Array.prototype.findIndex()

findIndex() 方法返回数组中满足提供的测试函数的第一个元素的索引。否则返回-1。

```
function isBigEnough(element) {
  return element >= 15;
}

[12, 5, 8, 130, 44].findIndex(isBigEnough);
// index of 4th element in the Array is returned,
// so this will result in '3'
```

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

语法

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

参数

callback

针对数组中的每个元素, 都会执行该回调函数, 执行时会自动传入下面三个参数:

element

当前元素。

index

当前元素的索引。

array

调用 `findIndex` 的数组。

thisArg

可选。执行 `callback` 时作为 `this` 对象的值.

描述

`findIndex` 方法对数组中的每个数组索引 `0..length-1` （包括）执行一次 `callback` 函数，直到找到一个 `callback` 函数返回真实值（强制为 `true` ）的值。如果找到这样的元素， `findIndex` 会立即返回该元素的索引。如果回调从不返回真值，或者数组的 `length` 为0，则 `findIndex` 返回-1。 与某些其他数组方法（如Array#some）不同，在稀疏数组中，即使对于数组中不存在的条目的索引也会调用回调函数。

回调函数调用时有三个参数：元素的值，元素的索引，以及被遍历的数组。

如果一个 `thisArg` 参数被提供给 `findIndex` , 它将会被当作 `this` 使用在每次回调函数被调用的时候。如果没有被提供，将会使用 `undefined` 。

`findIndex` 不会修改所调用的数组。

在第一次调用 `callback` 函数时会确定元素的索引范围，因此在 `findIndex` 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。被删除的元素不会被访问到。

示例

查找数组中首个质数元素的索引

以下示例查找数组中素数的元素的索引（如果不存在素数，则返回-1）。

```
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].findIndex(isPrime)); // -1, not found
console.log([4, 6, 7, 12].findIndex(isPrime)); // 2
```

Polyfill

```
// https://tc39.github.io/ecma262/#sec-array.prototype.findIndex
if (!Array.prototype.findIndex) {
```

```
  Object.defineProperty(Array.prototype, 'findIndex', {
    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 k.
        var kValue = o[k];
        if (predicate.call(thisArg, kValue, k, o)) {
          return k;
        }
        // e. Increase k by 1.
        k++;
      }

      // 7. Return -1.
      return -1;
    }
  });
}
```

如果您需要兼容不支持 `Object.defineProperty` 的JavaScript引擎，那么最好不要对 `Array.prototype` 方法进行 polyfill ，因为您无法使其成为不可枚举的。

规范

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

浏览器兼容

Desktop Mobile

Feature	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic Support	45	(Yes)	25	No	32	7.1

相关链接

- `Array.prototype.find()`
- `Array.prototype.indexOf()`