

输入 URL 到页面渲染的整个流程

之前我们学了那么多章节的内容，是时候找个时间将它们再次复习消化了。就借用这道经典面试题，将之前学习到的浏览器以及网络几章节的知识联系起来。

首先是 DNS 查询，如果这一步做了智能 DNS 解析的话，会提供访问速度最快的 IP 地址回来，这部分的内容之前没有写过，所以就在这里讲解下。

DNS

DNS 的作用就是通过域名查询到具体的 IP。

因为 IP 存在数字和英文的组合（IPv6），很不利于人类记忆，所以就出现了域名。你可以把域名看成是某个 IP 的别名，DNS 就是去查询这个别名的真正名称是什么。

在 TCP 握手之前就已经进行了 DNS 查询，这个查询是操作系统自己做的。当你在浏览器中想访问 `www.google.com` 时，会进行一下操作：

1. 操作系统会首先在本地缓存中查询 IP
2. 没有的话会去系统配置的 DNS 服务器中查询
3. 如果这时候还没得话，会直接去 DNS 根服务器查询，这一步查询会找出负责 `com` 这个一级域名的服务器
4. 然后去该服务器查询 `google` 这个二级域名
5. 接下来三级域名的查询其实是我们配置的，你可以给 `www` 这个域名配置一个 IP，然后还可以给别的三级域名配置一个 IP

以上介绍的是 DNS 迭代查询，还有种是递归查询，区别就是前者是由客户端去做请求，后者是由系统配置的 DNS 服务器做请求，得到结果后将数据返回给客户端。

PS: DNS 是基于 UDP 做的查询, 大家也可以考虑下为什么之前不考虑使用 TCP 去实现。

接下来是 TCP 握手, 应用层会下发数据给传输层, 这里 TCP 协议会指明两端的端口号, 然后下发给网络层。网络层中的 IP 协议会确定 IP 地址, 并且指示了数据传输中如何跳转路由器。然后包会再被封装到数据链路层的数据帧结构中, 最后就是物理层面的传输了。

在这一部分中, 可以详细说下 TCP 的握手情况以及 TCP 的一些特性。

当 TCP 握手结束后就会进行 TLS 握手, 然后就开始正式的传输数据。

在这一部分中, 可以详细说下 TLS 的握手情况以及两种加密方式的内容。

数据在进入服务端之前, 可能还会先经过负责负载均衡的服务器, 它的作用就是将请求合理的分发到多台服务器上, 这时假设服务端会响应一个 HTML 文件。

首先浏览器会判断状态码是什么, 如果是 200 那就继续解析, 如果 400 或 500 的话就会报错, 如果 300 的话会进行重定向, 这里会有个重定向计数器, 避免过多次的重定向, 超过次数也会报错。

浏览器开始解析文件, 如果是 gzip 格式的话会先解压一下, 然后通过文件的编码格式知道该如何去解码文件。

文件解码成功后会正式开始渲染流程, 先会根据 HTML 构建 DOM 树, 有 CSS 的话会去构建 CSSOM 树。如果遇到 script 标签的话, 会判断是否存在 async 或者 defer, 前者会并行进行下载并执行 JS, 后者会先下载文件, 然后等待 HTML 解析完成后顺序执行。

如果以上都没有，就会阻塞住渲染流程直到 JS 执行完毕。遇到文件下载的会去下载文件，这里如果使用 HTTP/2 协议的话会极大的提高多图的下载效率。

CSSOM 树和 DOM 树构建完成后会开始生成 Render 树，这一步就是确定页面元素的布局、样式等等诸多方面的东西

在生成 Render 树的过程中，浏览器就开始调用 GPU 绘制，合成图层，将内容显示在屏幕上了。

这一部分就是渲染原理中讲解到的内容，可以详细的说明下这一过程。并且在下载文件时，也可以说下通过 HTTP/2 协议可以解决队头阻塞的问题。

总的来说这一章节就是带着大家从 DNS 查询开始到渲染出画面完整的了解一遍过程，将之前学习到的内容连接起来。

当来这一过程远远不止这些内容，但是对于大部分人能答出这些内容已经很不错了，你如果想了解更加详细的过程，可以阅读这篇[文章](https://github.com/alex/what-happens-when) (<https://github.com/alex/what-happens-when>)。