ES5中新增的Array方法详细说明

这篇文章发布于 2013年04月25日，星期四，21:43，归类于 JS实例。 阅读 229741 次, 今日 45 次 44 条评论

by zhangxinxu from http://www.zhangxinxu.com
本文地址：http://www.zhangxinxu.com/wordpress/?p=3220

一、前言-索引

ES5中新增的不少东西，了解之对我们写JavaScript会有不少帮助，比如数组这块，我们可能就不需要去有板有眼地 `for` 循环了。

ES5中新增了写数组方法，如下：

1. forEach (js v1.6)
2. map (js v1.6)
3. filter (js v1.6)
4. some (js v1.6)
5. every (js v1.6)
6. indexOf (js v1.6)
7. lastIndexOf (js v1.6)
8. reduce (js v1.8)
9. reduceRight (js v1.8)

浏览器支持

- Opera 11+
- Firefox 3.6+
- Safari 5+
- Chrome 8+
- Internet Explorer 9+

对于让人失望很多次的IE6-IE8浏览器，Array原型扩展可以实现以上全部功能，例如 `forEach` 方法:

```
// 对于古董浏览器，如IE6-IE8

if (typeof Array.prototype.forEach != "function") {
  Array.prototype.forEach = function () {
    /* 实现 */
  };
}
```

二、一个一个来

1. **forEach**

   `forEach` 是Array新方法中最基本的一个，就是遍历，循环。例如下面这个例子：

   ```
   [1, 2 ,3, 4].forEach(alert);
   ```

等同于下面这个传统的`for`循环：

```
var array = [1, 2, 3, 4];

for (var k = 0, length = array.length; k < length; k++) {
  alert(array[k]);
}
```

Array在ES5新增的方法中，参数都是`function`类型，默认有传参，这些参数分别是？见下面：

```
[1, 2 ,3, 4].forEach(console.log);

// 结果：

// 1, 0, [1, 2, 3, 4]
// 2, 1, [1, 2, 3, 4]
// 3, 2, [1, 2, 3, 4]
// 4, 3, [1, 2, 3, 4]
```

显而易见，`forEach`方法中的`function`回调支持3个参数，第1个是遍历的数组内容；第2个是对应的数组索引，第3个是数组本身。

因此，我们有：

```
[].forEach(function(value, index, array) {
    // ...
});
```

对比jQuery中的`$.each`方法：

```
$.each([], function(index, value, array) {
    // ...
});
```

会发现，第1个和第2个参数正好是相反的，大家要注意了，不要记错了。后面类似的方法，例如`$.map`也是如此。

现在，我们就可以使用`forEach`卖弄一个稍显完整的例子了，数组求和：

```
var sum = 0;

[1, 2, 3, 4].forEach(function (item, index, array) {
  console.log(array[index] == item); // true
  sum += item;
});

alert(sum); // 10
```

再下面，更进一步，`forEach`除了接受一个必须的回调函数参数，还可以接受一个可选的上下文参数（改变回调函数里面的`this`指向）（第2个参数）。

```
array.forEach(callback,[ thisObject])
```

例子更能说明一切：

```
var database = {
  users: ["张含韵", "江一燕", "李小璐"],
  sendEmail: function (user) {
    if (this.isValidUser(user)) {
      console.log("你好，" + user);
    } else {
      console.log("抱歉，"+ user +"，你不是本家人");
    }
  },
  isValidUser: function (user) {
    return /^张/.test(user);
  }
};

// 给每个人法邮件
database.users.forEach(  // database.users中人遍历
  database.sendEmail,    // 发送邮件
  database               // 使用database代替上面标红的this
);

// 结果：
// 你好，张含韵
// 抱歉，江一燕，你不是本家人
// 抱歉，李小璐，你不是本家
```

如果这第2个可选参数不指定，则使用全局对象代替（在浏览器是为`window`），严格模式下甚至是`undefined`.

另外，forEach不会遍历纯粹“占着官位吃空饷”的元素的，例如下面这个例子：

```
var array = [1, 2, 3];

delete array[1]; // 移除 2
alert(array); // "1,,3"

alert(array.length); // but the length is still 3

array.forEach(alert); // 弹出的仅仅是1和3
```

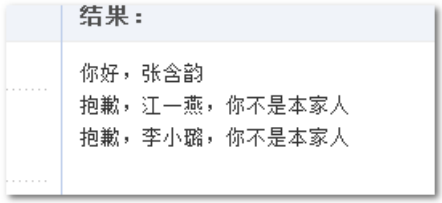
综上全部规则，我们就可以对IE6-IE8进行仿真扩展了，如下代码：

```
// 对于古董浏览器，如IE6-IE8

if (typeof Array.prototype.forEach != "function") {
  Array.prototype.forEach = function (fn, context) {
    for (var k = 0, length = this.length; k < length; k++) {
      if (typeof fn === "function" && Object.prototype.hasOwnProperty.call(this, k)) {
        fn.call(context, this[k], k, this);
      }
```

```
    }
  };
}
```

现在拿上面“张含韵”的例子测下我们扩展的 `forEach` 方法，您可能狠狠地点击这里：兼容处理的forEach方法demo

例如IE7浏览器下：



2. **map**

这里的 `map` 不是“地图”的意思，而是指“映射”。 `[].map();` 基本用法跟 `forEach` 方法类似：

```
array.map(callback,[ thisObject]);
```

`callback` 的参数也类似：

```
[].map(function(value, index, array) {
    // ...
});
```

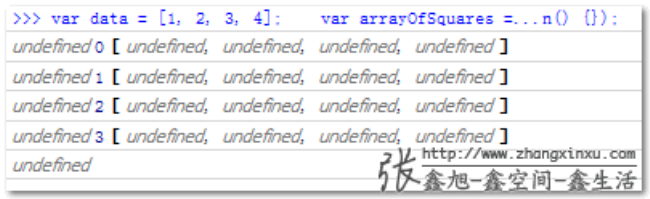
`map` 方法的作用不难理解，“映射”嘛，也就是原数组被“映射”成对应新数组。下面这个例子是数值项求平方：

```
var data = [1, 2, 3, 4];

var arrayOfSquares = data.map(function (item) {
  return item * item;
});

alert(arrayOfSquares); // 1, 4, 9, 16
```

`callback` 需要有 `return` 值，如果没有，就像下面这样：

```
var data = [1, 2, 3, 4];
var arrayOfSquares = data.map(function() {});

arrayOfSquares.forEach(console.log);
```

结果如下图，可以看到，数组所有项都被映射成了 `undefined` ：



在实际使用的时候，我们可以利用 `map` 方法方便获得对象数组中的特定属性值们。例如下面这个例子（之后的兼容demo也是该例子）：

```
var users = [
  {name: "张含韵", "email": "zhang@email.com"},
  {name: "江一燕",   "email": "jiang@email.com"},
  {name: "李小璐",  "email": "li@email.com"}
];

var emails = users.map(function (user) { return user.email; });

console.log(emails.join(", ")); // zhang@email.com, jiang@email.com, li@email.com
```

`Array.prototype` 扩展可以让IE6-IE8浏览器也支持 `map` 方法：

```
if (typeof Array.prototype.map != "function") {
  Array.prototype.map = function (fn, context) {
    var arr = [];
    if (typeof fn === "function") {
      for (var k = 0, length = this.length; k < length; k++) {
         arr.push(fn.call(context, this[k], k, this));
      }
    }
    return arr;
  };
}
```

您可以狠狠地点击这里：兼容map方法测试demo

结果显示如下图，IE6浏览器：

3. **filter**

   `filter` 为“过滤”、“筛选”之意。指数组 `filter` 后，返回过滤后的新数组。用法跟 `map` 极为相似：

   ```
   array.filter(callback,[ thisObject]);
   ```

   `filter` 的 `callback` 函数需要返回布尔值 `true` 或 `false`. 如果为 `true` 则表示，恭喜你，通过啦！如果为 `false`, 只能高歌“我只能无情地将你抛弃……”。

   可能会疑问，一定要是 `Boolean` 值吗？我们可以简单测试下嘛，如下：

   ```
   var data = [0, 1, 2, 3];
   var arrayFilter = data.filter(function(item) {
       return item;
   });
   console.log(arrayFilter); // [1, 2, 3]
   ```

   有此可见，返回值只要是弱等于 `== true/false` 就可以了，而非非得返回 `=== true/false`.

   因此，我们在为低版本浏览器扩展时候，无需关心是否返回值是否是纯粹布尔值（见下黑色代码部分）：

   ```
   if (typeof Array.prototype.filter != "function") {
     Array.prototype.filter = function (fn, context) {
       var arr = [];
       if (typeof fn === "function") {
   ```

```
        for (var k = 0, length = this.length; k < length; k++) {
           fn.call(context, this[k], k, this) && arr.push(this[k]);
        }
     }
     return arr;
  };
}
```

接着上面 `map` 筛选邮件的例子，您可以狠狠地点击这里：兼容处理后filter方法测试demo

主要测试代码为：

```
var emailsZhang = users
  // 获得邮件
  .map(function (user) { return user.email; })
  // 筛选出zhang开头的邮件
  .filter(function(email) {  return /^zhang/.test(email); });

console.log(emailsZhang.join(", ")); // zhang@email.com
```

> 实际上，存在一些语法糖可以实现 `map+filter` 的效果，被称之为“数组简约式(Array comprehensions)”。目前，仅FireFox浏览器可以实现，展示下又不会怀孕：
>
> ```
> var zhangEmails = [user.email for each (user in users) if (/^zhang/.test(user.email)) ];
>
> console.log(zhangEmails); // [zhang@email.com]
> ```

4. **some**

   `some` 意指“某些”，指是否“某些项”合乎条件。与下面的 `every` 算是好基友， `every` 表示是否“每一项”都要靠谱。用法如下：

   ```
   array.some(callback,[ thisObject]);
   ```

   例如下面的简单使用：

   ```
   var scores = [5, 8, 3, 10];
   var current = 7;

   function higherThanCurrent(score) {
     return score > current;
   }

   if (scores.some(higherThanCurrent)) {
     alert("朕准了！");
   }
   ```

   结果弹出了“朕准了”文字。 `some` 要求至少有1个值让 `callback` 返回 `true` 就可以了。显然， `8 > 7` ，因此 `scores.some(higherThanCurrent)` 值为 `true` .

   我们自然可以使用 `forEach` 进行判断，不过，相比 `some` ,不足在于， `some` 只有有 `true` 即返

回不再执行了。

IE6-IE8扩展如下：

```
if (typeof Array.prototype.some != "function") {
  Array.prototype.some = function (fn, context) {
 var passed = false;
 if (typeof fn === "function") {
      for (var k = 0, length = this.length; k < length; k++) {
    if (passed === true) break;
    passed = !!fn.call(context, this[k], k, this);
   }
    }
 return passed;
  };
}
```

于是，我们就有了“朕准了”的demo，您可以狠狠地点击这里：兼容处理后的some方法demo

5. **every**

跟 `some` 的基友关系已经是公开的秘密了，同样是返回Boolean值，不过， `every` 需要每一个妃子都要让朕满意，否则——“来人，给我拖出去砍了！”

IE6-IE8扩展（与 `some` 相比就是 `true` 和 `false` 调换一下）：

```
if (typeof Array.prototype.every != "function") {
  Array.prototype.every = function (fn, context) {
    var passed = true;
    if (typeof fn === "function") {
       for (var k = 0, length = this.length; k < length; k++) {
          if (passed === false) break;
          passed = !!fn.call(context, this[k], k, this);
      }
    }
    return passed;
  };
}
```

还是那个朕的例子，您可以狠狠地点击这里：是否every妃子让朕满意demo

```
if (scores.every(higherThanCurrent)) {
  console.log("朕准了！");
} else {
  console.log("来人，拖出去斩了！");
}
```

结果是：

6. **indexOf**

`indexOf` 方法在字符串中自古就有， `string.indexOf(searchString, position)` 。数组这里的 `indexOf` 方法与之类似。

```
array.indexOf(searchElement[, fromIndex])
```

返回整数索引值，如果没有匹配（严格匹配），返回 `-1`．`fromIndex` 可选，表示从这个位置开始搜索，若缺省或格式不合要求，使用默认值 `0`，我在FireFox下测试，发现使用字符串数值也是可以的，例如 `"3"` 和 `3` 都可以。

```
var data = [2, 5, 7, 3, 5];

console.log(data.indexOf(5, "x")); // 1 ("x"被忽略)
console.log(data.indexOf(5, "3")); // 4 (从3号位开始搜索)

console.log(data.indexOf(4)); // -1 (未找到)
console.log(data.indexOf("5")); // -1 (未找到，因为5 !== "5")
```

兼容处理如下：

```
if (typeof Array.prototype.indexOf != "function") {
  Array.prototype.indexOf = function (searchElement, fromIndex) {
    var index = -1;
    fromIndex = fromIndex * 1 || 0;

    for (var k = 0, length = this.length; k < length; k++) {
      if (k >= fromIndex && this[k] === searchElement) {
          index = k;
          break;
      }
    }
    return index;
  };
}
```

一个路子下来的，显然，轮到demo了，您可以狠狠地点击这里：兼容处理后indexOf方法测试demo

下图为ietester IE6下的截图：

7. **lastIndexOf**

   `lastIndexOf` 方法与 `indexOf` 方法类似：

```
array.lastIndexOf(searchElement[, fromIndex])
```

只是 `lastIndexOf` 是从字符串的末尾开始查找，而不是从开头。还有一个不同就是 `fromIndex` 的默认值是 `array.length - 1` 而不是 `0`．

IE6等浏览器如下折腾：

```
if (typeof Array.prototype.lastIndexOf != "function") {
  Array.prototype.lastIndexOf = function (searchElement, fromIndex) {
    var index = -1, length = this.length;
    fromIndex = fromIndex * 1 || length - 1;

    for (var k = length - 1; k > -1; k-=1) {
        if (k <= fromIndex && this[k] === searchElement) {
            index = k;
            break;
        }
    }
```

```
    return index;
  };
}
```

于是，则有：

```
var data = [2, 5, 7, 3, 5];

console.log(data.lastIndexOf(5)); // 4
console.log(data.lastIndexOf(5, 3)); // 1（从后往前，索引值小于3的开始搜索）

console.log(data.lastIndexOf(4)); // -1（未找到）
```

懒得截图了，结果查看可狠狠地点击这里：lastIndexOf测试demo

8. **reduce**

`reduce`是JavaScript 1.8中才引入的，中文意思为“减少”、“约简”。不过，从功能来看，我个人是无法与“减少”这种含义联系起来的，反而更接近于“迭代”、“递归(recursion)”，擦，因为单词这么接近，不会是ECMA-262 5th制定者笔误写错了吧~~🧐

此方法相比上面的方法都复杂，用法如下：

```
array.reduce(callback[, initialValue])
```

`callback`函数接受4个参数：之前值、当前值、索引值以及数组本身。`initialValue`参数可选，表示初始值。若指定，则当作最初使用的`previous`值；如果缺省，则使用数组的第一个元素作为`previous`初始值，同时`current`往后排一位，相比有`initialValue`值少一次迭代。

```
var sum = [1, 2, 3, 4].reduce(function (previous, current, index, array) {
  return previous + current;
});

console.log(sum); // 10
```

说明：

1. 因为`initialValue`不存在，因此一开始的`previous`值等于数组的第一个元素。
2. 从而`current`值在第一次调用的时候就是`2`.
3. 最后两个参数为索引值`index`以及数组本身`array`.

以下为循环执行过程：

```
// 初始设置
previous = initialValue = 1, current = 2

// 第一次迭代
previous = (1 + 2) =  3, current = 3

// 第二次迭代
previous = (3 + 3) =  6, current = 4

// 第三次迭代
previous = (6 + 4) =  10, current = undefined（退出）
```

有了`reduce`，我们可以轻松实现二维数组的扁平化：

```
var matrix = [
  [1, 2],
  [3, 4],
  [5, 6]
];

// 二维数组扁平化
var flatten = matrix.reduce(function (previous, current) {
  return previous.concat(current);
});

console.log(flatten); // [1, 2, 3, 4, 5, 6]
```

兼容处理IE6-IE8：

```
if (typeof Array.prototype.reduce != "function") {
  Array.prototype.reduce = function (callback, initialValue ) {
     var previous = initialValue, k = 0, length = this.length;
     if (typeof initialValue === "undefined") {
        previous = this[0];
        k = 1;
     }

    if (typeof callback === "function") {
      for (k; k < length; k++) {
         this.hasOwnProperty(k) && (previous = callback(previous, this[k], k, this));
      }
    }
    return previous;
  };
}
```

然后，测试整合，demo演示，您可以狠狠地点击这里：兼容IE6的reduce方法测试demo

IE6浏览器下结果如下图：

9. **reduceRight**

`reduceRight`跟`reduce`相比，用法类似：

```
array.reduceRight(callback[, initialValue])
```

实现上差异在于`reduceRight`是从数组的末尾开始实现。看下面这个例子：

```
var data = [1, 2, 3, 4];
var specialDiff = data.reduceRight(function (previous, current, index) {
  if (index == 0) {
    return previous + current;
  }
  return previous - current;
});

console.log(specialDiff); // 0
```

结果`0`是如何得到的呢?

我们一步一步查看循环执行:

```
// 初始设置
index = 3, previous = initialValue = 4, current = 3

// 第一次迭代
index = 2, previous = (4- 3) = 1, current = 2

// 第二次迭代
index = 1, previous = (1 - 2) = -1, current = 1

// 第三次迭代
index = 0, previous = (-1 + 1) = 0, current = undefined (退出)
```

为使低版本浏览器支持此方法，您可以添加如下代码:

```
if (typeof Array.prototype.reduceRight != "function") {
  Array.prototype.reduceRight = function (callback, initialValue ) {
    var length = this.length, k = length - 1, previous = initialValue;
    if (typeof initialValue === "undefined") {
        previous = this[length - 1];
        k--;
    }
    if (typeof callback === "function") {
       for (k; k > -1; k-=1) {
          this.hasOwnProperty(k) && (previous = callback(previous, this[k], k, this))
;
       }
    }
    return previous;
  };
}
```

您可以狠狠地点击这里：reduceRight简单使用demo

对比FireFox浏览器和IE7浏览器下的结果:

三、更进一步的应用

我们还可以将上面这些数组方法应用在其他对象上。

例如，我们使用forEach遍历DOM元素。

```
var eleDivs = document.getElementsByTagName("div");
Array.prototype.forEach.call(eleDivs, function(div) {
    console.log("该div类名是: " + (div.className || "空"));
});
```

可以输出页面所有`div`的类名，您可以狠狠地点击这里：Array新方法forEach遍历DOM demo

结果如下，IE6下，demo结果:

等很多其他类数组应用。

四、最后一点点了

本文为低版本IE扩展的Array方法我都合并到一个JS中了，您可以轻轻的右键这里或下载或查看：es5-array.js

以上所有未IE扩展的方法都是自己根据理解写的，虽然多番测试，难免还会有细节遗漏的，欢迎指出来。

参考文章：
JavaScript array “extras” in detail
Array.forEach

> 《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。

« CSS hover效果的逆向思维实现

我是如何理解”Another JavaScript quiz”中的题目 »

猜你喜欢

- 翻译：ECMAScript 5.1简介
- HTML <area><map>标签及在实际开发中的应用
- 看，for..in和for..of在那里吵架！
- 近期手机网页项目一些杂碎心得分享
- ECMAScript 5(ES5)中bind方法、自定义及小拓展
- 我对原型对象中this的一个懵懂错误认识
- 我是如何理解"Another JavaScript quiz"中的题目
- CSS实现兼容性的渐变背景(gradient)效果
- CSS实现跨浏览器的box-shadow盒阴影效果(2)
- 小tip:IE不支持CSS3多背景的替代解决方案
- canvas实现iPhoneX炫彩壁纸屏保外加pixi.js流体动效

分享到： 0

标签： ES5, every, filter, forEach, indexOf, lastIndexOf, map, reduce, reduceRight, some, 数组

发表评论（目前44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. 一只菜鸟前端攻城狮说道：
2018年08月12日 20:32

6的头皮发麻

回复

2. **Ricky**说道：
2018年06月14日 05:25

谢天谢地，我终于弄明白forEach了，感谢！

回复

3. 鲁西西说道：
2017年11月30日 00:41

reduceRight的解析错了，index应该分别是：2， 1， 0。希望大神能看到并纠正

回复

4. **libing_cheer**说道：
2017年09月14日 13:39

4. some
function higherThanCurrent(score) {
return score > current;
}
中的score应该为scores

回复

5. 大神方便加个微信吗说道：
2017年06月9日 16:53

好几年前的都给我受益匪浅，多谢

回复

6. 求助：**react**中这样的怎么做**map**遍历进行数据绑定？怎么嵌套说道：
2017年05月14日 17:43

<SubMenu key="sub1" title={{slidedata}}>
Option 1
Option 2
Option 3
Option 4

<SubMenu key="sub2" title={{slidedata}}>
Option 5
Option 6
Option 7

Option 8

……..

回复

7. **求指教说道：**
2017年05月13日 15:50

forEach (js v1.6)

括号里面的 js v1.6 是啥意思啊？？？ js version 1.6 什么鬼

回复

8. **aspwebchh说道：**
2016年12月23日 17:12

函数式编程的方法

回复

9. **牛奶007说道：**
2016年09月5日 20:27

大神，在reduce第一个例子中，在第二次迭代 current应该等于6吧。
《以下原文》
// 第二次迭代
previous = (3 + 3) = 6, current = 4

// 第三次迭代
previous = (6 + 4) = 10, current = undefined (退出)

回复

> **阿磊说道：**
> 2016年12月21日 13:34
>
> // 初始设置
> previous = initialValue = 1, current = 2
>
> // 第一次迭代
> previous = (1 + 2) = 3, current = 3
>
> // 第二次迭代
> previous = (3 + 3) = 6, current = 4
>
> // 第三次迭代
> previous = (6 + 4) = 10, current = undefined (退出)
>
> **********
> 你在好好琢磨下第一次迭代。current代表的是下一个数。
>
> 回复

10. **说道一下说道：**
2016年07月28日 14:36

大神，请问你们公司校招什么时候开始？招聘前端吗？谢谢！

回复

11. **杨金凯说道：**
2016年06月10日 20:31

大神，这个遍历既然指定了起始索引就没有必要在从头开始遍历了吧？
我自己写了一个类似的
Array.prototype.myIndexOf = function(val,index){
var i = -1;

```
index = Math.floor(index*1) || 0; //去掉小数位
for(; index<this.length;index++){
if(this[index] === val){
i = index;
break;
}
}
return i;
}
```

回复

12. **Guyw说道：**
2016年04月28日 17:29

学习了~

回复

13. **12cat说道：**
2016年03月18日 11:56

居然没有张含韵图，差评。

回复

14. **ChieveiT说道：**
2016年03月15日 18:38

好文，感谢分享。另外reduce的意思是“归约”，通过一个过程把一个结果集转化成一个结果，通常说的map-reduce算法中的reduce也是这个意思

回复

15. **ileason说道：**
2016年02月4日 17:10

很好，學到東西了

回复

16. **刺客说道：**
2015年12月16日 10:30

学习了！

回复

17. **我在山上砍柴说道：**
2015年11月21日 21:30

比书上还详细，费心了。

回复

18. **刺客说道：**
2015年10月15日 17:48

很详细！

回复

19. **Saku说道：**
2015年10月13日 21:54

学习了

回复

20. **matrix说道：**
2015年10月10日 13:47

Reduce 应该是归纳的意思~ ca

回复

21. **12sa说道：**
2015年09月24日 10:35

你好 你说的那个forEach中的第二个参数改变上下文环境的例子中，callback函数本身就是database.sendemail,是对象方法调用，本身的this指向不就是database对象吗？为什么会指向全局对象呢？全局对象不是在作为一个函数调用时才指向全局对象吗？

回复

22. **tim说道：**
2015年08月29日 16:18

why call it reduce: 直到reduce到只剩accumulator(你称为previous值),返回它！

回复

23. **ylxdzsw说道：**
2015年06月11日 21:09

Lisper一般把reduce翻译成“归约”

回复

24. **子不语说道：**
2015年05月11日 11:24

之所以叫 reduce , 是要跟 map 对应

回复

25. **tianyn说道：**
2015年04月14日 15:08

一个小小小问题：
原生的indexOf方法的第二个参数支持负数，表示从末尾向前计算，张老板忽略了。

回复

26. **code_bunny说道：**
2015年02月11日 17:14

还有个问题,在使用every的时候,即使是[7,8,,9]这个数组,如果判断是不是>6,它会忽略空项,所以得到的结果也是true,所以在扩展的时候也应该加个hasOwnProperty判断

回复

27. **code_bunny说道：**
2015年02月11日 16:15

hi:

.map方法的扩展,是不是也需要判断一下hasOwnProperty? 否则当数组中的其中一项没有的时候,在垃圾浏览器里会多一项出来,比如如果

是计算,就会多一项NaN…

回复

28. **Computer**说道：
2013年05月19日 22:45

漂亮…

回复

29. **canvast**说道：
2013年05月7日 10:14

forEach在IE6-8的扩展函数中应该不需要Object.prototype.hasOwnProperty.call(this, k)吧?
参考文章http://martinrinehart.com/frontend-engineering/engineers/javascript/arrays/array-loops.html?utm_source=javascriptweekly&utm_medium=email

回复

> 深红说道：
> 2016年07月23日 07:17
>
> Object.prototype.hasOwnProperty.call(this, k)不是为了跳过数组中那些被delete或未被赋值的项吗?
>
> 回复
>
> > 河说道：
> > 2018年03月28日 17:34
> >
> > Object.prototype.hasOwnProperty对delete后的数据有过滤作用，undefined的数据还是返回true呀
> >
> > 回复

30. 大超超。说道：
2013年05月7日 08:46

不明觉厉，ES5用的地方多不多呀？平时都是拿jQuery做js方面的开发。

回复

31. **airoschou**说道：
2013年05月6日 17:44

额，算是hack吗?

回复

32. 小方说道：
2013年05月1日 08:26

DYB 兄弟，不能这样用。console.log函数的形参类型可以有多个，比如这样 用console.log(‘你好%s’,’好人’)；这时，forEach(callback),callback(a[i],i,a);明显实参与形参不匹配了

回复

33. **bennyrice**说道：
2013年04月29日 23:53

forEach 那个例子里~~ 即使不指定database为执行上下文，红色的this也还是会是database本身吧？因为它出现在的那个函数是database的方法。默认会指向对象本身。

回复

张 鑫旭说道：
2013年05月2日 10:38

@bennyrice 不要误导小朋友。如果不指定 `database` 为执行上下文，指向的是 `database.email` 而不是 `database` . 会出错的哦！

回复

> goss说道：
> 2014年08月18日 16:43
>
> database.sendEmail并没有直接被调用，只是做了方法的传递，如果是database.sendEmail()就跟@bennyrice说的一样，以database为上下文；但是database.sendEmail，就要看具体调用了，很有可能是undefined。
>
> 回复

> 河说道：
> 2018年03月28日 17:41
>
> 不指定database的话，应该this 为window吧，我在chrome中得到的this也为window。
>
> 回复

> > hello world说道：
> > 2018年12月12日 22:55
> >
> > this为undefined, 只不过不是在严格模式下, 默认指向了window
> > 你试试看, 严格模式下就是undefined了

34. **bennyrice说道：**
2013年04月29日 23:16

@DYB 可以使用 [1,2,3].forEach(console.log,console) ~~~

回复

35. **DYB说道：**
2013年04月26日 21:35

很不幸。chrome的console不支持[1,2,3].forEach(console.log)

回复

36. **VVG说道：**
2013年04月25日 23:14

你再不得到张含韵她就老了！！！

回复

> 张 鑫旭说道：
> 2013年04月26日 08:53
>
> @VVG 兄弟，只有等老了才有机会啊！！
>
> 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 (77)
» 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

- 翻译：ECMAScript 5.1简介
- HTML <area><map>标签及在实际开发中的应用
- 看，for..in和for..of在那里吵架！
- 近期手机网页项目一些杂碎心得分享
- ECMAScript 5(ES5)中bind方法、自定义及小拓展
- 我对原型对象中this的一个懵懂错误认识
- 我是如何理解"Another JavaScript quiz"中的题目
- CSS实现兼容性的渐变背景(gradient)效果
- CSS实现跨浏览器的box-shadow盒阴影效果(2)
- 小tip:IE不支持CSS3多背景的替代解决方案
- canvas实现iPhoneX炫彩壁纸屏保外加pixi.js流体动效

Designed & Powerd by zhangxinxu
Copyright© 2009-2019 张鑫旭-鑫空间-鑫生活
鄂ICP备09015569号