# 前端不为人知的一面–前端冷知识集锦

链接：<http://web.jobbole.com/83473/>

**console.log("%c需要输出的信息 ", "css 代码");**

前端已经被玩儿坏了！像[console.log()可以向控制台输出图片](http://www.cnblogs.com/Wayou/p/chrome_dev_tool_style_console.html" \t "_blank)等炫酷的玩意已经不是什么新闻了，像用||操作符给变量赋默认值也是人尽皆知的旧闻了，今天看到Quora上一个[帖子](http://www.quora.com/Web-Development/What-are-the-most-interesting-HTML-JS-DOM-CSS-hacks-that-most-web-developers-dont-know-about?utm_source=html5weekly&utm_medium=email" \t "_blank)，瞬间又GET了好多前端技能，一些属于技巧，一些则是闻所未闻的冷知识，一时间还消化不过来。现分类整理出来分享给大家，也补充了一些平时的积累和扩展了一些内容。

Demo源码链接：**http://www.cnblogs.com/Wayou/p/chrome\_dev\_tool\_style\_console.html#a**

# HTML篇

## 浏览器地址栏运行JavaScript代码

这个很多人应该还是知道的，在浏览器地址栏可以直接运行JavaScript代码，做法是以 javascript: 开头后跟要执行的语句。比如：

JavaScript



|  |  |
| --- | --- |
| 1 | javascript:alert('hello from address bar :)'); |

将以上代码贴到浏览器地址栏回车后alert正常执行，一个弹窗神现。

需要注意的是如果是通过copy paste代码到浏览器地址栏的话，IE及Chrome会自动去掉代码开头的javascript:，所以需要手动添加起来才能正确执行，而Firefox中虽然不会自动去掉，但它根本就不支持在地址栏运行JS代码，sigh~

这一技术在我的另一篇博文《[让Chrome 接管邮件连接，收发邮件更方便了](http://www.cnblogs.com/Wayou/p/set_chrome_your_default_email_handler.html" \t "_blank)》中有使用到，利用在浏览器地址栏中执行JavaScript代码将Gmail设置为系统的邮件接管程序。

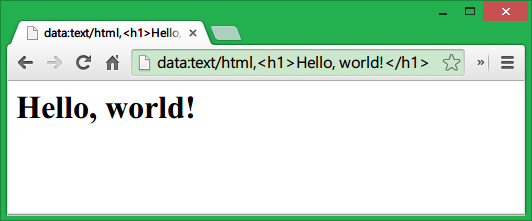
## 浏览器地址栏运行HTML代码

如果说上面那条小秘密知道的人还算多的话，这条秘笈知道的人就要少一些了，在非IE内核的浏览器地址栏可以直接运行HTML代码！

比如在地址栏输入以下代码然后回车运行，会出现指定的页面内容。



|  |  |
| --- | --- |
| 1 | data:text/html,<h1>Hello, world!</h1> |

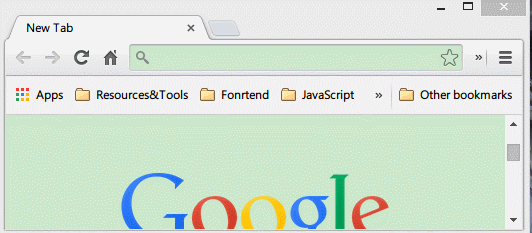


## 你造么，可以把浏览器当编辑器

还是浏览器地址栏上做文章，将以下代码贴到地址栏运行后浏览器变成了一个原始而简单的编辑器，与Windows自带的notepad一样，吼吼。



|  |  |
| --- | --- |
| 1 | data:text/html, <html contenteditable> |



归根结底多亏了HTML5中新加的 contenteditable 属性，当元素指定了该属性后，元素的内容成为可编辑状态。

推而广之，将以下代码放到console执行后，整个页面将变得可编辑，随意践踏吧~

JavaScript



|  |  |
| --- | --- |
| 1 | document.body.contentEditable='true'; |



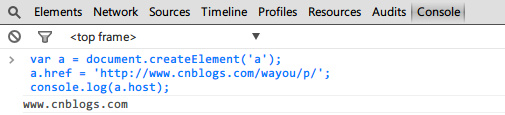
## 利用a标签自动解析URL

很多时候我们有从一个URL中提取域名，查询关键字，变量参数值等的需要，而万万没想到可以让浏览器方便地帮我们完成这一任务而不用我们写正则去抓取。方法就在JS代码里先创建一个 a 标签然后将需要解析的URL赋值给 a 的 href 属性，然后就得到了一切我们想要的了。

JavaScript



|  |  |
| --- | --- |
| 1  2  3 | var a = document.createElement('a');  a.href = 'http://www.cnblogs.com/wayou/p/';  console.log(a.host); |



利用这一原理，稍微扩展一下，就得到了一个更加健壮的解析URL各部分的通用方法了。下面代码来自[James的博客](http://james.padolsey.com/javascript/parsing-urls-with-the-dom/" \t "_blank)。

JavaScript



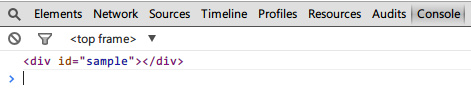
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | function parseURL(url) {      var a =  document.createElement('a');      a.href = url;      return {          source: url,          protocol: a.protocol.replace(':',''),          host: a.hostname,          port: a.port,          query: a.search,          params: (function(){              var ret = {},                  seg = a.search.replace(/^\?/,'').split('&'),                  len = seg.length, i = 0, s;              for (;i<len;i++) {                  if (!seg[i]) { continue; }                  s = seg[i].split('=');                  ret[s[0]] = s[1];              }              return ret;          })(),          file: (a.pathname.match(/\/([^\/?#]+)$/i) || [,''])[1],          hash: a.hash.replace('#',''),          path: a.pathname.replace(/^([^\/])/,'/$1'),          relative: (a.href.match(/tps?:\/\/[^\/]+(.+)/) || [,''])[1],          segments: a.pathname.replace(/^\//,'').split('/')      };  } |

## 页面拥有ID的元素会创建全局变量

在一张HTML页面中，所有设置了ID属性的元素会在JavaScript的执行环境中创建对应的全局变量，这意味着 document.getElementById 像人的阑尾一样显得多余了。但实际项目中最好老老实实该怎么写就怎么写，毕竟常规代码出乱子的机会要小得多。



|  |  |
| --- | --- |
| 1  2  3  4 | <div id="sample"></div>  <script type="text/javascript">          console.log(sample);  </script> |



## 加载CDN文件时，可以省掉HTTP标识

现在很流行的CDN即从专门的服务器加载一些通用的JS和CSS文件，出于安全考虑有的CDN服务器使用HTTPS方式连接，而有的是传统的HTTP，其实我们在使用时可以忽略掉这个，将它从URL中省去。



|  |  |
| --- | --- |
| 1 | <script src="//domain.com/path/to/script.js"></script> |

这一点在之前一篇译文博客《[jQuery编程最佳实践](http://www.cnblogs.com/Wayou/p/jquery_best_practise.html" \t "_blank)》中也有提到。

## 利用script标签保存任意信息

将script标签设置为 type='text' 然后可以在里面保存任意信息，之后可以在JavaScript代码中很方便地获取。



|  |  |
| --- | --- |
| 1  2  3 | <script type="text" id="template">  <h1>This won't display</h1>  </script> |

JavaScript



|  |  |
| --- | --- |
| 1 | var text = document.getElementById('template').innerHTML |

# CSS篇

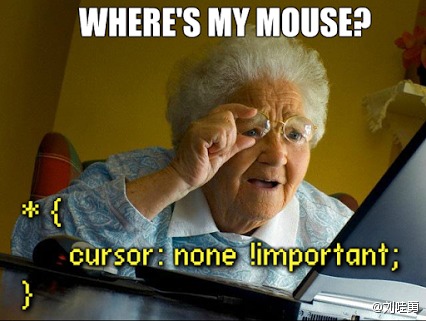
## 关于CSS的恶作剧

相信你看完以下代码后能够预料到会出现什么效果。

CSS



|  |  |
| --- | --- |
| 1  2  3 | \*{      cursor: none!important;  } |



## 简单的文字模糊效果

以下两行简单的CSS3代码可达到将文字模糊化处理的目的，出来的效果有点像使用PS的滤镜，so cool!

CSS



|  |  |
| --- | --- |
| 1  2  3  4 | p {      color: transparent;      text-shadow: #111 0 0 5px;  } |



## 垂直居中

有好多次博主都有这样的需求，垂直居中显示某个DIV，我们知道CSS中天然有水平居中的样式 text-align:center 。唯独这个垂直居中无解。

当然你可以将容器设置为 display:table ，然后将子元素也就是要垂直居中显示的元素设置为 display:table-cell ，然后加上来实现，但此种实现往往会因为 display：table 而破坏整体布局，那还不如直接用table标签了呢。

下面这个样式利用了translate来巧妙实现了垂直居中样式，需IE9+。

CSS



|  |  |
| --- | --- |
| 1  2  3  4  5 | .center-vertical {      position: relative;      top: 50%;      transform: translateY(-50%);  } |

相比而言，水平居中要简单得多，像上面提到的text-align:center，经常用到的技巧还有 margin:0 auto 。但对于margin大法也只在子元素宽度小于容器宽度时管用，当子元素宽度大于容器宽度时此法失效。

如法炮制，利用left和transform同样可实现水平居中，不过意义不大，毕竟text-align和margin差不多满足需求了。

CSS



|  |  |
| --- | --- |
| 1  2  3  4  5 | .center-horizontal {      position: relative;      left: 50%;      transform: translateX(-50%);  } |

## 多重边框

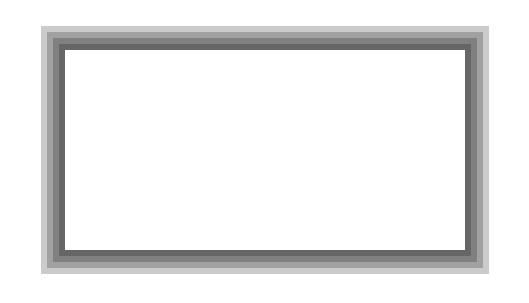
利用重复指定box-shadow来达到多个边框的效果

[在线演示](http://dabblet.com/gist/4034534)

CSS



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | /\*CSS Border with Box-Shadow Example\*/  div {      box-shadow: 0 0 0 6px rgba(0, 0, 0, 0.2), 0 0 0 12px rgba(0, 0, 0, 0.2), 0 0 0 18px rgba(0, 0, 0, 0.2), 0 0 0 24px rgba(0, 0, 0, 0.2);      height: 200px;      margin: 50px auto;      width: 400px  } |

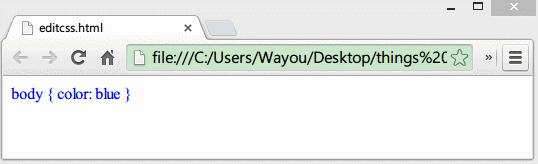


## 实时编辑CSS

通过设置style标签的 display:block 样式可以让页面的style标签显示出来，并且加上 contentEditable 属性后可以让样式成为可编辑状态，更改后的样式效果也是实时更新呈现的。此技巧在IE下无效。拥有此技能者，逆天也！



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <!DOCTYPE html>  <html>      <body>          <style style="display:block" contentEditable>           body { color: blue }          </style>      </body>  </html> |

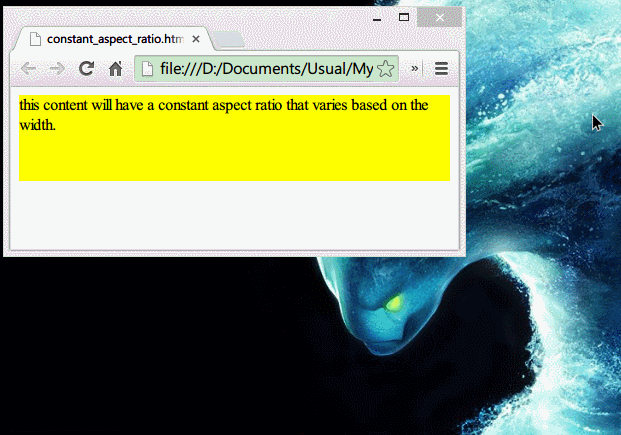


## 创建长宽比固定的元素

通过设置父级窗口的 padding-bottom 可以达到让容器保持一定的长度比的目的，这在响应式页面设计中比较有用，能够保持元素不变形。



|  |  |
| --- | --- |
| 1  2  3  4  5 | <div style="width: 100%; position: relative; padding-bottom: 20%;">      <div style="position: absolute; left: 0; top: 0; right: 0; bottom: 0;background-color:yellow;">          this content will have a constant aspect ratio that varies based on the width.      </div>  </div> |



## CSS中也可以做简单运算

通过CSS中的calc方法可以进行一些简单的运算，从而达到动态指定元素样式的目的。

CSS



|  |  |
| --- | --- |
| 1  2  3 | .container{  background-position: calc(100% - 50px) calc(100% - 20px);  } |

# JavaScript篇

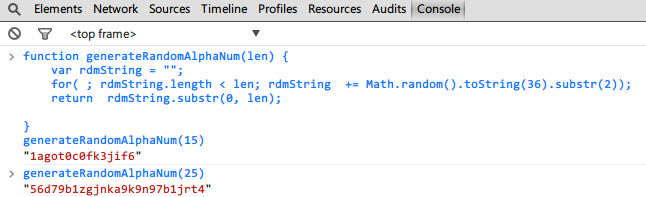
## 生成随机字符串

利用 Math.random 和 toString 生成随机字符串，来自前一阵子看到的一篇[博文](http://flippinawesome.org/2013/12/23/45-useful-javascript-tips-tricks-and-best-practices/" \t "_blank)。这里的技巧是利用了 toString 方法可以接收一个基数作为参数的原理，这个基数从2到36封顶。如果不指定，默认基数是10进制。略屌！

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5 | function generateRandomAlphaNum(len) {      var rdmString = "";      for (; rdmString.length < len; rdmString += Math.random().toString(36).substr(2));      return rdmString.substr(0, len);  } |



## 整数的操作

JavaScript中是没有整型概念的，但利用好位操作符可以轻松处理，同时获得效率上的提升。

|0 和 ~~ 是很好的一个例子，使用这两者可以将浮点转成整型且效率方面要比同类的 parseInt , Math.round  要快。在处理像素及动画位移等效果的时候会很有用。性能比较[见此](http://jsperf.com/math-floor-vs-math-round-vs-parseint/42" \t "_blank)。

JavaScript



|  |  |
| --- | --- |
| 1  2 | var foo = (12.4 / 4.13) | 0;//结果为3  var bar = ~~(12.4 / 4.13);//结果为3 |

顺便说句， !! 将一个值方便快速转化为布尔值 !!window===true  。

## 重写原生浏览器方法以实现新功能

下载的代码通过重写浏览器的 alert 让它可以记录弹窗的次数。

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | (function() {      var oldAlert = window.alert,          count = 0;      window.alert = function(a) {          count++;          oldAlert(a + "\n You've called alert " + count + " times now. Stop, it's evil!");      };  })();  alert("Hello World"); |

## 关于console的恶作剧

关于重写原生方法，这里有个恶作剧大家可以拿去寻开心。Chrome的 console.log 是支持对文字添加样式的，甚至log图片都可以。于是可以重写掉默认的log方法，把将要log的文字应用到CSS的模糊效果，这样当有人试图调用console.log()的时候，出来的是模糊不清的文字。好冷，我表示没有笑。

是从[这篇](https://plus.google.com/+AddyOsmani/posts/TanDFKEN9Kn)G+帖子的评论里看到的。使用之后的效果是再次调用log会输出字迹模糊不清的文字。

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4 | var \_log = console.log;  console.log = function() {    \_log.call(console, '%c' + [].slice.call(arguments).join(' '), 'color:transparent;text-shadow:0 0 2px rgba(0,0,0,.5);');  }; |



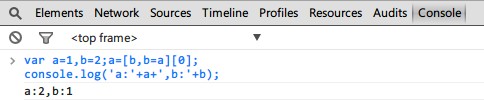
## 不声明第三个变量的值交换

我们都知道交换两个变量值的常规做法，那就是声明一个中间变量来暂存。但鲜有人去挑战不声明中间变量的情况，下面的代码给出了这种实现。蛮有创意 的。

JavaScript



|  |  |
| --- | --- |
| 1 | var a=1,b=2;a=[b,b=a][0]; |



## 万物皆对象

在JavaScript的世界，万物皆对象。除了 null 和 undefined ，其他基本类型数字，字符串和布尔值都有对应有包装对象。对象的一个特征是你可以在它身上直接调用方法。对于数字基本类型，当试图在其身上调用 toString 方法会失败，但用括号括起来后再调用就不会失败了，内部实现是用相应的包装对象将基本类型转为对象。所以 (1).toString() 相当于new Number(1).toString() 。因此，你的确可以把基本类型数字，字符串，布尔等当对象使用的，只是注意语法要得体。

同时我们注意到，JavaScript中数字是不分浮点和整形的，所有数字其实均是浮点类型，只是把小数点省略了而以，比如你看到的1可以写成1.，这也就是为什么当你试图1.toString()时会报错，所以正确的写法应该是这样：1..toString()，或者如上面所述加上括号，这里括号的作用是纠正JS解析器，不要把1后面的点当成小数点。内部实现如上面所述，是将1.用包装对象转成对象再调用方法。

## If语句的变形

当你需要写一个if语句的时候，不妨尝试另一种更简便的方法，用JavaScript中的逻辑操作符来代替。

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var day=(new Date).getDay()===0;  //传统if语句  if (day) {  alert('Today is Sunday!');  };  //运用逻辑与代替if  day&&alert('Today is Sunday!'); |

比如上面的代码，首先得到今天的日期，如果是星期天，则弹窗，否则什么也不做。我们知道逻辑操作存在短路的情况，对于逻辑与表达式，只有两者都真才结果才为真，如果前面的day变量被判断为假了，那么对于整个与表达式来说结果就是假，所以就不会继续去执行后面的 alert 了，如果前面day为真，则还要继续执行后面的代码来确定整个表达式的真假。利用这点达到了if的效果。

对于传统的if语句，如果执行体代码超过了1 条语句，则需要加花括号，而利用逗号表达式，可以执行任意条代码而不用加花括号。

JavaScript



|  |  |
| --- | --- |
| 1 | if(conditoin) alert(1),alert(2),console.log(3); |

上面if语句中，如果条件成立则执行三个操作，但我们不需要用花括号将这三句代码括起来。当然，这是不推荐的，这里是冷知识课堂:)

## 禁止别人以iframe加载你的页面

下面的代码已经不言自明了，没什么好多说的。

JavaScript



|  |  |
| --- | --- |
| 1 | if (window.location != window.parent.location) window.parent.location = window.location; |

## console.table

Chrome专属，IE绕道的 console 方法。可以将JavaScript关联数组以表格形式输出到浏览器 console ，效果很惊赞，界面很美观。

JavaScript



|  |  |
| --- | --- |
| 1  2  3 | //采购情况  var data = [{'品名': '杜雷斯', '数量': 4}, {'品名': '冈本', '数量': 3}];  console.table(data); |

