# JavaScript 基本知识归纳

## Array

var arr = [1,2,3,4,5];

1. Method

**Slice** 返回数组中部分元素的影像复制数组，不会改变原数组

例如：console.log(arr.slice(2,5)); // [3,4,5]

**Splice** 为数组新增或删除元素，改变原始数组的内容

例如：arr.splice(2,0,6) //[1,2,6,3,4,5 ]

arr.splice(2,2) //[1,2,5]

**Shift** 移除数组的首个元素，并返回被移除的元素，会改变数组长度。

例如：firstElement = arr.shift(); // arr=[2,3,4,5], firstElement = 1

**Unshift** 在数组首部增加一个或多个元素，返回新数组长度

例如：arr.unshift(7,8) // [7,8,1,2,3,4,5]

**Pop** 移除数组的最后一个元素，返回被移除的元素，会改变数组长度。

**Push** 在数组尾部增加一个或多个元素，返回新数组长度

**From** 创建类数组或可迭代对象的一个新的数组实例。(ES6)

例如：Array.from('foo') //[‘f’,’o’,’o’]

Array.from([1, 2, 3], x => x + x) //[2,4,6]

**Array.isArray()** 判断是否为数组 (ES6)

**Array.of()**  根据传入参数创建新数组实例。(ES6)

例如：Array.of(7); // [7]

Array.of(1, 2, 3); // [1, 2, 3]

区别于 Array 构造函数

Array(7); // [ , , , , , , ]

Array(1, 2, 3); // [1, 2, 3]

**Concat** 合并一个或多个数组，不改变原数组，返回合并后的新数组

例如：

例如：var alpha = ['a', 'b', 'c'];

var numeric = [1, 2, 3];

alpha.concat(numeric);

// result in ['a', 'b', 'c', 1, 2, 3]

**copyWithin** 复制数组中的部分位置元素至同一数组，并返回该数组

例如：

[1, 2, 3, 4, 5].copyWithin(0, 3);

// [4, 5, 3, 4, 5]

**Entries**  返回包含数组中每个元素的索引值Key/value 组成的数组迭代器。

var array1 = ['a', 'b', 'c'];

var iterator1 = array1.entries();

console.log(iterator1.next().value); //[0, "a"]

var a = ['a', 'b', 'c'];

var iterator = a.entries();

for (let e of iterator) {

console.log(e);

}

// [0, 'a']

// [1, 'b']

// [2, 'c']

**Keys** 返回包含数组每个元素索引值的数组迭代器(ES6)

例如：

var array1 = ['a', 'b', 'c'];

var iterator = array1.keys();

for (let key of iterator) {

console.log(key); // expected output: 0 1 2

}

**Values** 返回包含数组每个元素值的数组迭代器(ES6)

例如：

var arr = ['w', 'y', 'k', 'o', 'p'];

var iterator = arr.values();

for (let letter of iterator) {

console.log(letter);

}

**Every** 测试数组中的每个元素是否能通过传入的测试函数

例如：

[12, 5, 8, 130, 44].every(x => x >= 10); // false

[12, 54, 18, 130, 44].every(x => x >= 10); // true

**Some** 测试数组中的是否至少有一个元素能通过传入的测试函数

例如：

[2, 5, 8, 1, 4].some(x => x > 10); // false

[12, 5, 8, 1, 4].some(x => x > 10); // true

**Fill** 使用特定的元素填充数组中从起始至结束的位置(ES6)

[1, 2, 3].fill(4); // [4, 4, 4]

[1, 2, 3].fill(4, 1); // [1, 4, 4]

**Map** 对数组中所有的元素执行指定的函数，形成新的数组并返回

例如：

var numbers = [1, 4, 9];

var roots = numbers.map(Math.sqrt);

// roots is now [1, 2, 3]

// numbers is still [1, 4, 9]

**Reduce** 使用初始值和一个累计函数将数组中所有元素用累计函数作用在一个值上，并返回

例如：

arr.reduce(callback[, initialValue])

数组去重：

let arr = [1, 2, 1, 2, 3, 5, 4, 5, 3, 4, 4, 4, 4];

let result = arr.sort().reduce((accumulator, current) => {

const length = accumulator.length

if (length === 0 || accumulator[length - 1] !== current) {

accumulator.push(current);

}

return accumulator;

}, []);

console.log(result); //[1,2,3,4,5]

**reduceRight** 使用初始值和一个累计函数（从右向左）将数组中所有用累计函数作用在一个值上，并返回该值

var flattened = [[0, 1], [2, 3], [4, 5]].reduceRight(function(a, b) {

return a.concat(b);

}, []);

// flattened is [4, 5, 2, 3, 0, 1]

**Reverse** 颠倒数组内容

**Sort** 排序数组

**Find** 返回满足匹配函数的第一个元素，找不到返回 Undefined (ES6)

例如：

var array1 = [5, 12, 8, 130, 44];

var found = array1.find(function(element) {

return element > 10;

});

console.log(found);

// expected output: 12

**findIndex** 返回满足匹配函数的第一个元素的索引，找不到返回 -1 (ES6)

例如：

var array1 = [5, 12, 8, 130, 44];

function findFirstLargeNumber(element) {

return element > 13;

}

console.log(array1.findIndex(findFirstLargeNumber));

// expected output: 3

**Filter** 返回满足匹配函数的元素组成的新数组

var words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

const result = words.filter(word => word.length > 6);

console.log(result);

// expected output: Array ["exuberant", "destruction", "present"]

decodeURI() 解码encodeURI 编码的URI

decodeURIComponent() 解码encodeURIComponent编码的URI

encodeURI() 编码URI，

encodeURIComponent() 编码URI，编码更彻底，保留字符少

escape() 使用十六进制编码符替换特定的字符

unescape()

var set1 = ";,/?:@&=+$#"; // Reserved Characters

var set2 = "-\_.!~\*'()"; // Unreserved Marks

var set3 = "ABC abc 123"; // Alphanumeric Characters + Space

console.log(encodeURI(set1)); // ;,/?:@&=+$#

console.log(encodeURI(set2)); // -\_.!~\*'()

console.log(encodeURI(set3)); // ABC%20abc%20123 (the space gets encoded as %20)

console.log(encodeURIComponent(set1)); // %3B%2C%2F%3F%3A%40%26%3D%2B%24%23

console.log(encodeURIComponent(set2)); // -\_.!~\*'()

console.log(encodeURIComponent(set3)); // ABC%20abc%20123 (the space gets encoded as %20)

escape('abc123'); // "abc123"

escape('äöü'); // "%E4%F6%FC"

escape('ć'); // "%u0107"

// special characters

escape('@\*\_+-./'); // "@\*\_+-./"

## String

**charAt** 返回字符串中特定位置的字符组成的字符串（UTF-16编码的字符值）

例如：

var anyString = 'Brave new world';

console.log(anyString.charAt(2)) // a

**charCodeAt** 返回字符串中指定位置的UTF-16编码字符的0-65535之间的整数

例如：

'ABC'.charCodeAt(0); // returns 65

**charPointAt** 返回字符串中指定位置的Unicode码值

例如：

'ABC'.codePointAt(1); // 66

'\uD800\uDC00'.codePointAt(0); // 65536

'XYZ'.codePointAt(42); // undefined

**concat** 将参数字符串拼接在方法调用字符串上，并返回新字符串

例如：

var hello = 'Hello ';,

console.log(hello.concat('Kevin', '. Have a nice day.'));

/\* Hello, Kevin. Have a nice day. \*/

var greetList = ['Hello', ' ', 'Venkat', '!'];

"".concat(...greetList); // "Hello Venkat!"

"".concat({}); // [object Object]

"".concat([]); /// ""

"".concat(null); // "null"

"".concat(true); // "true"

"".concat(4, 5); // "45"

**endsWith** 判断字符串是否以某个子字符串结尾(ES6)

例如：

var str = 'To be, or not to be, that is the question.';

console.log(str.endsWith('question.')); // true

console.log(str.endsWith('to be')); // false

console.log(str.endsWith('to be', 19)); // true

**includes** 判断一个字符串是否包含另一个字符串，返回布尔值（ES6）

var str = 'To be, or not to be, that is the question.';

console.log(str.includes('To be')); // true

console.log(str.includes('question')); // true

console.log(str.includes('nonexistent')); // false

console.log(str.includes('To be', 1)); // false

console.log(str.includes('TO BE')); // false

padEnd

padStart

repeat

replace

search

slice

split

startWith

sub

substr

substring

toSource

toString

trim

trimStart

trimEnd

raw

DOM

DOM2

DOM3

事件

## Object

Object.assign()

Object.create()

Object.defineProperties()

Object.defineProperty()

Object.entries()

Object.keys()

Object.values()

Object.fromEntries()

Object.freeze()

Object.getOwnPropertyDescriptor()

Object.getOwnPropertyDescriptors()

Object.getOwnPropertyNames()

Object.getOwnPropertySymbols()

Object.getPrototypeOf()

Object.setPrototypeOf()

Object.is()

Object.isExtensible()

Object.isFrozen()

Object.isSealed()

Object.prototype.hasOwnProperty()

Object.prototype.isPrototypeOf()