Array.prototype.find()

find() 方法返回数组中满足提供的测试函数的第一个元素的值。否则返回 `undefined` 。

```
1 function isBigEnough(element) {
2   return element >= 15;
3 }
4
5 [12, 5, 8, 130, 44].find(isBigEnough); // 130
```

另请参见 `findIndex()` 方法，它返回数组中找到的元素的索引，而不是其值。

如果你需要找到一个元素的位置或者一个元素是否存在于数组中，使用 `Array.prototype.indexOf()` 或 `Array.prototype.includes()` 。

语法

```
arr.find(callback[, thisArg])
```

参数

callback

在数组每一项上执行的函数，接收 3 个参数：

element

当前遍历到的元素。

index

当前遍历到的索引。

array

数组本身。

thisArg 可选

可选，指定 `callback` 的 `this` 参数。

返回值

当某个元素通过 callback 的测试时，返回数组中的一个值，否则返回 `undefined` 。

描述

find 方法对数组中的每一项元素执行一次 `callback` 函数，直至有一个 callback 返回 `true` 。当找到了这样一个元素后，该方法会立即返回这个元素的值，否则返回 `undefined` 。注意 `callback` 函数会为数组中的每个索引调用即从 `0` 到 `lengh - 1` ，而不仅仅是那些被赋值的索引，这意味着对于稀疏数组来说，该方法的效率要低于那些只遍历有值的索引的方法。

callback 函数带有3个参数：当前元素的值、当前元素的索引，以及数组本身。

如果提供了 thisArg 参数，那么它将作为每次 callback 函数执行时的上下文对象，否则上下文对象为 `undefined` 。

find 方法不会改变数组。

在第一次调用 `callback` 函数时会确定元素的索引范围，因此在 `find` 方法开始执行之后添加到数组的新元素将不会被 callback 函数访问到。如果数组中一个尚未被callback函数访问到的元素的值被callback函数所改变，那么当callback函数访问到它时，它的值是将是根据它在数组中的索引所访问到的当前值。被删除的元素仍旧会被访问到。

示例

用对象的属性查找数组里的对象

```
var inventory = [
    {name: 'apples', quantity: 2},
    {name: 'bananas', quantity: 0},
    {name: 'cherries', quantity: 5}
];

function findCherries(fruit) {
    return fruit.name === 'cherries';
}

console.log(inventory.find(findCherries)); // { name: 'cherries', quantit
```

寻找数组中的质数

下面的例子展示了如何从一个数组中寻找质数（如果找不到质数则返回undefined）

```
function isPrime(element, index, array) {
  var start = 2;
  while (start <= Math.sqrt(element)) {
    if (element % start++ < 1) {
      return false;
    }
  }
  return element > 1;
}

console.log([4, 6, 8, 12].find(isPrime)); // undefined, not found
console.log([4, 5, 8, 12].find(isPrime)); // 5
```

当在回调中删除数组中的一个值时，当访问到这个位置时，其传入的值时 undefiend：

```
// Declare array with no element at index 2, 3 and 4
var a = [0,1,,,,5,6];

// Shows all indexes, not just those that have been assigned values
a.find(function(value, index) {
  console.log('Visited index ' + index + ' with value ' + value);
});

// Shows all indexes, including deleted
a.find(function(value, index) {

  // Delete element 5 on first iteration
  if (index == 0) {
    console.log('Deleting a[5] with value ' + a[5]);
    delete a[5];
  }
  // Element 5 is still visited even though deleted
  console.log('Visited index ' + index + ' with value ' + value);
});
```

垫片

本方法在ECMAScript 6规范中被加入，可能不存在于某些实现中。你可以通过以下代码来补充 `Array.prototype` .find。

```
// https://tc39.github.io/ecma262/#sec-array.prototype.find
if (!Array.prototype.find) {
  Object.defineProperty(Array.prototype, 'find', {
    value: function(predicate) {
     // 1. Let O be ? ToObject(this value).
      if (this == null) {
        throw new TypeError('"this" is null or not defined');
      }

      var o = Object(this);

      // 2. Let len be ? ToLength(? Get(O, "length")).
      var len = o.length >>> 0;

      // 3. If IsCallable(predicate) is false, throw a TypeError exceptic
      if (typeof predicate !== 'function') {
        throw new TypeError('predicate must be a function');
      }

      // 4. If thisArg was supplied, let T be thisArg; else let T be unde
      var thisArg = arguments[1];

      // 5. Let k be 0.
      var k = 0;

      // 6. Repeat, while k < len
      while (k < len) {
        // a. Let Pk be ! ToString(k).
        // b. Let kValue be ? Get(O, Pk).
        // c. Let testResult be ToBoolean(? Call(predicate, T, « kValue,
        // d. If testResult is true, return kValue.
        var kValue = o[k];
        if (predicate.call(thisArg, kValue, k, o)) {
          return kValue;
        }
        // e. Increase k by 1.
        k++;
      }

      // 7. Return undefined.
      return undefined;
    }
```

```
43    });
44  }
```

规范

Specification	Status	Comment
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.find	ST Standard	Initial definition.
ECMAScript Latest Draft (ECMA-262) Array.prototype.find	LS Living Standard	

浏览器兼容性

Desktop Mobile

Feature	Chrome	Firefox (Gecko)	Edge	Opera	Safari
Basic support	45.0	25.0 (25.0)	12	未实现	7.1

相关链接

- `Array.prototype.findIndex()` – find and return an index
- `Array.prototype.filter()` – find all matching elements
- `Array.prototype.every()` – test all elements together