# 翻译：ECMAScript 5.1简介

这篇文章发布于 2012年01月4日，星期三，22:01，归类于 外文翻译。 阅读 100009 次, 今日 4 次 3 条评论

> by zhangxinxu from http://www.zhangxinxu.com
> 本文地址：http://www.zhangxinxu.com/wordpress/?p=2148

> 原文地址：Introducing ECMAScript 5.1
> 翻译编辑：张鑫旭

## 内容导读：



## 简介

ECMAScript 5.1 (或仅 ES5) 是ECMAScript(基于JavaScript的规范)标准最新修正。 与HTML5规范进程本质类似，ES5通过对现有JavaScript方法添加语句和原生ECMAScript对象做合并实现标准化。ES5还引入了一个语法的严格变种，被称为”严格模式(strict mode)”。

本文我们将介绍一些有用的改变和添加。关于完整列表，请参考官方ECMAScript语言规范附录D和E (PDF下载, 3MB)，或者查看http://www.ecmascript.org/; 您还可以以HTML形式查看 – Michael[tm] Smith非官方的HTML版本说明。

## 浏览器支持

随着Opera 11.60的发布, 所有5大浏览器都支持ES5, 除了一些实现的bugs. 除非另有说明，本文中提到的的一切可以用在以下浏览器版本（或更高）：

- Opera 11.60
- Internet Explorer 9*
- Firefox 4
- Safari 5.1**
- Chrome 13

> * IE9不支持严格模式 — IE10 添加

> ** Safari 5.1 仍不支持 `Function.prototype.bind`, 尽管 `Function.prototype.bind` 现在已经被Webkit所支持。

对于旧版浏览器的支持信息，您可以查看Juriy Zaytsev很赞的 ECMAScript 5 兼容性表。

## ES5的严格模式

严格模式给作者提供了选择一个限制性更强语言变种的方式——给作者提供额外的可靠性给用户提供额外的安全性。在JS文件或是函数的顶部添加 `"use strict"` 即可启用严格模式。因为 `"use strict"` 就是个字符串，因此其会被旧版浏览器安全地忽视。

```
"use strict";
```

```
function strict(){
  "use strict";
  //...
}

function sloppy(){
  eval("window.foo = 'bar'");
}
```

在严格模式下运行脚本，不少导致提醒或buggy行为的事情会抛出错误，例如：

- 未声明的变量赋值抛出一个 `ReferenceError`，而不是创建一个全局变量。
- 不止一次对对象字面量分配相同的属性会抛出 `SyntaxError`.
- 使用 `with` 语句抛出 `SyntaxError`.

MDSN的严格模式文章有个关于所有这些差异很有用的总结表格。

| Language element | Restriction | Example |
| --- | --- | --- |
| Variable | Using a variable without declaring it. | `testvar = 4;` |
| Read-only property | Writing to a read-only property. | `var testObj`<br>`prop1: {`<br>`valu`<br>`writ`<br>`},` |



# JSON

ES5提供一个全局的 `JSON` 对象，用来序列化( `JSON.stringify` )和反序列化(JSON.parse)对象为JSON格式。

对于老的浏览器，可以考虑使用Douglas Crockford的json2.js, 可以让旧的浏览器实现同样的功能（原始支持功能测试后）。

**`JSON.parse(text [, reviver])`**

`JSON.parse` 接受文本(JSON格式)并转换成一个ECMAScript值。该可选的reviver参数是有带有 `key` 和 `value` 两个参数的函数，其作用于结果——让过滤和转换返回值成为可能。

```
>> var result = JSON.parse('{"a": 1, "b": "2"}');
Object

>> result.b
```

```
"2"
```

如果我们想确保解析的值是个整数，我们可以使用reviver方法。

```
var result = JSON.parse('{"a": 1, "b": "2"}', function(key, value){
  if (typeof value == 'string'){
    return parseInt(value);
  } else {
    return value;
  }
})

>> result.b
2
```

**`JSON.stringify(value [, replacer [, space]])`**

`JSON.stringify` 允许作者接受一个ECMAScript值然后转换成JSON格式的字符串。 在其最简单的形式中， `JSON.stringify` 接受一个值返回一个字符串，

```
>>> var mike = JSON.stringify({mike: "taylor"})
undefined

>> mike
'{"mike": "taylor"}'

>> typeof mike
"string"
```

如果我们需要改变值字符串化的方式，或是对我们选择的提供过滤，我们可以将其传给replacer函数。例如，我们想过滤出即将被字符串化的对象中值为13的属性：

```
var nums = {
  "first": 7,
  "second": 14,
  "third": 13
}

var luckyNums = JSON.stringify(nums, function(key, value){
  if (value == 13) {
    return undefined;
  } else {
    return value;
  }
});

>> luckyNums
'{"first": 7, "second": 14}'
```

如果replacer方法返回 `undefined` ，则键值对就不会包含在最终的JSON中。我们同样可以传递一个space参数以便获得返回结果的可读性帮助。space参数可以是个数字，表明了作缩进的JSON字符串或字符串每个水平上缩进的空格数。如果参数是个超过10的数值，或是超过10个字符的字符串，将导致取数值10或是截取前10个字符。

```
var luckyNums = JSON.stringify(nums, function(key, value) {
```

```
  if (value == 13) {
    return undefined;
  } else {
    return value;
  }
}, 2);

>> luckyNums
'{
  "first":7,
  "second":14
}'
```

## 附加对象

下面的方法是添加到 `Object` 上的构造器:

- `Object.getPrototypeOf`
- `Object.getOwnPropertyDescriptor`
- `Object.getOwnPropertyNames`
- `Object.create`
- `Object.defineProperty`
- `Object.defineProperties`
- `Object.seal`
- `Object.freeze`
- `Object.preventExtensions`
- `Object.isSealed`
- `Object.isFrozen`
- `Object.isExtensible`
- `Object.keys`

这些新增的好处之一是对象的属性有了更多控制，例如哪些是允许被修改的，哪些是可以枚举的，哪些是可以删除的等。这个的实现通过程序访问对象的*属性描述符(property descriptors)*. 例如:

```
var cat = {};

Object.defineProperty(cat, "name", {
  value: "Maru",
  writable: false,
  enumerable: true,
  configurable: false
});

Object.defineProperty(cat, "skill", {
  value: "exploring boxes",
  writable: true,
  enumerable: true,
  configurable: true
});
```

对于我们的 `cat` 对象, 其名字 `name` 不能被改变，但是会出现在 `for-in` 循环中。在其他方面，Maru擅长探索盒子(exploring boxes), 但是可以在将来改变，因为 `skill` 属性是 `writable` 和 `configurable` 的。

在之后的文章我们将详细探讨所有附加的对象。

## 额外的数组

以下方法添加到了Array `prototype` 对象上:

- `Array.prototype.indexOf`
- `Array.prototype.lastIndexOf`
- `Array.prototype.every`
- `Array.prototype.some`
- `Array.prototype.forEach`
- `Array.prototype.map`
- `Array.prototype.filter`
- `Array.prototype.reduce`
- `Array.prototype.reduceRight`

关于ES5数组"extras" Dmitry Soshnikov写过一篇有深度的参考文章。

Dmitry的文章中有一个没有提到，就是 `Array.isArray`，正如你看到的，这厮直接写在了 `Array` 构造器上，而不是 `prototype` 对象上。 `Array.isArray` 会按照你所期待的那样去做 — 这是一个根据参数的[[Class]]内部属性是否是"Array"返回 `true` 或 `false`.

```
Array.isArray("NO U")
>> false

Array.isArray(["NO", "U"])
>> true
```

在ES3中，唯一可靠的确定一个值是数组的方式就是使用"the Miller Device", 即比对一个数组其内在的 `[[Class]]` 属性。

```
Object.prototype.toString.apply(value) === '[object Array]'
```

## `Function.prototype.bind(thisArg [, arg1 [, arg2, …]])`

`Function.prototype.bind` 返回一个新的函数对象，该函数对象的*this*绑定到了 `thisArg` 参数上。从本质上讲，这允许你在其他对象链中执行一个函数。

```
function locate(){
  console.log(this.location);
}

function Maru(location){
  this.location = location;
}

var kitty = new Maru("cardboard box");

var locateMaru = locate.bind(kitty);

locateMaru();
```

在这个例子中，我们将Maru对象的上下文应用在`location`函数中。因为`location`是个全局对象的属性，其`this`值就是全局对象(`window`)。在这种情况下，我们向上寻找cat, 并不是`Location`对象，因为我们可以通过绑定的总是`kitty`的`this`值创建一个新方法`locateMaru`.

## 补充参考

- ECMAScript 5 对象和属性 by John Resig
- 理解JavaScript函数调用和"this" by Yehuda Katz
- JavaScript严格模式 by Angus Croll
- ECMA-262-5详细 介绍 by Dmitry Soshnikov
- ECMAScript 5 兼容性表 by Juriy Zaytsev



> 《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

（本篇完） // 想要打赏？点击这里。有话要说？点击这里。



« zSlide-基于CSS3/HTML5演示文档jQuery插件　　CSS radio/checkbox单复选框元素显隐技术 »

猜你喜欢

- ECMAScript 5(ES5)中bind方法、自定义及小拓展
- 我是如何理解"Another JavaScript quiz"中的题目
- ES5中新增的Array方法详细说明
- 翻译-高质量JavaScript代码书写基本要点
- 翻编-JavaScript有关的10个怪癖和秘密
- 近期手机网页项目一些杂碎心得分享
- 我对原型对象中this的一个懵懂错误认识
- js面向数据编程(DOP)一点分享
- 基于HTML模板和JSON数据的JavaScript交互
- JS前端创建html或json文件并浏览器导出下载
- 翻译 - CSS高峰会议内容精选



标签： bind, ECMAScript5, ES5, javascript, json, 对象, 数组

### 发表评论（目前3条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. **yanhaijing说道：**

2014年09月11日 14:59

推荐 http://yanhaijing.com/es5

回复

2. **aaa说道：**

2014年08月21日 02:30

while(;){}

回复

3. **maooson说道：**

2012年01月5日 08:59

“MDSN的严格模式文章有个关于所有这些差异很有用的总结表格”，应该是”MSDN”。

回复


最新文章

» 常见的CSS图形绘制合集
» 粉丝群第1期CSS小测点评与答疑
» 分享三个纯CSS实现26个英文字母的案例
» 小tips: 纯CSS实现打字动画效果
» CSS/CSS3 box-decoration-break属性简介
» CSS :placeholder-shown伪类实现Material Design占位符交互效果
» 从天猫某活动视频不必要的3次请求说起
» CSS vector-effect与SVG stroke描边缩放
» CSS ::backdrop伪元素是干嘛用的?
» 周知：CSS -webkit-伪元素选择器不再导致整行无效

今日热门

» 常见的CSS图形绘制合集 (193)
» 未来必热：SVG Sprite技术介绍 (120)
» 粉丝群第1期CSS小测点评与答疑 (115)
» HTML5终极备忘大全（图片版+文字版） (93)
» 让所有浏览器支持HTML5 video视频标签 (86)
» Selectivizr-让IE6~8支持CSS3伪类和属性选择器 (82)
» CSS3下的147个颜色名称及对应颜色值 (80)
» 视区相关单位vw, vh..简介以及可实际应用场景 (76)
» 写给自己看的display: flex布局教程 (76)
» 小tips: 纯CSS实现打字动画效果 (76)

今年热议

» 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)
» 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
» 看，for..in和for..of在那里吵架！ (60)

» 是时候好好安利下LuLu UI框架了！ (47)
» 原来浏览器原生支持JS Base64编码解码 (35)
» 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
» 炫酷H5中序列图片视频化播放的高性能实现 (31)
» CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
» windows系统下批量删除OS X系统.DS_Store文件 (26)
» 写给自己看的display: flex布局教程 (26)

## 猜你喜欢

- ECMAScript 5(ES5)中bind方法、自定义及小拓展
- 我是如何理解"Another JavaScript quiz"中的题目
- ES5中新增的Array方法详细说明
- 翻译-高质量JavaScript代码书写基本要点
- 翻编-JavaScript有关的10个怪癖和秘密
- 近期手机网页项目一些杂碎心得分享
- 我对原型对象中this的一个懵懂错误认识
- js面向数据编程(DOP)一点分享
- 基于HTML模板和JSON数据的JavaScript交互
- JS前端创建html或json文件并浏览器导出下载
- 翻译 - CSS高峰会议内容精选