**浏览器页面加载解析渲染机制**

<https://blog.csdn.net/baidu_30668495/article/details/83069761>

<https://blog.csdn.net/weixin_38080573/article/details/79440400>

<https://www.jianshu.com/p/516900e2c06c>

<https://baijiahao.baidu.com/s?id=1628316443220392498&wfr=spider&for=pc>

**一、为什么要了解浏览器加载、解析、渲染这个过程？**

了解浏览器如何进行加载，我们可以在引用外部样式文件，外部js时，将他们放到合适的位置，使浏览器以最快的速度将文件加载完毕。

了解浏览器如何进行解析，我们可以在构建DOM结构，组织css选择器时，选择最优的写法，提高浏览器的解析速率。

了解浏览器如何进行渲染，明白渲染的过程，我们在设置元素属性，编写js文件时，可以减少”重绘“”重新布局“的消耗。

这三个过程在实际进行的时候又不是完全独立，而是会有交叉。会造成一边加载，一边解析，一边渲染的工作现象。

**二、浏览器是如何进行加载、解析、渲染的呢？**

1.用户访问网页，DNS服务器（域名解析系统）会根据用户提供的域名查找对应的IP地址，找到后，系统会向对应IP地址的网络服务器发送一个http请求。

2.网络服务器解析请求，并发送请求给数据库服务器。

3.数据库服务器将请求的资源返回给网络服务器，网络服务器解析数据，并生成html文件，放入http response中，返回给浏览器。

4.浏览器解析 http response。

5.浏览器解析 http response后，需要下载html文件，以及html文件内包含的外部引用文件，及文件内涉及的图片或者多媒体文件。  
　解析html 构建dom树 -> 构建render树 -> 布局render树 -> 绘制render树 ：

**解析：**

　　1、将HTML构建成一个DOM树（DOM = Document Object Model 文档对象模型），DOM 树的构建过程是一个深度遍历过程：当前节点的所有子节点都构建好后才会去构建当前节点的下一个兄弟节点。

　　2、将CSS解析成CSS去构造CSSOM树( CSSOM = CSS Object Model CSS对象模型)

　　3、根据DOM树和CSSOM来构造 Rendering Tree（渲染树）。注意：Rendering Tree 渲染树并不等同于 DOM 树，因为一些像 Header 或 display:none 的东西就没必要放在渲染树中了。

　　4.有了Render Tree，浏览器已经能知道网页中有哪些节点、各个节点的CSS定义以及他们的从属关系。

　　5.下一步操作称之为Layout，顾名思义就是计算出每个节点在屏幕中的位置 layout render tree。

6.再下一步就是绘制，即遍历render树，并使用浏览器UI后端层绘制每个节点。

**性能优化中重绘、重排：**

（1）Reflow（回流/重排）：当它发现了某个部分发生了变化影响了布局，渲染树需要重新计算。

（2）Repaint（重绘）：改变了某个元素的背景颜色，文字颜色等，不影响元素周围或内部布局的属性，将只会引起浏览器的repaint，根据元素的新属性重新绘制，使元素呈现新的外观。重绘不会带来重新布局，并不一定伴随重排；

Reflow要比Repaint更花费时间，也就更影响性能。所以在写代码的时候，要尽量避免过多的Reflow。

reflow的原因：

（1）页面初始化的时候；

（2）操作DOM时；

（3）某些元素的尺寸变了；

（4）如果 CSS 的属性发生变化了。

减少 reflow/repaint

　（1）不要一条一条地修改 DOM 的样式。与其这样，还不如预先定义好 css 的 class，然后修改 DOM 的 className。

　（2）不要把 DOM 结点的属性值放在一个循环里当成循环里的变量。

　（3）为动画的 HTML 元件使用 fixed 或 absoult 的 position，那么修改他们的 CSS 是不会 reflow 的。

　（4）千万不要使用 table 布局。因为可能很小的一个小改动会造成整个 table 的重新布局。

# Http协议常见面试题

<https://blog.csdn.net/weixin_34380948/article/details/88715623>