JavaScript已经是目前最流行的语言了，它能做很多事情 - 网站界面，服务器端，游戏，操作系统，机器人等等很多很多。

不过，说实话，即使它这么疯狂流行，它的性能还没有达到它的极限。是的，它在改进，但是等到它在各个方面赶上本地应用之前，在做一个HYBIRD混合应用时，你还不得不使用一些伎俩来优化它的性能。

Firefox拥有目前最快的JavaScript解析器SpiderMonkey,

有各种各样的让JavaScript的速度更快的努力，其中一个是asm.js. Asm.js是JavaScript是由Emscripten 产生的一个子集，它为C/C++编绎成的JavaScript代码做了很多优化，编译型后的代码很难看，这就是为什么你不能自己写优化后的代码，但它运行非常快。我建议你阅读一下 [这篇文章](http://ejohn.org/blog/asmjs-javascript-compile-target/)

别扯了举个例子吧!

好了，我们的目标是写速度更快的JavaScript代码，这里有让你的代码跑得更快一些的小窍门，以及更好的内存效率。请注意，我不是严格讨论DOM和Web应用程序，它是关于JavaScript的，DOM只是一部分。

眼见为实，我要添加为第一个添加jsperf测试用例，使用的是Firefox38和Chrome39测试。

＃1不要类型转换

JavaScript是动态类型，但如果你想提高速度不要使用该功能。尽量保持变量的类型一致。这也适用于数组，尽管主要是由浏览器都进行了优化,但尽量不要混用不同类型的数组。这就是为何编译成 JavaScript的C/C++代码使用静态类型的原因之一。

2. {
3. var x = '2';
4. var y = 5;
5. x = 2;
6. x + y;
7. }

[测试用例](http://jsperf.com/casting-type/15)

另外: 字符串与数字类型间相互转换

比方说，你必须将字符串转换为数字，parseInt与parseFloat是最好的方法吗？让我们来看看。

2. parseFloat("100")
3. +"100"
4. // 整型
5. parseInt("100", 10)
6. "100"|0
7. "100" >> 0
8. "100" << 0
9. // 仅适用于正数
10. "100" >>> 0

[parseInt 测试](http://jsperf.com/type-cast/3)~ [parseFloat 测试](http://jsperf.com/type-cast/2)

Firefox对位操作进行了优化，运行的代码比parseInt和+运算速度快约99％。而Chrome显然对位运算符没有偏爱，他们比parseInt函数还慢62％。

parseFloat比+运算符在两种浏览器（Firefox 28％，Chrome 39％）上都要快。

因此，如果你在写Node/Chrome或Firefox的应用程序？我认为，一般使用parseInt函数是正确的。

＃2不要重新构造对象

重组对象不便宜，应该避免它：

不要使用delete运算符

删除操作比分配一个null属性慢很多。分配null在两个浏览器都快99％，但它不能修改对象的结构，但删除可以。

编辑：我认为这里有点误导，这并不意味着你不应该使用delete操作符，delete运算符有它自己的使用情况，它可以防止对象的内存泄漏。

[delete vs null](http://jsperf.com/object-structure-change)

不要以后再添加属性

尽量不要在以后再添加属性，最好从一开始就定义对象的架构。这在Firefox中快100％，在Chrome中快89％。

[动态属性VS预先定义结构](http://jsperf.com/object-dynamic-properties)

＃3字符串联连

字符串联连是一个非常昂贵的操作，但是应该用什么方法呢？当然不是Array.prototype.join。

+=运算符似乎比+快很多，他们在两种浏览器上比String.prototype.concat和Array.prototype.join都更快。Array.prototype.join是最慢的，符合市场预期。

[字符串连接测试](http://jsperf.com/string-concat-fast/9)

＃4正确的使用正则表达式

使用RegExp.prototype.exec是没有必要，不是吗？

然而，RegExp.prototype.test和String.prototype.search之间是有性能差异的，让我们来看看哪个方法更快：

[正则表达式的方法](http://jsperf.com/regex-methods-x-1)

RegExp.prototype.exec比String.prototype.match快了不少，但他们是不完全一样的东西，它们的区别超出了本文的范围，看这个 [问答](http://stackoverflow.com/questions/9214754/what-is-the-difference-between-regexp-s-exec-function-and-string-s-match-fun)。

RegEx.prototype.test更快，可能是因为它不返回找到匹配的索引。 String.prototype.search应仅用于找到所需的匹配的索引。

然而，你不应该使用正则表达式来查找另一个字符串的位置，你可以使用String.prototype.indexOf方法。

[String.prototype.search VS String.prototype.indexOf](http://jsperf.com/search-vs-indexof11/4)

另一个有趣的基准是 [String.prototype.indexOf VS RegExp.prototype.test](http://jsperf.com/test-vs-indexof-fast)，我个人预计后者要快，这是在Firefox中发生的事情，但在Chrome中，事实并非如此。 RegExp.prototype.test在Firefox中快32％，而在Chrome中String.prototype.indexOf快33％。在这种情况下，你自己选择喜欢的方式吧。

＃5限制声明/传递变量的范围（作用域）

假如你调用一个函数，浏览器必须做一些所谓的范围查找，它的昂贵程度取决于它要查找多少范围。尽量不要依辣全局/高范围的变量，尽量使局部范围变量，并将它们传递给函数。更少的范围查找，更少的牺牲速度。

这个测试告诉我们，从局部范围内传递和使用变量比从更高的声明范围查找变量快，无论是Chrome和Firefox。

[内部范围VS高范围VS全局](http://jsperf.com/closures-js)

＃6你不需要所有的东西都用jQuery

大多数开发者使用jQuery做一些简单的任务，我的意思在一些场合你没有必要使用jQuery，你觉得用$.val()始终是必要的吗？就拿这个例子：

1. $('input').keyup(function() {
3. if($(this).val() === 'blah') { ... }
5. });

这是学习如何使用JavaScript修改DOM的最重要原因之一，这样你可以编写更高效的代码。

用纯JavaScript100％完成同样的功能100%的速度更快，这是JSPerf基准 [测试](http://jsperf.com/jquery-vanilla-val)

1. $('input').keyup(function() {
3. if(this.value === 'blah') { ... }
5. });

0