**===**运算符比较，它不会自动转换数据类型，如果数据类型不一致，返回false，如果一致，再比较。

另一个例外是NaN这个特殊的Number与所有其他值都不相等，包括它自己,唯一判断标准是isNaN()

动态语言, var

## 字符串

**多行字符串** 例子:

`这是a一个

多行

字符创`

**模板字符串** :var name = '小明';var age = 20; var message = `你好, ${name}, 你今年${age}岁了!`;

字符串的**substing**和**indexOf**方法

## 数组

**数组**创建的两种方法:[1, 2, 3.14, 'Hello', null, true]; new Array(1, 2, 3); // 创建了数组[1, 2, 3]

数组的**slice()**方法:就是对应String的substring()版本  **splice()**方法是修改Array的“万能方法”，它可以从指定的索引开始删除若干元素，然后再从该位置添加若干元素 push()添加

**concat()**方法把当前的Array和另一个Array连接起来，并返回一个新的Array：

**join()**方法是一个非常实用的方法，它把当前Array的每个元素都用指定的字符串连接起来，然后返回连接后的字符串：arr.join('-');

## 对象:

var xiaoming = {

name: '小明'

};

xiaoming.age = 18; // 新增一个age属性

delete xiaoming.age; // 删除age属性

console.log('name' in xiaoming); // true

使用一个不存在的变量或者属性时候,不会报错,会显示**unfind**

var xiaohong = {

name: '小红',

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

};

xiaohong的属性名middle-school不是一个有效的变量，就需要用''括起来。访问这个属性也无法使用.操作符，必须用['xxx']来访问

## 条件判断

**JavaScript把null、undefined、0、NaN和空字符串''视为false，其他值一概视为true，a=1;if(a)为true**

## 循环

**for ... in**循环，它可以把一个对象的所有属性依次循环出来：

var o = {

name: 'Jack',

age: 20,

city: 'Beijing'

};

for (var key in o) {

console.log(key); // 'name', 'age', 'city'

}

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

for (var i in a) {

console.log(i); // '0', '1', '2'

console.log(a[i]); // 'A', 'B', 'C'

}

**Map**

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

m.set('Adam', 67);

**foreach:**

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

m.forEach(function (value, key, map) {

console.log(value);

});

## 方法

定义方法的两种方式:

function abs(x) {

if (x >= 0) {

return x;

} else {

return -x;

}

}

var abs = function (x) {

if (x >= 0) {

return x;

} else {

return -x;

}

}; //注意需要分号

...rest 和rest参数

## 解构赋值:为了简化代码而生(要es6浏览器才可以用)

var [x, y, z] = ['hello', 'JavaScript', 'ES6'];

var {name, age, passport} = person;//赋值会把对象person.name赋值给name,其他同理

**foreach**:在js中的遍历器:可以遍历数组/对象/map/set

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

m.forEach(function (value, key, map) {

console.log(value);

});

apply() 方法call()方法 ,控制函数的this指向, 方法.apply(xiaoming,[方法的参数])

//小明是方法的this想指向的对象

数组的**reduce**(函数名). /**map**(函数名)/**filter**(函数名)/**sort**(函数名) 可以让数组的元素进行制定函数的运算再返回一个新数组

## 闭包(不是很懂)

**箭头函数** x => x \* x 相当于function (x) {return x \* x;}

// 可变参数:

(x, y, ...rest) => {

var i, sum = x + y;

for (i=0; i<rest.length; i++) {

sum += rest[i];

}

return sum;

}

**Generator 函数和yield(不很懂)**

调用Generator函数时，并不会执行函数内部的代码，而是返回一个遍历器，该遍历器包含一个next方法。每次执行next方法，Generator函数体会开始执行，直到遇到yield语句，执行该语句并在此暂停

function \* gen(){

var result1 = yield 'hello';

var result2 = yield 'world';

return result1 + result2;

}

var g = gen();

g.**next**(1);

//{value : 'hello', done : **false**}

返回一个对象,value是返回值, done表示Generator函数体是否被执行完。

**Date对象** var now=new date(); var d = Date.parse('2015-06-24T19:49:22.875+08:00'); //d是时间戳

## Get Exp(不太会)(正则表达式)

## JSON

序列化: var s = JSON.stringify(xiaoming);//xiaoming是一个js对象

反序列化: JSON.parse('{"name":"小明","age":14}');

**构造函数:** function Student(name) {

this.name = name;

this.hello = function () {

alert('Hello, ' + this.name + '!');

}

}

var xiaoming = new Student('小明');

**location对象(浏览器对象)**  location.protocol; // 'http'

location.host; // 'www.example.com'

location.port; // '8080'

location.pathname; // '/path/index.html'

location.search; // '?a=1&b=2'

location.hash; // 'TOP'

location.reload;

**cookie:document(浏览器对象)对象的cookie属性.** document.cookie 一般用户信息存在cookie然后通过消息头传递

**dom:**

var p = document.getElementById('p-id');//获取元素id等于p-id的

// 设置HTML:

p.innerHTML = 'ABC <span style="color:red">RED</span> XYZ';

p.innerHTML = 'ABC'; // <p id="p-id">ABC</p>

// innerText 是设置文本

//增加节点

var

list = document.getElementById('list'),

haskell = document.createElement('p');

haskell.id = 'haskell';

haskell.innerText = 'Haskell';

list.appendChild(haskell);

删除节点

var parent = document.getElementById('parent');

parent.removeChild(parent.children[0]);

parent.removeChild(parent.children[1]);

删除本身

// 拿到待删除节点:

var self = document.getElementById('to-be-removed');

// 拿到父节点:

var parent = self.parentElement;

// 删除:

var removed = parent.removeChild(self);

removed === self; // true

## HTML5控件



<!-- HTML -->

<form id="test-form" onsubmit="return checkForm()">

<input type="text" name="test">

<button type="submit">Submit</button>

</form>

<script>

function checkForm() {

var form = document.getElementById('test-form');

// 可以在此修改form的input...

// 继续下一步:

return true;

}

</script>

//表单提交

// 把用户输入的明文变为**MD5**:

pwd.value = toMD5(pwd.value);