浏览器内核拿到内容后，渲染步骤大致可以分为以下几步：

1. 解析HTML，构建DOM树

2. 解析CSS，生成CSSOM树

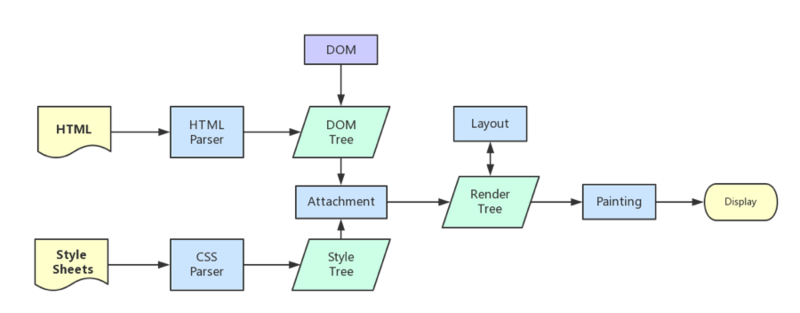
3. 合并DOM树和CSSOM树，生成render树

4. 布局render树（Layout/reflow），负责各元素尺寸、位置的计算

5. 绘制render树（paint），绘制页面像素信息

6. 浏览器会将各层的信息发送给GPU，GPU会将各层合成（composite），显示在屏幕上

第一步HTML解析，构建DOM



解析HTML到构建出DOM当然过程可以简述如下：

Bytes → characters → tokens → nodes → DOM

譬如假设有这样一个HTML页面：（以下部分的内容出自参考来源，修改了下格式）

<html>

<head>

<meta name="viewport" content="width=device-width,initial-scale=1">

<link href="style.css" rel="stylesheet">

<title>Critical Path</title>

</head>

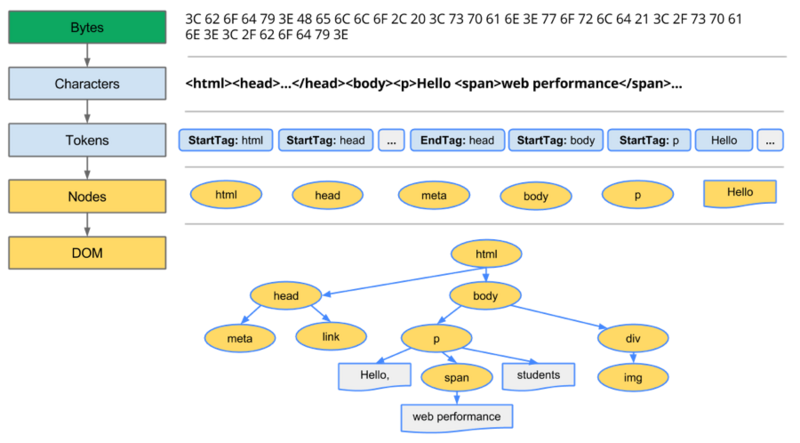
<body>

<p>Hello <span>web performance</span> students!</p>

<div><img src="awesome-photo.jpg"></div>

</body>

</html>

**重要的过程**

1. Conversion转换：浏览器将获得的HTML内容（Bytes）基于他的编码转换为单个字符

2. Tokenizing分词：浏览器按照HTML规范标准将这些字符转换为不同的标记token。每个token都有自己独特的含义以及规则集

3. Lexing词法分析：分词的结果是得到一堆的token，此时把他们转换为对象，这些对象分别定义他们的属性和规则

4. DOM构建：因为HTML标记定义的就是不同标签之间的关系，这个关系就像是一个树形结构一样

例如：body对象的父节点就是HTML对象，然后段略p对象的父节点就是body对象

## 第二步生成CSS规则

同理，CSS规则树的生成也是类似。简述为：

Bytes → characters → tokens → nodes → CSSOM

譬如style.css内容如下：

body { font-size: 16px }

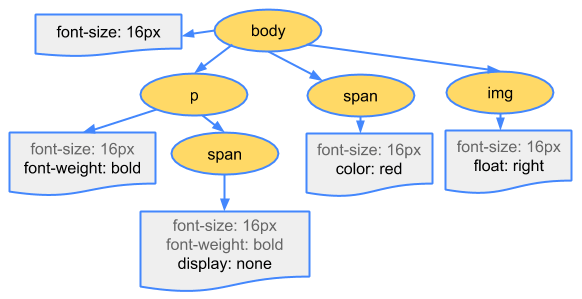
p { font-weight: bold }

span { color: red }

p span { display: none }

img { float: right }

那么最终的CSSOM树就是：



第三步构建渲染树

当DOM树和CSSOM都有了后，就要开始构建渲染树了

一般来说，渲染树和DOM树相对应的，但不是严格意义上的一一对应

因为有一些不可见的DOM元素不会插入到渲染树中，如head这种不可见的标签或者display: none等

