Array.prototype.find()

The `find()` method returns the **value** of the first element in the array that satisfies the provided testing function. Otherwise `undefined` is returned.

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

[12, 5, 8, 130, 44].find(isBigEnough); // 130
```

See also the `findIndex()` method, which returns the **index** of a found element in the array instead of its value.

If you need to find the position of an element or whether an element exists in an array, use `Array.prototype.indexOf()` or `Array.prototype.includes()`.

Syntax

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

Parameters

`callback`
Function to execute on each value in the array, 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 `find` was called upon.

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

Return value

A value in the array if an element passes the test; otherwise, `undefined`.

Description

The `find` method executes the `callback` function once for each index of the array until it finds one where `callback` returns a true value. If such an element is found, `find` immediately returns the value of that element. Otherwise, `find` returns `undefined`. `callback` is invoked for every index of the array from `0` to `length - 1` and is invoked for all indexes, not just those that have been assigned values. This may mean that it's less efficient for sparse arrays than other methods that only visit indexes that have been assigned a value.

`callback` is invoked with three arguments: the value of the element, the index of the element, and the Array object being traversed.

If a `thisArg` parameter is provided to `find`, it will be used as the `this` for each invocation of the `callback`. If it is not provided, then `undefined` is used.

`find` does not mutate the array on which it is called.

The range of elements processed by `find` is set before the first invocation of `callback`. Elements that are appended to the array after the call to `find` begins will not be visited by `callback`. If an existing, unvisited element of the array is changed by `callback`, its value passed to the visiting `callback` will be the value at the time that `find` visits that element's index; elements that are deleted are still visited.

Examples

Find an object in an array by one of its properties

```
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', quantity: 5 }
```

```
// { name: 'cherries', quantity: 5 }
```

Find a prime number in an array

The following example finds an element in the array that is a prime number (or returns `undefined` if there is no prime number).

```
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
```

The following examples show that non-existent and deleted elements are visited and that the value passed to the callback is their value when visited.

```
// 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);
});
```

Polyfill

This method has been added to the ECMAScript 2015 specification and may not be available in all JavaScript implementations yet. However, you can polyfill `Array.prototype.find` with the following snippet:

```
// 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 exception
      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 undef
      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, k
        // 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;
    }
  });
}
```

If you need to support truly obsolete JavaScript engines that don't support `Object.defineProperty`, it's best not to polyfill `Array.prototype` methods at all, as you can't make them non-enumerable.

Specifications

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

Browser compatibility

Desktop Mobile

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

See also

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