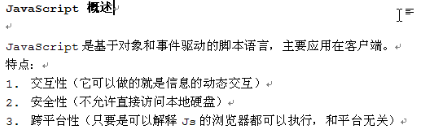
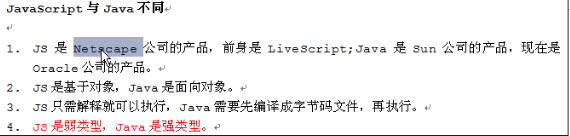
JavaScript介绍

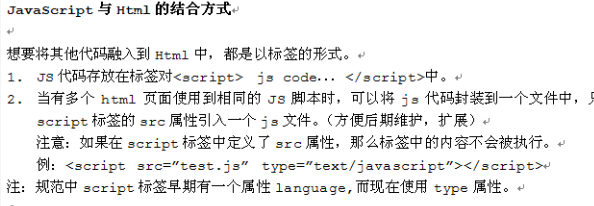
1. JavaScript也是一种程序设计语言，是**基于对象的**和**事件驱动**的脚本语言，主要应用在客户端。

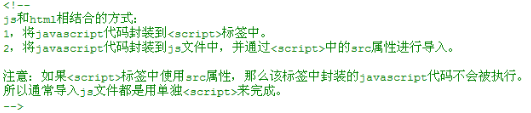


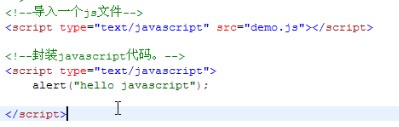
1. JavaScript和Java的不同：



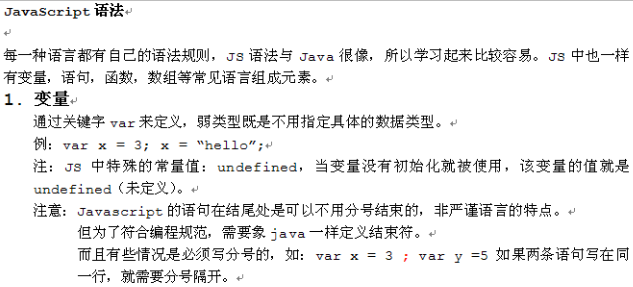
1. JavaScript和html语言的结合：

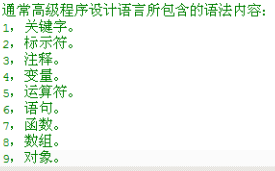






1. JavaScript语法：



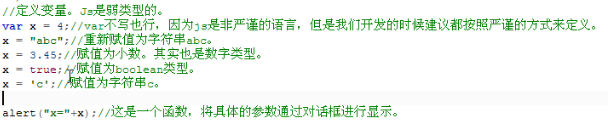


1. 关键字：被赋予了特殊意义的单词；
2. 标识符：自定义的名称，如变量名、函数名等；
3. 变量：用于标识内存中的一部分空间，用于存储数据，该空间的数据是可以变化的。 常量：
4. 运算符：数据运算符号；
5. 语句：
6. 函数：对特殊功能代码进行封装，提高代码的复用性；
7. 数组：对数据进行存储，便于操作；就是容器。
8. 对象：就是一个封装体，把函数和变量进行封装。
9. JavaScript语法的具体体现：
10. 定义变量：JS利用关键字var 定义变量；弱类型的：x什么数据都可以装。

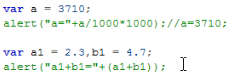
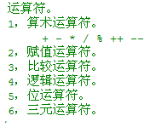
var x = 4; x = “abc”;

js是非严谨的，不写关键字var也可以。

Js中单引号和双引号都是代表字符串，都可以相互嵌套。



1. 运算符：



弱类型：3710除以1000得3.71 （java中得到3），2.3 加上4.7 等于7不是7.0











在JS中**单与&、单或|** 只是位操作（位与、位或），与java中不一样；

还有取反 ~



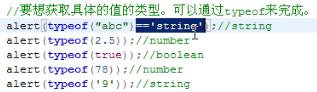


一些细节：

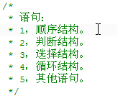
* 1. Undefined 未定义，它就是一个常量，如同null。

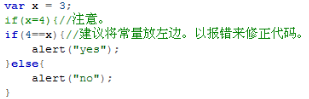


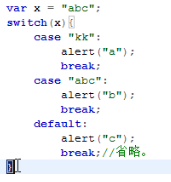
* 1. 获取变量的类型：



1. 语句：

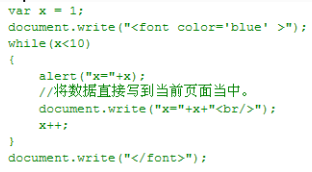






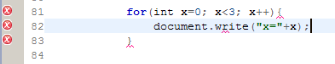
无论default语句放在哪里，都是执行完所有的case语句后，才执行default语句。

循环结构：





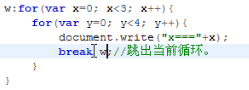
注意一定用var定义，不要利用int ，否则出现错误，很难发现。



1. Break语句和continue语句：

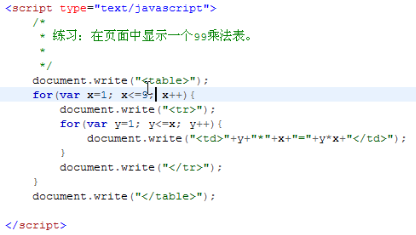
Break：跳出选择语句和循环语句；

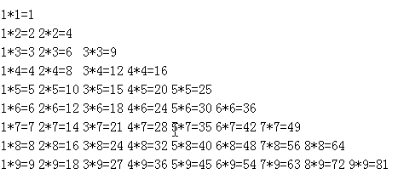
Continue：只用于循环语句，用于结束本次循环，继续执行下次循环。



利用标号，可以跳出外循环。

1. 练习题：在页面显示一个9X9乘法表。





1. JavaScript中的数组：
2. 数组 ：用于存储更多的数据，是一个容器。
3. 定义方法：有两种定义方法：
   1. var arr = []; var arr = [6,4,34,3]; //定义并初始化
   2. **利用JavaScript中的Array对象来完成定义；**

**var array = new Array()；**//等价于 var arr = [];

var array = new Array(5)；//定义一个长度为5的数组

var array = new Array(5,6,7)；//定义数组，并初始化

1. 数组类型：object



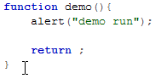
1. Java中数组只有一个属性length，JavaScript中数组具有很多属性。
2. **JavaScript的数组特点：**
   1. **长度是可变的；**
   2. **元素的类型是任意的；建议：在使用数组时，存储同一类型的数据，这样操作起来比较方便。**
3. JavaScript中的函数：就是一个功能的封装体。

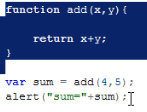
定义功能通常需要两个明确：

* 1. 功能的结果；
  2. 功能实现中参与运算的未知的内容；

定义函数的格式：（不需要定义返回值类型）





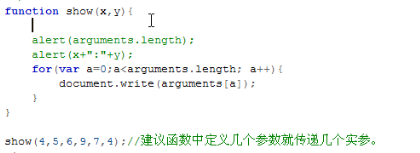
特别地，定义函数时，形参不需要定义类型var。

函数中的细节1 ：

JavaScript中的重载性，与java中的不一样，**JavaScript中只要是使用了函数名，就是调用了函数**，调用时实参和形参的个数可以不一致，数据类型只有var，所以数据类型任意传入。**但是，建议函数中定义几个形参，就传入几个实参，保持一致。**

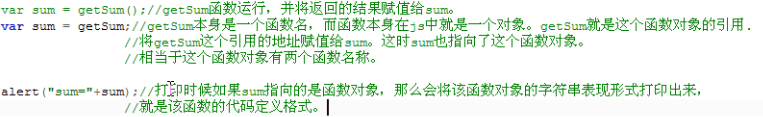
函数中的细节2：

函数中有一个数组在对传入的参数进行存储，这个数组就是arguments。

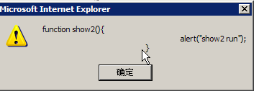


调用后就会输出 4,5,6,9,7,4

**调用函数要仔细了:**





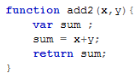


JavaScript中函数的其他表现形式：

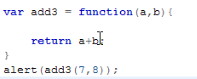
1. 动态函数：使用的JS中内置的一个对象Function对象；使用的不是很多。



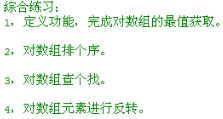
把参数列表和函数体通过字符串的形式穿进去，从而创建一个函数对象。



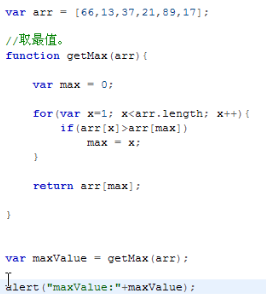
1. **匿名函数**:没有名字的函数，通常是函数的简写形式；



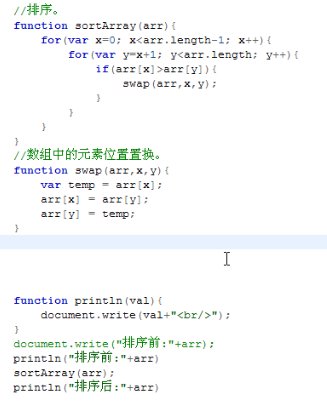
1. 综合练习题：



1. 获取数组元素中的最值：



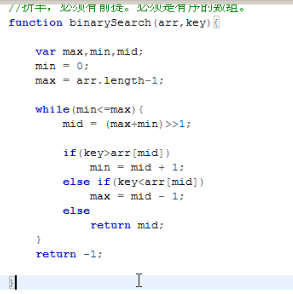
1. 对数组排序：



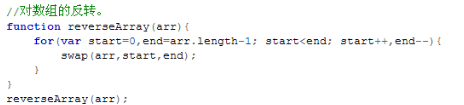


1. 数组查找：

二分查找：前提必须是有序数组。



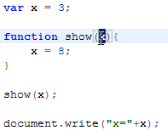
1. 对数组元素进行翻转：



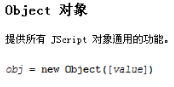
1. 全局变量和局部变量：

在函数体内定义的变量是局部变量，在脚本片段中定义的变量是全局变量，如for循环中定义的是全局变量。

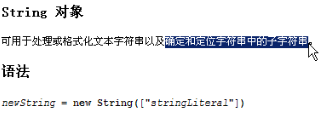


 **输出x = 3**

1. JavaScript中的常见对象：
2. Object对象

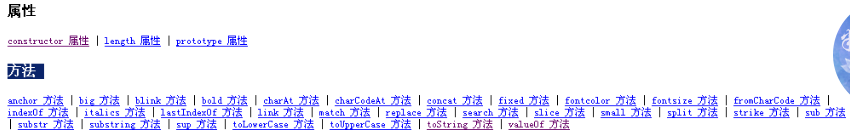


1. String 对象



一定是用var，不要使用string。





基本方法演示：



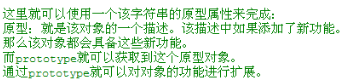
自定义方法：

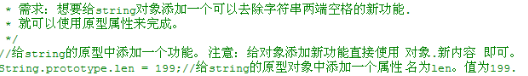


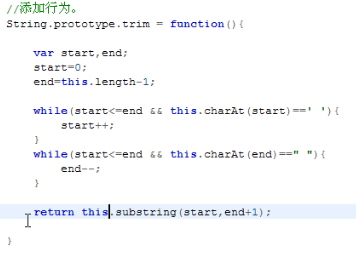
**JavaScript的原型属性：prototype属性。**





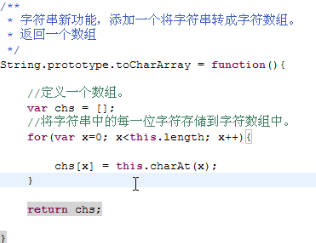




原型练习：



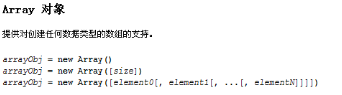
* + 1. 练习1



* + 1. 练习2：JavaScript中在函数内部可以定义对另外一个函数进行封装，java中是不允许的。



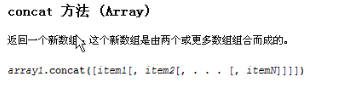
1. Array对象：
   1. Array对象：



* 1. 属性和方法:

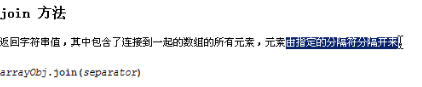


**concat方法：**

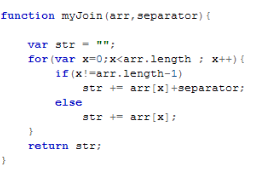




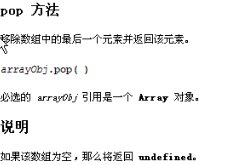
**Join 方法：**



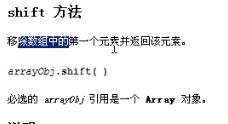
模拟join方法实现原理：

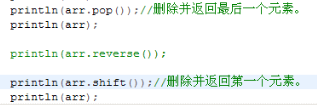


**pop方法：**



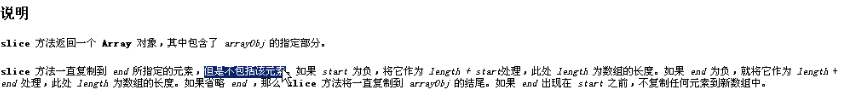
**shift方法：**





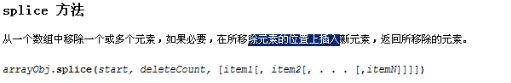
slice方法：



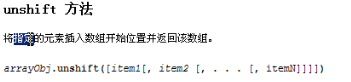




**splice方法：**

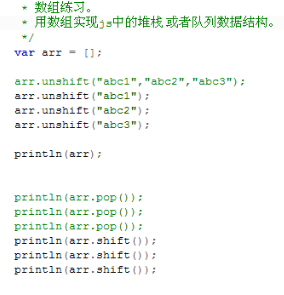


**unshift方法：**



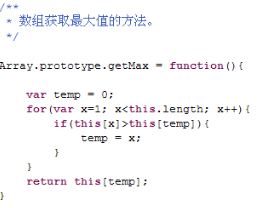


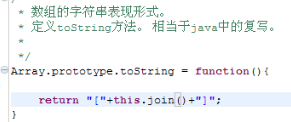
* 1. Array对象的练习：堆栈和队列结构。



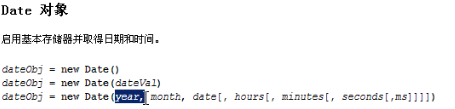
* 1. 数组的自定义功能：

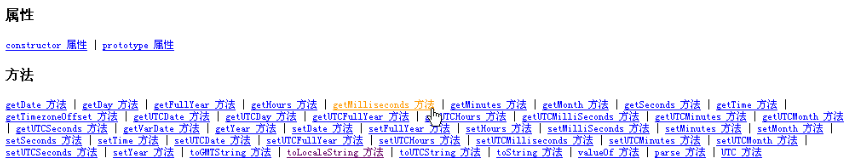






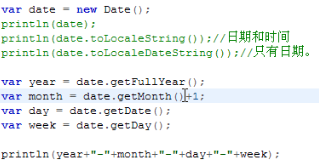
1. 类Date：

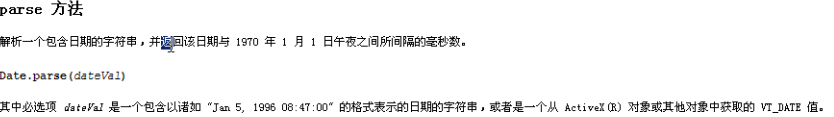


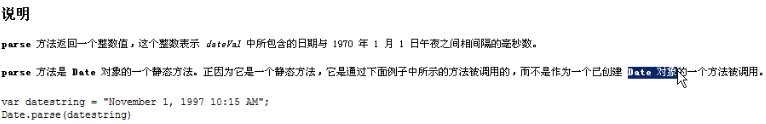


**month利用0-11表示1-12月份。**

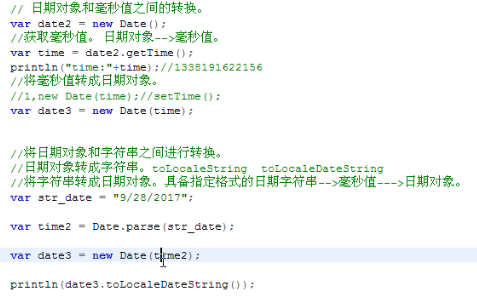












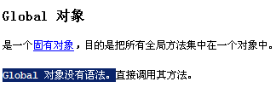
1. Math：



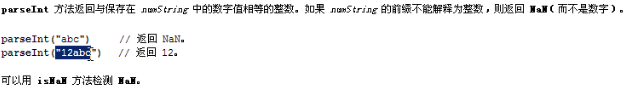


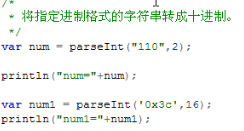


1. Global对象：其中的方法称为**全局方法**。

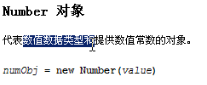


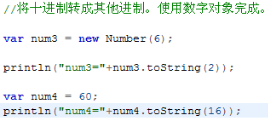


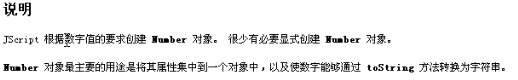




1. Number类：







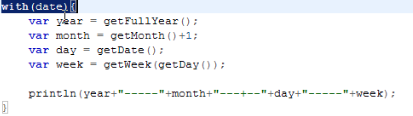
1. **JavaScript中的特有语句：**



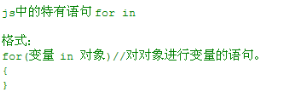
1. with

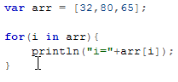


如：

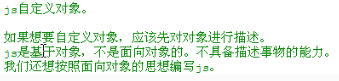


1. for in



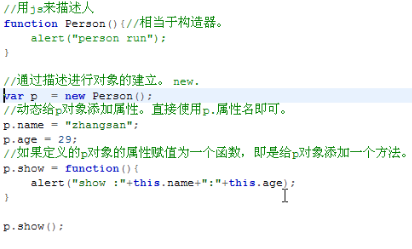


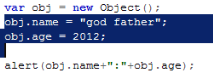
1. **JavaScript的自定义对象：**



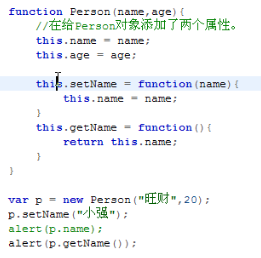


**实例：**



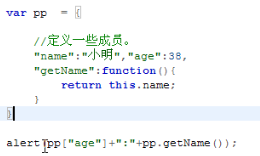


1. **先描述，后调用：**



1. **采用类似键值对的方式进行封装数据：**





1. **调用成员：**



**利用for in 调用对象的所有的属性值：**



1. **Map：**

