Daliy_Learning(2020-3-16)

CSS 中的 vertical-align 有哪些值？它在什么情况下才能生效？

vertical-align属性值：

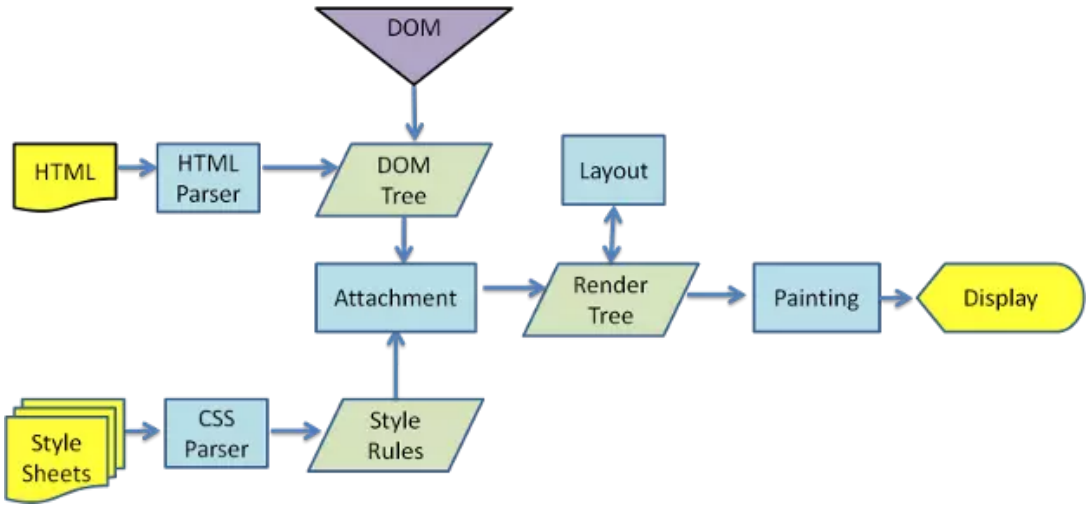
- 线类：baseline、top、middle、bottom
- 文本类：text-top、text-bottom
- 上标下标类：sub、super
- 数值百分比类：20px、2em、20%等（对于基线往上或往下偏移）

负值相对于基线往下偏移，正值往上偏移，事实上vertical-align:base-line等同于vertical-align:0。

vertical-align生效前提：

vertical-align属性只能应用于内联元素以及display值为table-cell的元素，因此vertical-align只能应用于display值为inline、inline-block、inline-table或table-cell的元素，span、strong、em、img、button、input等元素天然支持vertical-align属性，块级元素则不支持。需要注意浮动和绝对定位会让元素块状化，因此此元素绝对不会生效。如果使用百分比，那么其实依据的是line-height的值。

浏览器渲染过程



- 浏览器使用流式布局模型（Flow Based Layout）

- 浏览器会把HTML解析成DOM，把CSS解析成CSSOM，DOM和CSSOM合并就产生了Render Tree

- 有了RenderTree就能知道所有节点的样式，计算节点在页面上的大小和位置，把节点绘制到页面上

- 由于浏览器使用流式布局，对Render Tree的计算通常只需要遍历一次就可以完成，但table及其内部元素除外，通常需要多次计算且要花费3倍于同等元素的时间，这也是为什么要避免使用table布局的原因之一

浏览器渲染过程如下：

- 解析HTML，生成DOM树

- 解析CSS，生成CSSOM树

- 将DOM树和CSSOM树结合，生成渲染树(Render Tree)

- Layout(回流)：根据生成的渲染树，进行回流(Layout)，得到节点的几何信息（位置，大小）

- Painting(重绘)：根据渲染树以及回流得到的几何信息，得到节点的绝对像素

- Display：将像素发送给GPU，展示在页面上。（这一步其实还有很多内容，比如会在GPU将多个合成层合并为同一个层，并展示在页面中。而css3硬件加速的原理则是新建合成层等等）

何时触发回流和重绘

何时发生回流：

- 添加或删除可见的DOM元素
- 元素的位置发生变化
- 元素的尺寸发生变化（包括外边距、内边框、边框大小、高度和宽度等）
- 内容发生变化，比如文本变化或图片被另一个不同尺寸的图片所替代。
- 页面一开始渲染的时候（这肯定避免不了）
- 浏览器的窗口尺寸变化（因为回流是根据视口的大小来计算元素的位置和大小的）

何时发生重绘（回流一定会触发重绘）：

当页面中元素样式的改变并不影响它在文档流中的位置时（例如：color、background-color、visibility等），浏览器会将新样式赋予给元素并重新绘制它，这个过程称为重绘。

有时即使仅仅回流一个单一的元素，它的父元素以及任何跟随它的元素也会产生回流。现代浏览器会对频繁的回流或重绘操作进行优化，浏览器会维护一个队列，把所有引起回流和重绘的操作放入队列中，如果队列中的任务数量或者时间间隔达到一个阈值的，浏览器就会将队列清空，进行一次批处理，这样可以把多次回流和重绘变成一次。你访问以下属性或方法时，浏览器会立刻清空队列：

- clientWidth、clientHeight、clientTop、clientLeft
- offsetWidth、offsetHeight、offsetTop、offsetLeft
- scrollWidth、scrollHeight、scrollTop、scrollLeft

- width、height
- getComputedStyle()
- getBoundingClientRect()

以上属性和方法都需要返回最新的布局信息，因此浏览器不得不清空队列，触发回流重绘来返回正确的值。因此，我们在修改样式的时候，最好避免使用上面列出的属性，他们都会刷新渲染队列。如果要使用它们，最好将值缓存起来。

如何避免触发回流和重绘

CSS：

- 避免使用table布局。
- 尽可能在DOM树的最末端改变class。
- 避免设置多层内联样式。
- 将动画效果应用到position属性为absolute或fixed的元素上
- 避免使用CSS表达式（例如：calc()）
- CSS3硬件加速（GPU加速）

JavaScript：

- 避免频繁操作样式，最好一次性重写style属性，或者将样式列表定义为class并一次性更改class属性
- 避免频繁操作DOM，创建一个documentFragment，在它上面应用所有DOM操作，最后再把它添加到文档中

- 也可以先为元素设置display: none，操作结束后再把它显示出来。因为在display属性为none的元素上进行的DOM操作不会引发回流和重绘

- 避免频繁读取会引发回流/重绘的属性，如果确实需要多次使用，就用一个变量缓存起来

- 对具有复杂动画的元素使用绝对定位，使它脱离文档流，否则会引起父元素及后续元素频繁回流