javaScript的组成：

ECMAScript(5.1)：javaScript组成言的语法和基本对象

DOM：文档对象模型，描述处理网页内容的方法和接口

BOM：浏览器对象模型，描述与浏览器进行交互的方法和接口

javaScript特点：

跨平台的web脚本语言

向html页面添加交互行为

可与服务器进行通信

javaScriptxie写的位置

行间<input type="button" onclick="alert(1);" value="Click" />

内嵌 <script>//这里是js代码</script>

外链js文件<script src="demo.js"></script>

**变量**

变量用于保存数据，复用代码

声明变量的方式

使用关键字 var

声明变量 var 变量名;

赋值 变量名 = 数据;

声明并赋值 var 变量名= 数据;

变量的命名规则

1.允许是字母，数字，下划线(\_)，美元符($)任意组合而成

2.不允许数字开头

3.不允许使用关键字和保留字

4.关键字：当前语法中正在使用的单词 http://www.w3school.com.cn/js/pro\_js\_keywords.asp

5.保留字：将来可能在语法中使用的单词 http://www.w3school.com.cn/js/pro\_js\_reservedwords.asp

变量的命名风格

1、语义化

变量名要尽量贴合实际的意思

2、驼峰命名

小驼峰：从第二个单词开始，首字母大写

getElementById

大驼峰：首字母大写

Math() Date()

**JS属性操作**

**1.操作属性的方式**

点（.），中括号（[]） 在可变的情况下用[] 凡是操作页面中的内容都是字符串

**2.操作特殊属性的方式**

通过 [] 的方式获取

通过 驼峰 的方式（去掉 -，- 后的首字母大写）

用中括号的方式去操作不符合变量命名规则的属性

box.style['font-size']

如果要去掉横杠的话，把横杠后面的单词首字母大写

box.style['fontSize']

如果非要用点操作的话，要把横杠后面的单词首字母大写

box.style.fontSize

**3.HTML 的属性操作：读、写**

读操作：获取、找到

元素.属性名

写操作：“添加”、替换、修改

元素.属性名 = 新的值

**4.特殊的属性操作**

innerHTML :读写某个元素从开始标签到结束标签的所有内容。

class为js中的保留字，所以不用当作属性去操作，要获取元素的class，只能用className

在操作属性 src href 的时候，拿到的值是图片的绝对地址 不可以拿来做判断

颜色值也不可拿来做判断

目前改变input的type属性 在IE678下是不认的（以后可以用）

**流程控制语句**

**If判断语句**

1. 当我只有一种情况要去处理的时候 if (判断条件) {当判断条件成立的时候，执行这里的代码}

2. 当有两种不同情况要去处理的时候用 if (判断条件) {当判断条件成立（true）的时候，走这里的代码}

else{当判断条件不成立（false）的时候，走这里的代码}

3. 当有3种或者3种以上的不同情况要去处理 if (判断条件) {当判断条件成立的时候，走这里的代码}

else if(判断条件1){当判断条件1成立的时候，走这里的代码}

else if(判断条件2){当判断条件2成立的时候，走这里的代码 }

else{当以上所有条件都不成立的时候，走这里的代码}

程序首先会把if后括号中的判断条件转为true或者false，然后交给if去做判断。

一个等号代表赋值，两个等号代表判断是否相等

**for循环语句**

作用：根据一定的条件，重复的执行一行或多行代码

语法：for (条件初始化;条件判断;条件变化) {当条件判断成立（true）的时候，重复执行这里的代码}

运行流程

第一步：条件初始化

第二步：条件判断

第三步： 走大括号里的代码

第四步：条件变化

for走完第一次后，从第二次开始就不走第一步了，重复走第二步、第三步、第四步

何时结束：当条件判断不成立（false）for循环就会结束。 注意：给的条件一定要有不成立时候，否则就会出现死循环

for嵌套

嵌套的for中，外层的for会先执行一次，进到里层的for，里层的for如果没有执行完，外层的for是不会执行的。里层的for执行完成后，会执行外层for的下一步。嵌套的for中，两个变量一定不要一样

**for in**

用来遍历对象数据

语法

for(var 变量名 in 对象名){

变量名：对象每一个数据的名称

对象名[变量名]：对象每个数据名称对应的值

}

数组可以用for in去遍历，数组中的每个数据对的名字就是它的下标值

**while**

当条件成立的时候，去循环代码

语法

While (判断条件) {当判断条件成立的时候，执行这里的代码}

一定要在代码块里，给一个条件变化，不然会死循环

与for的区别

1、写法不同

2、for循环是次数循环

while是条件循环，不一定知道循环的次数

3、for循环是横向循环，while是纵向循环

遇到的循环的时候，首先要想到for，如果说用for做不出来了，再考虑用while

**break**

终止循环，不管后面有多少次循环，都不走，而且break后面的代码也不会执行。它用在循环体当中

**continue**

终止循环，跳过当次循环，continue后面的代码是不会执行的，但后面的循环还会执行

**switch**

根据不同的情况做不同的事情。根据变量的值做不同的事情

可以改成if语句

switch(变量){

case 值:

当值跟变量相等的时候，执行这里的代码

break;

case 值:

当值跟变量相等的时候，执行这里的代码

break;

default:

当上面的值都不与变量相等的时候，执行这里的代码

}

例子var n=3 ;

switch(n){

case 1:

alert('今天星期一');

break;

case 2:

alert('今天星期二');

break;

case 3:

alert('今天星期三');

break;

case 4:

alert('今天星期四');

break;

case 5:

alert('今天星期五');

break;

case 6:

alert('今天星期六');

break;

case 7:

alert('今天星期日');

break;

case 8:

alert('你傻呀，今天星期几都不知道');

break;

default:

alert('今天星期几，我也不知道');

}

default 当条件都不满足时执行这里的代码

swicth的穿透力非常强,它会执行条件成立代码，以及后面的代码，所以要用break，跳出当前的条件，后面的代码不会被执行

**获取元素的方法**

window 浏览器 document 文档，一个网页

1. 通过ID去获取一个元素 document/元素.getElementById('id名称')

2. 通过html标签获取一组元素 document/元素.getElementsByTagName('标签名称')

3. 通过class名获取一组元素 document/元素.getElementsByClassName('class名称') 获取到的也是一组元素的集合（类数组）

4. 通过CSS 选择器获取一个元素 document/元素.querySelector('css选择器');

5. 通过CSS 选择器获取一组元素 document/元素.querySelectorAll('css选择器')

**this**

this:关键字，代表指向，不能作为变量名字。当前对象，只能读，不能改

1. 函数被直接调用，this指向window

2. 当函数被事件调用，并且是以赋值的方式出现。谁调用了函数，this就指向谁

**自定义属性**

元素名.自定义属性名 常见的有自定义索引值 如li.index

**javaScript数据类型及转换**

**数据类型**

js中数据类型分为两大类，6种

一、基本（简单）数据类型

1、字符串 String 由一对引号包含起来了有0个或多个字符组成的串。引号一定是成对的

2、数字 Number 1）、由0-9之间的任意一个数字组合而成，分为整数和小数。取值范围从负无穷（-Infinity）到正无穷（+Infinity）。有最小值也有最大值 最小值：Number.MIN\_VALUE 最大值：Number.MAX\_VALUE

2）、NaN not a number 自己与自己都不相等 当运算失败的时候会出现

3、布尔值 Boolean true 真 false假

4、未定义 Undefined 有并且只有一个值undefined

5、空 Object有并且只有一个值null

二、复合（复杂）数据类型

6、对象 Object

由简单的数据类型和复合数据类型组合而成的。它的数据是放在了一个大括号中{}。数据有0个或者多个键值对组成，每一个数据之间用逗号隔开，最后一个数据不能用逗号，给报错

键值对key:value

key:数据名字，可以加引号也可以不加引号。当这个名字不符合变量命名规范的时候要加上引号

value:数据对应的值，js中的任意一种数据类型都可以

取某一个数据的值：可以用属性操作的方法去取每一个数据

**type of 检测数据类型**

typeof,返回这个数据对应的类型（字符串），跟系统自带的数据类型会有区别，首字母是小写的

格式：typeof 数据 ，typeof(数据)

ECMA中的数据类型和typeof的数据类型有点差别

ECMA中的6种数据类型: String、Number、Boolean、Undefined、 Null、 Object。

typeof中的6种数据类型: string、number、boolean、undefined、object、object、function。

判断一个数据类型的时候，用typeof==小写的对应类型

**数据类型转换**

把一种数据类型转换成另外一种数据类型

为何要转换：有的时候，我们需要进行数据之间的操作，这个时候，需要两种数据都是一样的类型才能进行操作，如果两种类型不同，可能得不到我们想要的结果

js可以把任意一种数据类型转换成以下几种，1.数字，2. 字符串，3. 布尔值

任意类型转数字

1、Number(数据类型)

Undefined类型转数字 结果转为NaN

Null类型转数字 结果转为0

Boolean类型转数字 true转成1 false转成0

String类型转数字 1）空字符串、空格字符串转为0

2）字符串中的内容为非数字转为NaN

3）字符串中的内容为纯数字，转为相应的数字

Number 数字转数字就为原来的数字

Object 对象转数字为NaN

空数组 转数字为 0

非空数组 转为数字为NaN

2、parseInt(字符串)

把字符串转为数字

字符串的第一位字符，必需为数字或者空格+数字，这个时候，它会一直往后找，直到找到一个非数字的时候，就停止查找，把找到的数字 返回出来。否则返回NaN

3、parseFloat(字符串)

把字符串转为浮点型数字。规则与parseInt一样

inNaN (数据类型) 判断一个数据是不是NaN，如果这个数据是NaN，返回true否则返回false

任意类型转字符串

String(数据类型)

**任意数据类型转布尔值**

Boolean(数据类型);

1、Undefined 转布尔值结果为false

2、Null 转布尔值结果为false

3、Boolean 对应的布尔值

4、数字 0 false ; 非0 true ; NaN false;

5、String 非空字符串(包括空格)转布尔值结果为true

6、Object 均转成 true （包括空对象）

**类型转换分类**

1、显示类型转换

明确告诉它要转换的类型 如：Number() String() Boolean() parseInt() parseFloat()

2、隐式类型转换

我们没有明确告诉他要转什么类型，但是在运算的时候，如果不转成另一种类型，运算就无法进行。js会自动把类型进行转换

**运算符**

算术运算符、赋值运算符、关系运算符、逻辑运算符

算术运算符

+ 加、- 减、 乘、/ 除、% 取模（求余数）、++、--

++，-- 两个符号可以放在操作数的前面也可以放在后面

操作一个数据的时候

++或--放在数值的后面与前面的效果是一样的

操作两个数据（给另一个变量赋值）的时候

操作符放在变量后面

++或--先把本身的值赋给另一个变量，然后自己再+1或-1

操作符放在变量前面

++或--先把本身的值+1结果赋给另一个变量，然后自己再+1或-1

var a=1;

a++;

++a;

console.log(a); / /a=2

var a=1;

var b=a++;

console.log(a,b); //a=2 b=1/

var c=1;

var d=++c;

console.log(c,d); //a=2 d=2

**赋值运算符**

=、+=、-=、=、/=、%=

**关系运算符**

<、>、<=、>=、==、!=、===、!==

结果都会转化成布尔值

<=、>= 满足一个条件就会返回true

=== 比较严格，它比较的不光是值，同时要比较类型。如果值与类型同时相同，返回true

!== 比较严格，它比较的不光是值，同时要比较类型。如果值与类型有一个不等，就返回true

undefined只与null在比较的时候会返回true,其它都返回false，自己与自己也相等

**逻辑运算符**

&& 与 并且的意思

当多个条件同时满足的时候，整个条件才会成立

当左侧条件成立，返回右侧的值。当左侧条件不成立，会返回左侧内容

|| 或 或者

如果有一个条件成立，那整个条件就都成立。如果多个条件都不成立，整个条件才不成立

当左侧条件成立，返回左侧的值。当左侧条件不成立，会返回右侧内容

! 否 取反

把值转成布尔值

**三元（目）运算符**

a（条件）? b（真）: c（假）

提升优先级的方法，加（）；

**函数**

**定义函数的语法**

function 函数名 (参数) { 函数体 } 命名函数

function () { 函数体} 匿名函数

**函数类型**

**1. 函数声明：**

function fn () {}; 小括号中可以放参数，也可以不放参数

调用

1）函数名字加上() 可以在任意地方调用

2）在函数命名的时候，在function之前加上 ! ~ + - (函数体)() 即自执行函数

**2. 函数表达式**

var fn = function(){}; function后面的名字可以有也可以没有

调用：

1) 变量名+()

2) 直接在末尾加上一对小括号 注意：必须要在函数表达式的后面调用

**函数参数**

从函数外面，向函数里面传递的数据，传递的数据可以在函数内部使用，那这个数据就叫参数

函数的参数可以是多个，中间用逗号隔开

**1、形参**

形式上的参数

声明函数时候的参数

它只能在函数内部去使用

可以认为，在这个函数内部的最上面，声明了一个变量，变量的名字叫abc（函数的参数）它的值，是由调用函数的时候传进来的值（实参）

**2、实参**

在函数调用的时候实际传入的参数

函数调用的时候传入的参数

实参必需与形参一一对应，如果定义函数的时候有形参，但是在调用函数的时候，没有给对应的实参。那么那个形参就为undefined

声明函数，它小括号里的参数叫形参，函数调用时小括号里的参数是实参。

**arguments**

代表了所有 实参 的集合，它与形参没有关系 类数组

arguments只能在函数内部使用

它有一个length属性，代表传进去参数的个数，可以用下标去取对应的实参

**函数返回值**

返回函数的运算结果

return 值

return只能用在函数里

能够阻止函数继续运行，return后面的代码是不会运行的

每个函数调用后，都会有返回值，如果说在函数内部使用了return，那这个函数返回的结果就是return后面的值。如果说函数内部没有使用return，那这个函数返回undefine

return用在这个函数我们要让它算出一个东西，在下面去使用它算出的值

**作用域**

变量与函数的可访问范围。它们是以函数块来划分，每个函数就是一个作用域

**全局作用域**

全局：整个文档

1、在函数外面声明的变量或者函数就是全局作用域，它们在任何地方都是可以访问到的(包括后面的script标签对)

2、如果一个变量没有var而直接赋值，那不管它是在函数内部还是在函数外声明，它都具有全局作用，在任何地方都可以访问到

所有的全局变量都是window的属性。window是一个全局对象，它在任何地方都可以访问到，那它的属性也是一样的

注意：

声明变量的时候，一定要加var，尽量避免使用全局变量，要用局部变量

只要是全局变量就是window的属性

**局部作用域**（函数作用域）

局部：函数内部

在函数里面声明的变量或者函数是局部作用域，它们只能在声明的函数里面使用，在外面是访问不到的

因为函数可以嵌套函数，在函数内部声明的变量与函数，它的子函数是可以访问到的

**作用域链**

变量与函数的查找规则

从里往外找

一个作用域，函数要找到一个变量，首先他会找自己内部有没有这个变量，如果自己内部有这个变量，它就会取这个变量的值。如果说没有，它就会一层一层的往外找，直到找到为止。如果外面也没有，就会报错

**预解析**

变量与函数都会有一个解析过程，在作用域中会提前把变量与函数解析，解析到当前作用域的最开始的位置

1、找到var、function两个关键字与参数（形参）

2、如果说是变量（var定义的与参数）给他们赋一个undefined值。找到函数的时候，会把整个函数块拿过来

3、把找到的这些东西，放到对应作用域的最开始的位置

4、逐行解析代码

**getComputedStyle**

获取计算后的样式,它是一个属性，得到的是样式的集合

获取到的属性的值是有误差的

IE678是不认识的

getComputedStyle(obj)

currentStyle

获取计算后的样式 IE678

obj.currentStyle

**定时器**

**setInterval**

**重复执行定时器** : setInterval(函数,每次执行时间)

参数：

1、函数：当时间到达后会执行这个函数

1、直接在函数里面去写处理的代码

2、把处理代码放在外面的函数中，在第一个参数的大括号里调用这个函数

3、把处理代码放在外面的函数中，把第一个参数写成函数的名字

2、时间：重复执行的时间

单位是毫秒

1s=1000ms

**setTimeout**

**延迟执行定时器**：setTimeout

setTimeout(function(){当时间到达后，就会执行这里的代码，只执行一次},时间);

定时器里的this是指向window

**清除定时器**

定时器的编号

定时器的返回值，返回一个数字。这个数字就代表对应的定时器

直接用定时器的编号是没办法全部清除定时器的，我们可以声明一个变量，把定时器的返回值存到这个变量当中，用变量代表它，就可以清 除掉对应的定时器

把定时器赋值给一个变量后，这个定时器也同样会自动执行，定时器的执行与变量没有关系

1、清除重复执行定时器

clearInterval(定时器的返回值);

2、清除延迟执行定时器

clearTimeout(定时器的返回值);

**日期对象**

js中的一个内置方法，用来操作日期与时间

语法

new Date()

参数

1、无参数

以本机的时间做为参考，返回一个时间对象

2、new Date(年,月,日,时,分,秒)

这个参数为数字

3、new Date('June 10,2013 12:12:12')

参数为字符串，方式是外国人表示时间的方法

4、 new Date(时间戳)

new Date().getTime();

时间戳：从1970年1月1日0时0分0秒到一个时间点之间的毫秒数

注意：和本机时间没有关联，不跟随本机时间动态变化

我们能根据本机时间去获取时间对象，但是不能设置本机的时间

**获取日期对象**的相关方法

日期对象.getFullYear() 获取年

日期对象.getMonth() 获取月，从0开始

日期对象.getDate() 获取日

日期对象.getDay() 获取星期 (星期从周日开始，从0开始计算)

日期对象.getHours() 获取小时

日期对象.getMinutes() 获取分钟

日期对象.getSeconds() 获取秒钟

日期对象.getMilliseconds() 获取毫秒

日期对象.getTime() 获取1970-01-01 00:00:00到目前日期对象的时间差（单位:毫秒）

注意：1s （秒）= 1000ms（毫秒）

**设置日期**的一些方法

日期对象.setFullYear() 设置年

日期对象.setMonth() 设置月 (月份从0开始计算)

日期对象.setDate() 设置日

日期对象.setHours() 设置小时

日期对象.setMinutes() 设置分钟

日期对象.setSeconds() 设置秒钟

日期对象.setMilliseconds() 设置毫秒

日期对象.setTime() 获取1970-01-01 00:00:00到目前日期对象的时间差（单位:毫秒）

注意：1s （秒）= 1000ms（毫秒）

setDate(0) 设置的时间如果是0或者是负数为自动回滚到过去的时间，如果设置的时间为超出的，就会跳到未来的时间

setMouth( , )可以放两个参数，同时设置月和日。

setFullYear( , , )可以放三个参数，同时设置年，月，日。

**字符串方法**

**字符串**

放在一对引号当中，由0个或者多个字符组成的串

字符串的属性

一个数据所具有的特性或者特征

length

代表了这个字符串里字符的个数

字符串的length属性不能修改

**方法**

**1.charAt**

作用：通过下标值找到对应的字符

语法：字符串.charAt(index)

参数：

index：索引值 （取值范围：0-字符串的length-1）

如果超出这个范围，比如小于0或者大于等于length，会返回一个空字符，

如果参数没有，默认为0

返回值：返回一个字符，传进去参数对应的字符

**2. charCodeAt**

作用：通过下标找到对应字符的Unicode编码

语法：字符串.charCodeAt(index)

参数：

index：索引值 （取值范围：0-字符串的length-1）

如果超出这个范围，比如小于0或者大于等于length，会返回一个NaN

如果参数没有，默认为0

返回值：

返回一个数字，参数对应的字符的Unicode编码

**3.indexOf**

作用：通过字符去找对应的下标值，找这个字符首次出现的位置（从左往右）

语法：字符串.indexOf(searchValue,fromIndex)

参数：

searchValue：要找的字符

fromIndex：开始查找的位置，默认值为0

返回值

如果查到的话，返回这个字符对应的下标值。如果没查到，返回-1

**4.slice**

作用：截取字符串

语法：字符串.slice(beginSlice,endSlice)

参数：

beginSlice：起始位置

endSlice：结束位置，没有这个参数默认为最后一个字符的位置

注意：1、不包含结束位置的字符 2、起始位置不能大于结束位置，否则返回一个空

如果两个参数都没有的话，返回整个字符串

返回值：返回一个截取后的字符串，原字符串不变

**5.split**

作用：把字符串按分隔符分割成数组

语法：字符串.split(separator,limit)

参数：

separator：分割符

分割后的数组当中是不包含这个参数的。它的左右必需切割成数组中的数据，如果没有的话，就为空

1、如果参数没有，或者参数这一个空格字符，会把整个字符串分割成一个数组

2、如果参数为一个空字符串的话，它会把字符串中的每一个字符都分成数组中的每一个数据

limit：分割后数组的长度，没有的话默认为全部

返回值：返回一个数组，截取后的数组，原字符串不变

**6.substring**

作用：截取字符串

语法：字符串.substring(indexStart,indexEnd)

参数：

indexStart：起始位置

indexEnd：结束位置，没有的话默认为最后一个字符的位置

不包含这个参数对应的字符

与slice的区别：

slice的参数可以为负数，slice的起始位置是不能大于结束位置的

substring的参数必需为正数，substring的起始位置可以大于结束位置，它会自动把两个位置调换

返回值：返回截取的字符串，原字符串不变

**7.substr**

作用：截取指定起始位置和长度的子字符串

语法：字符串.substr(start,length)

参数：

start：起始位置

length：截取的字符串长度，没有的话默认为起始位置到最后一个字符的长度

返回值：返回一个字符串，截取后的字符串。原字符串不变

**8.toLowerCase**

作用：把字符串转成全小写

语法：字符串.toLowerCase()

参数：它没有参数

返回值：返回一个字符串，这些字符串都是小写。原字符串不变

注意：js中是区分大小写的

**9.toUpperCase**

作用：把字符串转成全大写

语法：字符串.toUpperCase()

参数：它没有参数

返回值：返回一个字符串，这些字符串都是大写。原字符串不变

注意：js中是区分大小写的

**10.trim**

作用

去掉字符串前后的所有空格

语法

字符串.trim()

参数

它没有参数

返回值

返回一个字符串，这个字符串不包含首尾空格。原字符串不变

**JSON**

(JavaScript Object Notation) 是一种轻量级的数据交换格式

一种类似JS中的对象格式的字符串

JSON的类型是字符串，只不过长得像对象数据

JSON规则

1、放在一对引号当中，用{}或者[]包含起来

2、这里面的数据是以键值对的形式出现 key:value

3、它的键(key)必需用双引号包起来

4、每个数据之间用逗号隔开

JSON用途

用于和后端进行数据传输

JSON.parse

作用

把JSON类型转换成JS中对象类型

如果要转化的数据，不符合JSON格式，那么就会报错

语法：JSON.parse(JSONString)

JSON.stringify(object)

作用：把object转成JSON格式的字符串

**Math方法**

1**.向上取整** Math.ceil

语法：Math.ceil(数字)

参数：一个数字,如果这个数字是整数，就什么都不做

返回值：返回参数向上取整后的结果

2.**向下取整** Math.floor

作用：把一个数字向下取整

语法：Math.floor(数字)

参数：一个数字，如果这个数字是整数，什么都不做

返回值：返回参数向处取整后的结果

3. **四舍五入**Math.round

作用：把一个数字四舍五入

语法：Math.round(数字)

参数: 一个数字

返回值: 返回参数四舍五入后的结果

4. **数字的绝对值**Math.abs

作用：取一个数字的绝对值

语法：Math.abs(数字)

参数：一个数字

返回值：返回这个参数的绝对值

5. **随机数**Math.random

作用：返回一个0-1之间的随机数，不包含0和1

参数：它是没有参数的

常用的取整公式：

0 ~ x : Math.round(Math.random()x)

0 ~ x-1 : Math.floor(Math.random()x)

1 ~ x : Math.ceil(Math.random()x)||1

x ~ y : Math.round(Math.random()(y-x) + x)

**数组方法：**

**数组**

1、由0个或者多个有序数据组成的一组数据

2、数组的每个数据都有一个对应的下标，下标是按顺序排列的

3、数组有一个length属性，代表这个数组里数据的个数

创建数组

1、[]

2、new Array()

因为数组是一个对象数据

参数：放的就是数组中的每个数据，每个数据之间用逗号隔开

如果参数中只放了一个，并且这个参数是个数字。这个数字不是数组中的数据，而是数组的长度

修改数组

通过下标值取到某个数据，然后直接给它赋值

数组的长度即可以获取也可以修改。这个方法是非标准的，可能某天浏览器升级了，这个方法就会不支持

**1. arr.push()**

作用

添加数组，但是它添加的位置在数组的后面

语法

数组.push(element1,element2...)

参数

一个或多个元素，可以累加

返回值

返回新数组的长度，它是一个数字。原数组已经变了

**2.arr.pop()**

mixed：混合类型

作用：删除数组最后一个元素

语法：数组.pop()

参数：无

返回值：返回被删除的那数据，原数组已经变了，变成删除后的数据

**3.arr.unshift()**

作用：添加数组，但是它添加的位置在数组的前面

语法：数组.unshift(element1,element2...)

参数：一个或多个元素，可以累加

返回值：返回新数组的长度，它是一个数字。原数组已经变了

**4.arr.shift()**

作用：删除数组第0个数据

语法：数组.shift()

参数：无

返回值：返回被删除掉的那个数据，原数组改变了，变成删除后的数据

**5.arr.forEach()**

作用：遍历数组

语法：数组.forEach(function(currentValue,index,arr){

//每循环一次就执行这里的代码

})

参数：

forEach里的参数是回调函数

回调函数里的参数

currentValue 当前数组中的数据

index 当前数据对应的索引

arr 这个数组

**6.arr.sort()**

作用：数组排序

语法：

数组.sort(function(){

//排序的方法

})

参数：可选。用来指定按某种顺序进行排列的函数。如果省略，元素按照转换为的字符串的诸个字符的Unicode位点进行排序。

返回值:返回排序后的数组，原数组已经变成了排序后的数组

arr.sort(function(a,b){

return a-b; //从小到大排序

});

arr.sort(function(a,b){

return b-a; //从大到小排序

});

arr.sort(function(a,b){

return 0.5-Math.random(); //随机排序

});

**7.arr.concat()**

作用： 连接数组

语法： 数组.concat(value1,value2...)

参数： 参数可以是数组也可以是非数组

返回值：返回一个连接后的新数组，原数组不变

**8.arr.reverse**

作用： 反转数组

语法： 数组.reverse()

参数： 无

返回值：返回一个把原数组反转的数组，原数组变化

**9.arr.join**

作用

把数组变成字符串

语法

数组.join(separator)

可选，用于指定连接每个数组元素的分隔符。分隔符会被转成字符串类型；如果省略的话，默认为一个逗号。如果 seprator 是一个空字符串，那 么数组中的所有元素将被直接连接。

参数

separator 分隔符 它会存在字符串当中

如果参数为空字符串，那会把数组中的引号跟逗号全去掉

如果参数不为空，那会把数组中的逗号变成这个参数，并且把引号去掉

返回值

数组用指定的分隔符，拆分成一个字符串，原来的数组不变

把字符串切割成数组用：string.split

把数组拼接成字符串用：array.join

**10.arr.slice**

作用：截取数组中一部分

语法：数组.slice(begin,end)

参数：

begin：起始位置（起始下标的位置）

end：结束位置（结束下标的位置）

如果没有，默认到数组的最后一位

两个参数都没有，默认截取全部数组

返回值：返回一个截取后的数组，原数组不变

**11.arr.splice**

作用：替换数组

语法：数组.splice(start,deleteCount,item)

参数：

start 起始位置

deleteCount 删除的长度

item 替换的内容

如果没有这个参数，代表截取数组

如果有这个参数，会先取到前两个参数决定的位置，然后用这个参数顶替它

返回值：返回截掉的内容，原来的数组已经变化了，变成删除后内容