# JAVASCRIPT笔记

## 数据类型

### Number

JavaScript不区分整数和浮点数，统一用Number表示，以下都是合法的Number类型：

123; *// 整数123*

0.456; *// 浮点数0.456*

1.2345e3; *// 科学计数法表示1.2345x1000，等同于1234.5*

-99; *// 负数*

NaN; *// NaN表示Not a Number，当无法计算结果时用NaN表示*

Infinity; *// Infinity表示无限大，当数值超过了JavaScript的Number所能表示的最大值时，就表示为Infinity*

### 字符串

ES6新增多行字符串的表示方法 ` 使用反引号标识多行字符串，反引号位于数字1的左面 `

ES6新增模板字符串: var name=’java’; alert(‘你好，我叫${name}’);

字符串也可以通过索引访问字符，类似数组的检索方法

### 布尔值

**false** == 0; // **true**

**false** === 0; // **false**

**NaN与任何值都不相等，包括它自己**

NaN === NaN; *// false*

*判断NaN的唯一函数方法是 isNaN(NaN) //true*

### null和undefined

null表示一个“空”的值，它和0以及空字符串''不同，0是一个数值，''表示长度为0的字符串，而null表示“空”。

在其他语言中，也有类似JavaScript的null的表示，例如Java也用null，Swift用nil，Python用None表示。但是，在JavaScript中，还有一个和null类似的undefined，它表示“未定义”。

JavaScript的设计者希望用null表示一个空的值，而undefined表示值未定义。事实证明，这并没有什么卵用，区分两者的意义不大。大多数情况下，我们都应该用null。undefined仅仅在判断函数参数是否传递的情况下有用。

### 数组

Javascript数组可以包含任意类型的数值

[1, 2, 3.14, 'Hello', null, true];

**数组特性**：对lenth属性直接赋值会改变数组大小

索引越界不会提示异常，会改变数组大小

以上两点会改变数组大小，undefined会对空位填充

**数组的函数：**

Slice() 函数相当于字符串的substring() ,截取部分数组，范围前闭后包

Push()函数 在数组末尾增加元素 pop() 函数 将数组的末尾元素弹出，并返回该元素

Unshift() 在数组头部添加元素 shift() 弹出数组第一个元素

Sort() 对元素进行排序，直接在原数组中进行，按照默认顺序排序

Reverse() 将数组中元素进行翻转

Splice() 可以从指定索引开始删除若干元素，然后再添加若干元素

**var** arr = ['Microsoft', 'Apple', 'Yahoo', 'AOL', 'Excite', 'Oracle'];

*// 从索引2开始删除3个元素,然后再添加两个元素:*

arr.splice(2, 3, 'Google', 'Facebook'); *// 返回删除的元素 ['Yahoo', 'AOL', 'Excite']*

arr; *// ['Microsoft', 'Apple', 'Google', 'Facebook', 'Oracle']*

*// 只删除,不添加:*

arr.splice(2, 2); *// ['Google', 'Facebook']*

arr; *// ['Microsoft', 'Apple', 'Oracle']*

*// 只添加,不删除:*

arr.splice(2, 0, 'Google', 'Facebook'); *// 返回[],因为没有删除任何元素*

arr; *// ['Microsoft', 'Apple', 'Google', 'Facebook', 'Oracle']*

concat() 数组连接函数 将若干个数组连接起来返回新的数组

join() 将数组元素连接起来，参数接受连接符号，如果元素不是字符串，将自动转换为字符串后再连接

**\*使用字符串与数组连接会直接将数组内容转换为字符串显示**

var arr = ['小明', '小红', '大军', '阿黄'];

console.log('欢迎'+arr+'同学！');

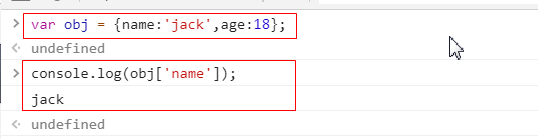
欢迎小明,小红,大军,阿黄同学！

### 对象

无序集合数据类型

键为字符串类型，值为任意类型

访问键的值：



对象属性中的键如果包含特殊字符（如“-”）,需要用‘’括起来：

**var** xiaohong = {

name: '小红',

'middle-school': 'No.1 Middle School'

};

访问属性不能用.来访问，使用[‘xxx’]

xiaohong['middle-school']; // 'No.1 Middle School'

xiaohong['name']; // '小红'

xiaohong.name; // '小红'

访问对象不存在的属性会返回 undefined

对象属性支持动态添加，obj.name=’xxx’ ,如果obj中原来没有name属性，那么这个语句会为obj添加属性name,并且值为xxx

使用in 检测对象中是否包含该属性

‘name’in obj //true

不过要小心，如果in判断一个属性存在，这个属性不一定是xiaoming的，它可能是xiaoming继承得到的：

'toString' **in** xiaoming; // **true**

因为toString定义在object对象中，而**所有对象最终都会在原型链上指向object**，所以xiaoming也拥有toString属性。

要判断一个属性是否是xiaoming自身拥有的，而不是继承得到的，可以用hasOwnProperty()方法：

var xiaoming = {

name: '小明'

};

xiaoming.hasOwnProperty('name'); // **true**

xiaoming.hasOwnProperty('toString'); // **false**

## ES6新增数据类型

传统js对象的键只支持字符串，引入新的数据类型以支持其他数据类型作为键

### Map

Map是一组键值对的结构，具有极快的查找速度。检索速度不会因为数据量增大而变慢

**var** m = **new** Map([['Michael', 95], ['Bob', 75], ['Tracy', 85]]);

m.get('Michael'); *// 95*

**var** m = **new** Map(); *// 空Map*

m.set('Adam', 67); *// 添加新的key-value*

m.has(‘Bob’) //是否存在键Bob

m.delete(‘Bob’) //删除键 再次调用get返回undefined

键一旦重复，会发生值覆盖

### Set

存储不重复的键

**var** s1 = **new** Set(); *// 空Set*

**var** s2 = **new** Set([1, 2, 3]); *// 含1, 2, 3*

s.add(4);

添加和删除方法与Map相同

### Map和Set的遍历 Iterable

遍历Array可以采用下标循环，遍历Map和Set就无法使用下标。为了统一集合类型，ES6标准引入了新的iterable类型，Array、Map和Set都属于iterable类型。

**var** a = ['A', 'B', 'C'];

**var** s = **new** Set(['A', 'B', 'C']);

**var** m = **new** Map([[1, 'x'], [2, 'y'], [3, 'z']]);

**for** (**var** x of a) { *// 遍历Array*

console.log(x);

}

**for** (**var** x of s) { *// 遍历Set*

console.log(x);

}

**for** (**var** x of m) { *// 遍历Map*

console.log(x[0] + '=' + x[1]);

}

更好的迭代方式是使用Iterator内置的forEach方法--- ES5.1开始引入

**1.数组**

var a = ['A', 'B', 'C'];

a.forEach(function (element, index, array) {

// element: 指向当前元素的值

// index: 指向当前索引

// array: 指向Array对象本身

console.log(element + ', index = ' + index);

});

**2.Set**

Set与Array类似，但Set没有索引，因此回调函数的前两个参数都是元素本身

3.Map

Map的回调函数参数依次为value、key和map本身：

中断循环：forEach循环的中断可以转变思路，先使用filter过滤掉不需要循环的元素，在将剩余的元素进行forEach循环

## Strict模式

在strict模式下运行的JavaScript代码，强制通过var申明变量，未使用var申明变量就使用的，将导致运行错误。防止不使用var声明全局变量造成操作结果互相影响。

## 条件判断

### If 判断括号内布尔值的原则

Javascript将 null 、undefined 、0 、NaN 、 和空字符串’’视为false,其他值视为true

### for…in循环

for( var element in obj) 可以遍历对象的所有属性（包括继承过来的属性）

要过滤掉对象继承的属性，用hasOwnProperty()实现

for(var u : obj){

if( obj.hasOwnProperty(u) ){

//得到对象属性（过滤掉了继承而来的属性）

}

}

## 附

var height = parseFloat(prompt('请输入身高(m):')); //接受输入

<https://www.liaoxuefeng.com/wiki/1022910821149312/1023021053637728>