### 浅谈前端（1）（DOM与css与html与js）

——————20190806韵方

（此篇文章非原创的篇幅较大，因为本人也是初学，以下内容可能来自各种网站）

首先我们来说说历史。

1990年代后期，微软和Netscape的浏览器大战，双方为在javaScript和Jscript一决生死，大规模的给浏览器增加强大的功能。导致许多功能除了微软平台和浏览器都不能正常的显示。于是万维网联盟就出来了，规定了一个统一的标准，结束了这场浏览器的大战，使得同一个网站在不同的浏览器中也可以显示，这就是DOM的由来。

那么，什么是DOM？说的抽象一点，他是万维网的标准，说的简单一点，你可以把他看成一棵树，不同的浏览器拿到这棵树会将它翻译成html，这棵树的样子如下图（图来自W3school）：



如果你学过html，那么当你看到这棵树的时候就基本意会到DOM是什么了，那么新的问题又来了，我们说的网页三要素——html／css／js和DOM到底有什么联系呢。

DOM对象的出现是为了方便javascript操作HTML标签，标准的HTML文档实质上就是一个是就一个树形结构的XML文档。DOM全名document object model 文档对象模型，也就是基于HTML文档出来的，所以操作DOM实际上就是在操作HTML文档对象(每一个标签).。CSS只能修饰HTML的样式而已，而javascript能把HTML与CSS相结合，也就是以前所说的DHTML（动态HTML）。在网上有这么一句话用来形容html，css，js和dom的关系我觉得比较恰当：



有了以上的理论基础，我们就可以开始尝试着去解释不同浏览器兼容性的问题，为什么有的css在这个浏览器可以在那个浏览器却不行，而它最根本的原因，就是浏览器的渲染过程不同，有的浏览器支持这样渲染而有的浏览器却不支持。

说到这里，一个很常见的词就出现在了大家的眼中，渲染。相比写过前端的或者学习计算机的都或多或少的听别人说过这个词，再解释这个词之前，我需要再引进一个词，解析。

DOM解析：就是把你所写的各种html标签，生成一个DOM TREE，可以认为就是生成了一个最原始的页面，一点样式都没有，毫无CSS修饰的那种；

DOM渲染：浏览器会把本身默认的样式+用户自己写得样式整合到一起，形成一个CSS TREE，而DOM渲染就是指DOM TREE 和 CSS TREE 结合到一起，生成一个Render TREE，呈现出一个带有样式的页面。

有了以上的概念，我们就可以来了解html整个加载的过程了

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。



看到以上的内容，大家对dom和css以及html的关系应该比较了解了，那么大家是否有新的疑问，那就是dom和js到底是什么关系呢？

在上面的比喻中，js被看作是编写dom的笔，那么js的作用就如同这只笔一样，他是用来操作dom的。比如dom是对象那么js就是语言，语言就是用啦操作对象的嘛。有了这个关系，就又带出了新的问题，既然js是用来操纵dom的，那么在dom渲染和解析的时候，js是如何被下载进去的呢，他的执行顺序是什么样的呢？

讲到这里就要讲讲计算机基础的执行单位，线程了，线程的概念就不与大家普及了，不知道的出门百度哈。整个html加载过程可以被看成是dom解析过程，dom渲染过程，css加载过程，js加载过程四个过程。html中每遇到< script >标签，页面就会重新渲染一次，因为要保证标签中的JS代码拿到的都是最新的样式。所以我们可以很容易的得出以下的结论：

1. JS的加载和执行会阻塞DOM的解析
2. JS的加载和执行会阻塞DOM的渲染

也就是说，遇见js加载时，所有的DOM的工作都会停下来，从这个结论出发，我们大概也就理解了为什么js会被放在最后body标签之前，因为他要保证js加载时dom的树渲染已经完成。那么，当有多个js存在且是以外链的方式存在是又是如何呢？

**浏览器都会按照script元素在页面中出现的先后顺序对他们依次进行解析（文件下载和代码执行）。换句话说，在第一个外链script下载 -> 执行，由于浏览器做了预解析的优化，在第一个js执行的过程中，后面的几个js会并行下载。第一个script执行完成 -> 第二个外链script下载完成 -> 执行，然后第三个，第四个……。这种方式会阻塞页面的渲染。**

也于是，解决这种阻塞的方式出现了，defer和async，对于这两个的区别，我这里引用了大佬阮一峰的解释：

**来自阮一峰博客对defer和async异同点说明：**

**defer与async的区别是：defer要等到整个页面在内存中正常渲染结束（DOM 结构完全生成，以及其他脚本执行完成），才会执行；async一旦下载完，渲染引擎就会中断渲染，执行这个脚本以后，再继续渲染。一句话，defer是“渲染完再执行”，async是“下载完就执行”。另外，如果有多个defer脚本，会按照它们在页面出现的顺序加载，而多个async脚本是不能保证加载顺序的。**