

# 数组

## 数组的本质

数组是一种特殊的对象，它的键名是固定的（默认是 0,1, 2...）。

JavaScript 规定对象的键名一律是字符串，所以数组的键名也是字符串。之所以可以用数值读取，是因为非字符串的键名会被转为字符串。

```
var arr = ["a", "b", "c"];

arr["0"]; // 'a'
arr[0]; // 'a'

let a3 = ["a", "b"];
for (let key in a3) {
  console.log(key, typeof key);
}
// 0 string
// 1 string
```

这点在赋值时也成立。

```
var a = [];

a[1.0] = 6;
a[1]; // 6
```

由于 1.00 转成字符串是 1，所以通过数字键 1 可以读取值。

数组成员只能用方括号`arr[0]`读取其成员。

```
var arr = [1, 2, 3];
arr.0 // Uncaught SyntaxError: Unexpected number
```

## 数组的 length 属性

### 1. 非连续数字键

数组的数字键不需要连续，`length`属性的值总是比最大的那个整数键大1。

```
var arr = ["a", "b"];
arr.length; // 2
```

```
arr[2] = "c";  
arr.length; // 3  
  
arr[9] = "d";  
arr.length; // 10  
  
arr[1000] = "e";
```

## 2. length 值范围

JavaScript 使用一个 32 位整数，保存数组的元素个数。数组的 `length` 属性的最大值是  $(2^{32} - 1)$ 。

```
[].length = Math.pow(2, 32)  
// Uncaught RangeError: Invalid array length  
  
[].length = Math.pow(2, 32) - 1  
// 4294967295
```

## 3. 非自然数键名不影响数组 length 属性

由于数组本质上是一种对象，所以可以为数组添加属性，但是这不影响 `length` 属性的值。

```
var a = [];  
  
a["p"] = "abc";  
a.length; // 0  
  
a[2.1] = "abc";  
a.length; // 0
```

`length` 属性的值就是等于最大的数字键加 1，而这个数组没有整数键，所以 `length` 属性保持为 0。

数组的键名是添加超出范围的数值，该键名会自动转为字符串。

```
var arr = [];  
arr[-1] = "a";  
arr[Math.pow(2, 32)] = "b";  
  
arr.length; // 0  
arr[-1]; // "a"  
arr[4294967296]; // "b"
```

最后两行之所以会取到值，是因为取键值时，数字键名会默认转为字符串。

## in 运算符

`in` 运算符适用于对象中，也适用于数组中。

```
var arr = ["a", "b", "c"];
2 in arr; // true
"2" in arr; // true
4 in arr; // false
```

如果数组的某个位置是空位，`in` 运算符返回 `false`。

```
var arr = [];
arr[100] = "a";
arr; // (101) [empty × 100, "a"]

100 in arr; // true
1 in arr; // false
```

## for...in 循环和数组的遍历

`for...in` 不仅会遍历数组所有的数字键，还会遍历非数字键。

```
var a = [1, 2, 3];
a.foo = true;

for (var key in a) {
  console.log(key);
}
// 0
// 1
// 2
// foo
```

在遍历数组时，也遍历到了非整数键 `foo`。所以，不推荐使用 `for...in` 遍历数组。

## 数组的空位

当数组的某个位置是空元素，即两个逗号之间没有任何值，我们就称这个数组存在空位。

```
var arr = [1, , 2];
arr.length; // 3
arr; // (3) [1, empty, 1];
arr[1]; // undefined
```

最后一个元素后面有没有逗号，结果是一样的。

```
var a1 = [1, 2];
a1; // (2) [1, 2]
```

当使用 `delete` 命令删除一个数组的成员后，这个位置也会成为空位，但是不影响 `length` 的属性的值。

```
var a66 = [1, 2, 3, 4];

delete a66[3]; // true

a66; // (4) [1, 2, 3, empty]

a66.length; // 4
```

**length属性不过滤空位。**

数组空位与数组成员 `undefined` 的值的区别

如果数组某个位置是空位和数组在该位置的成员的值是 `undefined` 是不一样的。如果是空位，使用数组的 `forEach` 方法、`for...in` 结构、以及 `Object.keys()` 方法进行遍历，空位都会跳过。

```
var a67 = [1, 2, , 4];

a67.forEach(function (x, i) {
  console.log(i + ", " + x);
});

// 0, 1
// 1, 2
// 3, 4

for (var key in a67) {
  console.log(key + ", " + a67[key]);
}

// 0, 1
// 1, 2
// 3, 4

Object.keys(a67);

// (3) [0, 1, 3]
```

当这个数组中第 2 个位置是空位时，第 2 个成员就被跳过了，不输出。

而如果成员值是 `undefined` 时，则不会跳过。

```

var a67 = [1, 2, , 4];

a67.forEach(function (x, i) {
    console.log(i + ", " + x);
});

// 0, 1
// 1, 2
// 2, undefined
// 3, 4

for (var key in a67) {
    console.log(key + ", " + a67[key]);
}

// 0, 1
// 1, 2
// 2, undefined
// 3, 4

Object.keys(a67);

// (3) [0, 1, 3]

```

## 类似数组的对象

如果一个对象的所有键名都是正整数或零，并且具有 `length` 属性，那这个对象就很像数组，成为“类似数组的对象”（array-like-object）。

```

var o = {
    0: "a",
    1: "b",
    length: 2,
};

o.length; // 2
o[0]; // "a"
o.pop(); // Uncaught TypeError: o44.pop is not a function

```

o 是对象，含有 `length` 属性，且其它键名是 0 或正整数，类似数组，是类似数组的对象。但不是真正的数组，所以没有 `pop()` 方法。

类似数组的对象包括：

- DOM 元素集
- 字符串
- arguments 对象
- DOM 对象

```
var elems = document.getElementsByTagName("div");
elems.length; // 45
elems instanceof Array; // false
```

- 字符串

```
var s = "a";
s.length; // 1
s instanceof Array; // false
```

- arguments 对象

```
var f = function () {
    return arguments;
};
var arrayLike = f("a", "b");
arrayLike.length; // 2
arrayLike instanceof Array; // false
```

数组的 `slice` 方法可以将“类似数组的对象”变成真正的数组。

```
var arr = Array.prototype.slice.call(arrayLike);
arr; // (2) ["a", "b"]

var arr = Array.prototype.slice.call(s);
arr; // ["a"]

var arr = Array.prototype.slice.call(elems);
arr instanceof Array; // true
```