# 数组（Array）

数组也是一个对象，功能和普通对象（Object）的功能类似的

\* - 数组中也可以保存不同的元素（相当于属性值），不同的是普通对象是通过一个字符串作为属性名来操作属性的，而数组是通过索引来操作元素的。

\* - 索引（index）

\* - 索引就是从0开始的整数

\* - 数组比普通对象的存储效率要高，在开发中我们经常使用数组来存储一些数据

创建一个数组对象

var arr = new Array();

使用typeof检查一个数组对象时会返回object

console.log(typeof arr); //object

向数组中添加元素

\* - 语法：数组[索引] = 值;

arr[0] = 10;

arr[1] = 20;

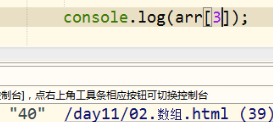
arr[2] = 30;

arr[3] = 40;

arr[4] = 30;

读取数组中的元素

\* - 语法：数组[索引]



如果读取一个数组中没有的元素，会返回undefined，和对象类似。所以修改\删除数组中的元素方法和对象中的方法是一样的。

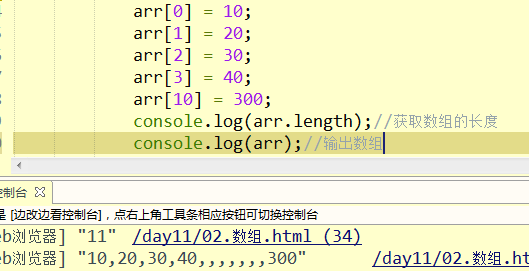
## length：

获取数组的长度：

\* - length属性可以获取或设置数组的长度

\* - 对于连续的数组通过length属性获取到的就是数组中元素的个数

\* - 对于非连续的数组length属性获取到的就是数组的最大索引+1（不要使用非连续数组）

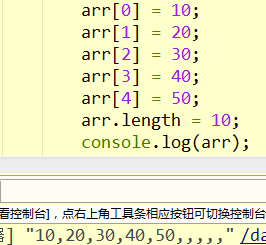
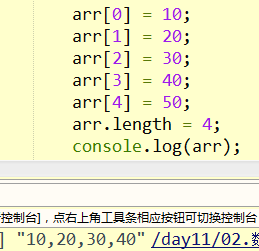


对于非连续数组，会在数组中创建出空的元素

修改数组的length属性

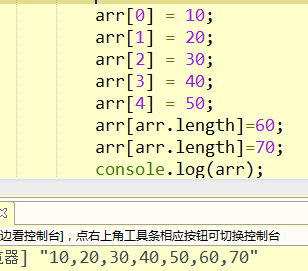
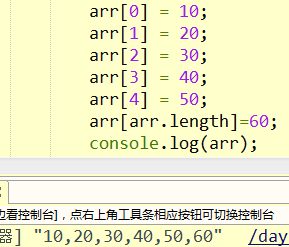
如果将长度修改为小于之前的长度的值，则多余的部分会被删除

如果将长度修改为大于之前的长度的值，则数组中会创建出空的元素



向数组的最后添加一个元素

语法：数组[数组.length] = 值;



## 数组的字面量

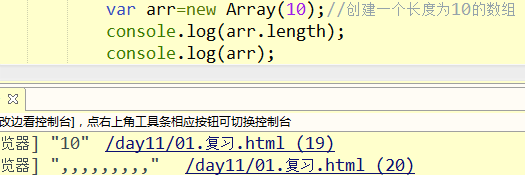
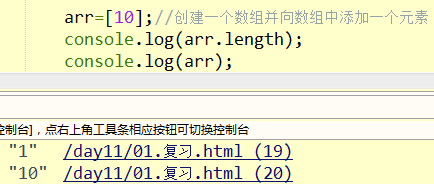
var arr = [] ;

相比于var arr = new Array();更方便；而且使用字面量，可以在创建数组的同时，向数组中添加元素 语法：var 数组 = [元素1,元素2...元素N];

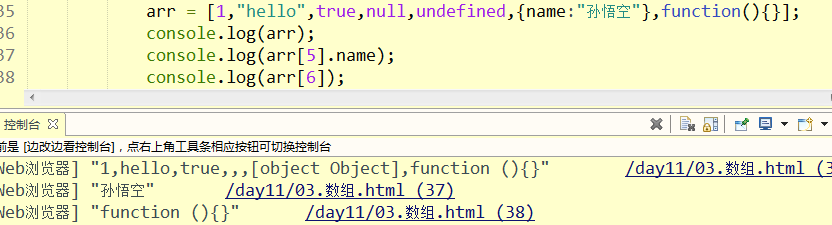
var arr=[1,2,3,4,5];

采用构造函数也可以：var arr=new Array(1,2,3,4,5);和字面量是一样的，但字面量方便

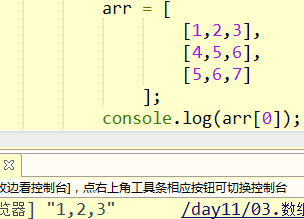
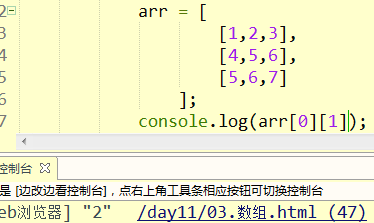
区别：



在数组中可以保存任意类型的元素，只不过null和undefined是空的



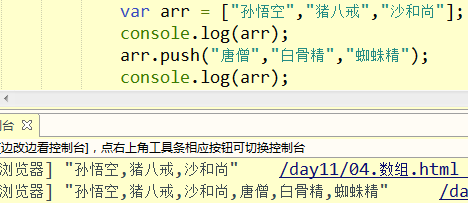
数组中也可以保存数组，如果数组中保存的依然是数组，则我们称这个数组为二维数组

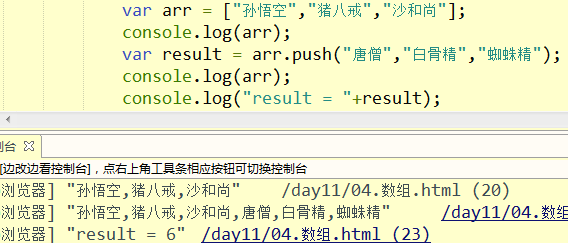
## push() pop() unshift() shift()

### push()

\* - 该方法可以用来向数组的末尾添加新的元素,并返回数组的新的长度



可以添加一个或多个元素

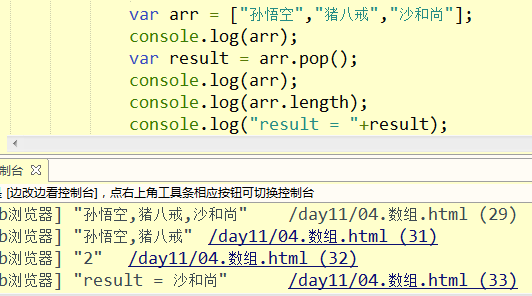


并且会返回数组的新的长度

### pop()

\* - 用来删除并返回数组的最后一个元素

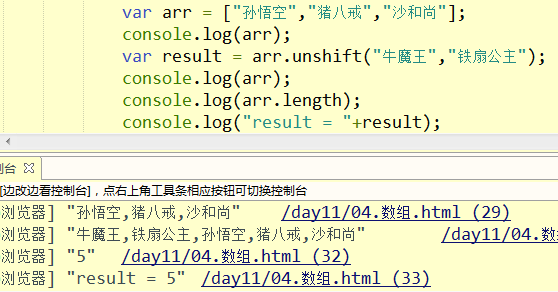
\* - 元素删除以后，会自动修正数组的长度



调用一次删除最后一个，再调用一次再删除新数组中最后一个

### unshift()

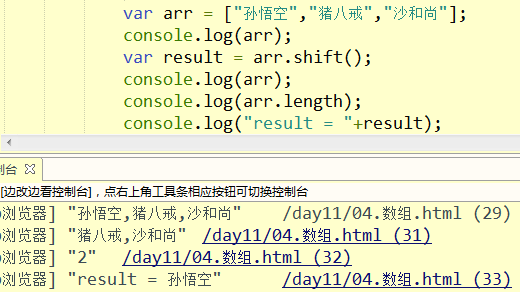
\* - 向数组的最前边添加元素，并返回新的长度



### shift()

\* - 删除数组最前边的元素，并返回被删除的元素

\* - 删除完元素以后，会自动调整数组的长度



## 遍历数组

遍历数组就是指将数组中的每一个元素都获取到

\* - 一般情况下遍历数组都是使用for循环来遍历

var arr = ["孙悟空","猪八戒","沙和尚","唐僧","白骨精","蜘蛛精"];

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

console.log(arr[i]);

}

### 练习

function Person(name , age){

this.name = name;

this.age = age;

}

//向原型中添加toString

Person.prototype.toString = function(){

return "Person[name="+this.name+", age="+this.age+"]";

};

//创建几个Person的实例

var per1 = new Person("孙悟空",18);

var per2 = new Person("猪八戒",28);

var per3 = new Person("白骨精",16);

var per4 = new Person("蜘蛛精",14);

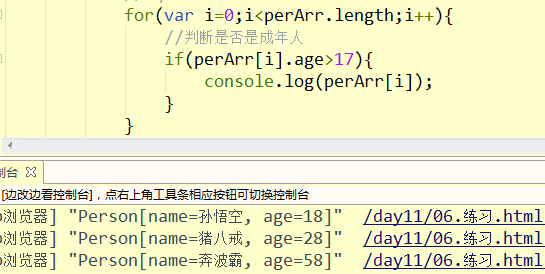
var per5 = new Person("红孩儿",8);

var per6 = new Person("奔波霸",58);

//将这些对象保存到一个数组中

var perArr = [per1 , per2 , per3 , per4 , per5 , per6];

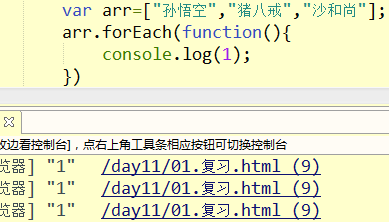
//对perArr数组进行遍历，并将其中成年人的信息输出



也可以使用**forEach()**来遍历数组

forEach()中需要一个函数作为参数（可以使用匿名函数），

arr.forEach(function(){});



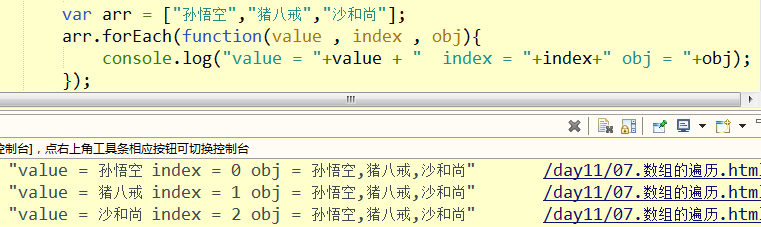
但我们没有调用函数，像这种由我们定义而不由我们调用的函数，我们称为**回调函数**，forEach()中的回调函数，由浏览器调用，数组中有几个元素函数就会被调用几次，每次遍历到一个元素都会调用一次函数，浏览器会通过实参的形式来将遍历到的内容传递进函数

浏览器在调用回调函数时传递了三个实参：

\* 第一个参数：正在遍历的元素

\* 第二个参数：正在遍历的元素的索引

\* 第三个参数：正在遍历的数组对象



**forEach()方法不支持IE8及以下的浏览器，所以如果需要兼容IE8则不要使用**

## slice() splice()

### slice()

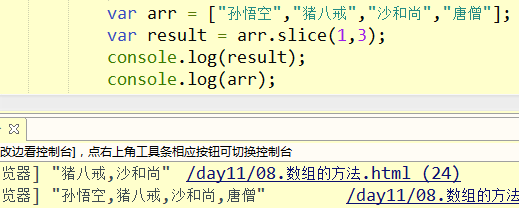
\* - 可以用来截取数组中的指定元素

\* - 参数：

\* 第一个参数，截取开始的位置(包括开始的位置)

\* 第二个参数，截取结束的位置(不包括结束的位置)

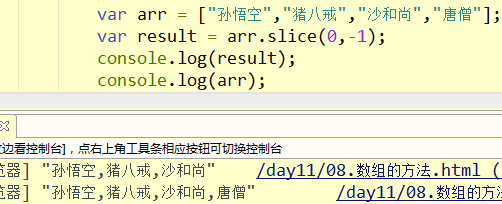
\* - 该方法不会影响到原数组，而是将截取到的内容封装为一个新的数组返回



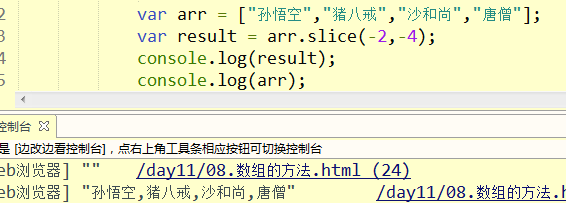
\* - 可以传递一个负数作为参数，如果是负数，则从后往前数

\* -1 表示 倒数第一

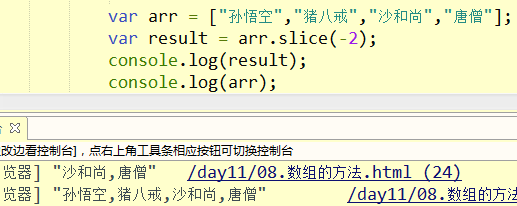
\* -2 表示 倒数第二



截取都是从左往右截取，如果第一个参数大于第二个参数，则会返回空串



\* - 如果省略第二个参数，则从开始位置一直截取到最后



### splice()

\* - 可以用来删除数组中的元素，并添加新元素

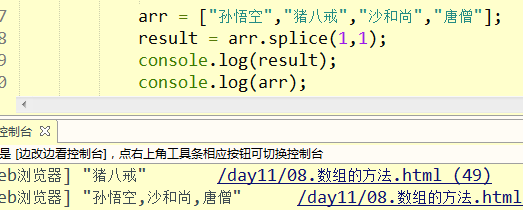
\* - 它会影响到原数组，一旦执行原数组中的元素将会被删除

\* - 它会将被删除的元素作为返回值返回

\* - 参数：

\* 第一个参数，开始位置的索引

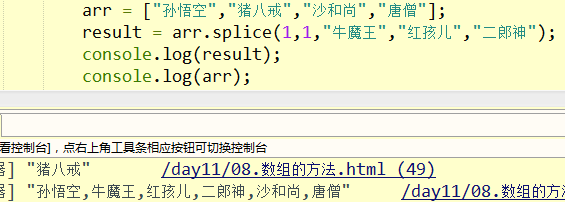
\* 第二个参数，表示删除的个数



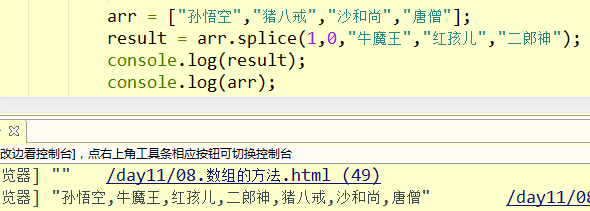
\* 第三个参数，这个方法还可以传递多个参数

\* 在第三个参数以后，可以跟多个元素作为参数，

\* 这些元素将会被插入到开始位置之前的位置



当第二个参数为0且第三个参数后跟多个元素时，相当于在第一个参数所对应元素后面插入多个元素而不删除那个元素

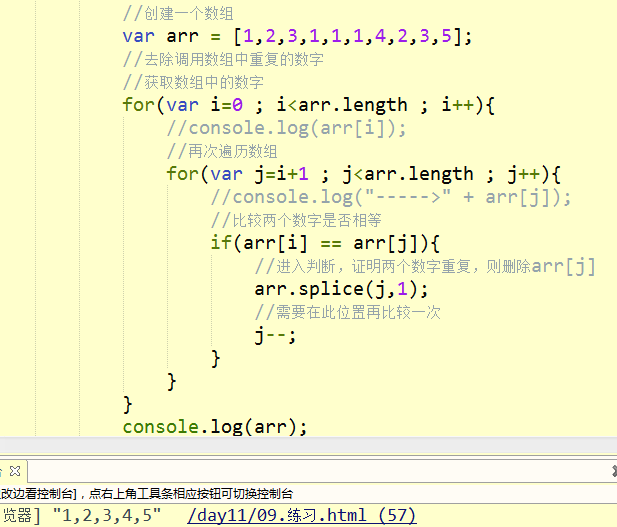


### 练习

//创建一个数组

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

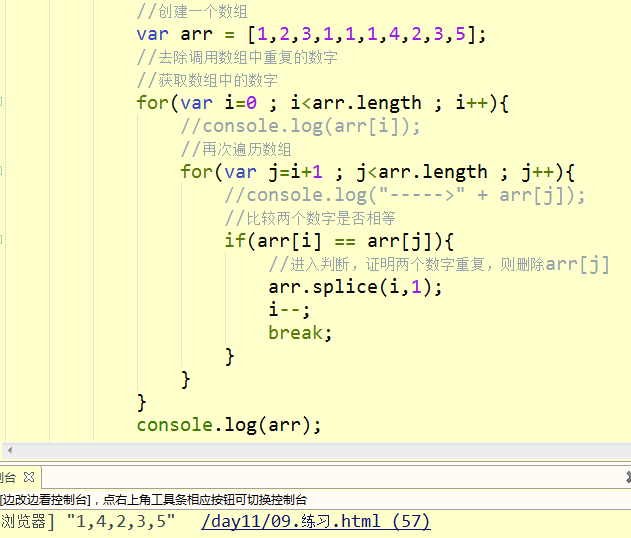
//去除调用数组中重复的数字



关于j--的解释：当arr[i]==arr[j]时，会删除arr[j],此时后面的参数会往前补位，由于此时删除了arr[j]，arr[j+1]会变成arr[j]，但arr[i]已经和arr[j]比较过了，所以arr[i]再不会和新arr[j]比较，如果arr[j]和arr[j+1]是相同的参数，就会出现重复数字，需要将j的值-1，再和新arr[j]比较

这种方法是将后面重复的删除

另一种方法：删除i：



关于i--的解释：当arr[i]==arr[j]时，会删除arr[i], arr[i+1]会变成arr[i],如果没有i--，

新arr[i]会接着和arr[j]比较，而不会和arr[j]以前的比较，如果以前有和arr[i]相同的数字也不会删除，所以需要i--，当嵌套的for循环完以后，外面的for又i++，相当于i的值没变，此时arr[i]其实是arr[i+1],这样新arr[i]又会重新开始依次比较，使用break可以提高运算速度，当arr[i]==arr[j]时，删除arr[i],i--,退出嵌套循环，新arr[i]又会重新开始依次比较,防止新arr[i]接着和arr[j]比较，节省时间。

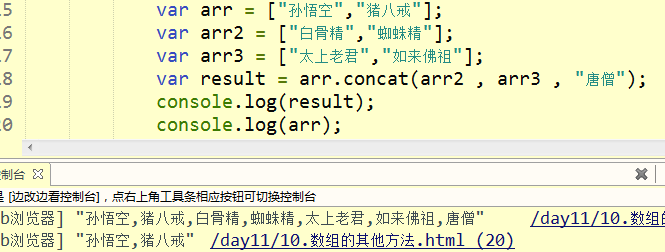
## 数组的其他方法

### concat()

\* - 可以用来连接两个或多个数组

\* - 不会影响原数组，而是将新的数组的返回

\* - 可以将多个数组作为concat参数传递，也可以传递单个的元素

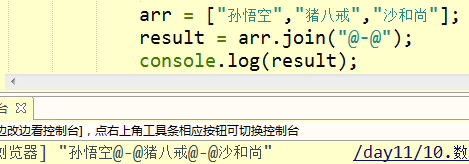


### join()

\* - 可以将数组中的所有的元素连接为一个字符串



\* - 该方法可以接收一个字符串作为参数，这个字符串将会作为连接符来连接元素



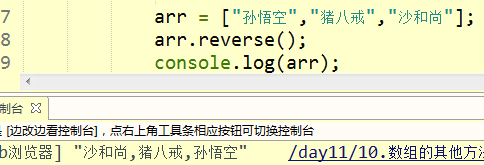
\* 如果不传，默认使用逗号连接， 如果想让元素中间什么也没有，传一个空串即可

### reverse()

\* - 该方法可以用来反转数组

\* - 前边的去后边，后边的去前边

\* - 它会影响到原数组，直接在原数组上反转

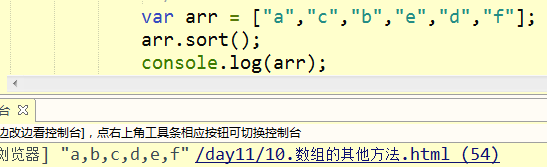


### sort()

\* - 可以用来对数组中的元素进行排序

\* - sort()会影响到原数组

\* - 默认它会按照元素的Unicode编码进行排序



\* - 如果希望使用sort对一个数字型的数组进行排序可能会得到不可预期的结果，比如10的Unicode编码比2的小

\* 因为它同样也会按照数字的Unicode编码进行排序

\* - 我们也可以自己来指定排序的规则

如果需要自定义排序规则，需要将一个函数作为参数传递给sort()方法

\* 函数的结构：function(a , b){}

\* 浏览器会多次以数组中的元素作为实参来调用该函数，

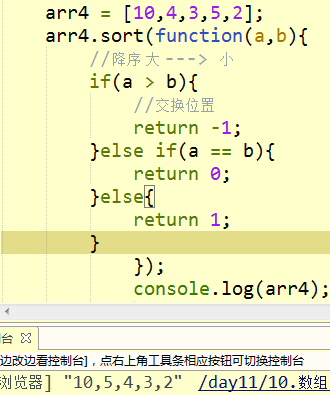
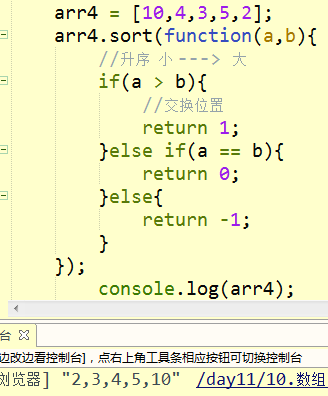
\* a和b具体是谁不确定，但是a一定是在b前边

\* 浏览器将会根据函数的返回值，来决定两个元素是否交互位置

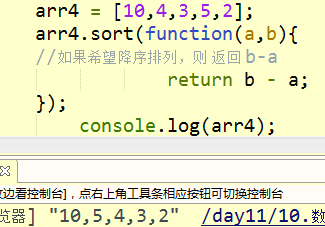
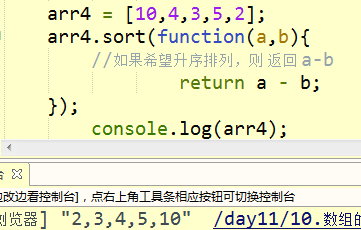
\* 如果返回一个大于0的值，则两个元素互相交换位置

\* 如果返回一个等于0的值，则证明两个元素相等，位置不变

\* 如果返回一个小于0的值，两个元素的位置也不变



**简便写法**：



# 函数的补充

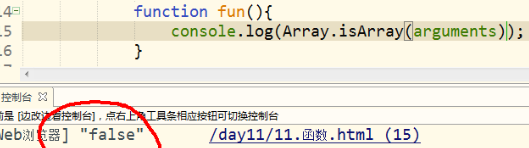
## arguments

除了this，在调用函数时，浏览器还传递了另一个隐含的参数，叫做arguments，arguments是一个类数组对象，但不是数组。

判断一个对象是不是数组：



除了这种方法，Array中也有一个方法：Array.isArray()

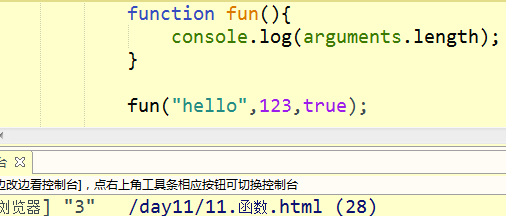


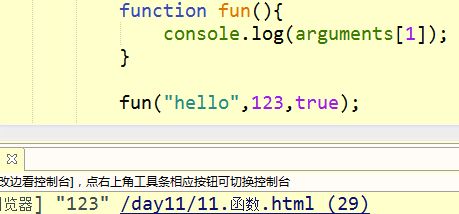
在arguments里边保存着函数执行时的实参

\* 实参会保存到arguments对象中指定的位置,

\* 通过arguments即使不定义形参也可以使用参数

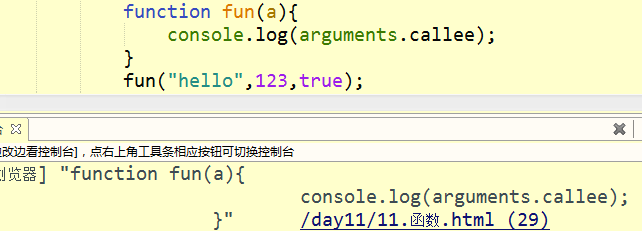
既然arguments是一个类数组对象，有些地方就和数组很像

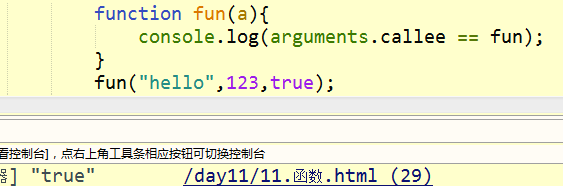




属性： callee

\* - arguments中有一个属性 callee，代表的是当前正在调用的函数对象



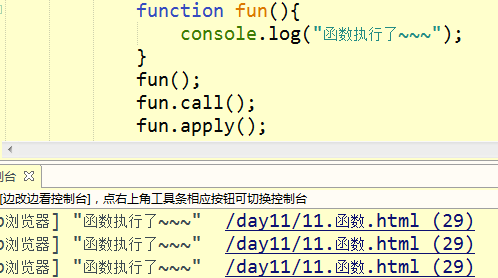


函数对象的方法：

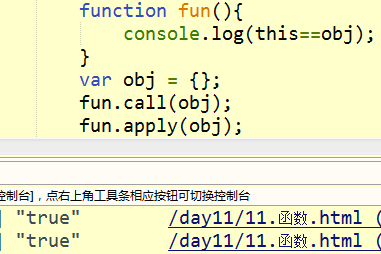
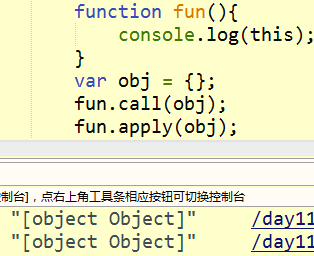
\* call

\* apply

\* - 当调用函数对象的call和apply时都会导致函数立即执行，就相当于调用了函数一样



如果通过call和apply去调用一个函数，则call和apply中的第一个参数将会是函数中的this，作用就是修改函数里的this。call和apply不指定参数时，this就是window



this是函数的上下文对象，根据函数的调用方式不同，this的值也不同

1.以函数的形式调用，this是window

2.以方法的形式调用，this是调用方法的对象

3.以构造函数的形式调用，this是新创建的那个对象

4.以call()和apply()调用，this是方法的第一个参数

区别：

\* call的参数，需要一个一个列出来

\* apply的参数，需要封装为一个数组传递

在call方法中可以将实参作为call的第二个以后的参数，传递给函数

fun.call(obj,1,2);

apply的参数不能一个一个的传，需要保存到一个数组中统一传递

fun.apply(obj , [1 , 2]);

# Date

## Date

是一个用来表示时间的对象

创建一个Date对象

如果直接调用构造函数，则会创建一个时间对象，对象中封装的是对象的创建时间，而不是输出时间

var d = new Date();

无论什么时候输出console.log(d);,时间都是一样的，都是执行到var d = new Date();的时间

创建一个表示指定时间的对象

//在构造函数传递一个表示时间的字符串，来指定时间

//年份全都写四位，避免歧义

//字符串格式：月份/日/年份空格时:分:秒

var d2 = new Date("11/23/2003 14:20:00");

## 方法

getDate()可以获取当前日期对象是几号

var date = d2.getDate();

console.log(date); //23

getDay()可以获取当前日期对象是周几

\* 返回值 0-6

\* 0 表示 周日

\* 1 表示周一

\* …

getMonth()可以获取当前日期对象的月份

\* 返回值 0-11

\* 0 表示 一月

\* 1 表示二月

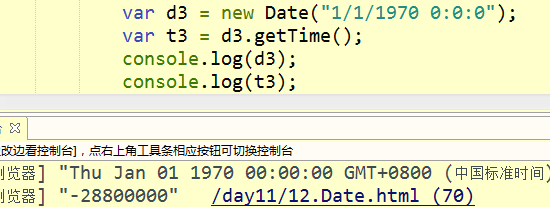
\* …

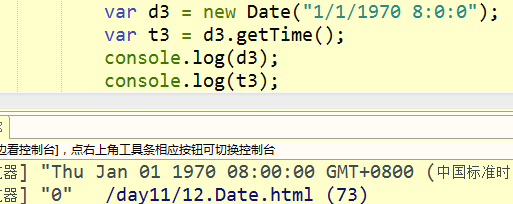
getFullYear()获取当前日期对象的年份

getTime()

\* - 获取当前日期对象的时间戳

\* - 时间戳指从格林威治标准时间1970年1月1日0时0分0秒到现在所花费的 毫秒数（由于中国在东八区，时间比标准时间早8小时 ）



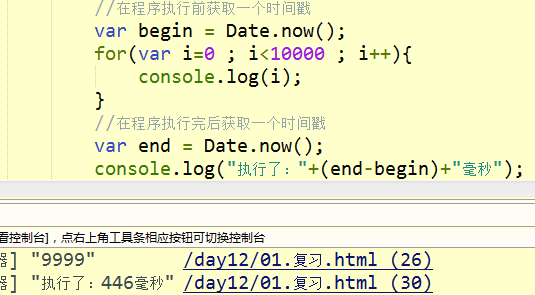


\* - 在计算机的底层，所有的时间都是以时间戳的形式保存的

Date.now()可以获取代码执行时的时间戳。now 是通过Date构造函数直接调用的，不用创建对象

console.log(Date.now());

可以利用Date.now()来计算程序执行了多少毫秒



上面446ms的时间都是输出的时间，实际上for循环的时间是很快的

