**显示信息的命令**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <title>常用console命令</title>      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  </head>  <body>      <script type="text/javascript">          console.log('hello');          console.info('信息');          console.error('错误');          console.warn('警告');      </script>  </body>  </html> |

最常用的就是console.log了。

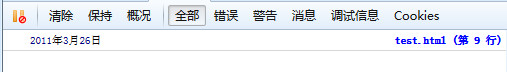
**占位符**

console上述的集中度支持printf的占位符格式，支持的占位符有：字符（%s）、整数（%d或%i）、浮点数（%f）和对象（%o）:

| **占位符** | **作用** |
| --- | --- |
| %s | 字符串 |
| %d or %i | 整数 |
| %f | 浮点数 |
| %o | 可展开的DOM |
| %O | 列出DOM的属性 |
| %c | 根据提供的css样式格式化字符串 |

|  |
| --- |
| <script type="text/javascript">       console.log("%d年%d月%d日",2011,3,26);  </script> |

效果：



%o、%O都是用来输出Object对象的，对普通的Object对象，两者没区别，但是打印dom节点时就不一样了：

|  |
| --- |
| // 格式成可展开的的DOM，像在开发者工具Element面板那样可展开  console.log('%o',document.body.firstElementChild);  // 像JS对象那样访问DOM元素，可查看DOM元素的属性  // 等同于console.dir(document.body.firstElementChild)  console.log('%O',document.body.firstElementChild); |

%c占位符是最常用的。使用%c占位符时，对应的后面的参数必须是CSS语句，用来对输出内容进行CSS渲染。常见的输出方式有两种：文字样式、图片输出。

**文字输出**

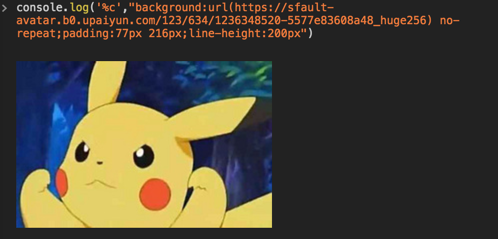
|  |
| --- |
| console.log("%cHello world,欢迎您！","color: red; font-size: 20px");  //输出红色的、20px大小的字符串：Hello world,欢迎您！ |

除了普通文本，还能输出如[知乎](https://www.zhihu.com/)的console面板一样的字符画。这些字符画是可以在线生成的：

* [picascii](http://picascii.com/)
* [mg2txt](http://www.degraeve.com/img2txt.php)
* [Ascii generator](http://www.network-science.de/ascii/)

大概方法：使用在线工具生成字符画，然后复制到sublime中，将每行开头的换行删除，且替换成\n。最后只有一行代码，即保证没有换行，最后再丢到console.log(“”)代码中即可，当然，也可以添加结合%c做出更酷炫的效果(console输出默认是不换行的)。

**图片输出**

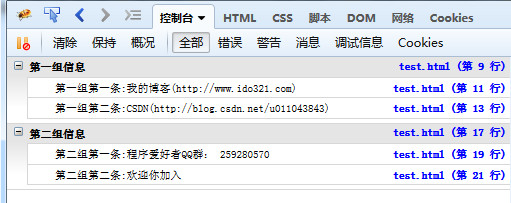


由于 console不能定义img，因此用背景图片代替。此外，console不支持width和height，利用空格和font-size代替；还可以使用padding和line-height代替宽高。

不想这么麻烦，可以试试[console-image](https://github.com/dunxrion/console.image)这个插件。

**信息分组**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>       <title>常用console命令</title>       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />   </head>   <body>       <script type="text/javascript">           console.group("第一组信息");         　　　　console.log("第一组第一条:我的博客([http://www.ido321.com](http://www.ido321.com/))");         　　　　console.log("第一组第二条:CSDN(<http://blog.csdn.net/u011043843>)");         　　console.groupEnd();     　　    console.group("第二组信息");         　　　　console.log("第二组第一条:程序爱好者QQ群： 259280570");         　　　　console.log("第二组第二条:欢迎你加入");     　　    console.groupEnd();       </script>   </body>  </html> |

效果：  


**查看对象的信息**

console.dir()可以显示一个对象所有的属性和方法。

|  |
| --- |
| <script type="text/javascript">          var info = {              blog:"[http://www.ido321.com](http://www.ido321.com/)",              QQGroup:259280570,              message:"程序爱好者欢迎你的加入"          };          console.dir(info);  </script> |

效果：  


**显示某个节点的内容**

console.dirxml()用来显示网页的某个节点（node）所包含的html/xml代码。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <title>常用console命令</title>      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  </head>  <body>      <div id="info">          <h3>我的博客：www.ido321.com</h3>          <p>程序爱好者:259280570,欢迎你的加入</p>      </div>      <script type="text/javascript">          var info = document.getElementById('info');          console.dirxml(info);      </script>  </body>  </html> |

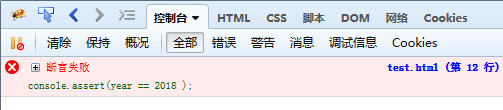
效果：  


**判断变量是否是真**

console.assert()用来判断一个表达式或变量是否为真。如果结果为否，则在控制台输出一条相应信息，并且抛出一个异常。

|  |
| --- |
| <script type="text/javascript">      　　var result = 1;      　　console.assert( result );      　　var year = 2014;      　　console.assert(year == 2018 );  </script> |

1是非0值，是真；而第二个判断是假，在控制台显示错误信息

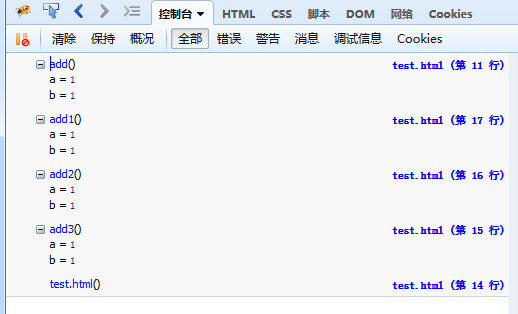


**追踪函数的调用轨迹**

console.trace()用来追踪函数的调用轨迹。

|  |
| --- |
| <script type="text/javascript">  /\*函数是如何被调用的，在其中加入console.trace()方法就可以了\*/  　　function add(a,b){          console.trace();  　　　　return a+b;  　　}  　　var x = add3(1,1);  　　function add3(a,b){return add2(a,b);}  　　function add2(a,b){return add1(a,b);}  　　function add1(a,b){return add(a,b);}  </script> |

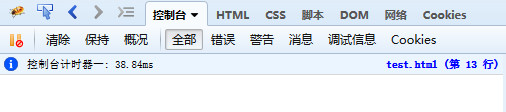
控制台输出信息：



**计时功能**

console.time()和console.timeEnd()，用来显示代码的运行时间。

|  |
| --- |
| <script type="text/javascript">  　　console.time("控制台计时器一");  　　for(var i=0;i<1000;i++){  　　　　for(var j=0;j<1000;j++){}  　　}  　　console.timeEnd("控制台计时器一");  </script> |

运行时间是38.84ms  


**console.profile()的性能分析**

性能分析（Profiler）就是分析程序各个部分的运行时间，找出瓶颈所在，使用的方法是console.profile()。

|  |
| --- |
| <script type="text/javascript">       　　function All(){               alert(11);   　　　　     for(var i=0;i<10;i++){                   funcA(1000);                }   　　　　    funcB(10000);   　　    }         　　function funcA(count){       　　　　for(var i=0;i<count;i++){} 　　}="" 　　function="" funcb(count){="" 　　　　for(var="" i="0;i<count;i++){}" 　　console.profile('性能分析器');="" 　　all();="" 　　console.profileend();="" <="" script>="" <="" pre="">  <p>输出如图：<br>  <img src="<http://www.jqhtml.com/wp-content/uploads/2017/11/console1122-9.png>"></p>  <h2><a name="9"></a>其它</h2>  <p><a href="<http://www.alloyteam.com/2013/11/console-log/>" target="\_blank">从console.log说起</a></p>  <p class="source">原文链接：<a href="<https://github.com/dwqs/blog/issues/32>" target="\_blank"><https://github.com/dwqs/blog/issues/32></a></p>                      </count;i++){}> |