

## 09 页面渲染机制（一）之 DOM 树和 CSSOM 树的构建

更新时间：2019-06-21 16:58:31



我要扼住命运的咽喉，它妄想使我屈服，这绝对办不到。生活是这样美好，活他一千辈子吧！

——贝多芬

页面渲染机制这部分内容总共分成两个小节，这两节里我们准备聊一下页面的渲染的过程，包括页面的加载、DOM 树的构建、CSSOM 树的构建、渲染树的构建和最后的渲染过程等。浏览器的渲染机制和网页的优化息息相关，只有知道了页面是怎么渲染出来的，才能在写代码的时候使用最合理的方式，比如知道了 CSS 文件的解析过程后就知道为什么要把 CSS 文件放在 HTML 的前边，知道为什么要少用@import 了。这一部分的内容，我们会先介绍渲染的整体过程，然后再把这个过程中比较重要的部分做详细介绍。

### 页面的加载和渲染全过程

当我们在浏览器里输入一个 URL 后，最终会呈现一个完整的网页。这中间会经历如下的过程：

#### 1. HTML 的加载

输入 URL 后，最先拿到的是 HTML 文件。HTML 是一个网页的基础，所以要在最开始的时候下载它。HTML 下载完成以后就会开始对它进行解析。

#### 2. 其他静态资源下载

HTML 在解析的过程中，如果发现 HTML 文本里面夹杂的一些外部的资源链接，比如 CSS、JS 和图片等时，会立即启用别的线程下载这些静态资源。这里有个特殊的是 JS 文件，当遇到 JS 文件的时候，HTML 的解析会停下来，等 JS 文件下载结束并且执行完，HTML 的解析工作再接着来。这样做是因为 JS 里可能会出现修改已经完成的解析结果，有白白浪费资源的风险，所以 HTML 解析器干脆等 JS 折腾完了再干。

#### 3. DOM 树构建

在 HTML 解析的同时，解析器会把解析完的 HTML 转化成 DOM 对象，再进一步构建 DOM 树。

#### 4. CSSOM 树构建

当 CSS 下载完，CSS 解析器就开始对 CSS 进行解析，把 CSS 解析成 CSS 对象，然后把这些 CSS 对象组装起

来，构建出一棵 CSSOM 树。

## 5. 渲染树构建

DOM 树和 CSSOM 树都构建完成以后，浏览器会根据这两棵树构建出一棵渲染树。

## 6. 布局计算

渲染树构建完成以后，所有元素的位置关系和需要应用的样式就确定了。这时候浏览器会计算出所有元素的大小和绝对位置。

## 7. 渲染

布局计算完成以后，浏览器就可以在页面上渲染元素了。比如从 (x1, y1) 到 (x2, y2) 的正方形区域渲染成蓝色。经过渲染引擎的处理后，整个页面就显示在了屏幕上。

上面讲了一下浏览器从加载到渲染的大概过程，这部分内容是想让同学们对加载有个大概的印象，接下来我们把这个过程中比较重要的部分再详细讲解下。

# DOM 树的构建

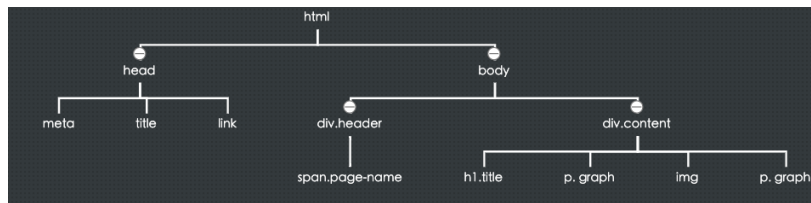
页面中的每一个 HTML 标签，都会被浏览器解析成一个对象，我们称它为文档对象（Document Object）。HTML 的本质是一个嵌套结构，在解析的时候会把每个文档对象用一个树形结构组织起来，所有的文档对象都会挂在一个叫做 Document 的东西上，这种组织方式就是 HTML 最基础的结构—文档对象模型（DOM），这棵树里面的每个文档对象就叫做 DOM 节点。

在 HTML 加载的过程中，DOM 树就在开始构建了。构建的过程是先把 HTML 里每个标签都解析成 DOM 节点（每个标签的属性、值和上下文关系等都在这个文档对象里），然后使用深度遍历的方法把这些对象构造成一棵树。

我们以下面的 HTML 文件为例：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" href="/index.css">
</head>
<body>
  <div class="header">
    <span class="page-name">文章详情页</span>
  </div>
  <div class="content">
    <h1 class="title">文章标题</h1>
    <div class="article">
      <p class="graph">>吃葡萄不吐葡萄皮</p>
      
      <p class="graph"><吃葡萄倒吐葡萄皮</p>
    </div>
  </div>
</body>
</html>
```

在构建 DOM 树的时候，就是从最外层 HTML 节点开始，按深度优先的方式构建。之所以用深度优先，是因为 HTML 在加载的时候是自上而下的，最先加载的是根节点<html>，然后是根节点的第一个子节点<head>，再然后是 head 的第一个子节点<meta>...head 构建完成后再去构建 body 部分的内容，以此类推。使用深度优先的方式构建这棵树就和文档的加载顺序吻合了。最后，上面这个 html 结构就会生成如下样式的一棵 DOM 树：



## CSSOM 树的构建

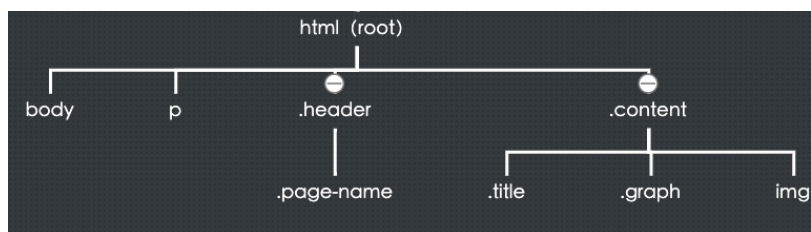
在浏览器构建 DOM 树的同时，如果样式也加载完成了，那么 CSSOM 树也在同步地构建。CSS 树和 DOM 类似，它的树形结构记录着所有样式的信息。

我们以给上面的 HTML 加上如下的样式：

```

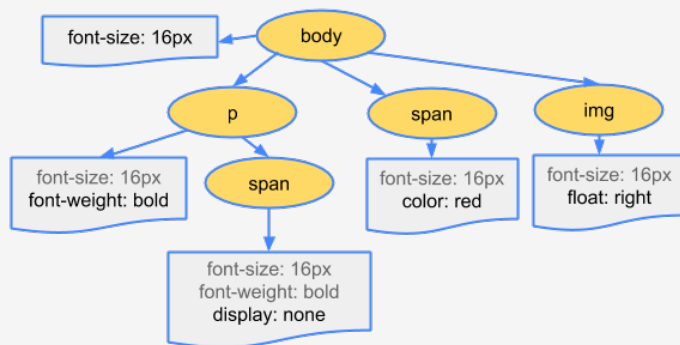
body{
  font-size: 16px;
}
// 去掉所有p元素的内外边距
p{
  margin: 0;
  padding: 0;
}
// 页面头部行高50px，文本垂直居中，隐藏
.header{
  height: 50px;
  line-height: 50px;
  display: none;
  text-align: center;
}
.header .page-name{
  font-size: 20px;
}
// 文本区域左右两边留10px空白
.content{
  padding: 0 10px;
}
.content .title{
  font-size: 20px;
}
// 内容区行高30px
.content .graph{
  line-height: 30px;
}
// 文章中的图片用作块级元素，水平居中
.content img{
  display: block;
  margin: 0 auto;
}
  
```

我们就以这一组样式为例，这样一组样式中有公用的样式 `p` 和 `body`，有标题栏 `.header` 部分的样式，还有内容区 `.content` 部分的样式。这样通过解析器的构造，可以得到类似下面这样的一棵 CSSOM 树：



### Tips:

1、这棵树是一个示意图，并不是浏览器里构造 CSSOM 树的真实的数据结构，而且各种浏览器内核实现 CSSOM 树的方式也不全都相同。这部分内容可以参考 [Google Web Fundamentals](#)，它把 CSSOM 树描述成自上而下建立的结构，类似这样：



我们课程里也是按着这个文档里 CSSOM 的模型来示意的，不同的是我把 HTML 节点作为根节点。这是因为考虑到给 HTML 标签设置样式的时候同样会生效，所以 HTML 标签应该也存在于 CSSOM 树中。

2、CSSOM 树和 DOM 树是独立的两个数据结构，它们没有一一对应关系。DOM 树描述的是 HTML 标签的层级关系，CSSOM 树描述的是选择器之间的层级关系。

3、在 CSS 中存在样式的继承机制，有些属性在父节点设置后，在其后代节点都会具备这个样式。比如我们在 HTML 上设置一个“font-size:20px;”，那么页面里基本所有的标签都可以继承到这个属性了。当然不是所有标签和属性都可以有继承特性的，比如 border 这种属性就不是可继承的。如果 border 可继承了，那么在一个父元素里设置上以后，所有子元素都会有个边框，这显然是不合理的。所以在大部分情况下，通过这种推理，就能知道哪些样式是可以继承的，而哪些不行。

### 小结

这一节讲了渲染的大致过程、DOM 树的构建和 CSSOM 树的构建。到这个阶段，渲染需要的基础工作就准备完成了。这节里的内容结构如下：



这一节里需要同学们了解 DOM 树和 CSSOM 树的结构以及它们的区别。下一节将会讲解渲染树的构成以及后面的布局、渲染过程。