



分类：[IT技术](#)

上一篇：[IETF：互联网精神的典范](#)

下一篇：[《美丽新世界》读后感](#)

# Firebug控制台详解

作者：阮一峰

日期：2011年3月26日

[Firebug](#)是网页开发的利器，能够极大地提升工作效率。

但是，它不太容易上手。我曾经翻译过一篇[《Firebug入门指南》](#)，介绍了一些基本用法。今天，继续介绍它的高级用法。

=====

## Firebug控制台详解

作者：阮一峰



控制台（Console）是Firebug的第一个面板，也是最重要的面板，主要作用是显示网页加载过程中产生各类信息。

### 一、显示信息的命令

Firebug内置一个console对象，提供5种方法，用来显示信息。

最简单的方法是`console.log()`，可以用来取代`alert()`或`document.write()`。比如，在网页脚本中使用`console.log("Hello World")`，加载时控制台就会自动显示如下内容。

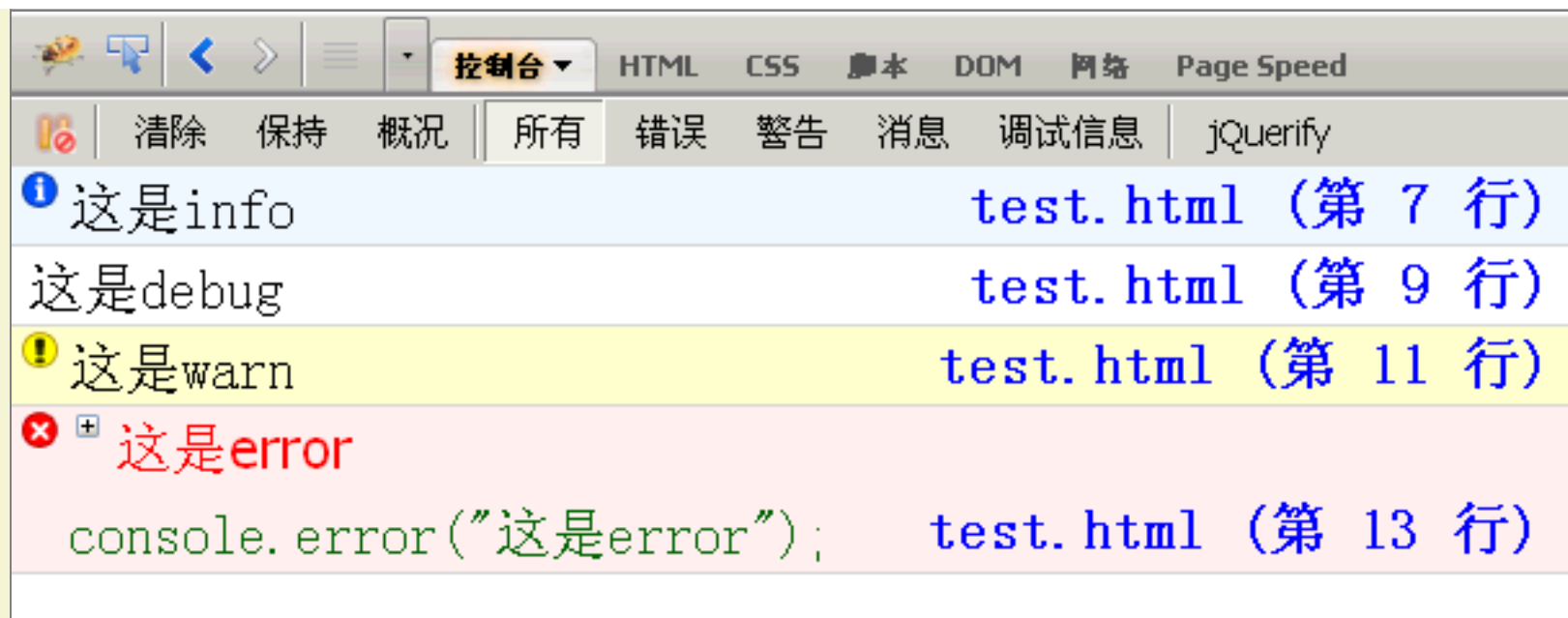


另外，根据信息的不同性质，console对象还有4种显示信息的方法，分别是一般信息console.info()、除错信息console.debug()、警告提示console.warn()、错误提示console.error()。

比如，在网页脚本中插入下面四行：

```
console.info("这是info");  
  
console.debug("这是debug");  
  
console.warn("这是warn");  
  
console.error("这是error");
```

加载时，控制台会显示如下内容。



可以看到，不同性质的信息前面有不同的图标，并且每条信息后面都有超级链接，点击后跳转到网页源码的相应行。

## 二、占位符

console对象的上面5种方法，都可以使用printf风格的占位符。不过，占位符的种类比较少，只支持字符（%s）、整数（%d或%i）、浮点数（%f）和对象（%o）四种。

比如，

```
console.log("%d年%d月%d日",2011,3,26);
```

```
console.log("圆周率是%f",3.1415926);
```

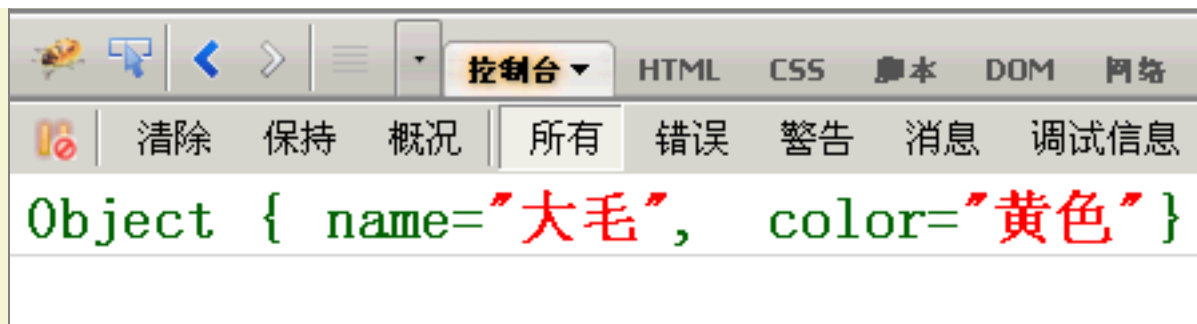


%o占位符，可以用来查看一个对象内部情况。比如，有这样一个对象：

```
var dog = {} ;  
  
dog.name = "大毛" ;  
  
dog.color = "黄色";
```

然后，对它使用o%占位符。

```
console.log("%o",dog);
```



### 三、分组显示

如果信息太多，可以分组显示，用到的方法是`console.group()`和`console.groupEnd()`。

```
console.group("第一组信息");
```

```
    console.log("第一组第一条");
```

```
    console.log("第一组第二条");
```

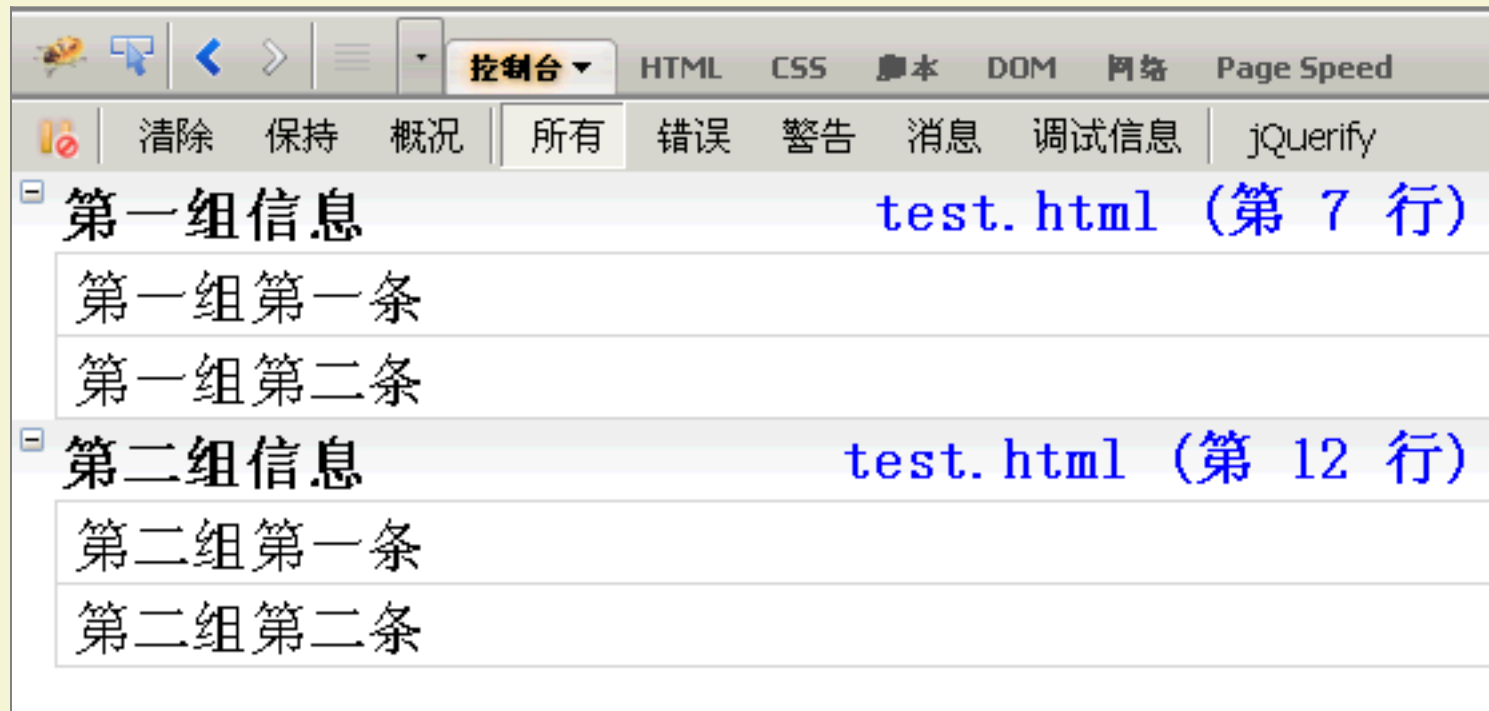
```
console.groupEnd();
```

```
console.group("第二组信息");
```

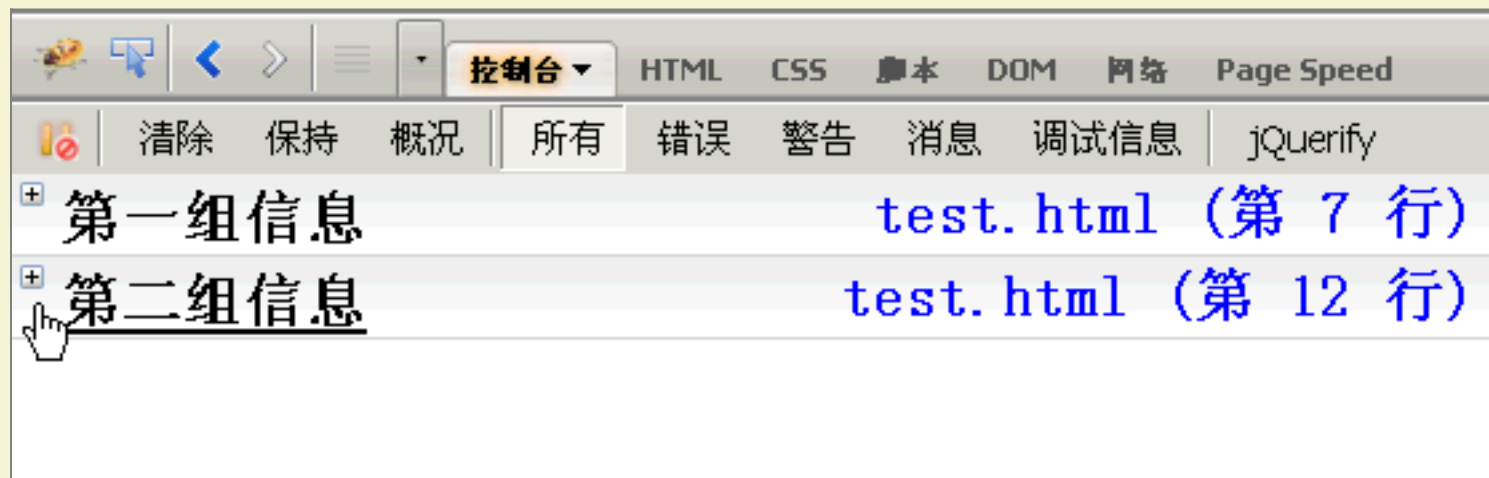
```
    console.log("第二组第一条");
```

```
    console.log("第二组第二条");
```

```
console.groupEnd();
```



点击组标题，该组信息会折叠或展开。



#### 四、`console.dir()`

`console.dir()`可以显示一个对象所有的属性和方法。

比如，现在为第二节的`dog`对象，添加一个`bark()`方法。

```
dog.bark = function(){alert("汪汪汪");};
```

然后，显示该对象的内容，

```
console.dir(dog);
```





## 五、**console.dirxml()**

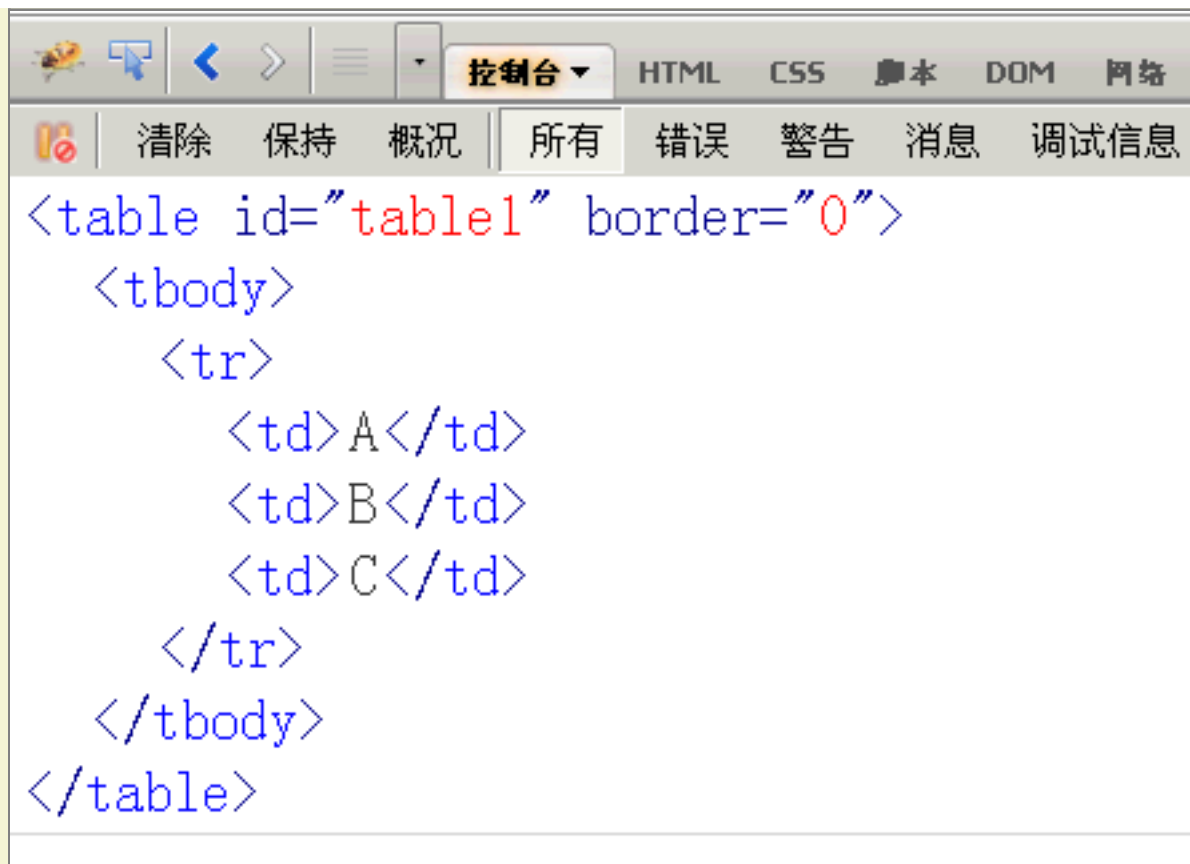
`console.dirxml()`用来显示网页的某个节点（node）所包含的html/xml代码。

比如，先获取一个表格节点，

```
var table = document.getElementById("table1");
```

然后，显示该节点包含的代码。

```
console.dirxml(table);
```



The screenshot shows a web browser's developer console. The top toolbar includes icons for a bug, a cursor, and navigation arrows, along with tabs for '控制台' (Console), 'HTML', 'CSS', '脚本' (Scripts), 'DOM', and '网络' (Network). The '控制台' tab is active, displaying a list of messages: '清除' (Clear), '保持' (Keep), '概况' (Summary), '所有' (All), '错误' (Errors), '警告' (Warnings), '消息' (Messages), and '调试信息' (Debugging Information). The main area of the console shows the following HTML code:

```
<table id="table1" border="0">
  <tbody>
    <tr>
      <td>A</td>
      <td>B</td>
      <td>C</td>
    </tr>
  </tbody>
</table>
```

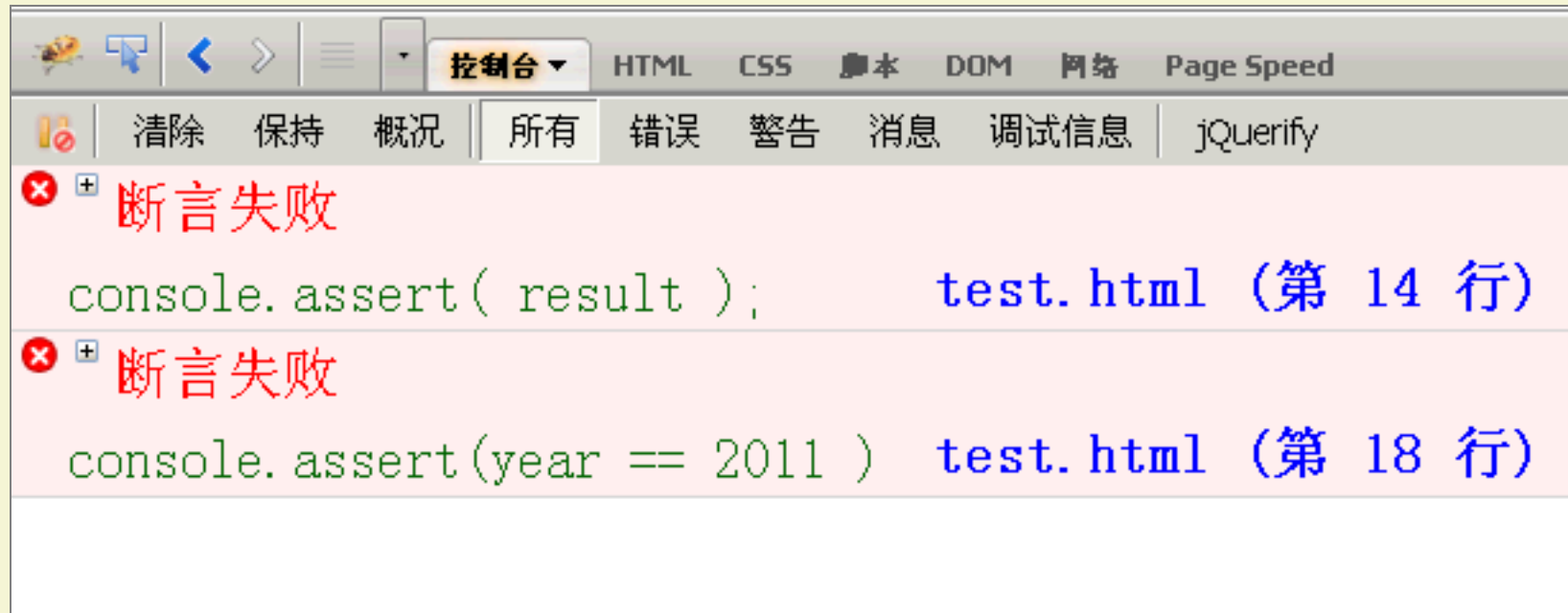
## 六、**console.assert()**

`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。

比如，下面两个判断的结果都为否。

```
var result = 0;
```

```
console.assert( result );  
  
var year = 2000;  
  
console.assert(year == 2011 );
```



## 七、**console.trace()**

**console.trace()**用来追踪函数的调用轨迹。

比如，有一个加法器函数。

```
function add(a,b){  
  
    return a+b;  
  
}
```

我想知道这个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;  
  
}
```

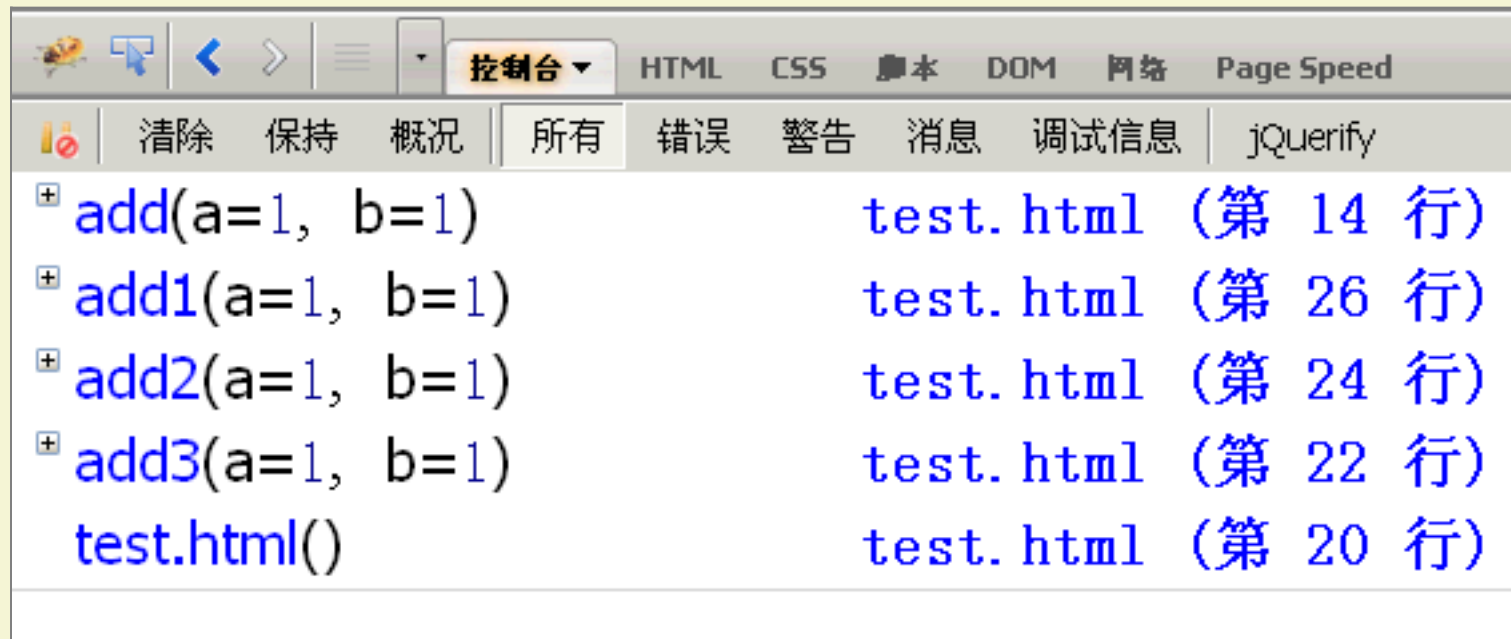
假定这个函数的调用代码如下：

```
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);}
```

运行后，会显示add()的调用轨迹，从上到下依次为add()、add1()、add2()、add3()。

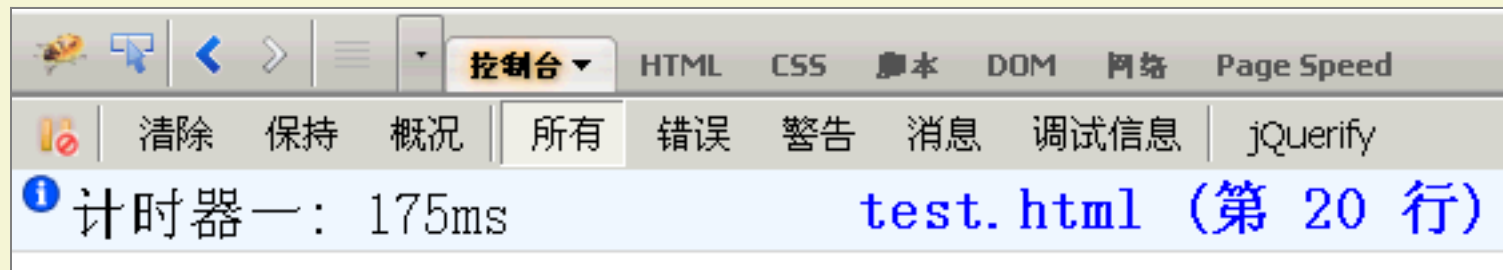


## 八、计时功能

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

```
console.time("计时器一");
```

```
for(var i=0;i<1000;i++){  
  
    for(var j=0;j<1000;j++){  
  
    }  
  
    console.timeEnd("计时器一");
```



## 九、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()`。

假定有一个函数`Foo()`，里面调用了另外两个函数`funcA()`和`funcB()`，其中`funcA()`调用10次，`funcB()`调用1次。

```
function Foo(){
```

```
    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++){

    }

}
```


然后，就可以分析Foo()的运行性能了。

```
console.profile('性能分析器一');

Foo();
```

```
console.profileEnd();
```

控制台会显示一张性能分析表，如下图。



The screenshot shows a browser's developer console with the 'Performance' tab selected. The title bar indicates '性能分析器一 (2.656 毫秒, 12 次调用)'. Below the title bar is a table with 8 columns: 函数 (Function), 调用 (Calls), 百分比 (Percentage), 占用时间 (Time), 时间 (Time), 平均时间 (Average Time), 最小时间 (Minimum Time), and 最大时间 (Maximum Time). The table lists three functions: funcA, funcB, and Foo. funcA has 10 calls, taking 52.37% of the time. funcB has 1 call, taking 46.27% of the time. Foo has 1 call, taking 1.36% of the time.

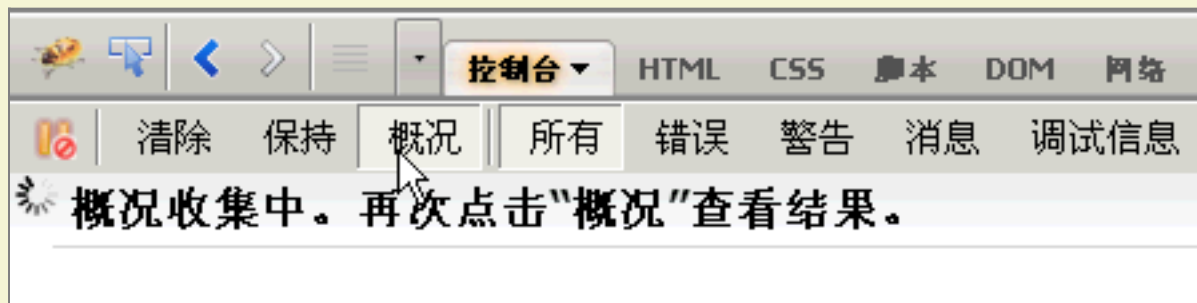
函数	调用	百分比	占用时间	时间	平均时间	最小时间	最大时间
funcA	10	52.37%	1.391ms	1.391ms	0.139ms	0.123ms	0.284ms
funcB	1	46.27%	1.229ms	1.229ms	1.229ms	1.229ms	1.229ms
Foo	1	1.36%	0.036ms	2.656ms	2.656ms	2.656ms	2.656ms

标题栏提示，一共运行了12个函数，共耗时2.656毫秒。其中funcA()运行10次，耗时1.391毫秒，最短运行时间0.123毫秒，最长0.284毫秒，平均0.139毫秒；funcB()运行1



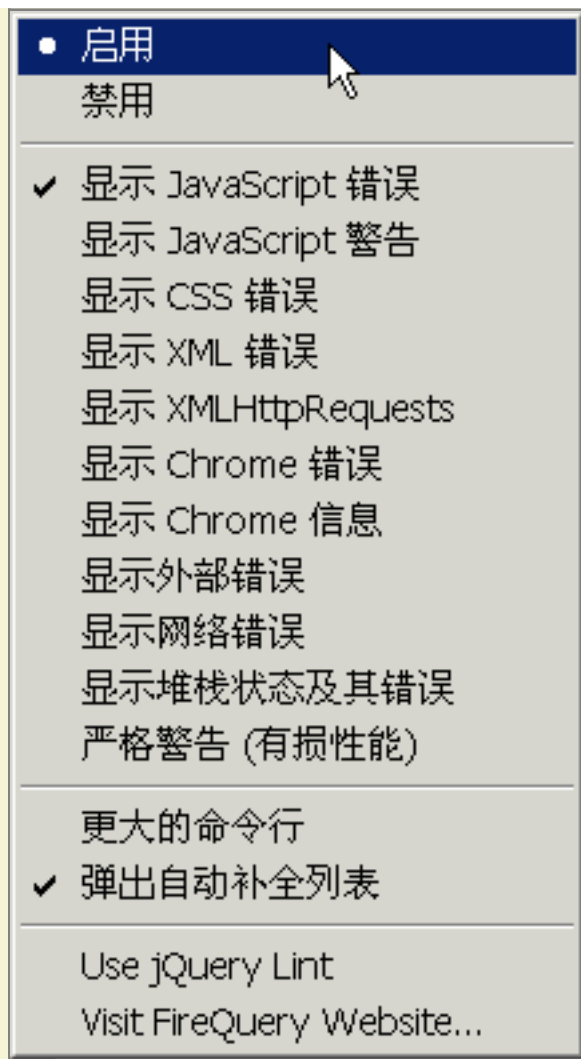
次，耗时1.229ms毫秒。

除了使用`console.profile()`方法，firebug还提供了一个"概况"（Profiler）按钮。第一次点击该按钮，"性能分析"开始，你可以对网页进行某种操作（比如ajax操作），然后第二次点击该按钮，"性能分析"结束，该操作引发的所有运算就会进行性能分析。



## 十、属性菜单

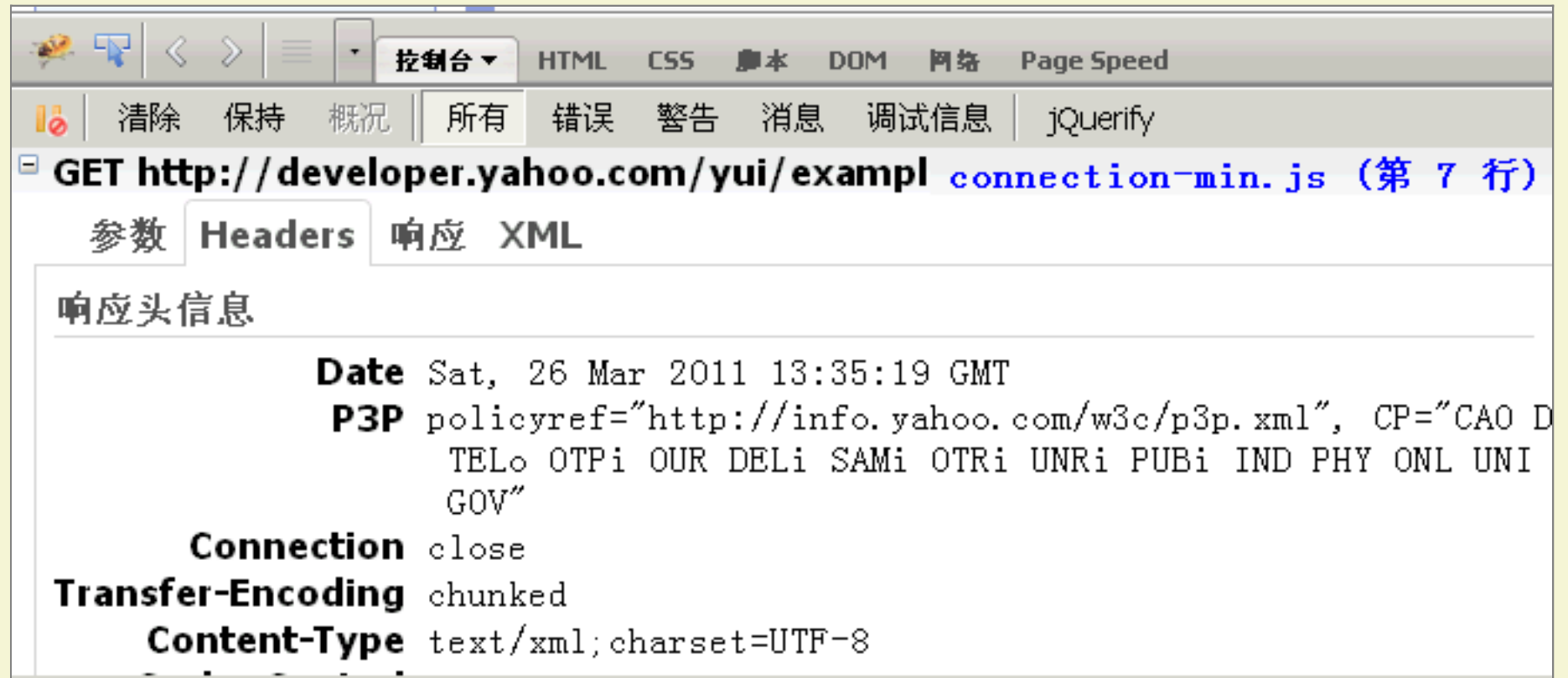
控制台面板的名称后面，有一个倒三角，点击后会显示属性菜单。



默认情况下，控制台只显示Javascript错误。如果选中Javascript警告、CSS错误、XML错误都送上，则相关的提示信息都会显示。

这里比较有用的是"显示XMLHttpRequests"，也就是显示ajax请求。选中以后，网页的所有ajax请求，都会在控制台面板显示出来。

比如，点击一个[YUI示例](#)，控制台就会告诉我们，它用ajax方式发出了一个GET请求，http请求和响应的头信息和内容主体，也都可以看到。



[参考文献]

- \* [Firebug Tutorial - Logging, Profiling and CommandLine \(Part I\)](#)
- \* [Firebug Tutorial - Logging, Profiling and CommandLine \(Part II\)](#)

(完)

## 文档信息

---

- 版权声明：自由转载-非商用-非衍生-保持署名 | Creative Commons BY-NC-ND 3.0

- 原文网

址：[http://www.ruanyifeng.com/blog/2011/03/firebug\\_console\\_tutorial.html](http://www.ruanyifeng.com/blog/2011/03/firebug_console_tutorial.html)

- 最后修改时间：2011年3月26日 22:14



## 相关文章

---

- **2011.03.12:** [四位计算机的原理及其实现](#)

你是否想过，计算机为什么会加减乘除？或者更直接一点，计算机的原理到底是


什么？

- **2011.03.09:** [URL的井号](#)

去年9月，twitter改版。

## 功能链接

---

- 前一篇：[IETF：互联网精神的典范](#)
- 后一篇：[《美丽新世界》读后感](#)
- 更多内容请访问：[首页](#) » [档案](#) » [IT技术](#)
- 站内搜索：
- **Feed**订阅：

GO !

## 广告

---

## 留言（ 11条 ）

---

幸然 说：

这篇文章太好了，一直在用firebug,却不知道这些代码功能。

2011年3月27日 00:44 | [档案](#) | [引用](#)

---

南 靖男 说：

装了FireQuery，但没装FireCookie和FirePHP。

2011年3月27日 14:56 | [档案](#) | [引用](#)

---

and1coder 说：

确实有很多特性没有用过而且又非常的实用:D

2011年3月27日 16:13 | [档案](#) | [引用](#)

---

梧桐 说：

非常有用的文章

2011年3月27日 18:34 | [档案](#) | [引用](#)

---

**freetao** 说：

不错不错，mark一下。

2011年3月27日 18:38 | [档案](#) | [引用](#)

---

**lazycai** 说：

刚刚升级Firefox 4，看到默认有审查元素功能，就把firebug删了。

看来firebug还是很强大的

2011年3月28日 10:53 | [档案](#) | [引用](#)

---

城主 说：

还是仅限于前端开发吧~不过的确很强大

2011年3月28日 12:51 | [档案](#) | [引用](#)

---

**Controlsea** 说：

更详细的了解到了firebug强大的一面～赞！！

2011年3月28日 15:22 | 档案 | 引用

---

**w** 说：

[http://getfirebug.com/wiki/index.php/Console\\_API](http://getfirebug.com/wiki/index.php/Console_API)

2011年3月29日 11:07 | 档案 | 引用

---

**wxianfeng** 说：

```
var dog = {} ;  
    dog.name = "大毛" ;  
    dog.color = "黄色";  
console.log("%o",dog)  
console.dir(dog)
```

学了一招 console 下直接查看 对象方法 ， 不需要再 添加 watch 了 ...

2011年3月29日 19:44 | 档案 | 引用

---

**John.Yu** 说：

```
console.info("这是info");
```



```
console.debug("这是debug");
```

```
console.warn("这是warn");
```

```
console.error("这是error");
```

是否可以象Log4J可以控制输出的等级呢？

2011年3月30日 18:57 | 档案 | 引用

---

## 我要发表看法

您的留言（HTML标签部分可用）

您的大名：

«-必填

电子邮件：

«-必填，不公开

个人网址：

«-我信任你，不会填写广告链接

记住个人信息？☐

发表

«- 点击按钮

---

联系方式 | [ruanyifeng.com](http://ruanyifeng.com) 2003 - 2011 