Array.prototype.filter()

In This Article

The `filter()` method creates a new array with all elements that pass the test implemented by the provided function.

```
var words = ["spray", "limit", "elite", "exuberant", "

var longWords = words.filter(function(word){
  return word.length > 6;
});

// Filtered array longWords is ["exuberant", "destruct
```

ES6 version

```
const words = ["spray", "limit", "elite", "exuberant",

let longWords = words.filter(word => word.length > 6);

// Filtered array longWords is ["exuberant", "destruct
```

Syntax

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

Parameters

`callback`
Function is a predicate, to test each element of the array. Return `true` to keep the element, `false` otherwise, taking three arguments:

`element`
The current element being processed in the array.

`index`
The index of the current element being processed in the array.

`array`
The array `filter` was called upon.

`thisArg` Optional
Optional. Value to use as `this` when executing `callback`.

Return value

A new array with the elements that pass the test.

Description

`filter()` calls a provided `callback` function once for each element in an array, and constructs a new array of all the values for which `callback` returns a value that coerces to `true`. `callback` is invoked only for indexes of the array which have assigned values; it is not invoked for indexes which have been deleted or which have never been assigned values. Array elements which do not pass the `callback` test are simply skipped, and are not included in the new array.

`callback` is invoked with three arguments:

1. the value of the element
2. the index of the element
3. the Array object being traversed

If a `thisArg` parameter is provided to `filter`, it will be used as the callback's `this` value. Otherwise, the value `undefined` will be used as its `this` value. The `this` value ultimately observable by `callback` is determined according to the usual rules for determining the `this` seen by a function.

`filter()` does not mutate the array on which it is called.

The range of elements processed by `filter()` is set before the first invocation of `callback`. Elements which are appended to the array after the call to `filter()` begins will not be visited by `callback`. If existing elements of the

array are changed, or deleted, their value as passed to `callback` will be the value at the time `filter()` visits them; elements that are deleted are not visited.

Examples

Filtering out all small values

The following example uses `filter()` to create a filtered array that has all elements with values less than 10 removed.

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

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

Filtering invalid entries from JSON

The following example uses `filter()` to create a filtered json of all elements with non-zero, numeric `id`.

```
var arr = [
  { id: 15 },
  { id: -1 },
  { id: 0 },
  { id: 3 },
  { id: 12.2 },
  { },
  { id: null },
  { id: NaN },
  { id: 'undefined' }
];

var invalidEntries = 0;

function isNumber(obj) {
  return obj!== undefined && typeof(obj) === 'number'
}

function filterByID(item) {
  if (isNumber(item.id)) {
    return true;
  }
```

```
    invalidEntries++;
    return false;
  }

  var arrByID = arr.filter(filterByID);

  console.log('Filtered Array\n', arrByID);
  // Filtered Array
  // [{ id: 15 }, { id: -1 }, { id: 0 }, { id: 3 }, { id

  console.log('Number of Invalid Entries = ', invalidEnt
  // Number of Invalid Entries = 4
```

Searching in array

Following example uses filter() to filter array content based on search criteria

```
  var fruits = ['apple', 'banana', 'grapes', 'mango', 'o

  /**
   * Array filters items based on search criteria (query
   */
  function filterItems(query) {
    return fruits.filter(function(el) {
        return el.toLowerCase().indexOf(query.toLowerCas
    })
  }

  console.log(filterItems('ap')); // ['apple', 'grapes']
  console.log(filterItems('an')); // ['banana', 'mango',
```

ES2015 Implementation

```
  const fruits = ['apple', 'banana', 'grapes', 'mango',

  /**
   * Array filters items based on search criteria (query
   */
  const filterItems = (query) => {
    return fruits.filter((el) =>
      el.toLowerCase().indexOf(query.toLowerCase()) > -1
    );
  }
```

```
console.log(filterItems('ap')); // ['apple', 'grapes']
console.log(filterItems('an')); // ['banana', 'mango',
```

Polyfill

`filter()` was added to the ECMA-262 standard in the 5th edition; as such it may not be present in all implementations of the standard. You can work around this by inserting the following code at the beginning of your scripts, allowing use of `filter()` in ECMA-262 implementations which do not natively support it. This algorithm is exactly equivalent to the one specified in ECMA-262, 5th edition, assuming that `fn.call` evaluates to the original value of `Function.prototype.bind()`, and that `Array.prototype.push()` has its original value.

```
if (!Array.prototype.filter)
  Array.prototype.filter = function(func, thisArg) {
    'use strict';
    if ( ! ((typeof func === 'Function' || typeof func
        throw new TypeError();

    var len = this.length >>> 0,
        res = new Array(len), // preallocate array
        t = this, c = 0, i = -1;
    if (thisArg === undefined)
      while (++i !== len)
        // checks to see if the key was set
        if (i in this)
          if (func(t[i], i, t))
            res[c++] = t[i];
    else
      while (++i !== len)
        // checks to see if the key was set
        if (i in this)
          if (func.call(thisArg, t[i], i, t))
            res[c++] = t[i];

    res.length = c; // shrink down array to proper siz
    return res;
  };
```

Specifications

Specification	Status	Comment

ECMAScript 5.1 (ECMA-262) The definition of 'Array.prototype.filter' in that specification.	Standard	Initial definition. Implemented in JavaScript 1.6.
ECMAScript 2015 (6th Edition, ECMA-262) The definition of 'Array.prototype.filter' in that specification.	Standard	
ECMAScript Latest Draft (ECMA-262) The definition of 'Array.prototype.filter' in that specification.	Living Standard	

Browser compatibility

Desktop Mobile

Feature	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic Support	(Yes)	(Yes)	1.5	9	(Yes)	(Yes)

See also

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

Was this article helpful?

Tags: Array ECMAScript 5 JavaScript Method polyfill Prototype Reference

Contributors to this page: BeauAngel15, demir-delic, daraclare, fscholz, ufef, Axngm,anonyco, Johann-S, JoshMcCall, erikadoyle, nickmessing, robbiejaeger, getify, kdex,mkutny, SandipNirmal, luc4leone, uxitten, RainbowDange

Last updated by: BeauAngel15, Oct 23, 2017, 2:07:21 PM

See also

Standard built-in objects

`Array`

Properties

`Array.length`

Array.prototype

Array.prototype[@@unscopables]

Methods

Array.from()

Array.isArray()

Array.observe()

Array.of()

Array.prototype.concat()

Array.prototype.copyWithin()

Array.prototype.entries()

Array.prototype.every()

Array.prototype.fill()

Array.prototype.filter()

Array.prototype.find()

Array.prototype.findIndex()

Array.prototype.forEach()

Array.prototype.includes()

Array.prototype.indexOf()

Array.prototype.join()

Array.prototype.keys()

Array.prototype.lastIndexOf()

Array.prototype.map()

Array.prototype.pop()

Array.prototype.push()

Array.prototype.reduce()

Array.prototype.reduceRight()

Array.prototype.reverse()

Array.prototype.shift()

Array.prototype.slice()

Array.prototype.some()

Array.prototype.sort()

Array.prototype.splice()

Array.prototype.toLocaleString()

Array.prototype.toSource()

Array.prototype.toString()

Array.prototype.unshift()

Array.prototype.values()

Array.prototype[@@iterator]()

Array.unobserve()

get Array[@@species]

Inheritance:

Function

- **Properties**
- **Methods**

`Object`

- **Properties**
- **Methods**