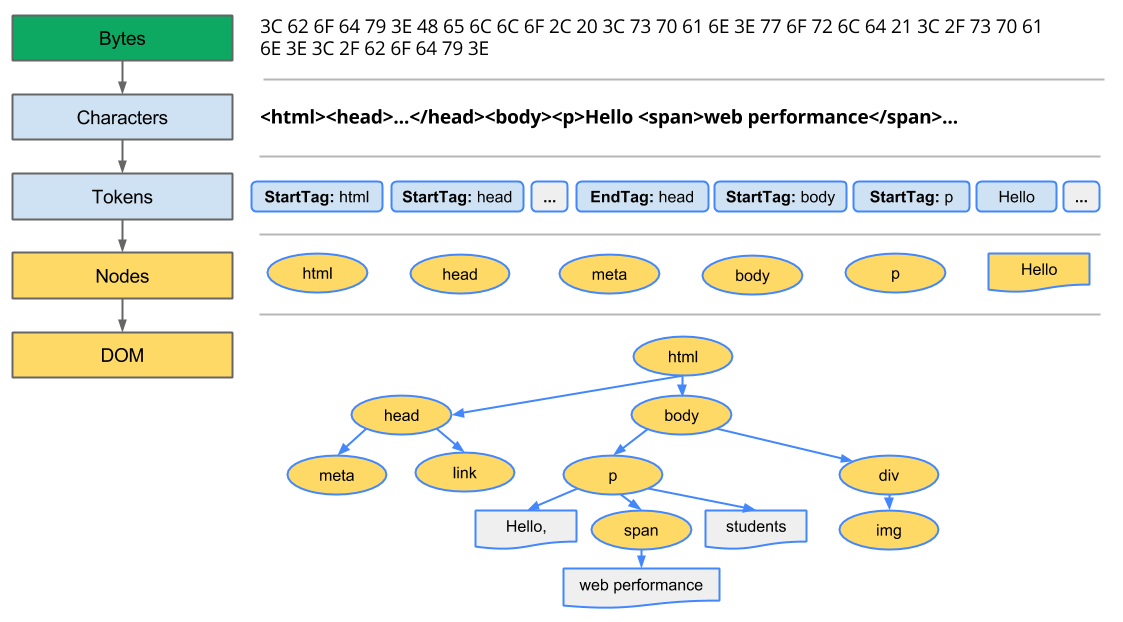
## **几个概念**

(1) DOM：Document Object Model，浏览器将HTML解析成树形的数据结构，简称DOM。

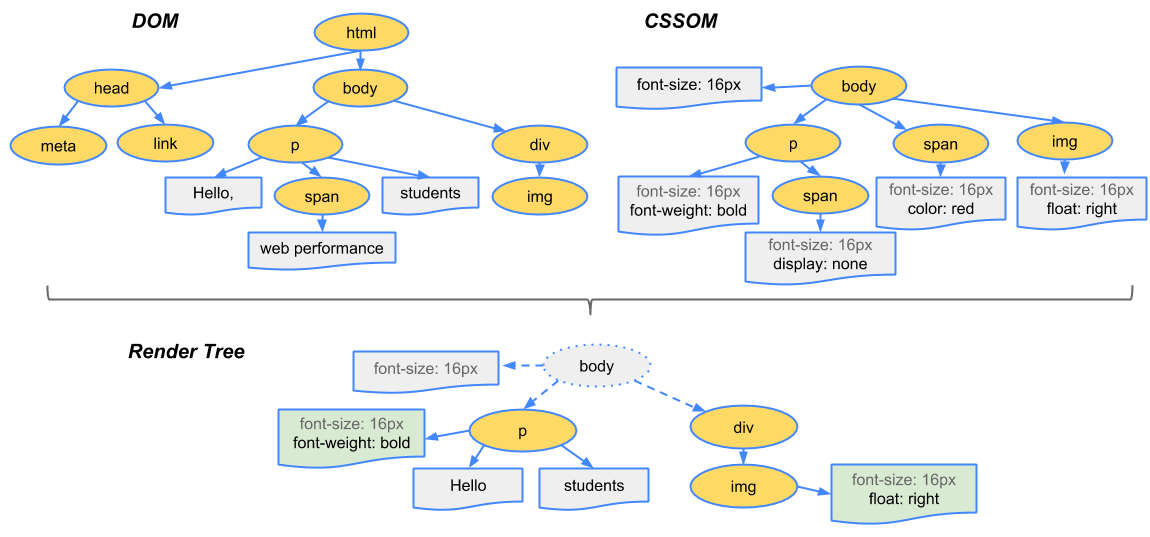
(2) CSSOM：CSS Object Model，浏览器将CSS代码解析成树形的数据结构。

(3) DOM 和 CSSOM 都是以Bytes → characters → tokens → nodes → object model. 这样的方式生成最终的数据。如下图所示：



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。

(4) Render Tree：DOM 和 CSSOM 合并后生成 Render Tree，如下图：



Render Tree 和 DOM 一样，以多叉树的形式保存了每个节点的css属性、节点本身属性、以及节点的孩子节点。

注意：display:none 的节点不会被加入Render Tree，而visibility: hidden 则会，所以，如果某个节点最开始是不显示的，设为display:none是更优的。

## **浏览器的渲染过程**

(1) ****Create/Update DOM And request css/image/js****：浏览器请求到HTML代码后，在生成DOM的最开始阶段（应该是 Bytes → characters 后），并行发起css、图片、js的请求，无论他们是否在HEAD里。

注意：发起js文件的下载request并不需要DOM处理到那个script节点，比如：简单的正则匹配就能做到这一点，虽然实际上并不一定是通过正则：）。这是很多人在理解渲染机制的时候存在的误区。

(2) ****Create/Update Render CSSOM****: CSS文件下载完成，开始构建CSSOM。

(3) ****Create/Update Render Tree****：所有CSS文件下载完成，CSSOM构建结束后，和 DOM 一起生成 Render Tree。

(4) ****Layout****：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。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。

(5) ****Painting****：Layout后，浏览器已经知道了哪些节点要显示（which nodes are visible）、每个节点的CSS属性是什么（their computed styles）、每个节点在屏幕中的位置是哪里（geometry）。就进入了最后一步：Painting，按照算出来的规则，通过显卡，把内容画到屏幕上。

以上五个步骤前3个步骤之所有使用 “Create/Update” 是因为DOM、CSSOM、Render Tree都可能在第一次Painting后又被更新多次，比如JS修改了DOM或者CSS属性。

Layout 和 Painting 也会被重复执行，除了DOM、CSSOM更新的原因外，图片下载完成后也需要调用Layout 和 Painting来更新网页。