Array.prototype.fill()

fill() 方法用一个固定值填充一个数组中从起始索引到终止索引内的全部元素。

```
var numbers = [1, 2, 3]
numbers.fill(1);

// results in [1, 1, 1]
```

语法

```
arr.fill(value)
arr.fill(value, start)
arr.fill(value, start, end)
```

参数

value
用来填充数组元素的值。

start 可选
起始索引，默认值为0。

end 可选
终止索引，默认值为 `this.length` 。

返回值

修改后的数组。

描述

具体要填充的元素区间是 [`start` , `end`), 一个半开半闭区间.

`fill` 方法接受三个参数 `value` , `start` 以及 `end` . `start` 和 `end` 参数是可选的, 其默认值分别为 `0` 和 `this` 对象的 `length` 属性值 .

如果 `start` 是个负数, 则开始索引会被自动计算成为 `length+start`, 其中 `length` 是 `this` 对象的 `length` 属性值 . 如果 `end` 是个负数, 则结束索引会被自动计算成为 `length+end` .

fill 方法故意被设计成通用方法, ~~也就是说它不需要~~ ~~`this`~~ ~~值必须是个数组对象, 类数组对象也是可以调用该方法的.~~它需要this值是个对象，类数组对象调用会报错

fill 方法是个可变方法, 它会改变调用它的 `this` 对象本身, 然后返回它, 而并不是返回一个副本.

示例

```
[1, 2, 3].fill(4)               // [4, 4, 4]
[1, 2, 3].fill(4, 1)            // [1, 4, 4]
[1, 2, 3].fill(4, 1, 2)         // [1, 4, 3]
[1, 2, 3].fill(4, 1, 1)         // [1, 2, 3]
[1, 2, 3].fill(4, -3, -2)       // [4, 2, 3]
[1, 2, 3].fill(4, NaN, NaN)     // [1, 2, 3]
Array(3).fill(4);               // [4, 4, 4]
[].fill.call({length: 3}, 4) // {0: 4, 1: 4, 2: 4, length: 3}
```

Polyfill

```
if (!Array.prototype.fill) {
  Object.defineProperty(Array.prototype, 'fill', {
    value: function(value) {

      // Steps 1-2.
      if (this == null) {
        throw new TypeError('this is null or not defined');
      }

      var O = Object(this);

      // Steps 3-5.
      var len = O.length >>> 0;

      // Steps 6-7.
      var start = arguments[1];
      var relativeStart = start >> 0;

      // Step 8.
      var k = relativeStart < 0 ?
```

```
        Math.max(len + relativeStart, 0) :
        Math.min(relativeStart, len);

      // Steps 9-10.
      var end = arguments[2];
      var relativeEnd = end === undefined ?
        len : end >> 0;

      // Step 11.
      var final = relativeEnd < 0 ?
        Math.max(len + relativeEnd, 0) :
        Math.min(relativeEnd, len);

      // Step 12.
      while (k < final) {
        O[k] = value;
        k++;
      }

      // Step 13.
      return O;
    }
  });
}
```

如果你确实需要维护已过时的不支持 `Object.defineProperty` 的 JavaScript 引擎，那么最好完全不向 `Array.prototype` 添加方法，因为你不能使它不可枚举。

规范

规范	状态	注释
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.fill	ST Standard	最初定义。
ECMAScript Latest Draft (ECMA-262) Array.prototype.fill	LS Living Standard	

浏览器兼容性

Desktop Mobile

特性	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
基础支持	45 [1]	31 (31)	未实现	未实现	未实现

相关

- `Array`
- `TypedArray.prototype.fill()`