

Date: 2015-07-15

Title: 进阶CSS从零开始突破

Excerpt: 原本是想好好写一些基础的东西，可是写着写着，感觉其实网络上有太多这些东西了，所以太监了，写不动了……暂时就先这样吧，以后看情况是否继续写……

进阶**CSS**从零开始突破

一、写在前面

一直就在想自己是不是可以写一点什么，但一直又不知道写什么好，就自己的能力而言，相对比较擅长的也就是这个**CSS**了，不过说实在的，对于这个东西真的还是比较简单的，看看手册上，无非也就是那么一些属性，多用几次就好了。虽然话是这么说吧，不过还是经常在网络上（QQ群、QA社区等）看到很多朋友问的都是很基础的东西，想想或许应该是基础不是十分扎实吧，又或者是其他什么原因吧，再者就是有一些想要进入前端队伍的朋友但又没接触过这些东西。

那么我想说，这一份文档就是给相对而言对基础了解不够多的朋友，以及想要进入前端队伍的朋友看的。从最简单的**CSS**开始去了解一个页面。

这份文档其实我很早就想写了，虽然之前写过《CSS那些事儿》这本书，但这毕竟是以前的东西，而且关键销量也不好，最后市面见不到了，后来也曾想过再尝试写一本，朋友们也鼓励让我写，但最后都拒绝了。而现在要写这个，主要几点是：

1. 我太无聊了想给自己找点事情做做，顺带稍微巩固一下**CSS**；
2. 在这个浮躁的前端行业中，太多人遗忘了**CSS**是做什么用的，经常看到所谓精通CSS的朋友其实很多东西是不会的；
3. 写这个电子文档我觉得我可以在追求相对严谨的基础上随心所欲去表达，而不用受到各方面的限制；

哦了，废话扯得也差不多了，很快就要开始进入正题了，希望这很基础很基础的文档会有人喜欢……

二、学习之前的准备

在开始学习之前，我们需要准备一点东西在自己的电脑中，无论你的系统是 windows 还是 OS X 系列，甚至是 linux 之类的，这都不是关键，关键是有以下软件存在：

- **编辑器**：对于编辑器没什么要求，因为我们写 **HTML** 和 **CSS** 都很简单，只要能输入字符，保存成 `.html` 或者 `.htm` 以及 `.css` 就可以了，不过目前好像比较流行 **Sublime Text**，至于为什么流行，我也不知道，反正写 **HTML** 和 **CSS** 只要有代码高亮这个最基本的要求，其他都是后期大家自己的习惯和追求了；
- **浏览器**：对于浏览器要说的就太多了，各个版本的 **IE**、还有谷歌的 **Chrome**、以及还有 **Firefox**、**Opera** 等等。这里需要提的一点是，虽然现在 **IE6** 使用率已经降得很低了，但是从目前的情况来看，我们可以稍微了解一点，然后还有可能的话，也会提到一点关于 **CSS3** 的东西，那么就需要用高版本的浏览器来看效果，比如不断在更新的 **Chrome** 等；
 - `-webkit-`：浏览器私有前缀，代表的是 **WebCore** 内核的 **webkit** 浏览器，主要是 **Chrome** 和 **Safari** 中；
 - `-moz-`：浏览器私有前缀，主要代表是 **Gecko** 内核的 **Mozilla** 浏览器，主要是 **Firefox** 中；
 - `-ms-`：浏览器私有前缀，主要代表的是 **Trident** 内核的 windows 下的 **IE** 浏览器；
 - `-o-`：浏览器私有前缀，主要代表的是 **Presto** 内核的 **Opera** 浏览器；
- **第三方插件**：这个或许可能就不提了，在网络上搜索相关的前端开发插件会有一堆的结果出现，关键还是看个人的“手感”如何，而且后面提到的都是一些基础的代码编写，用浏览器自带的就 OK 了；
- **手册**：相关的 **HTML 手册** 和 **CSS 手册** 还是有必要保留一份，便于及时翻阅查看，这里稍微罗列几个仅供参考；
 - CSS 参考手册 <http://www.w3school.com.cn/cssref/>
 - CSS 参考手册 <http://css.doyoe.com/>
 - CSS 参考 <https://developer.mozilla.org/zh-CN/docs/Web/CSS/Reference>

- HTML 标签列表 <http://www.w3school.com.cn/tags/>

Nº2.1 我们不谈div+css

现在很多人还是一直在说什么 **div+css**，这并不是我们所要谈的，在一个页面中不是只有一个 **div** 标签存在，一个页面是有很多个不同的**HTML**标签来描述的，然后通过**CSS**来增强页面的视觉效果。简单来说，在一个Word文档中，大家应该都会去选择不同级别的标题来描述，也会去给一些重要的文字去加粗加强，最后还会去选择不同的颜色以及段落缩进来加强一个Word文档的表现效果。其实这就跟我们在网页中要去表现的是一样的。

所以，当我们要去编写一个页面的时候，要去考虑什么地方是段落，什么地方是标题，什么地方我们需要加强突出显示，而不是盲目去使用 **div** 标签。具体每个标签有什么作用只要大家去看看 **HTML** 标签列表 <http://www.w3school.com.cn/tags/> 就可以了解了。

了解**HTML**标签不是这次的重点，也不要太在意不同浏览器给每个标签所设定的最初的效果，因为我们最终是可以通过**CSS**去覆盖修改的。当然，也不是所有的都可以覆盖，比如一些表单的样式，这个后续会提到。

Nº2.2 我们要懂得搜索

在后续的网页制作过程中我们总是会遇到各种各样的问题，对于这类问题，就目前来说，已经有很多前人经历过，并且做过总结，所以，善于使用搜索对我们初学来说是一个很大的捷径。当然，如果平时有看一些博客并做过笔记的话，那么相信在这些属于你个人的总结中也会得到很多的帮助。

那么当我们遇到问题的时候应该怎么去搜索呢，其实对于这块，我最大的感受就是，你觉得遇到什么问题你就直接在百度或者谷歌的搜索框中输入你想要知道的答案就差不多了。

比如我们遇到在**IE6**中看到两个 **div** 标签的间距特别大，相对于其他浏览器似乎是有两倍的间距大小时，或者你不知道是两倍的间距时，可以这么搜索，输入：

为什么两个div之间IE6中的间距特别大

那么就可以看到有很多的结果出现了，将近**31200**个答案，然后分别去看一下，基本上也就能明白一些了。

The screenshot shows a Google search results page. At the top, the query "为什么两个div之间IE6中的间距特别大" is entered. Below the search bar, there are several navigation links: 网页 (Web), 新闻 (News), 贴吧 (Baidu Tieba), 知道 (Zhihu), 音乐 (Music), 图片 (Images), 视频 (Videos), 地图 (Maps), 文库 (Library), and 更多» (More). A red arrow points to the search result count "百度为您找到相关结果约31,200个". The first result is titled "怎么解决ie6中div与div之间间距较大的问题? - HTML / C... 大家论坛" and includes a snippet of code: "2011年12月13日 - 用了padding但是还是不行,他只隔开了内容与元素之间的距离,我用margin隔开两个div之间的距离,在ie7、8还有火狐都行,就是在ie6中距离会变得很大,到底怎... club.topsage.com/thread... - 百度快照 - 87%好评". The second result is "【求助】怎么解决ie6中div与div之间间距较大的问题?_css吧_百度贴吧" with a snippet: "10条回复 - 发帖时间: 2011年12月13日 怎么解决ie6中div与div之间间距较大的问题?请各位帮帮忙,谢谢啦! () yanghao...我估计是你的两个div之间有空格。比如这么写:<div id="div1">aaaa</div><... tieba.baidu.com/p/1322... - 百度快照 - 78%好评". The third result is "IE6 两个div有间隙的问题(IE 3px bug) _Div+CSS教程_CS..._脚本之家" with a snippet: "2014年7月25日 - 在IE7中两个div是紧挨着的,但是在IE6中会出现两个div之间出现3px左右的间隙... 在IE7中两个div是紧挨着的,但是在IE6中会出现两个div之间出现3px... www.jb51.net/css/191..... - 百度快照 - 75%好评". The fourth result is "一个关于ie6中上下的div和div之间出现了间距的问题_百度知道" with a snippet: "3个回答 - 最新回答: 2010年05月29日 - 30人觉得有用 最佳答案: 楼主,您好,我是专门对CSS HACK研究的人,希望我的答案能帮助你解决这个问题 你在你定义的类里边添加_line-height:1... 更多关于为什么两个div之间IE6中的间距特别大的问题>> zhidao.baidu.com/link?... - 百度快照 - 80%好评".

当然如果知道这是什么原因导致的，而不知道怎么去解决，只是听说过的话，那么就更简单的，输入你所你知道的那个原因，相信你也能找到解决方案了。

其实搜索也就这么一回事，并不是一定要知道什么关键词才可以去搜索，在国内，百度还是很强大的。当然啦，如果可以的话，其实谷歌也是一个很不错的选择。

Nº2.3 还要懂得如何提问

或许可能在多次搜索的结果之后还是没找到解决方案，同时这个时候身边没有同事或者朋友可以帮忙，那么可以考虑一些有同行在的专业性的QQ群或者问答社区。不过在这个时候要去提问的话，最好能做到这么几点，大家也才能更好的帮助你。

- 描述清楚具体的问题；
- 配图说明，配图可选择你在某个浏览器中你看到的正确的效果和在某个浏览器中你看到的不正确的效果，或者是你喜欢达到的效果和你目前所见到的效果截图；
- 适当的代码说明，最好是有一个**demo**之类的效果可以提供给他人；

基本上有图有代码，大家能帮你解决的效率会高很多，而千万不要只是丢一张图或者一小段你认为是相关的代码，这样是不合理的。对于**CSS**而言，是存在样式属性的继承性和覆盖性的，所以还是需要自己适当的剥离一下代码，或者做成一个简单的**demo**给他人看效果。

三、理论概念

在很多书籍中，前面都会跟大家讲**HTML**和**CSS**的一些发展史，然后再讲什么**web标准**之类的理论性的东西，对于这类东西，看来看去也就那样，无非就是说各大浏览器厂商什么什么，W3C组织成员怎么怎么啦之类的blablabla一堆。这些东西如果大家有兴趣，就百度一下，或者看一下[维基百科](#)也可以，历史课我一向不及格，肯定说不好什么。

Nº3.1 CSS特性

提到**CSS**的特性，可以从**CSS**的全称上了解到一些。

- 英文全称：Cascading Style Sheets
- 中文全称：层叠样式表

从这个名称中我们可以得到一个关键词就是层叠，拥有层叠特性的**CSS**可以让我们在编码的时候去自由组合，从而得到一些属性的**继承**、**叠加**，以及我们可以去**覆盖**一些样式属性。

Nº3.1.1 继承

所谓继承无非就是儿子得到老子的一些东西，那么在**HTML**中我们所要知道就是父级和子级的关系存在。

```
<div>这个div是最外层的父级
  <p>这个p是a的父级，div的子级
    <a href="">这a呢是p的子级，也可以看成是div的子级</a>
  </p>
</div>
```

就这么一个简单的层级嵌套，我们可以得到的信息就是图中所描述的一样：**div**包含了**p**，然后**p**又包含了**a**，就这样一层层嵌套下来，形成了父级与子级的关系。那么在我们写**CSS**的时候，这个嵌套对样式又有什么影响呢？

假设我们写了一个样式如下：

```
div {  
    color: #FF0000;  
}
```

这个代表着的是我们要让 `div` 标签中的文本颜色是红色，那么我们得到的结果就会是这样：

这个div是最外层的父级

这个p是a的父级，div的子级 这a呢是p的子级，也可以看成是div的子级

可以看到p标签中的文字颜色也变成了红色，这就是继承。细心的你或许看到了，那为什么 `a` 标签里的文字没有继承 `div` 的红色呢，`a` 难道不是 `div` 的子级元素吗？不，`a` 是 `div` 的自己元素，而且 `a` 其实也继承 `div` 的红色属性，只不过这个时候的 `a` 有另外一种情况发生了，这就是下面要说的覆盖。

Nº3.1.2 覆盖

所谓的覆盖就是通过选择符的优先权重级别（后续会提到这个权重）来加强当前标签的**CSS**属性，前面提到的 `a` 标签就是有这个情况，虽然我们没有定义这个标签的样式，但是浏览器在自带的默认样式中有一些样式代码，从而导致 `a` 看起来没有继承 `div` 的属性。

```
a:-webkit-any-link {           user agent stylesheet  
    color: -webkit-link;  
    text-decoration: underline;  
    cursor: auto;  
}  
  
Inherited from div  
div {                         test.html:7  
    color: #ff0000;  
}
```

从这个截图中我们可以看到，`div` 标签中的红色被删除 `color: #ff0000;`，而上面还有一个 `color: -webkit-link;` 属性存在。其实这个就是因为**CSS**的覆盖特性导致的。

而如果我们把 `color: #FF0000` 定义在 `a` 标签上，那么情况就又变了，这个时候 `a` 的文字颜色变成红色了，样式覆盖的对象就变了。

```
a {  
    color: #ff0000;  
}  
  
a:-webkit-any-link { user agent stylesheet  
    color: -webkit-link;  
    text-decoration: underline;  
    cursor: auto;  
}
```

这里 `a:-webkit-any-link` 的属性相对比较特殊，牵扯到了 `user agent stylesheet`（用户代理样式），这个主要是浏览器默认的样式，需要针对性去定义才能覆盖原有的样式，而无法通过继承的方式去修改。

No3.1.3 叠加

所谓叠加从字面上其实我们也可以猜到一二了，简单来说就是把几个属性值累加在一起。比如我们的 **HTML** 结构是这样的：

```
<p class="text">在这个p标签上还有一个类名是 text </p>  
<p>这个p标签上是没有类名的</p>
```

那么假如我们给 `p` 标签定义了一个样式，让文字变成红色：

```
p {  
    color: #FF0000; /* p标签的文字颜色是红色 */  
}
```

我们得到的效果就是这样的：

在这个p标签上还有一个类名是 text
这个p标签上是没有类名的

这个时候我们再给 `.text` 这个类名增加一些样式：

```
.text {  
    font-weight: bold; /* .text这个类名的标签中的文字加粗 */  
    text-decoration: underline; /* .text这个类名的标签中的文字  
    有下划线 */  
}
```

这样我们就可以看到有 `.text` 这个类名的 `p` 标签上的文字样式就相对应的增加了，而没有 `.text` 这个类名的 `p` 标签还是保持着一个简单的红色文字，并没有加粗和下划线的出现。

在这个p标签上还有一个类名是 text

这个p标签上是没有类名的

Nº3.2 基本语法

在**CSS**也不知道是不是该叫语法，毕竟**CSS**不属于编程语言，只是没有逻辑的描述型语言。语法这个词或许真的是有点不合适，一般有人叫“规则”，不过无论叫什么，暂且跳过，我们就先叫语法吧。**CSS**的基本语法很简单，主要是有两部分组成：选择符和属性声明。

- 选择符有多种表现形式，后续我们会谈到一些常用和不常用的一些选择符，至于一些目前还在开发阶段的可用性几乎为零的选择符就不会去提到；
- 属性声明主要是有属性和属性值两种组成，并且通过冒号 `:` 隔开，属性值后面用分号 `;` 结束；



选择符的表现方式有多种，暂且先跳过，后续做说明。对于属性声明这块，在 `{}` 花括号内的最后一个属性值是可以不需要在后面带上分号 `;` 的，就比如：`.className { color: #FF0000 }` 和 `.className { color: #FF0000; }` 是相同的，唯一的差别就是最后一个分号 `;` 是否存在。

Nº3.2.1 注释

任何一种代码都存在着注释，合理的注释能够给后期自己看代码的时候带来很多便利，如果是在团队中多人合作的话，也能让其他成员更清晰知道某段或者某行的代码作用。那么在**CSS**中的注释是什么呢，很简单，只需要使

用 `/* */` 包含就可以了。

以 `/*` 开始，`*/` 结束，这中间你可以用多行的形式来描述，也可以直接一行的方式来描述，并且注释的内容也可以在 `{}` 内部出现，只要不断开属性与属性值，基本上来说就没事，要不很有可能会让别人浏览器误以为是某些特殊性的属性声明，导致解析错误。

正确的注释方式：

```
/*
这里将开始一段多行注释
看到的内容都是注释
不会影响到任何东西，因此可以做详细的css说明
*/
p {color:red;}
/* 然后也可以是单行的一个注释，做简单的说明 */
strong {color:green; /* 如果你愿意的话，也可以在这里加注释 */}
font-size:12px;}
```

合理的使用注释，能够给你的代码带来更大的可读性，但也别盲目使用无意义的注释。

最后发布代码上线的时候，可以使用第三方工具将你的注释删除，从而减小代码文件。文件越小，用户那边加载肯定也是越快。

Nº3.3 引用方式

当我们在编写页面的时候，调用**CSS**的方法一般来说有四种，不过常用的就三种方式：

- `link` 标签的引入；
- `style` 标签的插入；
- **HTML**中的 `style` 属性方式插入；
- **CSS**中的 `<import>` 方式导入，这种方式目前已经没人去使用了；

Nº3.3.1 `link` 标签的引入

这种方式引入的**CSS**一般是引入一个 `.css` 的文件，如：

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

直接引入一个 `style.css` 的文件，所有的**CSS**样式都是在这个文件中，通过选择符来匹配**HTML**中相对应的标签元素。

需要注意的是，`link` 标签一般都是在**HTML**代码中的 `head` 标签内，如：

```
<head>
<meta charset="UTF-8">
<title>Document</title>
<link rel="stylesheet" href="style.css">
</head>
```

而在 `head` 标签内还包含了其他更多的一些标签，大部分都是一些说明性或者是网页某些特殊功能性的标签，当然也可以包含调用 `JavaScript` 的标签，这里就不多提了，关键是大家知道 `link` 标签主要是在什么地方出现就可以了。当然，在 `link` 标签中还有一些属性，比如 `media` 之类的，我们也就暂且跳过了，有兴趣的朋友可以打开这个网页看一下http://w3school.com.cn/tags/tag_link.asp。

No3.3.2 `style` 标签的插入

在**HTML**页面代码中直接使用 `style` 标签插入，并包含相对应的**CSS**内容即可，后续该文档中提到的方式都是以这种方式来做说明。这种方式比较适合单页面或者CSS内容比较少的情况下去使用，因为当我们使用了 `style` 标签插入的话，仅对当前页面有效，而通过 `link` 方式调用的话，只要有调用的页面都会有效，在做网站的时候，还是推荐使用 `link` 的方式引入**CSS**文件。

```
<head>
<meta charset="UTF-8">
<title>Document</title>
<style>
p {
    color: #FF0000; /* p标签的文字颜色是红色 */
}
</style>
</head>
```

那么这个 `style` 标签的插入呢，一般我们也是放在 `head` 标签内，然后在 `style` 标签中就是写我们所需要的**CSS**内容就可以了。

No3.3.3 **HTML**中的 `style` 属性方式插入

一般来说不太建议这样操作，**CSS**最初的作用是是为了能够更好的可复用性的去修饰**HTML**标签，让页面变得更漂亮，而如果我们使用这种方式的话，那么就只能针对当前的某个标签起作用，对其他标签是没有任何效果，比如：

```
<p>这里是有没有style属性的</p>
<p style="color:red;font-size:24px;">增加style属性来改变样式
</p>
```

当页面中出现这样的**HTML**代码时，虽然我们看到 `style` 属性中的内容跟我们在其他两种方式中写的**CSS**很类似（缺少了 `{}` 花括号和选择符），但还是会有作用。可这个方式仅仅只能针对该 `style` 属性值所在的**HTML**标签有效，效果如下：

这里是有没有style属性的

增加style属性来改变样式

虽然效果有，可维护性就变得很差了，所以这种方式能不用就不要用了。

Nº3.3.4 CSS中的 `@import` 方式导入

评价最差的一种方式，大家知道一下就可以了，能用也不要，改为 `link` 标签引入的方式吧。有兴趣的朋友看这里<http://www.jb51.net/css/67778.html>了解一下就好了。

Nº3.4 权重优先级

鉴于**CSS**的特性，在我们考虑使用怎么触发**CSS**属性的时候，也就是我们通过哪种选择符的方式让**HTML**中相对应标签有样式时，要稍微有所考虑，否则你会发现最终的效果或许并不是你想要的，但是样式其实已经起作用了，但却被别的给覆盖了。那么这就是我们所要讲的权重优先级。

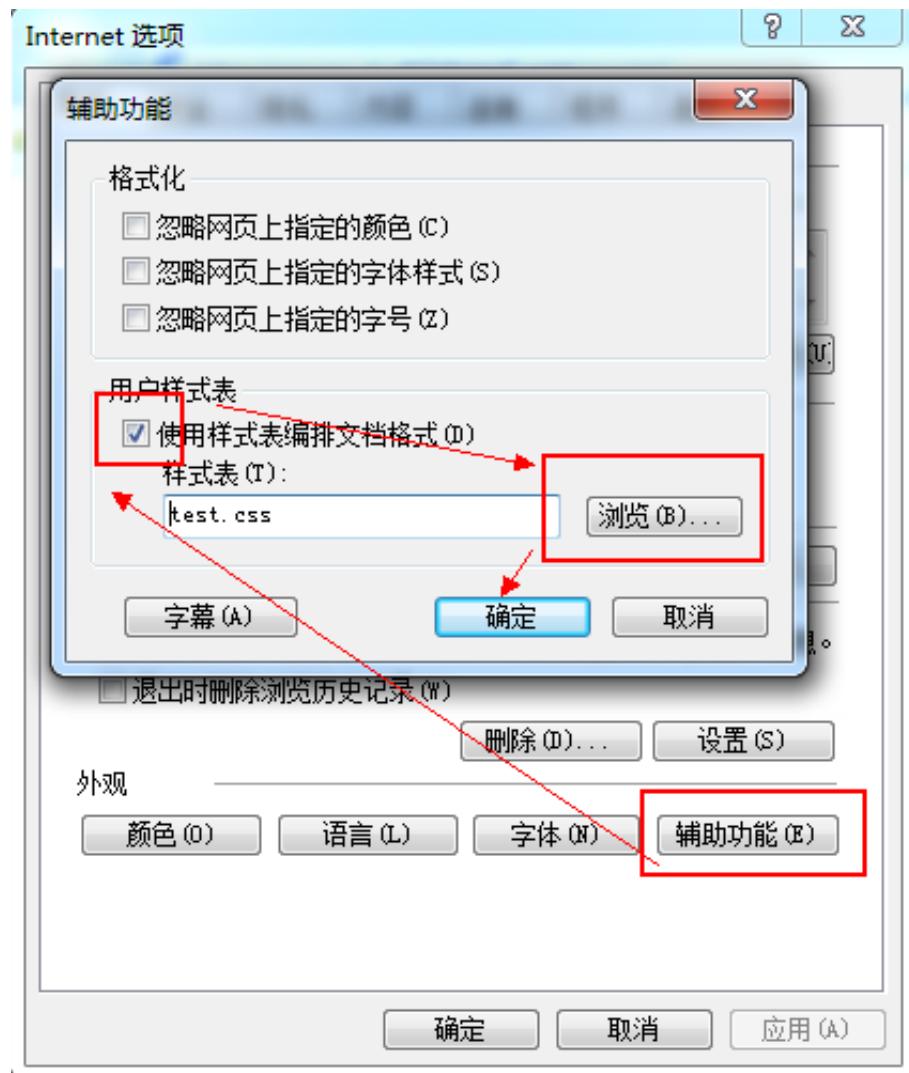
Nº3.4.1 CSS的来源

所谓的**CSS**来源，其实就是浏览器在解析网页的时候，会从什么地方获取**CSS**代码来修饰我们的页面。这来源主要有三种途径：

1. **user agent stylesheet** 用户代理的默认CSS，也就是浏览器默认的一些样式，在前面我们提到的覆盖特性中有一个截图可以看到右上角是有一个灰色的标记所代表就是浏览器的内置样式表；

```
a:-webkit-any-link {           user agent stylesheet
  color: -webkit-link;
  text-decoration: underline;
  cursor: auto;
}
Inherited from div
div {                           test.html:7
  color: #ff0000;
}
```

2. **author style sheets** 开发者定义的CSS，也就是我们作为一个开发人员所编写的代码在被引入到页面后的样式；
3. **user style sheets** 用户自定义的CSS，用户通过浏览器的第三方插件或者浏览器自带的一个插入样式表的功能所加载的样式，这类用户一般有这么几种：
 - 安装第三方浏览器插件来改变某些网站的样式，比如 <https://userstyles.org/styles/72212/v6> 这个通过 Stylish 插件增加了样式后，针对新浪微博首页做了一些处理，去掉一些广告顺带改变了一下页面布局；
 - 安装某些软件通过HTML中的一些关键词屏蔽了广告；
 - 用户获取的CSS文件或者自己编写的CSS样式，直接通过浏览器的菜单来加载。记得很早之前浏览器中是有看到过，但现在我却没在 OS X 系统中发现这个菜单，或许是记错了，也或许在 windows 系统中可以找到，有兴趣的朋友可自行查找一下，比如 windows 下的IE选项设置中；



简单来说，就是用户如果通过其他方式加载了**CSS**样式，那么可以修改任何网页的页面布局，而作为网页开发者来说，是可以覆盖部分浏览器内置的样式。

注意：这里提到开发者的样式可以覆盖部分浏览器内置的样式，主要体现在表单元素中，比如 `select` 标签是最显著的，尤其是在**IE**浏览器中，特别是低版本的**IE**浏览器。

No3.4.2 需要注意的 `!important` 特殊性

这种处理是对 `user` 和 `author` 权限的平衡，一般情况下 `author` 的权重高于 `user`，这是对 `author` 创作的尊重；但对于特殊用户（例如：色盲、色弱、近视用户），他们具有特殊需求（例如 大字号、特定的颜色对比），这种需求可以通过 `user !important` 声明来获得最高优先级。

所以，对于前面提到**CSS**来源的优先级，在基于 **important** 的前提下，优先级排列应该是这样：

1. user agent declarations
2. user *normal* declarations
3. author *normal* declarations
4. author **important** declarations
5. user **important** declarations

纠正：来源 [hbkdsm](#) 的提示，以上的顺序有误，纠正如下，有兴趣的朋友可以打开这个页面查看 <https://github.com/linxz/blog/issues/2>

那么这个 **!important** 我们在样式中是怎么使用的呢，其实很简单的，只要在属性值后面加上就可以了，比如下面这个**HTML**中，第二个 **p** 标签中的文字是红色的，第一个是默认的颜色：

```
<p>这里是有没有style属性的</p>
<p style="color:red;font-size:24px;">增加style属性来改变样式
</p>
```

那么这个时候我们希望通过在样式修改，于是在 **style** 标签中插入了这么一段：

```
p { color: blue; }
```

然而并没有什么卵用，第二个 **p** 标签中的文字还是红色，而第一个却变成了蓝色，这并不是我们所想要的。

这里是有没有style属性的

增加style属性来改变样式

这是为什么呢，其实这里就涉及到一个选择符方面的权重优先级，等一下我们再说。那么这个时候我们如果想要所有的 **p** 标签文字都变成蓝色的话，就需要通过 **!important** 的方式来提升权重优先级，所以样式部分应该是这样写：

```
p { color: blue !important; }
```

那么这个时候我们在页面中看到的就是这样了：

这里是有没有style属性的

增加style属性来改变样式

No3.4.3 选择符的权重优先级

选择符（或者叫选择器）在**CSS**中的作用是非常大的，每个**HTML**标签的样式都是通过选择符来操作的，只有在**CSS**选择符匹配到相对应的元素才会有样式。具体关于选择符的内容我们后续再谈，这里先知道这么几种涉及到权重基本计算选择符：

- 行内样式，也就是**HTML**标签元素的 `style` 属性，如
`<p style="color:red"></p>`；
- ID选择符，通过**HTML**标签元素的 `id` 属性来匹配，如
`<p id="idName"></p>`；
- 类选择符，通过**HTML**标签元素的 `class` 属性来匹配，如
`<p class="className"></p>`；
- 标签选择符，通过**HTML**标签元素的标签名来匹配，如 `<p><p>`；
- 通配符选择符，在**CSS**中直接使用 `*` 星号来匹配所有**HTML**标签元素，如
`* {color:red}`；

在编写**CSS**的时候，如果一个不小心就会发现页面的样式并不是我们所想要的，那么有部分可能性就是因为没有注意到以上几种选择符的权重优先级。那么我们应该怎么去计算这些权重呢，其实很简单。

通配符选择符

这个是权重最低的一种，虽然我们很少会去用，但偶尔还是会去用的，比如在写**CSS reset**部分的时候：

```
* {margin: 0; padding: 0;}
```

总之吧，无论是在什么时候出现，虽然会把样式作用到**HTML**标签，优先级是最低最低的，如果用数字表示的话，那就是 **0**，一个永远不会被累加的数字。暂且就这么感受吧。

行内样式

这个是权重最高的一种，在页面中经常会看到一些通过**JavaScript**方式将一些**CSS**样式写入到**HTML**的 `style` 属性中，从而去覆盖其他样式。对于这种方式，如果不是因为**JavaScript**写入的话，平时我们能不这么写的话，那就不要这样写，否则要修改样式就只能通过修改**HTML**文件来实现，而不能利用独立的**CSS**文件。

简单来说，用了行内样式，由于权重过高，而且**CSS**复用性几乎为0，后期维护成本比较大。

选择符权重计算方式

前面提到的通配符选择符我们是用了一个 **0** 来表示，那么对于其他几个，我们也同样用数字来表示，然后按照固定模式来计数累加就可以。假设初始值都为 **1** 的时候，那么就有可能是这样的一个情况：

行内	<code>id</code>	<code>class</code>	标签	通配符
1	1	1	1	1

这里只是假设同时都有样式被匹配到某个**HTML**标签中的时候，就比如：

```
<p id="linxz" class="lin_xz" style="margin: 5px;">增加style  
属性来改变样式</p>
```

然后在**CSS**部分写的是：

```
* {margin: 0;}  
#linxz {margin: 10px;}  
.lin_xz {margin: 20px;}  
p {margin: 30px;}
```

这个时候我们会发现最终的样式其实就是只有 `margin: 5px;` 这个。

不过在我们编码的时候，并不会这样去写。因为**HTML**标签是可以嵌套的，会有多层次级父级元素，那么就有可能看到这样的一个**HTML**结构（300来层的嵌套，夸张一下）：

```

<div id="father_000" class="father_000">
    <div id="father_001" class="father_001">
        .....
        <div id="father_300" class="father_300">
            <p id="linxz" class="lin_xz" style="margin: 5px;">增加style属性来改变样式</p>
        </div>
        .....
    </div>
</div>

```

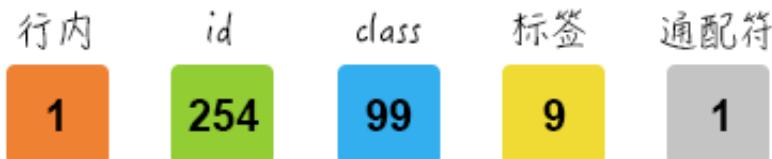
然后我们在样式写：

```

* {margin: 0;}
#father_000 #father_001 ..... #father_300 #linxz {margin: 10px;}
.father_000 .father_001 ..... .father_300 .lin_xz {margin: 20px;}
div div ..... div p {margin: 30px;}

```

这样很夸张的嵌套也并不会改变什么，只是让某条**CSS**规则在权重增加很多，但无法跨越过自身的级别去超越权重，大概情况就像这样。



再多的数量也只是会在自身区域中增加，就如上图所示，**254** 个 `id` 的规则也抵不过一条行内样式的规则，同理 **99** 个 `class` 的类选择符也抵不过一个 `id` 的ID选择符。用**CSS**代码来描述的就是：

```

* {margin: 0;}
#linxz {margin: 10px;} /* 这个规则会大于其他选择符，但比行内样式权重低 */
.father_000 .father_001 ..... .father_300 .lin_xz {margin: 20px;}
div div ..... div p {margin: 30px;}

```

权重计算中例外的 `!important`

在前面我也提到过 `!important` 这个特殊的关键词，在选择符计算中表现是最为明显的。上面所提到的权重计算都是比较正常的计算，但如果有一些特殊情况出现的时候，比如我们希望 `id` 中的样式能够覆盖行内样式，那么就需要在 `id` 选择符的规则中增加一个 `!important`。

```
#linxz {margin: 10px !important;} /* 这时候的id选择符优先级是最高 */
```

如果有多条属性要提升权重优先级的话，那么就要增加多个：

```
#linxz {  
    margin: 10px !important;  
    color: red !important;  
    font-size: 20px !important;  
    padding: 20px; /* 当没有 !important 的时候，权重计算还是跟之前一样 */  
}
```

关于权重的内容如果有兴趣的话，可以到网络上搜一搜，会找到更多的一些内容，然后再深入了解就可以了。