Array.prototype.filter()

filter() 方法创建一个新数组, 其包含通过所提供函数实现的测试的所有元素。

```
function isBigEnough(value) {
  return value >= 10;
}

var filtered = [12, 5, 8, 130, 44].filter(isBigEnough);

// filtered is [12, 130, 44]

// ES6 way

const isBigEnough = value => value >= 10;

let [...spraed]= [12, 5, 8, 130, 44];

let filtered = spraed.filter(isBigEnough);

// filtered is [12, 130, 44]
```

语法

```
var new_array = arr.filter(callback[, thisArg])
```

参数

callback

用来测试数组的每个元素的函数。调用时使用参数 (element, index, array)。
返回true表示保留该元素（通过测试），false则不保留。

thisArg

可选。执行 `callback` 时的用于 `this` 的值。

返回值

一个新的通过测试的元素的集合的数组

描述

`filter` 为数组中的每个元素调用一次 `callback` 函数，并利用所有使得 `callback` 返回 true 或 等价于 true 的值 的元素创建一个新数组。 `callback` 只会在已经赋值的索引上被调用，对于那些已经被删除或者从未被赋值的索引不会被调用。那些没有通过 `callback` 测试的元素会被跳过，不会被包含在新数组中。

`callback` 被调用时传入三个参数：

1. 元素的值
2. 元素的索引
3. 被遍历的数组

如果为 `filter` 提供一个 `thisArg` 参数，则它会被作为 `callback` 被调用时的 `this` 值。否则， `callback` 的 `this` 值在非严格模式下将是全局对象，严格模式下为 `undefined` 。
The `this` value ultimately observable by `callback` is determined according to the usual rules for determining the `this` seen by a function.

`filter` 不会改变原数组。

`filter` 遍历的元素范围在第一次调用 `callback` 之前就已经确定了。在调用 `filter` 之后被添加到数组中的元素不会被 `filter` 遍历到。如果已经存在的元素被改变了，则他们传入 `callback` 的值是 `filter` 遍历到它们那一刻的值。被删除或从来未被赋值的元素不会被遍历到。

示例

例子：筛选排除掉所有的小值

下例使用 `filter` 创建了一个新数组，该数组的元素由原数组中值大于 10 的元素组成。

```
function isBigEnough(element) {
  return element >= 10;
}
var filtered = [12, 5, 8, 130, 44].filter(isBigEnough);
// filtered is [12, 130, 44]
```

兼容旧环境（Polyfill）

`filter` 被添加到 ECMA-262 标准第 5 版中，因此在某些实现环境中不被支持。可以把下面的代码插入到脚本的开头来解决此问题，该代码允许在那些没有原生支持 `filter` 的实现环境中使用它。该算法是 ECMA-262 第 5 版中指定的算法，假定 `fn.call` 等价于 `Function.prototype.call` 的初始值，且 `Array.prototype.push` 拥有它的初始值。

```
if (!Array.prototype.filter)
{
  Array.prototype.filter = function(fun /* , thisArg*/)
  {
    "use strict";

    if (this === void 0 || this === null)
      throw new TypeError();

    var t = Object(this);
    var len = t.length >>> 0;
    if (typeof fun !== "function")
      throw new TypeError();

    var res = [];
    var thisArg = arguments.length >= 2 ? arguments[1] : void 0;
    for (var i = 0; i < len; i++)
    {
      if (i in t)
      {
        var val = t[i];

        // NOTE: Technically this should Object.defineProperty at
        //       the next index, as push can be affected by
        //       properties on Object.prototype and Array.prototype.
        //       But that method's new, and collisions should be
        //       rare, so use the more-compatible alternative.
        if (fun.call(thisArg, val, i, t))
          res.push(val);
      }
    }

    return res;
  };
}
```

规范

Specification	Status	Comment
ECMAScript 5.1 (ECMA-262) Array.prototype.filter	ST Standard	Initial definition. Implemented in JavaScript 1.6
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.filter	ST Standard	

浏览器兼容性

Desktop Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	(Yes)	1.5 (1.8)	9	(Yes)	(Yes)

相关链接

- `Array.prototype.forEach()`
- `Array.prototype.every()`
- `Array.prototype.some()`