《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

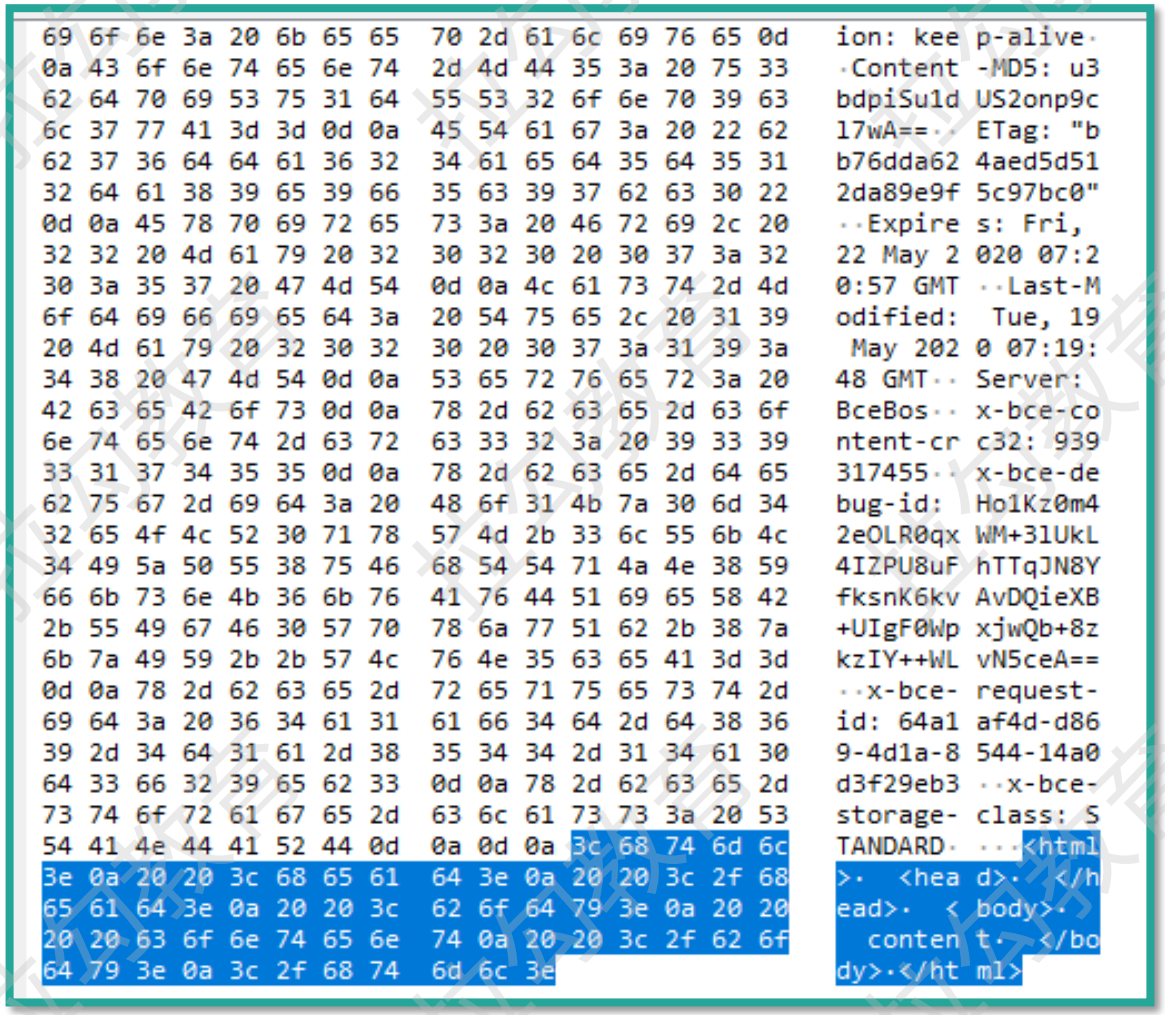
第06讲：浏览器如何渲染页面？

举例


```
<html>
 <head>
 </head>
 <body>
  lagou
 </body>
</html>
```

从 HTML 到 DOM

1. 字节流解码



```
69 6f 6e 3a 20 6b 65 65  70 2d 61 6c 69 76 65 0d   ion: kee p-alive·
0a 43 6f 6e 74 65 6e 74  2d 4d 44 35 3a 20 75 33   ·Content -MD5: u3
62 64 70 69 53 75 31 64  55 53 32 6f 6e 70 39 63   bdpiSu1d US2onp9c
6c 37 77 41 3d 3d 0d 0a  45 54 61 67 3a 20 22 62   l7wA==·· ETag: "b
62 37 36 64 64 61 36 32  34 61 65 64 35 64 35 31   b76dda62 4aed5d51
32 64 61 38 39 65 39 66  35 63 39 37 62 63 30 22   2da89e9f 5c97bc0"
0d 0a 45 78 70 69 72 65  73 3a 20 46 72 69 2c 20   ··Expire s: Fri,
32 32 20 4d 61 79 20 32  30 32 30 20 30 37 3a 32   22 May 2 020 07:2
30 3a 35 37 20 47 4d 54  0d 0a 4c 61 73 74 2d 4d   0:57 GMT ··Last-M
6f 64 69 66 69 65 64 3a  20 54 75 65 2c 20 31 39   odified:  Tue, 19
20 4d 61 79 20 32 30 32  30 20 30 37 3a 31 39 3a    May 202 0 07:19:
34 38 20 47 4d 54 0d 0a  53 65 72 76 65 72 3a 20   48 GMT·· Server:
42 63 65 42 6f 73 0d 0a  78 2d 62 63 65 2d 63 6f   BceBos·· x-bce-co
6e 74 65 6e 74 2d 63 72  63 33 32 3a 20 39 33 39   ntent-cr c32: 939
33 31 37 34 35 35 0d 0a  78 2d 62 63 65 2d 64 65   317455·· x-bce-de
62 75 67 2d 69 64 3a 20  48 6f 31 4b 7a 30 6d 34   bug-id:  Ho1Kz0m4
32 65 4f 4c 52 30 71 78  57 4d 2b 33 6c 55 6b 4c   2eOLR0qx WM+3lUkL
34 49 5a 50 55 38 75 46  68 54 54 71 4a 4e 38 59   4IZPU8uF hTTqJN8Y
66 6b 73 6e 4b 36 6b 76  41 76 44 51 69 65 58 42   fksnK6kv AvDQieXB
2b 55 49 67 46 30 57 70  78 6a 77 51 62 2b 38 7a   +UIgF0Wp xjwQb+8z
6b 7a 49 59 2b 2b 57 4c  76 4e 35 63 65 41 3d 3d   kzIY++WL vN5ceA==
0d 0a 78 2d 62 63 65 2d  72 65 71 75 65 73 74 2d   ··x-bce- request-
69 64 3a 20 36 34 61 31  61 66 34 64 2d 64 38 36   id: 64a1 af4d-d86
39 2d 34 64 31 61 2d 38  35 34 34 2d 31 34 61 30   9-4d1a-8 544-14a0
64 33 66 32 39 65 62 33  0d 0a 78 2d 62 63 65 2d   d3f29eb3 ··x-bce-
73 74 6f 72 61 67 65 2d  63 6c 61 73 73 3a 20 53   storage- class: S
54 41 4e 44 41 52 44 0d  0a 0d 0a 3c 68 74 6d 6c   TANDARD· ···<html
3e 0a 20 20 3c 68 65 61  64 3e 0a 20 20 3c 2f 68   >·  <hea d>·  </h
65 61 64 3e 0a 20 20 3c  62 6f 64 79 3e 0a 20 20   ead>·  < body>·
20 20 63 6f 6e 74 65 6e  74 0a 20 20 3c 2f 62 6f     conten t·  </bo
64 79 3e 0a 3c 2f 68 74  6d 6c 3e                  dy>·</ht ml>
```

从 HTML 到 DOM

2. 输入流预处理

上一步解码得到的字符流数据

在进入解析环节之前还需要进行一些预处理操作

比如将换行符转换成统一的格式

最终生成**规范化**的字符流数据

这个把字符数据进行统一格式化的过程称之为“**输入流预处理**”

从 HTML 到 DOM

3. 令牌化

- 将字符数据转化成令牌（Token）
- 解析 HTML 生成 DOM 树

3. 令牌化

- 将字符数据转化成令牌（Token）

  每次接收一个或多个输入流中的字符

  然后根据当前状态和这些字符来更新下一个状态

  也就是说在不同的状态下接收同样的字符数据

  可能会产生不同的结果

从 HTML 到 DOM

3. 令牌化

- 初始化为“数据状态”（Data State）
- 匹配到字符 <，状态切换到 “标签打开状态”（Tag Open State）
- 匹配到字符 !，状态切换至 “标签声明打开状态”（Markup Declaration Open State）

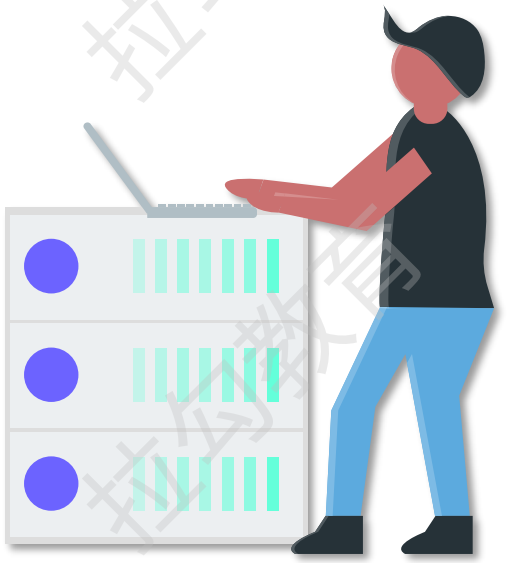
  后续 7 个字符可以组成字符串 DOCTYPE

  跳转到 “DOCTYPE 状态”（DOCTYPE State）
- 匹配到字符为空格

  当前状态切换至 “DOCTYPE 名称之前状态”（Before DOCTYPE Name State）
- 匹配到字符串 html

  创建一个新的 DOCTYPE 标记，标记的名字为 “html”

  然后当前状态切换至 “DOCTYPE 名字状态”（DOCTYPE Name State）



从 HTML 到 DOM

3. 令牌化

- 匹配到字符 >，跳转到 “数据状态” 并且释放当前的 DOCTYPE 标记
- 匹配到字符 <，切换到 “标签打开状态”
- 匹配到字符 h

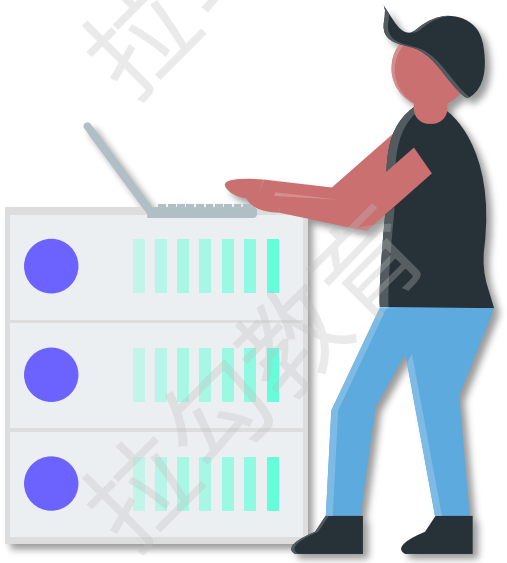
  创建一个新的起始标签标记，设置标记的标签名为空

  当前状态切换至 “标签名称状态” （Tag Name State）
- 从字符 h 开始解析

  将解析的字符一个一个添加到创建的起始标签标记的标签名中

  直到匹配到字符 >，此时当前状态切换至 “数据状态” 并释放当前标记

  当前标记的标签名为 “html”



从 HTML 到 DOM

3. 令牌化

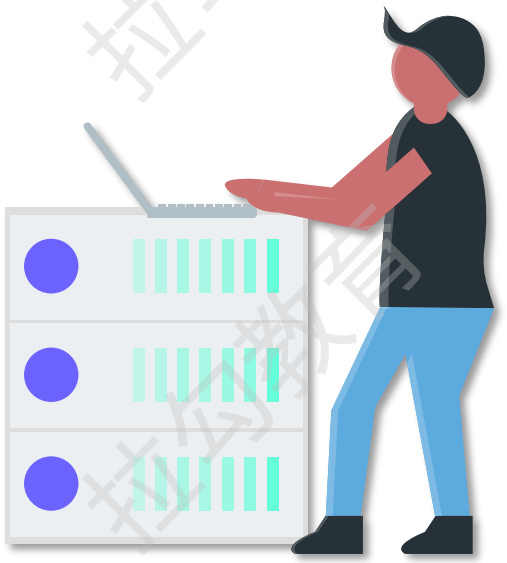
- 解析后续的 <body> 的方式与 <html> 一致

  创建并释放对应的起始标签标记，解析完毕后，当前状态处于 “数据状态”
- 匹配到字符串 “标记”

  针对每一个字符，创建并释放一个对应的字符标记

  解析完毕后，当前状态仍然处于 “数据状态”
- 匹配到字符 <，进入 “标签打开状态”
- 匹配到字符 /，进入 “结束标签打开状态” （End Tag Open State）
- 匹配到字符 b，创建一个新的结束标签标记

  设置标记的标签名为空，当前状态切换至“标签名称状态” （Tag Name State）



从 HTML 到 DOM

3. 令牌化

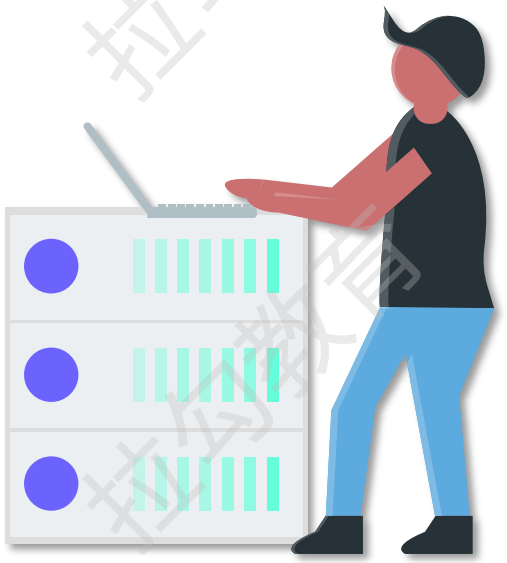
- 重新从字符 b 开始解析

  将解析的字符一个一个添加到创建的结束标签标记的标签名中

  直到匹配到字符 >，此时当前状态切换至 “数据状态” 并释放当前标记

  当前标记的标签名为 “body”
- 解析 </html> 的方式与 </body> 一样
- 所有的 html 标签和文本解析完成后，状态切换至 “数据状态”

  一旦匹配到文件结束标志符（EOF），则释放 EOF 标记



从 HTML 到 DOM

3. 令牌化

开始标签:html
开始标签:head
结束标签:head
开始标签:body
字符串:lagou
结束标签:body
结束标签:html

从 HTML 到 DOM

补充 1：遇到 script 标签时的处理

- 如果遇到的是**内联代码**，也就是在 script 标签中直接写代码

  解析过程会暂停

  执行权限会转给 JavaScript 脚本引擎

  待 JavaScript 脚本执行完成之后再交由渲染引擎继续解析

从 HTML 到 DOM

补充 1：遇到 script 标签时的处理

- 脚本内容中调用了改变 DOM 结构的 **document.write() 函数**

  此时渲染引擎会回到第二步，将这些代码加入字符流，重新进行解析

- 如果是**外链脚本**

  那么渲染引擎会根据标签属性来执行对应的操作

从 HTML 到 DOM

4. 构建 DOM 树

浏览器在创建解析器的同时会创建一个 **Document 对象**

在树构建阶段，Document 会作为根节点被不断地修改和扩充

标记步骤产生的令牌会被送到树构建器进行处理

HTML 5 标准中定义了每类令牌对应的 **DOM 元素**

当树构建器接收到某个令牌时

就会创建该令牌对应的 DOM 元素并将该元素插入到 DOM 树中

从 HTML 到 DOM

4. 构建 DOM 树

为了纠正元素标签嵌套错位的问题和处理未关闭的元素标签

树构建器创建的新 DOM 元素还会被插入到一个**开放元素栈**中

从 HTML 到 DOM

4. 构建 DOM 树

- 进入初始状态 “initial” 模式
- 树构建器接收到 DOCTYPE 令牌后

  树构建器会创建一个 DocumentType 节点附加到 Document 节点上

  DocumentType 节点的 name 属性为 DOCTYPE 令牌的名称

  切换到 “before html” 模式
- 接收到令牌 html 后

  树构建器创建一个 html 元素并将该元素作为 Document 的子节点

  插入到 DOM 树中和开放元素栈中

  切换为 “before head” 模式

从 HTML 到 DOM

4. 构建 DOM 树

- 虽然没有接收到 head 令牌

  但仍然会隐式地创建 head 元素并加入到 DOM 树和开放元素栈中

  切换到“in head”模式
- 将开放元素栈中的 head 元素弹出，进入 “after head”模式
- 接收到 body 令牌后

  会创建一个 body 元素插入到 DOM 树中同时压入开放元素栈中

  当前状态切换为 “in body” 模式

从 HTML 到 DOM

4. 构建 DOM 树

- 接收到字符令牌，创建 Text 节点，节点值为字符内容“标记”

  将 Text 节点作为 body 元素节点插入到 DOM 树中
- 接收到结束令牌 body

  将开放元素栈中的 body 元素弹出

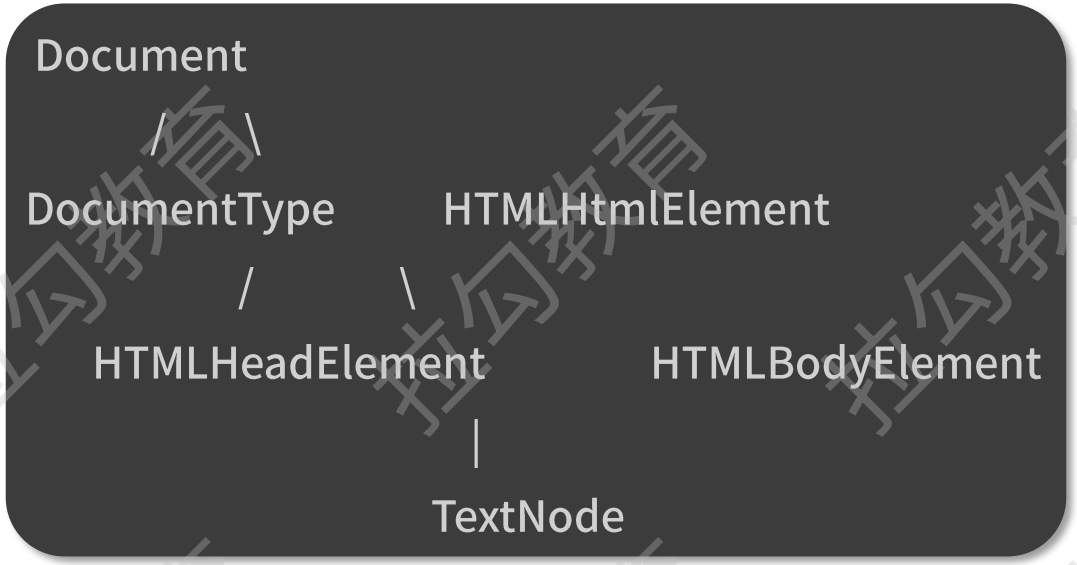
  切换至 “after body” 模式
- 接收到结束令牌 html

  将开放元素栈中的 html 元素弹出

  切换至 “after after body” 模式
- 接收到 EOF 令牌，树构建器停止构建，html 文档解析过程完成

从 HTML 到 DOM

4. 构建 DOM 树



从 HTML 到 DOM

4. 构建 DOM 树

从 HTML 到 DOM

补充 2：从 CSS 到 CSSOM

- CSS 解析的过程与 HTML 解析过程步骤一致，最终也会生成**树状结构**

- CSSOM 树的节点具有**继承特性**

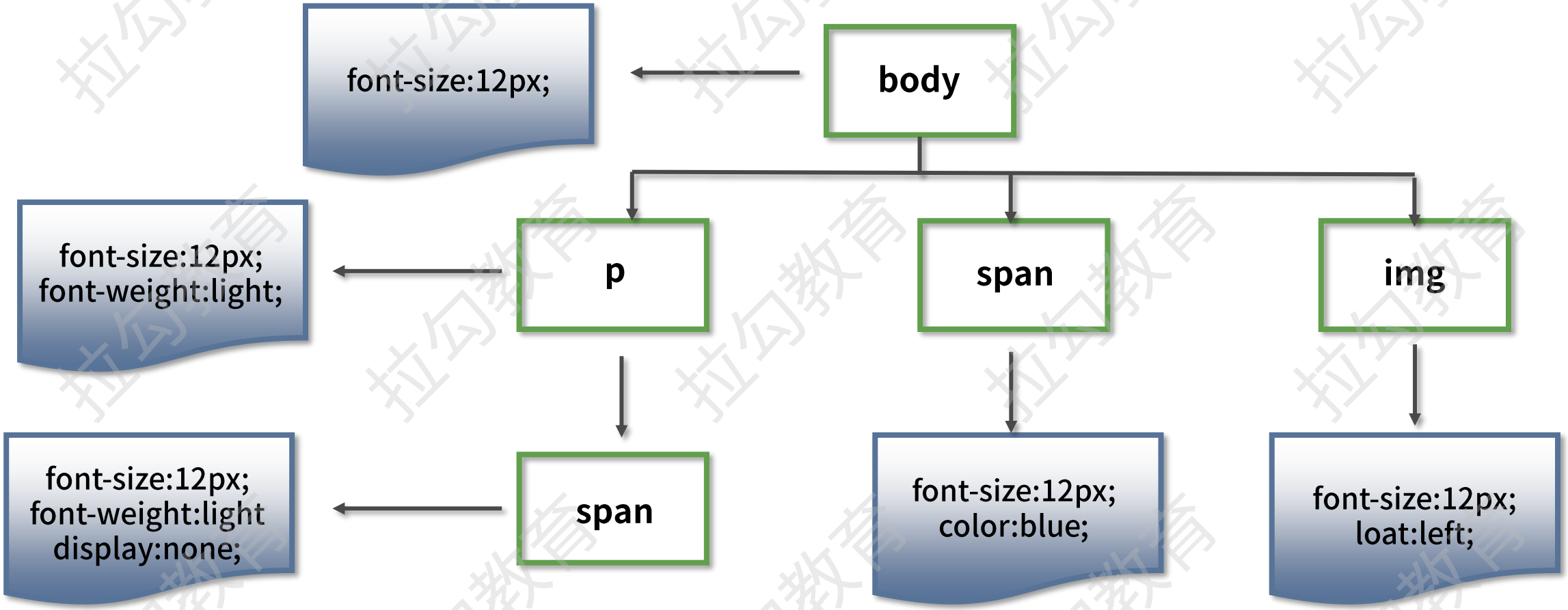
  也就是会先继承父节点样式作为当前样式，然后再进行补充或覆盖

从 HTML 到 DOM

补充 2：从 CSS 到 CSSOM

```
body { font-size: 12px }
p { font-weight: light }
span { color: blue }
p span { display: none }
img { float: left }
```

从 HTML 到 DOM



从 DOM 到渲染

5. 构建渲染树

DOM 树包含的结构内容与 CSSOM 树包含的样式规则都是**独立**的

为了更方便渲染，先需要将它们**合并**成一棵渲染树

从 DOM 树的**根节点开始**遍历

然后在 CSSOM 树上找到每个节点对应的样式

遍历过程中会自动忽略那些不需要渲染的节点（比如脚本标记、元标记等）

以及不可见的节点（比如设置了“display:none”样式）

同时也会将一些需要显示的伪类元素加入到渲染树中

6. 布局

布局就是计算元素的大小及位置

计算元素布局是一个比较复杂的操作

因为需要考虑的因素有很多，包括字体大小、换行位置等

这些因素会影响段落的大小和形状

进而影响下一个段落的位置

从 DOM 到渲染

6. 布局

布局完成后会输出对应的“**盒模型**”

它会精确地捕获每个元素的确切位置和大小

将所有相对值都转换为屏幕上的绝对像素

从 DOM 到渲染

7. 绘制

绘制就是将渲染树中的每个节点转换成屏幕上的实际像素的过程

注意

如果没有弄清楚绘制顺序，那么很可能会导致页面被错误地渲染

从 DOM 到渲染

7. 绘制

绘制过程中的第一步就是**遍历布局树，生成绘制记录**

然后渲染引擎会根据绘制记录去绘制相应的内容

对于**无动画效果**的情况

只需要考虑空间维度，生成不同的图层

然后再把这些图层进行合成

最终成为我们看到的页面

总结

浏览器渲染引擎生成页面的 7 个步骤

前面 4 个步骤为 DOM 树的生成过程

后面 3 个步骤是利用 DOM 树和 CSSOM 树来渲染页面的过程

字节 → 字符 → 令牌 → 树 → 页面

总结

在构建渲染树的时候

渲染引擎需要遍历 DOM 树节点并从 CSSOM 树中找到匹配的样式规则

在匹配过程中是通过自上而下还是自下而上的方式呢？

为什么？

Next：加餐01《手写 CSS 预处理》