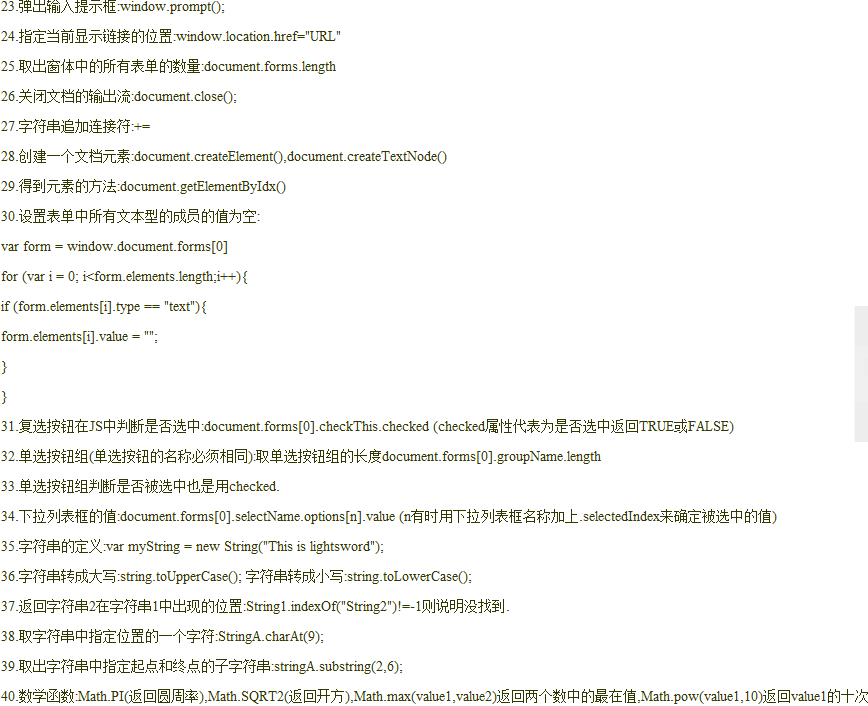
课程内容：

1、基础归纳：

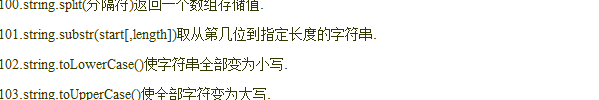












2、基础对象

·数组 Array

·日期 Date

·数学 Math

·字符串 String

**数组**

**1、数组的创建**

var name= new Array();　//创建一个数组

name[0]="zhangsan";   //给数组赋值

name[1]="lisi";

var arrayObj = new Array([size]);　//创建一个数组并指定长度，注意不是上限，是长度

var name=["zhangsan","lisi"];       //创建一个数组并赋值

var name=new Array("zhangsan","lisi");

    要说明的是，虽然第二种方法创建数组指定了长度，但实际上所有情况下数组都是变长的，也就是说即使指定了长度为5，仍然可以将元素存储在规定长度以外的，注意：这时长度会随之改变。

**2、数组的元素的访问**

var testGetArrValue=arrayObj[1]; //获取数组的元素值

arrayObj[1]= "这是新值"; //给数组元素赋予新的值

**3、数组元素的添加**

arrayObj. push([item1 [item2 [. . . [itemN ]]]]);// 将一个或多个新元素添加到数组结尾，并返回数组新长度

arrayObj.unshift([item1 [item2 [. . . [itemN ]]]]);// 将一个或多个新元素添加到数组开始，数组中的元素自动后移，返回数组新长度

arrayObj.splice(insertPos,0,[item1[, item2[, . . . [,itemN]]]]);//将一个或多个新元素插入到数组的指定位置，插入位置的元素自动后移，返回""。

**4、数组元素的删除**

arrayObj.pop(); //移除最后一个元素并返回该元素值

arrayObj.shift(); //移除最前一个元素并返回该元素值，数组中元素自动前移

arrayObj.splice(deletePos,deleteCount); //删除从指定位置deletePos开始的指定数量deleteCount的元素，数组形式返回所移除的元素

**5、数组的截取和合并**

arrayObj.slice(start, [end]); //以数组的形式返回数组的一部分，注意不包括 end 对应的元素，如果省略 end 将复制 start 之后的所有元素

arrayObj.concat([item1[, item2[, . . . [,itemN]]]]); //将多个数组（也可以是字符串，或者是数组和字符串的混合）连接为一个数组，返回连接好的新的数组

**6、数组的拷贝**

arrayObj.slice(0); //返回数组的拷贝数组，注意是一个新的数组，不是指向

arrayObj.concat(); //返回数组的拷贝数组，注意是一个新的数组，不是指向

**7、数组元素的排序**

arrayObj.reverse(); //反转元素（最前的排到最后、最后的排到最前），返回数组地址

arrayObj.sort(); //对数组元素排序，返回数组地址

**8、数组元素的字符串化**

arrayObj.join(separator); //返回字符串，这个字符串将数组的每一个元素值连接在一起，中间用 separator 隔开。

toLocaleString 、toString 、valueOf：可以看作是join的特殊用法，不常用

**二、数组对象的3个属性**

**1、length 属性**

    Length属性表示数组的长度，即其中元素的个数。因为数组的索引总是由0开始，所以一个数组的上下限分别是：0和length-1。和其他大多数语言 不同的是，JavaScript数组的length属性是可变的，这一点需要特别注意。当length属性被设置得更大时，整个数组的状态事实上不会发生 变化，仅仅是length属性变大；当length属性被设置得比原来小时，则原先数组中索引大于或等于length的元素的值全部被丢失。下面是演示改 变length属性的例子：

var arr=[12,23,5,3,25,98,76,54,56,76];

//定义了一个包含10个数字的数组

alert(arr.length); //显示数组的长度10

arr.length=12; //增大数组的长度

alert(arr.length); //显示数组的长度已经变为12

alert(arr[8]); //显示第9个元素的值，为56

arr.length=5; //将数组的长度减少到5，索引等于或超过5的元素被丢弃

alert(arr[8]); //显示第9个元素已经变为"undefined"

arr.length=10; //将数组长度恢复为10

alert(arr[8]); //虽然长度被恢复为10，但第9个元素却无法收回，显示"undefined"

    由上面的代码我们可以清楚的看到length属性的性质。但length对象不仅可以显式的设置，它也有可能被隐式修改。JavaScript中可以使用 一个未声明过的变量，同样，也可以使用一个未定义的数组元素（指索引超过或等于length的元素），这时，length属性的值将被设置为所使用元素索 引的值加1。例如下面的代码：

var arr=[12,23,5,3,25,98,76,54,56,76];

alert(arr.length);

arr[15]=34;

alert(arr.length);

    代码中同样是先定义了一个包含10个数字的数组，通过alert语句可以看出其长度为10。随后使用了索引为15的元素，将其赋值为15，即 arr[15]=34，这时再用alert语句输出数组的长度，得到的是16。无论如何，对于习惯于强类型编程的开发人员来说，这是一个很令人惊讶的特 性。事实上，使用new Array()形式创建的数组，其初始长度就是为0，正是对其中未定义元素的操作，才使数组的长度发生变化。

    由上面的介绍可以看到，length属性是如此的神奇，利用它可以方便的增加或者减少数组的容量。因此对length属性的深入了解，有助于在开发过程中灵活运用。

**2、prototype 属性**

返回对象类型原型的引用。prototype 属性是 object 共有的。

objectName.prototype

objectName 参数是object对象的名称。

说明：用 prototype 属性提供对象的类的一组基本功能。 对象的新实例“继承”赋予该对象原型的操作。

    对于数组对象，以以下例子说明prototype 属性的用途。

    给数组对象添加返回数组中最大元素值的方法。要完成这一点，声明一个函数，将它加入 Array.prototype， 并使用它。

function array\_max( )

{

   var i, max = this[0];

   for (i = 1; i < this.length; i++)

   {

   if (max < this[i])

   max = this[i];

   }

   return max;

}

Array.prototype.max = array\_max;

var x = new Array(1, 2, 3, 4, 5, 6);

var y = x.max( );

该代码执行后，y 保存数组 x 中的最大值，或说 6。

**3、constructor 属性**

表示创建对象的函数。

object.constructor //object是对象或函数的名称。

说明：constructor 属性是所有具有 prototype 的对象的成员。它们包括除 Global 和 Math 对象以外的所有 JScript 固有对象。constructor 属性保存了对构造特定对象实例的函数的引用。

例如：

x = new String("Hi");

if (x.constructor == String) // 进行处理（条件为真）。

或

function MyFunc {

// 函数体。

}

y = new MyFunc;

if (y.constructor == MyFunc) // 进行处理（条件为真）。

对于数组来说：

y = new Array();

**日期对象**

js中提供了Date对象专门用于操作日期和时间。若想操作日期和时间

必须首先创建Date对象实例。

如  var objDate = new Date(); 还有多种形式

如：

dateObj = new Date()

dateObj = new Date(dateVal)  //dateValw为unix时间戳即表示指定日期与1970年1月1日午夜间全球标准时间的毫秒数

dateObj = new Date(year, month,day,hours,minutes,seconds)

dateObj = new Date(year, month,day)

然后就是获取日期时间，获取日期时间有很多方法，下面php点点通将为你一一介绍：

getFullYear()  返回 Date 对象中用本地时间表示的年份值

getMonth()   返回 Date 对象中用本地时间表示的月份值。

注意： 月份为0（表示1月份）~11

getDate()  返回 Date 对象中用本地时间表示的天数。

getDay()   返回 Date 对象中所表示日期的星期数

getHour()  返回 Date 对象中用本地时间表示的小时值。

getMinutes()  返回 Date 对象中用本地时间表示的分钟值。

getSeconds()   返回 Date 对象中用本地时间表示的秒数。

getMilliseconds()  返回 Date 对象中用本地时间表示的毫秒值。

当然js也能获取UTC（全球标准时间） 表示的日期。因为不常用就不介绍了！

js的Date对象被创建了以后所表示的日期和时间不是不变的，Date对象提供了一套方法进行人为设置。

下面来一一介绍：

setFullYear()  设置 Date 对象表示的年份值

setMonth()   设置 Date 对象表示的月份值。注意： 月份为0（表示1月份）~11

setDate()  设置 Date 对象表示的天数。

setDay()   设置 Date 对象所表示日期的星期数

setHours()  设置 Date 对象表示的小时值。

setMinutes()  设置 Date 对象表示的分钟值。

setSeconds()   设置 Date 对象表示的秒数。

setMilliseconds()  设置 Date 对象表示的毫秒值。

当然日期的表示是有习惯和格式的，date对象中提供了一些方法表示将日期时间转化为字符串，常用的如下：

toDateString()   //格式如："Mon Mar 26 2012"

toGMTString()   //格林威标准时间

toUTCString()  //全球标准时间

toLocaleString()  //本地时间格式

toString()  //Date对象转化为字符串

toTimeString()  //date对象的时间转化为字符串

Date对象中的其他方法：

parse()  解析一个包含日期的字符串，并返回该日期与1970年1月1日午夜之间所间隔的毫秒数。

valueOf()   返回指定对象的原始值。

UTC()  返回全球标准时间 (UTC)(或 GMT)的1970年1月1日到所指定日期之间所间隔的毫秒数。

**数学对象**

js中Math对象包含了所有这些能力，这个对象与JavaScript其他对象不同，因为用户不能生成所使用对象的副本，相反脚本汇集了简单Math对 象的属性和方法。Math对象实际上存在于每个窗口和框架中，但是对脚本没有影响，程序员把这种固定对象叫做静态对象。Math对象（开头是一个大写M） 是这些属性和方法引用的一部分，Math对象的属性通常是常数。

**Math函数与属性使用语法**

Math.方法名(参数1,参数2,...);

Math.属性;

**说明**

Math函数可以没有参数，比如Math.random()函数，或有多个参数，比如Math.max()函数

Math的相关函数全部为静态函数，应该直接调用。

**JavaScript中Math函数和属性非常多，下面按功能分类分别列出常用的几种**

**Math三角函数与属性**

* Math.sin() -- 返回数字的正弦值
* Math.cos() -- 返回数字的余弦值
* Math.tan() -- 返回数字的正切值
* Math.asin() -- 返回数字的反正弦值
* Math.acos() -- 返回数字的反余弦值
* Math.atan() -- 返回数字的反正切值
* Math.atan2() -- 返回由x轴到点(x,y)的角度(以弧度为单位)
* Math.PI 属性 -- 返回圆的周长与其直径的比值(圆周率π)，约等于3.1415926

**Math自然对数相关函数与属性**

* Math.exp() -- 返回E(自然对数的底数)的x次幂(指数)
* Math.log() -- 返回数字的自然对数
* Math.E 属性 -- 返回自然对数的底数，E约等于2.718
* Math.LN2 属性 -- 返回2的自然对数loge2，约等于0.693
* Math.LN10 属性 -- 返回10的自然对数loge2，约等于2.302
* Math.LOG2E 属性 -- 返回以2为底的E的对数log2e，约等于1.442
* Math.LOG10E 属性 -- 返回以10为底的E的对数log10e，越等于0.434

**Math四舍五入类函数**

* Math.abs() -- 返回数字的绝对值
* Math.ceil() -- 返回大于等于数字参数的最小整数(取整函数)，对数字进行上舍入
* Math.floor() -- 返回小于等于数字参数的最大整数，对数字进行下舍入
* Math.round() -- 返回数字最接近的整数，四舍五入

**Math最大最小类函数**

* Math.max() -- 返回数个数字中较大的值
* Math.min() -- 返回数个数字中较小的值

**Math幂指类函数**

* Math.pow() -- 返回底数的指定次幂
* Math.sqrt() -- 返回数字的平方根
* Math.SQRT1\_2 属性 -- 返回0.5的平方根，或2的平方根除1，约等于0.707
* Math.SQRT2 属性 -- 返回2的平方根，约等于1.414

**Math随机数函数**

* Math.random() -- 返回0和1之间的伪随机数

**字符串对象**

**一、JavaScript 中 slice 、substr 和 substring的区别：**

1: String.slice(start,end): 一个新的字符串。包括字符串 stringObject 从 start 开始（包括 start）到 end 结束（不包括 end）为止的所有字符.  
  
2: String.substring(start,end) 这个就有点特别了，它是先从start，end里找出一个较小的值. 然后从字符串的开始位置算起，截取较小值位置和较大值位置之间的

字符串,截取出来的字符串的长度为较大值与较小值之间的差。  
一个新的字符串，该字符串值包含 stringObject 的一个子字符串，其内容是从 start 处到 stop-1 处的所有字符，其长度为 stop 减 start。  
  
3: String.substr(start,end) 这个就是我们常用的从指定的位置(start)截取指定长度(end)的字符串.  
一个新的字符串，包含从 stringObject 的 start（包括 start 所指的字符） 处开始的 lenght 个字符。如果没有指定 lenght，那么返回的字符串包含从 start   
到 stringObject 的结尾的字符。  
String 对象的方法 slice()、substring() 和 substr() (不建议使用)都可返回字符串的指定部分。slice() 比 substring() 要灵活一些，因为它允许使用负数作为参数。slice() 与 substr() 有所不同，因为它用两个字符的位置来指定子串，而 substr() 则用字符位置和长度来指定子串。'

**二、String对象中常用的方法：**

**1、charCodeAt方法**返回一个整数，代表指定位置字符的Unicode[编码](http://www.cr173.com/k/bmzh/)。  
strObj.charCodeAt(index)  
说明：  
index将被处理字符的从零开始计数的编号。有效值为0到字符串长度减1的数字。  
如果指定位置没有字符，将返回NaN。  
例如：  
var str = "ABC";  
str.charCodeAt(0);  
结果：65  
**2、fromCharCode方法**从一些Unicode字符串中返回一个字符串。  
String.fromCharCode([code1[,code2...]])  
说明：  
code1，code2...是要转换为字符串的Unicode字符串序列。如果没有参数，结果为空字符串。  
例如：  
String.fromCharCode(65,66,112);  
结果：ABp  
**3、charAt方法**返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串。  
strObj.charAt(index)  
说明：  
index想得到的字符的基于零的索引。有效值是0与字符串长度减一之间的值。  
例如：  
var str = "ABC";  
str.charAt(1);  
结果：B  
**4、slice方法**返回字符串的片段。  
strObj.slice(start[,end])  
说明：  
start下标从0开始的strObj指定部分其实索引。如果start为负，将它作为length+start处理，此处length为字符串的长度。  
end小标从0开始的strObj指定部分结束索引。如果end为负，将它作为length+end处理，此处length为字符串的长度。  
例如：  
012345  
var str = "ABCDEF";  
str.slice(2,4);  
结果：CD  
**5、substring方法**返回位于String对象中指定位置的子字符串。  
strObj.substring(start,end)  
说明：  
start指明子字符串的起始位置，该索引从0开始起算。  
end指明子字符串的结束位置，该索引从0开始起算。  
substring方法使用start和end两者中的较小值作为子字符串的起始点。如果start或end为NaN或者为负数，那么将其替换为0。  
例如：  
012345  
var str = "ABCDEF";  
str.substring(2,4); // 或 str.substring(4,2);  
结果：CD  
**6、substr方法**返回一个从指定位置开始的指定长度的子字符串。  
strObj.substr(start[,length])  
说明：  
start所需的子字符串的起始位置。字符串中的第一个字符的索引为0。  
length在返回的子字符串中应包括的字符个数。  
例如：  
012345  
var str = "ABCDEF";  
str.substr(2,4);  
结果：CDEF  
**7、indexOf方法**返回String对象内第一次出现子字符串位置。如果没有找到子字符串，则返回-1。  
strObj.indexOf(substr[,startIndex])  
说明：  
substr要在String对象中查找的子字符串。  
startIndex该整数值指出在String对象内开始查找的索引。如果省略，则从字符串的开始处查找。  
例如：  
01234567  
var str = "ABCDECDF";  
str.indexOf("CD"，1); // 由1位置从左向右查找 123...  
结果：2  
**8、lastIndexOf方法**返回String对象中字符串最后出现的位置。如果没有匹配到子字符串，则返回-1。  
strObj.lastIndexOf(substr[,startindex])  
说明：  
substr要在String对象内查找的子字符串。  
startindex该整数值指出在String对象内进行查找的开始索引位置。如果省略，则查找从字符串的末尾开始。  
例如：  
01234567  
var str = "ABCDECDF";  
str.lastIndexOf("CD",6); // 由6位置从右向左查找 ...456  
结果：5  
**9、search方法返回**与正则表达式查找内容匹配的第一个字符串的位置。  
strObj.search(reExp)  
说明：  
reExp包含正则表达式模式和可用标志的正则表达式对象。  
例如：  
var str = "ABCDECDF";  
str.search("CD"); // 或 str.search(/CD/i);  
结果：2  
**10、concat方法返回字符串值**，该值包含了两个或多个提供的字符串的连接。  
str.concat([string1[,string2...]])  
说明：  
string1，string2要和所有其他指定的字符串进行连接的String对象或文字。  
例如：  
var str = "ABCDEF";  
str.concat("ABCDEF","ABC");  
结果：ABCDEFABCDEFABC  
**11、将一个字符串分割为子字符串**，然后将结果作为字符串数组返回。  
strObj.split([separator[,limit]])  
说明：  
separator字符串或 正则表达式 对象，它标识了分隔字符串时使用的是一个还是多个字符。如果忽略该选项，返回包含整个字符串的单一元素数组。  
limit该值用来限制返回数组中的元素个数。  
例如：  
var str = "AA BB CC DD EE FF";  
alert(str.split(" "，3));  
结果：  
AA,BB,CC  
**12、toLowerCase方法**返回一个字符串，该字符串中的字母被转换成小写。  
例如：  
var str = "ABCabc";  
str.toLowerCase();  
结果：abcabc  
**13、toUpperCase方法返回一个字符串**，该字符串中的所有字母都被转换为大写字母。  
例如：  
var str = "ABCabc";  
str.toUpperCase();

结果：ABCABC

虽然js String对象已经提供像slice、replace、indexOf和substring等方法,但在实际项目应用中会对其进行扩展，以达到实用、方便目的。注释很详细，废话少说，代码如下：

**三、String对象方法扩展：**  
/\*\*  
\*字符串-格式化  
\*/  
String.prototype.format = function(){  
var args = arguments;//获取函数传递参数数组,以便在replace回调函数内使用  
var regex = /\{(\d+)\}/g;//匹配并捕获所有 形如:{数字} 字串  
return this.replace(regex,function(m,i){//参数=匹配子串+第几次匹配+匹配字串位置+源字符串  
return args[i];  
});  
}  
/\*\*  
\*字符串-去掉前后空白字符  
\*/  
String.prototype.trim = function(){  
return this.replace(/(^\s\*)|(\s\*$)/g, "");  
}  
/\*\*  
\*字符串-去掉前空白字符  
\*/  
String.prototype.ltrim = function(){  
return this.replace(/(^\s\*)/g, "");  
}  
/\*\*  
\*字符串-去掉后空白字符  
\*/  
String.prototype.rtrim = function(){  
return this.replace(/(\s\*$)/g, "");  
}  
/\*\*  
\*字符串-获取以ASCII编码字节数 英文占1字节 中文占2字节  
\*/  
String.prototype.lenASCII=function(){  
return this.replace(/[^\x00-\xff]/g,'xx').length;//将所有非\x00-\xff字符换为xx两个字符,再计算字符串  
}  
/\*\*  
\*字符串-获取以UNICODE编码字节数 一个字符均占2个字节  
\*/  
String.prototype.lenUNICODE=function(){  
return this.length\*2;  
}  
ps：若对js提供类型对象或自定义对象进行方法扩展，应利用原型prototype这个对象属性进行扩展，具体方式以下:  
  
String.prototype.trim=function(){  
//...代码略  
};  
String.prototype.ltrim=function(){  
//...代码略  
};

* 1、concat  
  将两个或多个字符的文本组合起来，返回一个新的字符串。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar a = "hello";
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar b = ",world";
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar c = a.concat(b);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifalert(c);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//c = "hello,world"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 2、indexOf  
  返回字符串中一个子串第一处出现的索引（从左到右搜索）。如果没有匹配项，返回 -1 。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar index1 = a.indexOf("l");
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//index1 = 2
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar index2 = a.indexOf("l",3);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//index2 = 3
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 3、charAt  
  返回指定位置的字符。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar get\_char = a.charAt(0);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//get\_char = "h"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 4、lastIndexOf  
  返回字符串中一个子串最后一处出现的索引（从右到左搜索），如果没有匹配项，返回 -1 。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar index1 = lastIndexOf('l');
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//index1 = 3
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar index2 = lastIndexOf('l',2)
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//index2 = 2
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 5、match  
  检查一个字符串匹配一个正则表达式内容，如果么有匹配返回 null。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar re = new RegExp(/^\\w+$/);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar is\_alpha1 = a.match(re);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//is\_alpha1 = "hello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar is\_alpha2 = b.match(re);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//is\_alpha2 = null
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 6、substring  
  返回字符串的一个子串，传入参数是起始位置和结束位置。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar sub\_string1 = a.substring(1);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//sub\_string1 = "ello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar sub\_string2 = a.substring(1,4);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//sub\_string2 = "ell"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 7、substr  
  返回字符串的一个子串，传入参数是起始位置和长度
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar sub\_string1 = a.substr(1);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//sub\_string1 = "ello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar sub\_string2 = a.substr(1,4);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//sub\_string2 = "ello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 8、replace  
  用来查找匹配一个正则表达式的字符串，然后使用新字符串代替匹配的字符串。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar result1 = a.replace(re,"Hello");
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//result1 = "Hello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar result2 = b.replace(re,"Hello");
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//result2 = ",world"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 9、search  
  执行一个正则表达式匹配查找。如果查找成功，返回字符串中匹配的索引值。否则返回 -1 。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar index1 = a.search(re);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//index1 = 0
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar index2 = b.search(re);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//index2 = -1
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 10、slice  
  提取字符串的一部分，并返回一个新字符串（与 substring 相同）。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar sub\_string1 = a.slice(1);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//sub\_string1 = "ello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar sub\_string2 = a.slice(1,4);
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//sub\_string2 = "ell"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 11、split  
  通过将字符串划分成子串，将一个字符串做成一个字符串数组。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar arr1 = a.split("");
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//arr1 = [h,e,l,l,o]
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 12、length  
  返回字符串的长度，所谓字符串的长度是指其包含的字符的个数。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar len = a.length();
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//len = 5
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 13、toLowerCase  
  将整个字符串转成小写字母。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar lower\_string = a.toLowerCase();
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//lower\_string = "hello"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* 14、toUpperCase  
  将整个字符串转成大写字母。
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifvar upper\_string = a.toUpperCase();
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif//upper\_string = "HELLO"
* http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif