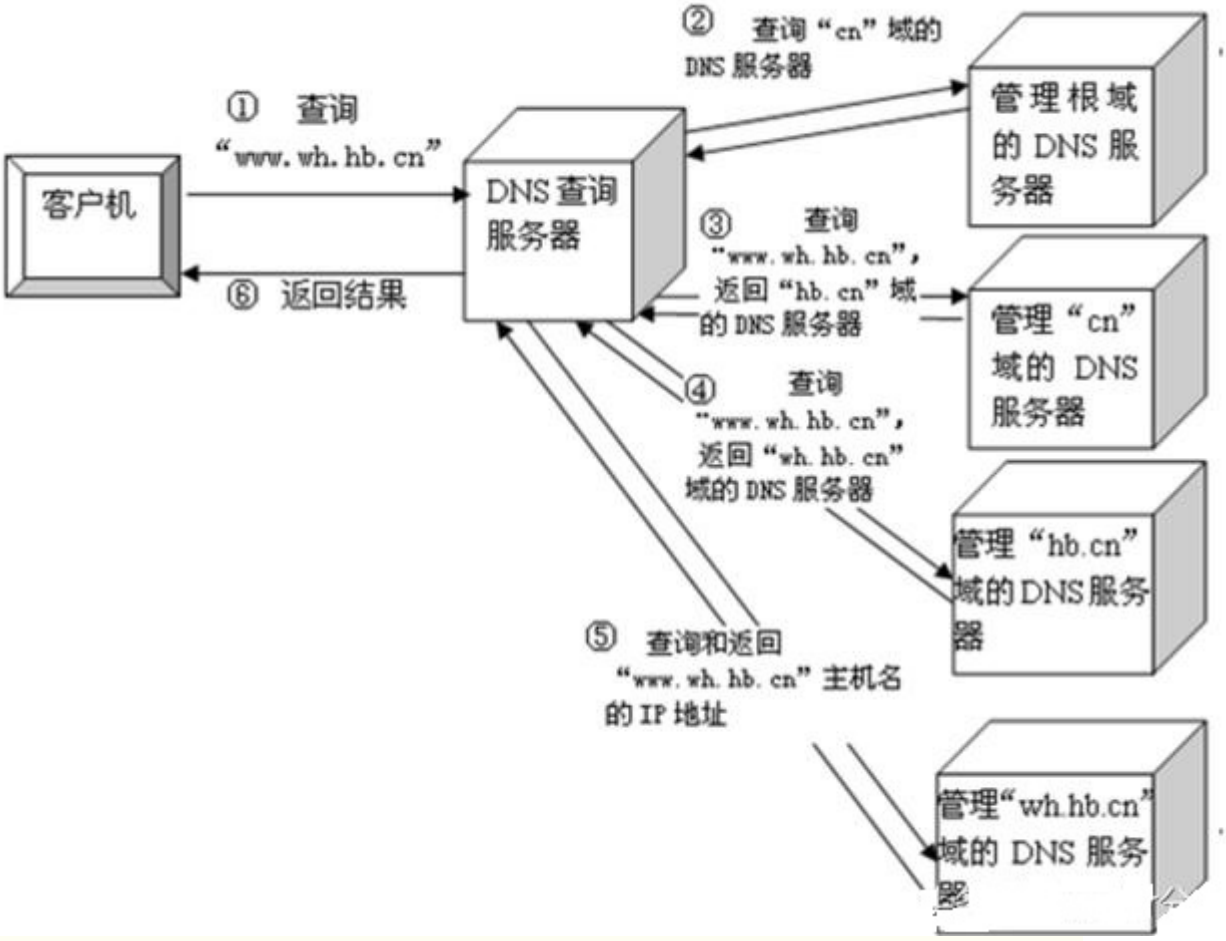
我们知道在地址栏输入的域名并不是最后资源所在的真实位置，域名只是与 IP 地址的一个映射。网络服务器的 IP 地址那么多，我们不可能去记一串串的数字，因此域名就产生了，域名解析的过程实际是将域名还原为 IP 地址的过程。

首先浏览器先检查本地 hosts 文件是否有这个网址映射关系，如果有就调用这个 IP 地址映射，完成域名解析。

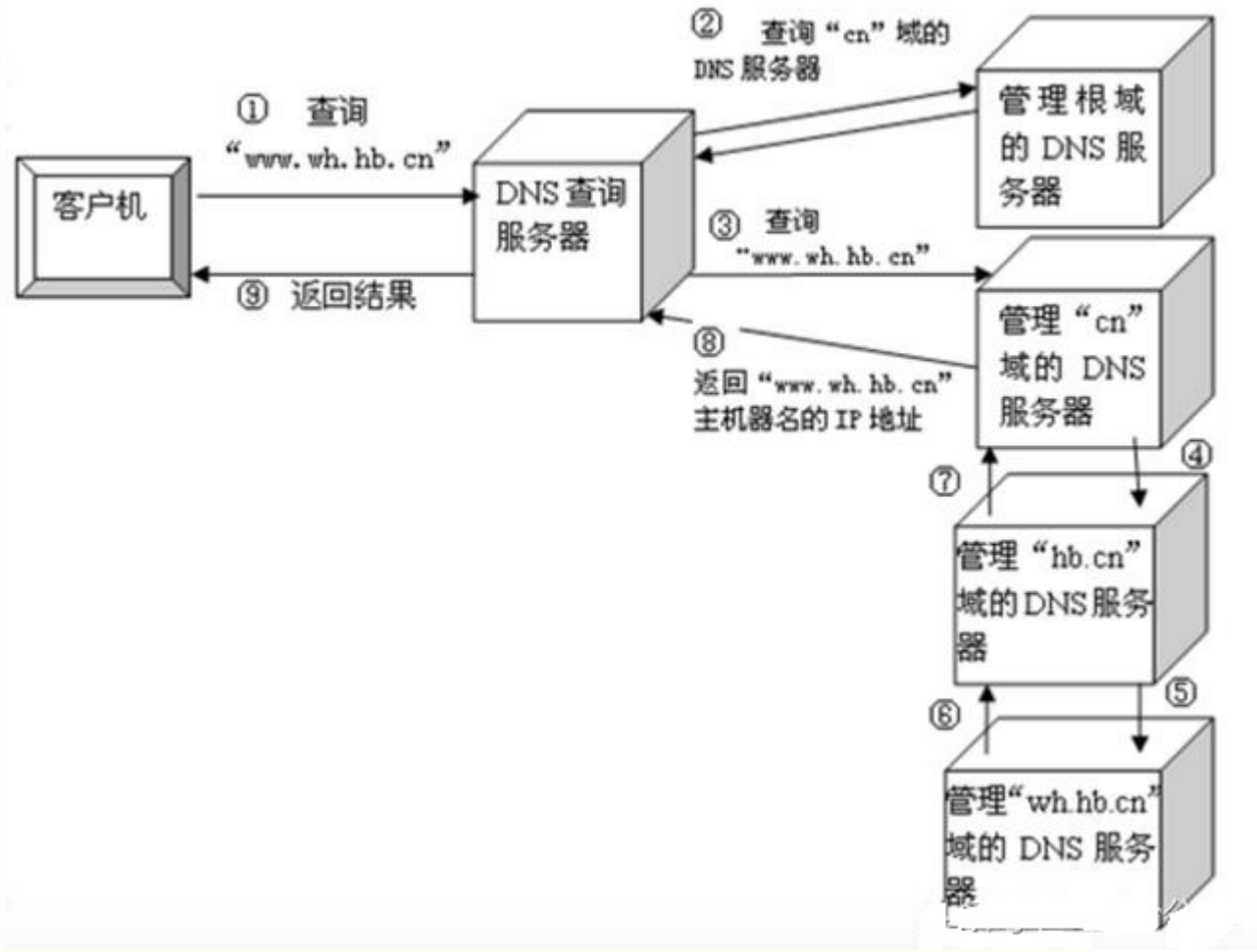
如果没找到则会查找本地 DNS 解析器缓存，如果查找到则返回。

如果还是没有找到则会查找本地 DNS 服务器，如果查找到则返回。

最后迭代查询，按根域服务器 ->顶级域,.cn->第二层域，hb.cn ->子域，www.hb.cn 的顺序找到 IP 地址。



递归查询，按上一级 DNS 服务器->上上级->....逐级向上查询找到 IP 地址。



四、TCP 连接

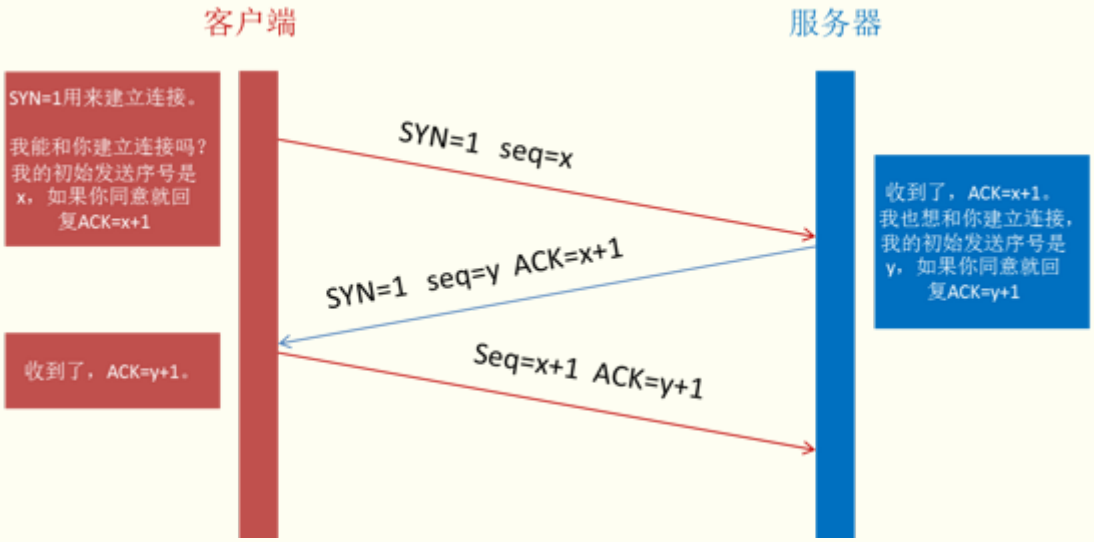
在通过第一步的 DNS 域名解析后，获取到了服务器的 IP 地址，在获取到 IP 地址后，便会开始建立一次连接，这是由 TCP 协议完成的，主要通过三次握手进行连接。

第一次握手： 建立连接时，客户端发送 syn 包（syn=j）到服务器，并进入 SYN_SENT 状态，等待服务器确认；

第二次握手： 服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；

第三次握手： 客户端收到服务器的 SYN+ACK 包，向服务器发送确认包 ACK(ack=k+1），此包发送完毕，客户端和服务器进入 ESTABLISHED（TCP 连接成功）状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据。



五、浏览器向服务器发送 HTTP 请求

完整的 HTTP 请求包含请求起始行、请求头部、请求主体三部分。

▼ General
Request URL: https://www.google-analytics.com/analytics.js
Request Method: GET
Status Code: 200 (from disk cache)
Remote Address: 203.208.51.68:443

▼ Response Headers
age: 559
alt-svc: quic=":443"; ma=2592000; v="37,36,35"
cache-control: public, max-age=7200
content-encoding: gzip
content-length: 12156
content-type: text/javascript
date: Wed, 29 Mar 2017 05:34:51 GMT
expires: Wed, 29 Mar 2017 07:34:51 GMT
last-modified: Sat, 18 Mar 2017 01:34:54 GMT
server: Golfe2
status: 304
timing-allow-origin: *
vary: Accept-Encoding
x-content-type-options: nosniff

▼ Request Headers
⚠ Provisional headers are shown
Referer: http://www.cnblogs.com/yuteng/articles/1904215.html
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chro

六、浏览器接收响应

服务器在收到浏览器发送的 HTTP 请求之后，会将收到的 HTTP 报文封装成 HTTP 的 Request 对象，并通过不同的 Web 服务器进行处理，处理完的结果以 HTTP 的 Response 对象返回，主要包括状态码，响应头，响应报文三个部分。

状态码主要包括以下部分

1xx：指示信息–表示请求已接收，继续处理。

2xx：成功–表示请求已被成功接收、理解、接受。

3xx：重定向–要完成请求必须进行更进一步的操作。

4xx：客户端错误–请求有语法错误或请求无法实现。

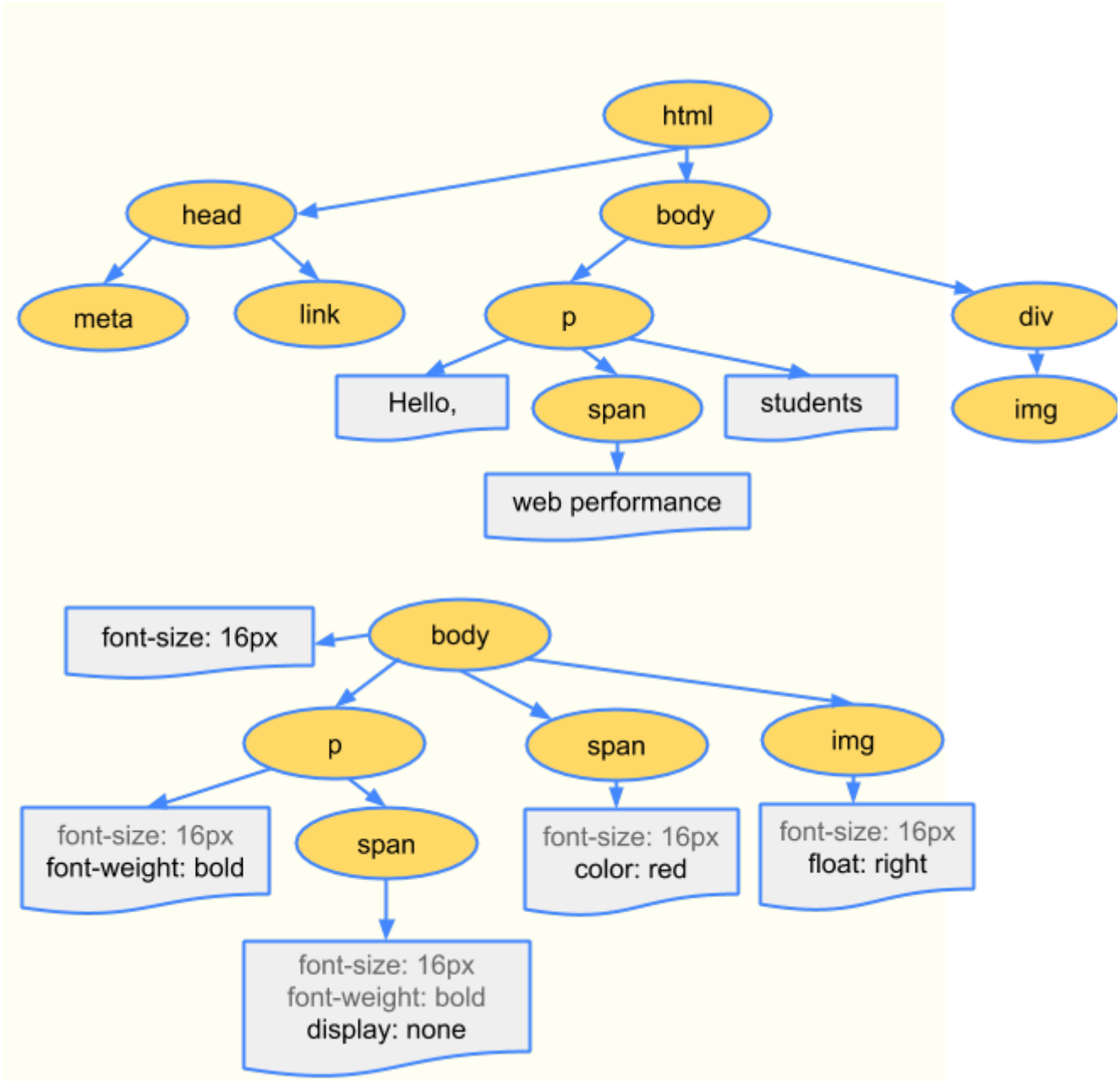
5xx：服务器端错误–服务器未能实现合法的请求。

响应头主要由 Cache-Control、 Connection、Date、Pragma 等组成。

响应体为服务器返回给浏览器的信息，主要由 HTML，css，js，图片文件组成。

七、页面渲染

如果说响应的内容是 HTML 文档的话，就需要浏览器进行解析渲染呈现给用户。整个过程涉及两个方面：解析和渲染。在渲染页面之前，需要构建 DOM 树和 CSSOM 树。



在浏览器还没接收到完整的 HTML 文件时，它就开始渲染页面了，在遇到外部链入的脚本标签或样式标签或图片时，会再次发送 HTTP 请求重复上述的步骤。在收到 CSS 文件后会对已经渲染的页面重新渲染，加入它们应有的样式，图片文件加载完立刻显示在相应位置。在这一过程中可能会触发页面的重绘或重排。这里就涉及了两个重要概念：Reflow 和 Repaint。

Reflow，也称作 Layout，中文叫回流，一般意味着元素的内容、结构、位置或尺寸发生了变化，需要重新计算样式和渲染树，这个过程称为 Reflow。

Repaint，中文重绘，意味着元素发生的改变只是影响了元素的一些外观之类的时候（例如，背景色，边框颜色，文字颜色等），此时只需要应用新样式绘制这个元素就 OK 了，这个过程称为 Repaint。

所以说 Reflow 的成本比 Repaint 的成本高得多的多。DOM 树里的每个结点都会有 reflow 方法，一个结点的 reflow 很有可能导致子结点，甚至父点以及同级结点的 reflow。

下面这些动作有很大可能会是成本比较高的：

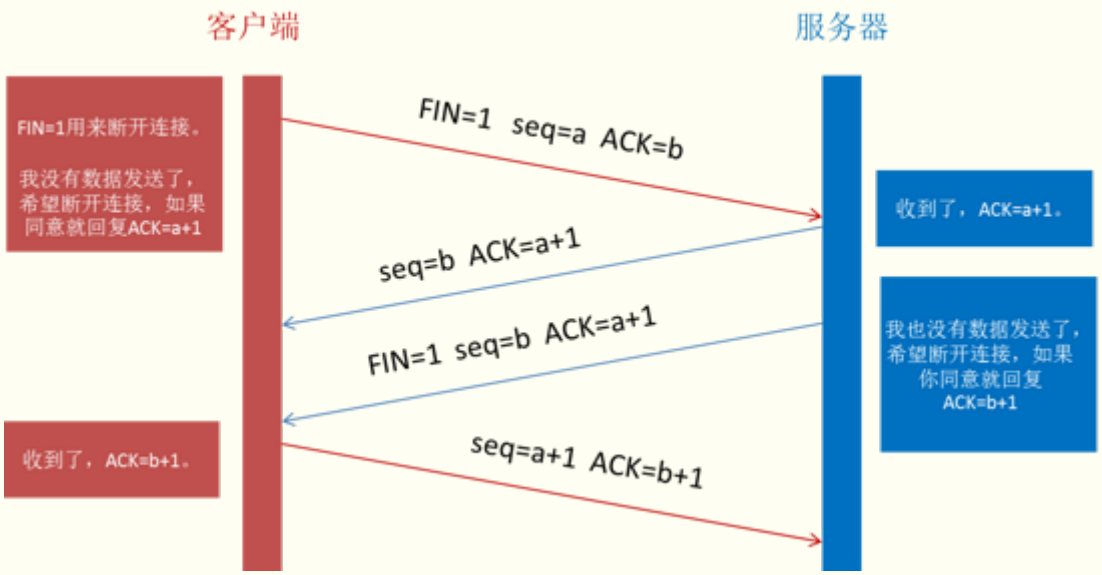
- 增加、删除、修改 DOM 结点时，会导致 Reflow 或 Repaint
- 移动 DOM 的位置，或是搞个动画的时候
- 内容发生变化
- 修改 CSS 样式的时候
- Resize 窗口的时候（移动端没有这个问题），或是滚动的时候
- 修改网页的默认字体时

基本上来说，reflow 有如下的几个原因：

- Initial，网页初始化的时候
- Incremental，一些 js 在操作 DOM 树时
- Resize，其些元件的尺寸变了
- StyleChange，如果 CSS 的属性发生变化了
- Dirty，几个 Incremental 的 reflow 发生在同一个 frame 的子树上

八、关闭 TCP 连接或继续保持连接

通过四次挥手关闭连接(FIN ACK, ACK, FIN ACK, ACK)。



第一次挥手是浏览器发完数据后，发送 FIN 请求断开连接。

第二次挥手是服务器发送 ACK 表示同意，如果在这一次服务器也发送 FIN 请求断开连接似乎也没有不妥，但考虑到服务器可能还有数据要发送，所以服务器发送 FIN 应该放在第三次挥手中。

这样浏览器需要返回 ACK 表示同意，也就是第四次挥手。

至此从浏览器地址栏输入 URL 到页面呈现到你面前的整个过程就分析完了，上面内容如有错误欢迎留言交流。

----------转自（***希望star***）