第 5 章

规则 5——将样式表放在顶部

Rule 5: Put Stylesheets at the Top

Yahoo!的一个负责主要门户的团队向他们的页面中添加了大量 DHTML 特性，并尝试确保它们不会对响应时间产生负面影响。其中一个复杂的 DHTML 特征是在发送 Email 消息时弹出一个 `DIV`，这不属于实际呈现页面的一部分——它只在页面加载完毕后，用户单击按钮来发送 Email 消息时才会访问。由于在呈现页面时不需要使用它，前端工程师将弹出式 `DIV` 所需的 CSS 放在了外部样式表中，并在页面的底部添加了相应的 `LINK` 标签，期望将其包含在页面的末尾能够使其加载更快。

这背后的逻辑是有意义的。其他很多组件（图片、样式表、脚本，等等）是呈现页面所必需的。由于组件（通常）是按照它们在文档中出现的顺序下载的，将有 DHTML 特性的样式表放在最后可以使得很多重要的组件首先被下载，从而得到一个加载很快的页面。

果真如此吗？

在 Internet Explorer（仍然是最流行的浏览器）中，实际产生的页面比原来的明显缓慢。在尝试寻找一种能为页面加速的方法时，我们发现将 DHTML 特征的样式表放在文档顶部——`Head` 中——能使页面加载得更快。这与我们所期望的相矛盾。将样式表放在前面，会延迟页面中其他重要组件的加载，怎么会改善页面加载时间呢？对此更深入的研究导致了规则 5 的出现。

逐步呈现

Progressive Rendering

关心性能的前端工程师都希望页面能逐步地加载，也就是说，我们希望浏览器能够尽快显示内容。这对于有很多内容的页面以及 Internet 连接很慢的用户来说尤其重要。为用户提

供可视化回馈的重要性已经有了很好的研究和记载。Jakob Nielson——可用性工程师先驱——在其 Web 文章（注 1）中从进度指示器的角度强调了可视化回馈的重要性。

> 进度指示器有三个主要优势——它们让用户知道系统没有崩溃，只是正在为他或她解决问题；它们指出了用户大概还需要等多久，以便用户能够在漫长的等待中做些其他事情；最后，它们能给用户提供一些可以看的东西，使得等待不再是那么无聊。最后一点优势不可低估，这也是为什么推荐使用图形进度条而不是仅仅以数字形式显示预期的剩余时间。

在我们这里，HTML 页面就是进度指示器。当浏览器逐步地加载页面时，页头、导航栏、顶端 logo 等，所有这些都会为等待页面的用户提供视觉反馈。这改善了整体用户体验。

将样式表放在文档底部会导致在浏览器中阻止内容逐步呈现。为避免当样式变化时重绘页面中的元素，浏览器会阻塞内容逐步呈现。规则 5 对于加载页面所需的实际时间没有太多影响，它影响更多的是浏览器对这些组件顺序的反应。实际上，用户**感觉缓慢**的页面反而是可视化组建加载得**更快**的页面。在浏览器和用户等待位于底部的样式表时，浏览器会延迟显示任何可视化组件。下一节给出的示例展示了这一现象，我将其称之“白屏”。

sleep.cgi

在为这一现象创建示例时，我开发了一个工具，它在展示延迟的组件如何影响 Web 页面上十分有效——*sleep.cgi*。这是一个简单的 Perl CGI 程序，它需要下面这些参数——

`sleep`

将响应延迟多长时间（按秒计）。默认值为 `0`。

`type`

返回的组件类型。可取的值只有 `gif`、`js`、`css`、`html` 和 `swf`。默认值为 `gif`。

`expires`

下面三个值之一：`-1`（返回一个已经过时的 `Expires` 头）、`0`（不返回 `Expires` 头）和 `1`（返回一个未来的 `Expires` 头）。默认值为 `1`。

注 1：Jakob Nielson，“Response Times: The Three Important Limits”，*http://www.useit.com/papers/responsetime.html*。

`last`

取 `-1` 返回一个 `Last-Modified` 头，其时间戳和文件的时间相等。取 `0` 则不返回 `Last-Modified` 头。默认值为 `-1`。

`redir`

取 `1` 则产生一个 302 响应，重定向回同样的、不带 `redir=1` 的 URL。

第一个示例需要一些慢速的图片和一个慢速的样式表。这通过下面得到 *sleep.cgi* 的请求来实现——

```
<img src="/bin/sleep.cgi?type=gif&sleep=2&expires=-1&last=0">
<link rel="stylesheet" href="/bin/sleep.cgi?type=css&sleep=1&expires=-1&last=0">
```

图片和样式表都使用 `expires=-1` 选项时，会得到一个拥有已过期的 `Expires` 头的响应。这样就避免了组件被缓存，你可以重复运行这个测试，每次都能得到同样的体验（我还为每个组件的 URL 添加了唯一的时间戳，进一步防止缓存）。为了减少这个测试中的变量，我指定了 `last=0` 来从响应中移除 `Last-Modified` 头。图片需要两秒的延迟（`sleep=2`），而样式表只需要一秒的延迟（`sleep=1`）。这确保了看到的延迟不是由样式表的响应时间造成的，而是由阻塞行为造成的（这也是该页面要测试的）。

放大组件的响应时间，使得我们可以形象地看到它们在页面加载和响应时间上的效果。我公开了该 Perl 代码，这样其他人也可以使用该代码完成他们的测试（*http://stevesouders.com/hpws/sleep.txt*）。请将代码复制到一个可执行文件中并为其命名为 *sleep.cgi*，然后将其放到 Web 服务器上的一个可执行目录中即可。

白屏
Blank White Screen

这一节展示了两个 Web 页面，它们只在一个方面有所不同——样式表是在页面的顶部还是底部。我们可以看到这给用户体验带来了多么大的差别！

将 CSS 放在底部
CSS at the Bottom

第一个示例展示了将样式表放在 HTML 文档底部所带来的不足。

将 CSS 放在底部的示例

http://stevesouders.com/hpws/css-bottom.php

注意将样式表放在文档底部是如何延迟页面加载的。这个问题很难跟踪到，因为它只发生在 Internet Explorer 中，并且依赖于页面是如何加载的。在使用过这个页面之后，你会发现偶然发生页面加载缓慢。当发生这种现象时，页面会完全空白，直到页面所有的内容同时涌上屏幕，如图 5-1 所示，逐步呈现被禁止了。这是一种不好的用户体验，因为没法向用户确保他的请求正在被正确地处理，用户会因为不知道发生了什么而离开。这就是用户离开你的网站投奔竞争对手的原因。

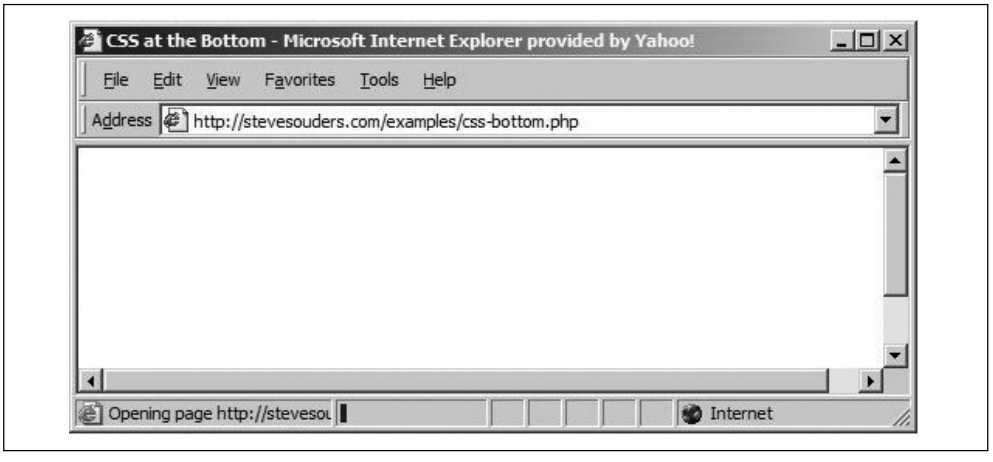

图 5-1：白屏

在 Internet Explorer 中，将样式表放在文档底部会导致白屏问题的情形有以下几种：

在新窗口中打开时

单击示例页面中的“new window”链接，在新窗口中打开“CSS at the Bottom”。用户通常在跨站导航时打开新窗口，如从搜索结果页导航到实际的目标页。

重新加载时

单击刷新按钮是另外一种导致白屏的方式，这是一种常见的用户操作。在页面加载时最小化然后恢复窗口就能看到白屏。

作为主页

将浏览器默认页设置为 *http://stevesouders.com/hpws/css-bottom.php* 并打开新的浏览器窗口就会导致白屏。规则 5 对于那些希望其网站能被用作主页的团队来说尤其重要。

将 CSS 放在顶部
CSS at the Top

为了避免白屏，请将样式表放在文档顶部的 HEAD 中。经过这样修改后的示例网站称作“CSS at the Top”，解决了所有错误情况。不管页面是如何加载的——在新窗口打开、重新加载者作为主页——页面都是逐步呈现的。

将 CSS 放在顶部的示例

http://stevesouders.com/hpws/css-top.php

搞定！只有一点比较复杂的地方需要指出。

将样式表包含在文档中有两种方式：使用 LINK 标签和@import 规则。使用 LINK 标签的示例如下所示：

```
<link rel="stylesheet" href="styles1.css">
```

下面是使用带有@import 规则的 STYLE 标签示例：

```
<style>
@import url("styles2.css");
</style>
```

一个 STYLE 块可以包含多个@import 规则，但@import 规则必须放在所有其他规则之前。我曾遇到过没有注意到这一点的情形，开发人员需要花时间去尝试检查为什么@import 规则中的样式表没有加载。出于这一原因，我更喜欢使用 LINK 标签（需要了解的东西较少）。除了语法更简单外，使用 LINK 标签来代替@import 还能带来性能上的收益。@import 规则有可能会导致白屏现象，即便把@import 规则放在文档的 HEAD 标签中也是如此，如下面的示例所示。

将 CSS 放在顶端并使用@import 的示例

http://stevesouders.com/hpws/css-top-import.php

使用@import 规则会导致组件下载时的无序性。图 5-2 展示了前面三个示例的 HTTP 流量。每个页面包含八个 HTTP 请求：

- 一个 HTML 页面
- 六个图片
- 一个样式表

css-bottom.php 和 *css-top.php* 中的组件是按照它们出现的顺序下载的。然而，尽管 *css-top-import.php* 将样式表放在了文档顶部的 HEAD 中，样式表依然是**最后**下载的，因为它使用了@import。结果，它产生了白屏问题，就像 *css-bottom.php* 一样。

图 5-2 还表明了每个页面加载的整体时间（包括所有页面组建）是一样的——约 7.3 秒。令人惊讶的是，感觉缓慢的页面——*css-bottom.php* 和 *css-top-import.php*——实际下载页面所必需的组件的时间是最短的。它们完成下载 HTML 页面和所有 6 个图片用了 6.3 秒，而 *css-top.php* 用了 7.3 秒来下载页面所必需的组件。css-top.php 多花了 1 秒钟时间，因为它要首先下载样式表，尽管这并不是页面呈现时所必需的。这就使 6 个图片的下载时间延迟了大约 1 秒。尽管花了更多时间来下载所需的组件，但用户感觉 *css-top.php* 显示得更快，因为页面是逐步呈现的。

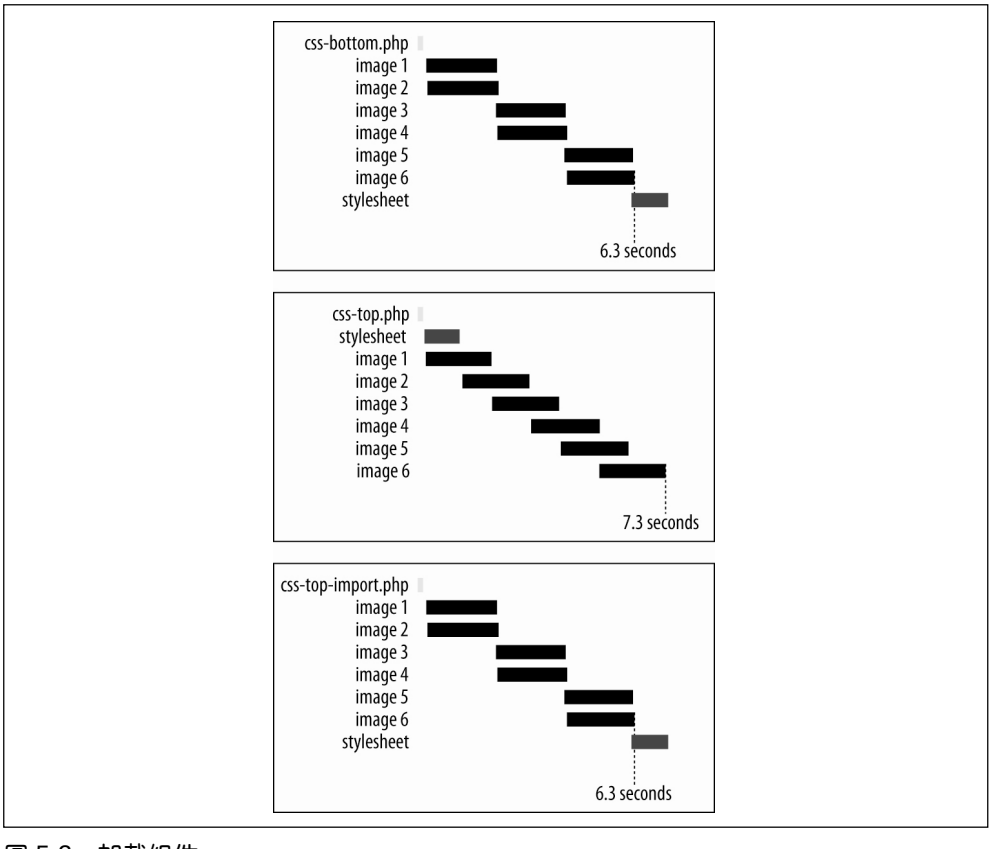

图 5-2：加载组件

太棒了！我们知道怎么做了——**使用 LINK 标签将样式表放在文档的 HEAD 中**。但如果你像我一样，你就会问自己，“为什么浏览器以这种方式工作呢？”

无样式内容的闪烁
Flash of Unstyled Content

白屏现象源自于浏览器的行为。要知道我们的样式表在呈现页面时几乎用不到——它只影响发送 Email 消息时的 DHTML 特性。尽管 Internet Explorer 已经得到了所需的组件，它依然要等到样式表下载完毕之后再呈现它们。样式表在页面中的位置并不影响下载时间，但是会影响页面的呈现。David Hyatt 就为什么浏览器会这样做进行了很好的解释（注 2）。

> 如果样式表仍在加载，构建呈现树就是一种浪费，因为在所有样式表加载并解析完毕之前无需绘制任何东西。否则，在其准备好之前显示内容会遇到 FOUC（无样式内容的闪烁，Flash of Unstyled Content）问题。

下面的例子展示了这一问题。

无样式内容的 CSS 闪烁的示例

http://stevesouders.com/hpws/css-fouc.php

在这个例子中，文档为样式表使用了一个 CSS 规则，但样式表被（不正确地）放在了底部。当页面逐步加载时，文字首先显示，然后是图片。最后，在样式表正确地下载并解析之后，已经呈现的文字和图片要用新的样式重绘了。这就是“无样式内容的闪烁”。应该避免这个问题。

白屏是浏览器在尝试修改前端工程师所犯的错误——将样式表放在文档比较靠后的位置。白屏是对 FOUC 问题的弥补。浏览器可以延迟呈现，直到所有的样式表都下载完之后，这就导致了白屏。反之，浏览器可以逐步呈现，但要承担闪烁的风险。这里没有完美的选择。

前端工程师应该做什么？
What's a Frontend Engineer to Do?

那么如何同时避免白屏和无样式内容的闪烁呢？

在“无样式内容的 CSS 闪烁”示例中，闪烁并不总是发生，它取决于你的浏览器以及如何加载页面。在本章前面的内容中，我曾介绍过白屏仅当在新窗口中加载页面、重新加载和作为主页时在 Internet Explorer 中发生。在这些情况下，Internet Explorer 选择了白屏。然而，如果你单击链接、使用书签或键入 URL，Internet Explorer 选择第二种方式——承担 FOUC 风险。

注 2：David Hyatt，“Surfin' Safari”博客，*http://weblogs.mozillazine.org/hyatt/archives/2004_05.html#005496*。

Firefox 则是一致的——它总是选择第二种方式（FOUC）。所有三个例子在 Firefox 中的行为都是一样的——它们会逐步呈现。对于第一个例子，Firefox 的行为更考虑用户感受，因为样式表对于呈现页面来说并不是必需的，但在“无样式内容的 CSS 闪烁”示例中，用户就不那么幸运了。用户会切实体验到 FOUC 问题，因为 Firefox 是逐步呈现的。

当浏览器的行为不同时，前端工程师应该做些什么呢？

你可以在 HTML 规范中找到答案（*http://www.w3.org/TR/html4/struct/links.html#h-12.3*）：

> 和 A 不一样，[LINK]只能出现在文档的 HEAD 节中，但其出现次数是任意的。

由于历史原因，浏览器支持违反 HTML 规范的页面，这是为了让那些老旧的、不规整的页面也能够浏览，但在处理样式表方面，Internet Explorer 和 Firefox 都要求 Web 开发社区遵循规范。这就导致即使要承担降低用户体验的风险，违反了规范的页面（将 `LINK` 放到 `HEAD` 节的外面）仍然能够呈现。

在努力改善 Web 上最频繁访问的页面时，Yahoo!的门户团队最初将样式表放到了页面底部，结果适得其反。他们发现最佳解决方案就是遵循 HTML 规范，将它放在顶部。两种情况——白屏和无样式内容的闪烁——都不再是风险。如果你的样式表不要求呈现页面，可以想办法在文档加载完毕后动态加载进来，如第 8 章中“加载后下载”一节所述。否则，不管你的样式表在呈现页面时是否必需，都应该遵守这个规则。

使用 LINK 标签将样式表放在文档 HEAD 中