**[翻译]简化CSS选择器**

* 原文：[Simplifying CSS Selectors](http://www.stevesouders.com/blog/2009/06/18/simplifying-css-selectors/)
* 原作者：[Steve Souders](http://stevesouders.com/bio.php)
* 翻译：[ytzong](http://www.99css.com)

本文是《[EvenFaster Web Sites: Performance Best Practices for Web Developers (Paperback)](http://www.amazon.com/gp/product/0596522304?ie=UTF8&tag=stevsoud-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=0596522304)》的最后一章。上篇帖子《[Performance Impact of CSS Selectors](http://www.stevesouders.com/blog/2009/03/10/performance-impact-of-css-selectors/)》（[中文版](http://www.99css.com/2009/06/performance-impact-of-css-selectors.html)）最后提出了一段假设：

对大多数网站而言，优化CSS选择器活得的性能提升很小，不值得去计较。有些配合Javascript交互的CSS规则会明显的拖慢页面。这是应该关注的焦点。所以我开始关注现实中影响页面性能的CSS样式相关的小问题。

我收到了很多反馈。David Hyatt的文章《[Writing Efficient CSS for use in the Mozilla UI](https://developer.mozilla.org/en/Writing_Efficient_CSS)》披露：

样式系统渲染一条规则是从最右边开始之后依次向左移动。在你的小子树(subtree)持续检测的时候，样式系统将继续向左侧移动直到它不匹配CSS规则或匹配错误。

由此得出，我们优化工作的重点应该是：匹配大量页面元素的最右侧的CSS选择器。我上篇博文测试的CSS选择器看起来很费性能，但是按这条新观点审视，我们发觉这其实不值得担心，比如：

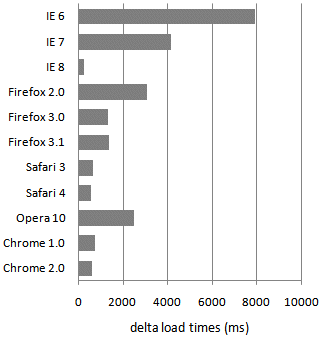
DIV DIV DIV P A.class0007 {}

这个选择器有5层，看起来很复杂，但是我们来看最右侧的选择器 A.class0007 ，我们发现，在整个页面中需要浏览器逆向匹配的只有一个元素。

优化CSS选择器的关键点在于最右侧的选择器，也叫做*key selector*(巧合？)。有一个更昂贵的选择器

A.class0007 \* {}

尽管这个选择器看起来更简单，但对浏览器匹配而言更昂贵。因为浏览器要从右至左，开始后要检查匹配 \* 的所有元素。这意味着浏览器会尝试匹配页面中的所有元素。下图为[普通选择器](http://stevesouders.com/efws/css-selectors/universal.php)与先前的[后代选择器](http://stevesouders.com/efws/css-selectors/descendant.php)加载时间的对比：



它清晰的反映出一个匹配很多元素的key selector会严重的拖慢页面。其他可能会大量增加浏览器工作的key selector包括：

A.class0007 DIV {}

#id0007 > A {}

.class0007 [href] {}

DIV:first-child {}

不是所有的CSS选择器伤害性能，尽管看起来如此。CSS选择器的关键点在于泛匹配的key selector。这对于含有大量DOM元素、CSS规则，更高 reflow 的Web2.0应用更加重要。

Tags: [性能](http://www.99css.com/?tag=%e6%80%a7%e8%83%bd), [翻译](http://www.99css.com/?tag=%e7%bf%bb%e8%af%91)