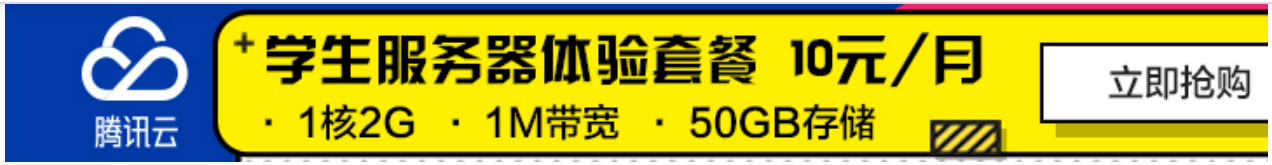






classicemi RP 1.2k 发布于 Orange的前端笔记 关注专栏
2014-04-24 发布

# 你真的了解 console 吗

javascript 前端 27.4k 次阅读 · 读完需要 17 分钟

对于前端开发者来说，在开发过程中需要监控某些表达式或变量的值的时候，用 debugger 会显得过于笨重，取而代之则是会将值输出到控制台上方便调试。最常用的语句就是 `console.log(expression)` 了。

然而对于作为一个全局对象的 `console` 对象来说，大多数人了解得还并不全面，当然我也是，经过我的一番学习，现在对于这个能玩转控制台的 JS 对象有了一定的认识，想与大家分享一下。

console 对象除了 `console.log()` 这一最常被开发者使用的方法之外，还有很多其他的方法。灵活运用这些方法，可以给开发过程增添许多便利。

## console 的方法

**console.assert(expression, object[, object...])**

接收至少两个参数，第一个参数的值或返回值为 `false` 的时候，将会在控制台上输出后续参数的值。例如：

```
console.assert(1 == 1, object); // 无输出，返回 undefined
console.assert(1 == 2, object); // 输出 object
```

### console.count([label])

输出执行到该行的次数，可选参数 label 可以输出在次数之前，例如：

```
(function() {
  for (var i = 0; i < 5; i++) {
    console.count('count');
  }
})();
// count: 1
// count: 2
// count: 3
// count: 4
// count: 5
```

### console.dir(object)

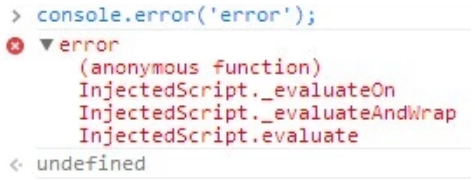
将传入对象的属性，包括子对象的属性以列表形式输出，例如：

```
var obj = {
  name: 'classicemi',
  college: 'HUST',
  major: 'ei'
};
console.dir(obj);
```

输出：


**console.error(object[, object...])**

用于输出错误信息，用法和常见的 `console.log` 一样，不同点在于输出内容会标记为错误的样式，便于分辨。输出结果：

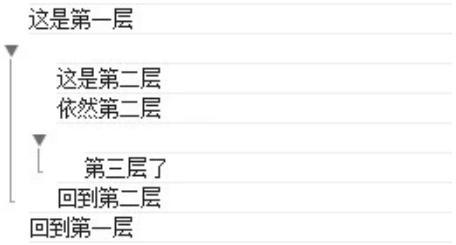



**console.group**

这是个有趣的方法，它能够让控制台输出的语句产生不同的层级嵌套关系，每一个 `console.group()` 会增加一层嵌套，相反要减少一层嵌套可以使用 `console.groupEnd()` 方法。语言表述比较无力，看代码：

```
console.log('这是第一层');
console.group();
console.log('这是第二层');
console.log('依然第二层');
console.group();
console.log('第三层了');
console.groupEnd();
console.log('回到第二层');
console.groupEnd();
console.log('回到第一层');
```

输出结果：




和 `console.group()` 相似的方法是 `console.groupCollapsed()` 作用相同，不同点是嵌套的输出内容是折叠状态，在有大段内容输出的时候使用这个方法可以使输出版面不至于太长。。。吧

**console.info(object[, object...])**

此方法与之前说到的 `console.error` 一样，用于输出信息，没有什么特别之处。

```
console.info('info'); // 输出 info
```

**console.table()**

可将传入的对象，或数组以表格形式输出，相比传统树形输出，这种输出方案更适合内部元素排列整齐的对象或数组，不然可能会出现很多的 undefined。

```
var obj = {
  foo: {
    name: 'foo',
    age: '33'
  },
  bar: {
    name: 'bar',
    age: '45'
  }
};

var arr = [
  ['foo', '33'],
```

```
  ['bar', '45']
];

console.table(obj);
console.table(arr);
```

| (index) | name | age |
|---|---|---|
| foo | "foo" | "33" |
| bar | "bar" | "45" |

VM261:18

| (index) | 0 | 1 |
|---|---|---|
| 0 | "foo" | "33" |
| 1 | "bar" | "45" |

VM261:19

也可以

**console.log(object[, object...])**

这个不用多说，这个应该是开发者最常用的吧，也不知道是谁规定的。。。

```
console.log('log'); // 输出 log
```

**console.profile([profileLabel])**

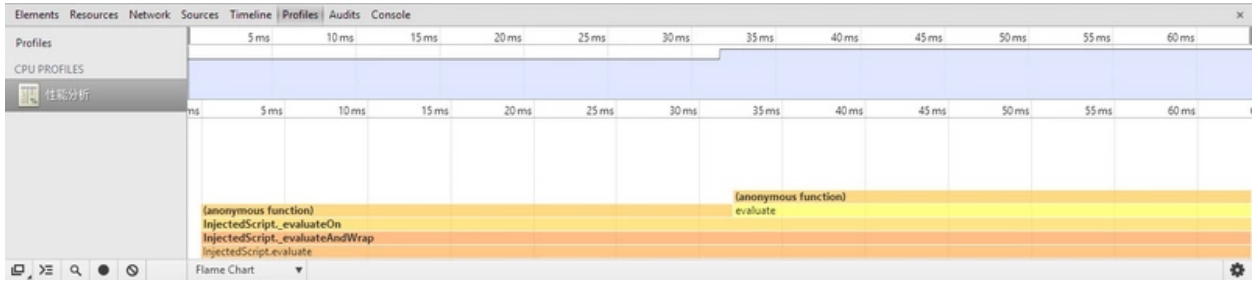
这是个挺高大上的东西，可用于性能分析。在 JS 开发中，我们常常要评估段代码或是某个函数的性能。在函数中手动打印时间固然可以，但显得不够灵活而且有误差。借助控制台以及 `console.profile()` 方法我们可以很方便地监控运行性能。

例如下面这段代码：

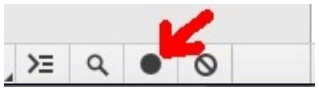
```
function parent() {
  for (var i = 0; i < 10000; i++) {
    childA()
  }
```

```
}

function childA(j) {
  for (var i = 0; i < j; i++) {}
}

console.profile('性能分析');
parent();
console.profileEnd();
```

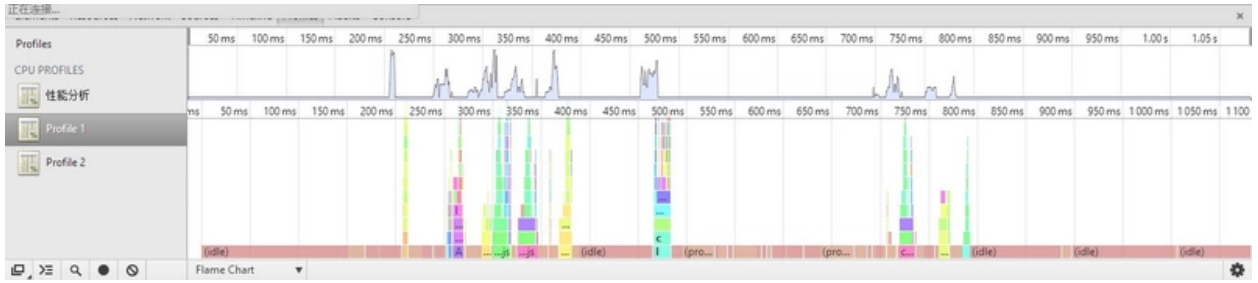
然后我们可以在 Profiles 面板下看到上述代码运行过程中的消耗时间。




页面加载过程的性能优化是前端开发的一个重要部分，使用控制台的 profiles 面板可以监控所有 JS 的运行情况使用方法就像录像机一样，控制台会把运行过程录制下来。如图，工具栏上有录制和停止按钮。



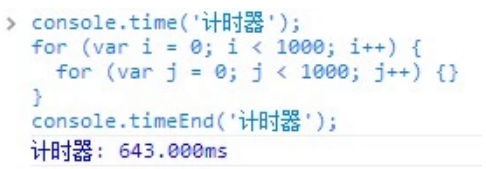
录制结果：



### console.time(name)

计时器，可以将成对的 `console.time()` 和 `console.timeEnd()` 之间代码的运行时间输出到控制台上，`name` 参数可作为标签名。

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




（刚才实际多写了一层循环，结果电脑风扇呜呜转啊，浏览器直接无响应了。。。）

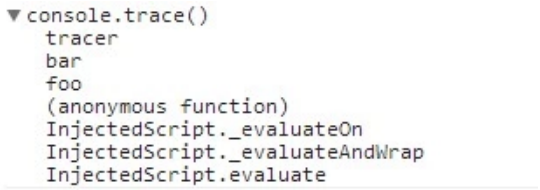
### console.trace()

`console.trace()` 用来追踪函数的调用过程。在大型项目尤其是框架开发中，函数的调用轨迹可以十分复杂，`console.trace()` 方法可以将函数的被调用过程清楚地输出到控制台上。

```
function tracer(a) {
  console.trace();
  return a;
}

function foo(a) {
  return bar(a);
}

function bar(a) {
  return tracer(a);
}
```

```
var a = foo('tracer');
```

输出：




**console.warn(object[, object...])**

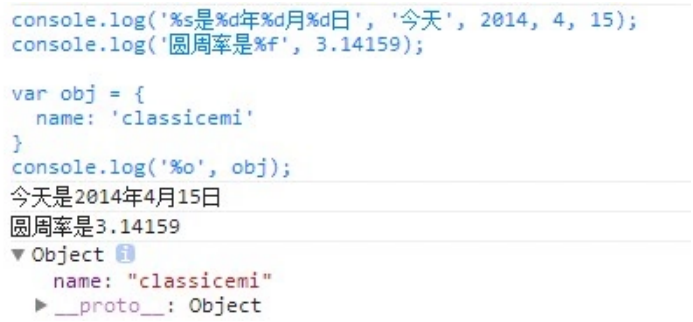
输出参数的内容，作为警告提示。

```
console.warn('warn'); // 输出 warn
```

**占位符**

`console` 对象上的五个直接输出方法，`console.log()`，`console.warn()`，`console.error()`，`console.exception()` (等同于 `console.error()`)和 `console.info()`，都可以使用占位符。支持的占位符有四种，分别是字符( `%s` )、整数( `%d` 或 `%i` )、浮点数( `%f` )和对象( `%o` )。

```
console.log('%s是%d年%d月%d日', '今天', 2014, 4, 15);
console.log('圆周率是%f', 3.14159);

var obj = {
  name: 'classicemi'
}
console.log('%o', obj);
```




还有一种特殊的标示符 %c ，对输出的文字可以附加特殊的样式，当进行大型项目开发的时候，代码中可能有很多其他开发者添加的控制台语句。开发者对自己的输出定制特别的样式就可以方便自己在眼花缭乱的输出结果中一眼看到自己需要的内容。想象力丰富的童鞋也可以做出有创意的输出信息，比如常见的招聘信息和个人介绍啥的。

输出结果：

```
console.log('%cMy name is classicemi.', 'color: #fff; background: #f40; font-size: 24
```


%c 标示符可以用各种 CSS 语句来为输出添加样式，再随便举个栗子， background 属性的 url() 中添加图片路径就可以实现图片的输出了，各位前端童鞋快施展你们的 CSS 神技来把控制台玩坏吧~~

## 参考文献

1. https://developer.mozilla.org/en-US/docs/Web/API/console
2. http://www.ruanyifeng.com/blog/2011/03/firebug_console_tutorial.html

3. http://blog.segmentfault.com/civerzhu/1190000000425386
4. http://blog.mariusschulz.com/2013/11/13/advanced-javascript-debugging-with-consoletable

赞 | 34 收藏 | 153


## 你可能感兴趣的

- **Web开发工具系列（1）---Firebug中的console API** zhangding firefox javascript
- **javascript中console命令详解** specialCoder console javascript
- **如何禁止JavaScript对象重写?** Fundebug javascript
- **JavaScript 错误处理与调试——“调试技术与常见的IE错误”的注意要点** Oliveryoung javascript
- **如何使用JavaScript控制台改进工作流程** songfens typescript sass html5 html javascript
- **JS进阶篇--JS之console.log详解以及兄弟姐们邻居方法扩展** 风雨后见彩虹 javascript console.log
- **关于console的小技巧** aimiy javascript
- **调试技术** zhangding javascript

## 7 条评论

默认排序 | 时间排序

**StephenLee** · 2014年04月25日

再补充几个： 1. `console.debug()` 用于输出输出debug的信息。 2. `console.timeStamp()` 用于标记运行时的timeline信息。 3. `console.memory` 用于显示此刻使用的内存信息（这是一个属性而不是方法）。 4. `console.clear()` 清空控制台的内容（当然你可以用快捷键ctrl+L）。 5. `$0` 可以在控制台输出在Elements面板选中的页面元素。 6. `$_` 表示上一次在控制台键入的命令，你也可以用快捷键“上方向键”来达到

同样的效果。 7. `$x` 可以用xPath的语法来获取页面上的元素。 `btw`，博主，我修改多次是因为我的回复不能换行，所以在尝试换行，不好意思。经询问，发现这是换行的 `bug`，打扰博主了。:)

赞 +2　回复

**公子** · 2014年04月25日

再补充几个： 1. `console.debug()` 用于输出输出debug的信息。 1. `console.timeStamp()` 用于标记运行时的timeline信息。 1. `console.memory` 用于显示此刻使用的内存信息（这是一个属性而不是方法）。 1. `console.clear()` 清空控制台的内容（当然你可以用快捷键 `ctrl+L` ）。 1. `$0` 可以在控制台输出在Elements面板选中的页面元素。 1. `$_` 表示上一次在控制台键入的命令，你也可以用快捷键“上方向键”来达到同样的效果。 1. `$x` 可以用xPath的语法来获取页面上的元素。 你好我是来测试bug的，如有打扰请见谅

赞 +1　回复

**MingjunYang** · 2014年04月25日

很牛逼的东西!~~

赞　回复

**JustQyx** · 2014年04月25日

好东西啊，之前我就只知道 console.log，现在想想太无知了。

赞　回复

**公子** · 2014年04月25日

您好我是来测试bug的 您好我是来测试bug的

赞　回复

**classicemi** 作者 · 2014年04月25日

感谢补充~~

赞　回复

**read_lee** · 2015年01月14日

涨见识了~

赞 回复

**呦呦烈鸟** · 2016年02月26日

涨姿势了，

赞 回复

**HECQ** · 2018年04月27日

如何使用console对象在js代码中，生成profile，之后保存到本地文件中呢。

赞 回复

文明社会，理性评论

发布评论


移动客户端

Copyright © 2011-2019 SegmentFault. 当前呈现版本 19.02.27
浙ICP备 15005796号-2  浙公网安备 33010602002000号  杭州堆栈科技有限公司版权所有

CDN 存储服务由 又拍云 赞助提供