第一节、创建数组、数组操作

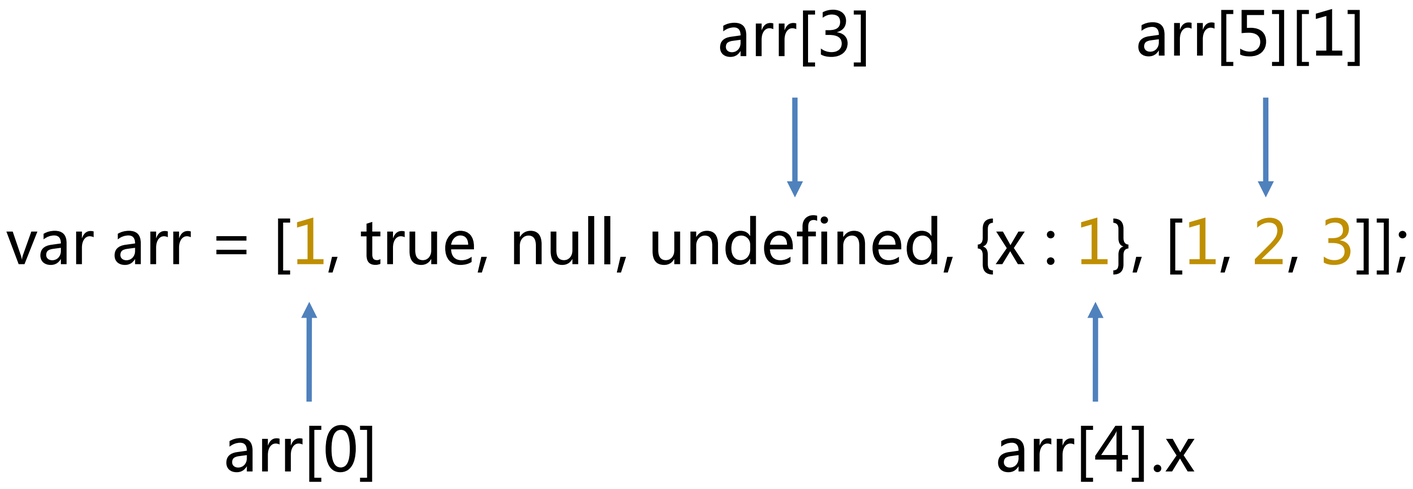
JavaScript深入浅出

数组

@Bosn

数组概述

数组是值的有序集合。每个值叫做元素，每个元素在数组中都有数字位置编号，也就是索引。JS中的数组是弱类型的，数组中可以含有不同类型的元素。数组元素甚至可以是对象或其它数组。



创建数组-字面量

```
var BAT = ['Alibaba', 'Tencent', 'Baidu'];
var students = [{name : 'Bosn', age : 27}, {name : 'Nunnly', age : 3}];
var arr = ['Nunnly', 'is', 'big', 'keng', 'B', 123, true, null];
var arrInArr = [[1, 2], [3, 4, 5]];

var commasArr1 = [1, , 2]; // 1, undefined, 2
var commasArr2 = [,,]; // undefined * 2
```

size from 0 to 4,294,967,295(2^23 -1)

```
> new Array(4294967295)
<· ▶ Array[4294967295]
> new Array(4294967296)
⊗   ▶ Uncaught ▶ RangeError: Invalid array length
```

创建数组-new Array

```
var arr = new Array();
var arrWithLength = new Array(100); // undefined * 100
var arrLikesLiteral = new Array(true, false, null, 1, 2, "hi");
// 等价于[true, false, null, 1, 2, "hi"];
```

数组元素读写

```
var arr = [1, 2, 3, 4, 5];
arr[1]; // 2
arr.length; // 5


arr[5] = 6;
arr.length; // 6


delete arr[0];
arr[0]; // undefined
```

数组元素增删

动态的，无需指定大小

```
var arr = [];
arr[0] = 1;
arr[1] = 2;
arr.push(3);
arr; // [1, 2, 3]

arr[arr.length] = 4; // equal to arr.push(4);
arr; // [1, 2, 3, 4]

arr.unshift(0);
arr; // [0, 1, 2, 3, 4];
```

```
delete arr[2];
arr; // [0, 1, undefined, 3, 4]
arr.length; // 5
2 in arr; // false

arr.length -= 1;
arr; // [0, 1, undefined, 3, 4],  4 is removed

arr.pop(); // 3 returned by pop
arr; // [0, 1, undefined], 3 is removed

arr.shift(); // 0 returned by shift
arr; // [1, undefined]
```

数组迭代

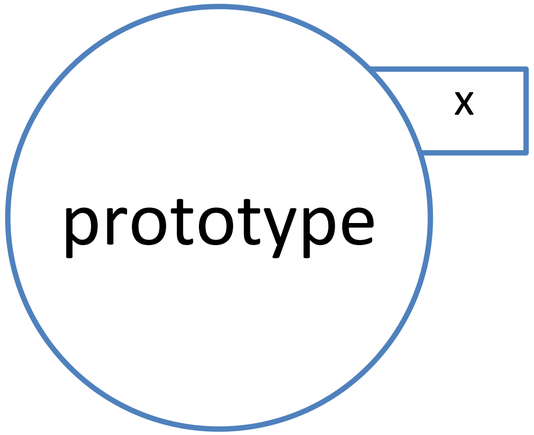
```
var i = 0, n = 10;
var arr = [1, 2, 3, 4, 5];
for (; i < n; i++) {
    console.log(arr[i]); // 1, 2, 3, 4, 5
}

for(i in arr) {
    console.log(arr[i]); // 1, 2, 3, 4, 5
}
```

```
Array.prototype.x = 'inherited';

for(i in arr) {
    console.log(arr[i]); // 1, 2, 3, 4, 5, inherited
}

for(i in arr) {
    if (arr.hasOwnProperty(i)) {
        console.log(arr[i]); // 1, 2, 3, 4, 5
    }
}
```



第二节、二维数组、稀疏数组

二维数组


```
var arr = [[0, 1], [2, 3], [4, 5]];                // result:
var i = 0, j = 0;                                  // row 0
var row;                                           // 0
for (; i < arr.length; i++) {                      // 1
    row = arr[i];                                  // row 1
    console.log('row ' + i);                       // 2
    for (j = 0; j < row.length; j++) {             // 3
        console.log(row[j]);                       // row 2
    }                                              // 4
}                                                  // 5
```

稀疏数组

稀疏数组并不含有从0开始的连续索引。一般length属性值比实际元素个数大。

```
var arr1 = [undefined];
var arr2 = new Array(1);
0 in arr1; // true
0 in arr2; // false
arr1.length = 100;
arr1[99] = 123;
99 in arr1; // true
98 in arr1; // false
```

```
var arr = [,,];
0 in arr; // false
```

第三节、数组方法

数组方法

{} => Object.prototype
[] => Array.prototype

Array.prototype.join
Array.prototype.reverse
Array.prototype.sort
Array.prototype.concat
Array.prototype.slice 切片
Array.prototype.splice 胶接
Array.prototype.forEach (ES5)
Array.prototype.map (ES5)
Array.prototype.filter (ES5)
Array.prototype.every (ES5)
Array.prototype.some (ES5)
Array.prototype.reduce/reduceRight (ES5)
Array.prototype.indexOf/lastIndexOf （ES5）
Array.isArray (ES5)

Array.prototype.join

将数组转为字符串

```
var arr = [1, 2, 3];
arr.join(); // "1,2,3"
arr.join("_"); // "1_2_3"

function repeatString(str, n) {
    return new Array(n + 1).join(str);
}
repeatString("a", 3); // "aaa"
repeatString("Hi", 5); // "HiHiHiHiHi"
```

Array.prototype.reverse

将数组逆序

```
var arr = [1, 2, 3];
arr.reverse(); // [3, 2, 1]
arr; // [3, 2, 1]
```

原数组被修改

Array.prototype.sort

排序

```
var arr = ["a", "d", "c", "b"];
arr.sort(); // ["a", "b", "c", "d"]

arr = [13, 24, 51, 3];
arr.sort(); // [13, 24, 3, 51]
arr; // [13, 24, 3, 51]

arr.sort(function(a, b) {
    return a - b;
}); // [3, 13, 24, 51]
```

原数组被修改

```
arr = [{age : 25}, {age : 39}, {age : 99}];
arr.sort(function(a, b) {
    return a.age - b.age;
});
arr.forEach(function(item) {
    console.log('age', item.age);
});
// result:
// age 25
// age 39
// age 99
```

Array.prototype.concat

数组合并

```
var arr = [1, 2, 3];
arr.concat(4, 5); // [1, 2, 3, 4, 5]
arr; // [1, 2, 3]
```

原数组未被修改

```
arr.concat([10, 11], 13); // [1, 2, 3, 10, 11, 13]

arr.concat([1, [2, 3]]); // [1, 2, 3, 1, [2, 3]]
```

Array.prototype.slice

返回部分数组

```
var arr = [1, 2, 3, 4, 5];
arr.slice(1, 3); // [2, 3]
arr.slice(1); // [2, 3, 4, 5]
arr.slice(1, -1); // [2, 3, 4]
arr.slice(-4, -3); // [2]
```

原数组未被修改

Array.prototype.splice

数组拼接

```
var arr = [1, 2, 3, 4, 5];
arr.splice(2); // returns [3, 4, 5]
arr; // [1, 2];
```

原数组被修改

```
arr = [1, 2, 3, 4, 5];
arr.splice(2, 2); // returns [3, 4]
arr; // [1, 2, 5];

arr = [1, 2, 3, 4, 5];
arr.splice(1, 1, 'a', 'b'); // returns [2]
arr; // [1, "a", "b", 3, 4, 5]
```

```
var arr = [1, 2, 3, 4, 5];
arr.forEach(function(x, index, a){
    console.log(x + '|' + index + '|' + (a === arr));
});
// 1|0|true
// 2|1|true
// 3|2|true
// 4|3|true
// 5|4|true
```

Array.prototype.map

数组映射

```
var arr = [1, 2, 3];
arr.map(function(x) {
    return x + 10;
}); // [11, 12, 13]
arr; // [1, 2, 3]
```

原数组未被修改

Array.prototype.filter

数组过滤

```
var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
arr.filter(function(x, index) {
    return index % 3 === 0 || x >= 8;
}); // returns [1, 4, 7, 8, 9, 10]
arr; // [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

原数组未被修改

Array.prototype.every & some

数组判断

```
var arr = [1, 2, 3, 4, 5];
arr.every(function(x) {
    return x < 10;
}); // true

arr.every(function(x) {
    return x < 3;
}); // false
```

```
var arr = [1, 2, 3, 4, 5];
arr.some(function(x) {
    return x === 3;
}); // true

arr.some(function(x) {
    return x === 100;
}); // false
```

Array.prototype.reduce&reduceRight

```
var arr = [1, 2, 3];
var sum = arr.reduce(function(x, y) {
    return x + y
}, 0); // 6
arr; //[1, 2, 3]
```

原数组未被修改

```
arr = [3, 9, 6];
var max = arr.reduce(function(x, y) {
    console.log(x + "|" + y);
    return x > y ? x : y;
});
// 3|9
// 9|6
max; // 9
```

```
max = arr.reduceRight(function(x, y) {
    console.log(x + "|" + y);
    return x > y ? x : y;
});
// 6|9
// 9|3
max; // 9
```

Array.prototype.indexOf&lastIndexOf

数组检索

```
var arr = [1, 2, 3, 2, 1];
arr.indexOf(2); // 1
arr.indexOf(99); // -1
arr.indexOf(1, 1); // 4
arr.indexOf(1, -3); // 4
arr.indexOf(2, -1); // -1
arr.lastIndexOf(2); // 3
arr.lastIndexOf(2, -2); // 3
arr.lastIndexOf(2, -3); // 1
```

Array.isArray

判断是否为数组

```
Array.isArray([]); // true
```

```
[] instanceof Array; // true
({}).toString.apply([]) === '[object Array]'; // true
[].constructor === Array; // true
```

第四节、数组小结

数组 VS. 一般对象

相同

都可以继承
数组是对象，对象不一定是数组
都可以当做对象添加删除属性

不同

数组自动更新length
按索引访问数组常常比访问一般对象属性明显迅速。
数组对象继承Array.prototype上的大量数组操作方法

字符串和数组

```
var str = "hello world";
str.charAt(0); // "h"
str[1]; // e

Array.prototype.join.call(str, "_");
// "h_e_l_l_o_ _w_o_r_l_d"
```

小结

数组概念

创建数组、数组增删改查操作

二维数组、稀疏数组

数组方法

数组 VS. 一般对象

数组 VS. 字符串

谢谢