第7章 WMLScript标准库规范

7.1 范围

协议（Wireless Application Protocol，WAP）是WAP论坛经过不断努力得到的成果，它提供了一个业界技术规范，以便开发出适用于各种无线通信网络的应用和业务。 WAP论坛的工作范围就是为各种业务和应用制定一系列的技术规范。无线市场正在快速增长，新的用户不断增多，新的业务不断涌现。为了使运营商和生产者能够从容的面对先进业务、多种类业务和快速灵活的业务生成等诸多的挑战， WAP规定了一系列传输层、会话层和应用层协议。有关WAP体系结构更多的信息，请参阅“无线应用协议体系结构规范” (Wireless Application Protocol Architecture Specification) [WAP]。

本规范定义了WMLScript [WMLScript]标准函数库的接口，这些接口提供了对 WAP客户端核心功能的存取。 WMLScript是一种能够提供编程功能的语言，可以用在基于 WAP的应用开发之中，它是WAP开发平台的一部分，可用于向客户端增加脚本支持。

ECMAScript [ECMA262]和WMLScript的主要区别之一是：在送到客户端之前，WMLScript被编译成字节码，用这种方法，可以使的窄带通信信道得到最佳的利用，客户端需要的内存量最小。由于这样的原因， JavaScript语言许多先进的特性被移走，以使语言更精炼，易于编译成比特码，并且容易学习。

为使WMLScript更有效，WMLScript加入了对库的支持，以代替从 ECMAScript中移走的某些功能。这种特性提供了访问内部函数的功能，同时还为将来提供了不需要更多开销的扩展方法。

下面的章节对函数库进行了描述，这些函数库提供了访问 WAP客户端核心函数的功能。除了浮点库之外，所有库都用在客户端的脚本环境之中。 浮点库不是必需的，它只支持能进行浮点运算的客户端。

7.2 符号说明

本规范用下列信息描述函数库：

名　称	库的名字。库的命名格式遵循 [WMLScript] 规范中的定义。 例如：Lang，String
库标识符(ID)	WMLScript编译器使用专门为这些库保留的数字标识符，标识符的取值范围分为两段。
0～32767	用于标准库。
32768～65535	留给将来使用。
描述	函数库及其使用约定的简单描述。

本规范用下列信息描述库函数：

函数	规定函数名及其参数的个数。 函数命名格式遵循 [WMLScript] 规范中的

定义。

函数名区分大小写字母。

例如：abs (*value*)

用法：var a = 3*Lang.abs(length)；

函数标识符(ID)	WMLScript编译器使用专为函数保留的数字标识符。这些标识符的取值范围为0～255。
描述	描述函数的特性及参数。
参数	指出函数参数的类型。 例如：value = Number
返回值	指出返回值的类型。 例:如：String或invalid
特例	描述可能出现的特殊情况、错误代码以及相应的返回值。对于所有函数都可能出现的标准错误，不在这里描述（关于错误处理的更多信息，见7.3.3 节）。 例如：若*value1*≤0，*value*2<0且不是整数，则返回invalid。
示例	给出例子，说明函数的使用方法。 var a =-3； var b =Lang.abs (a)； // b = 3

7.3 WMLScript的顺从性

WMLScript标准库函数提供了一个扩展 WMLScript语言的机制,因此，定义的库函数必须遵循WMLScript的习惯和规则。

7.3.1 支持的数据类型

下面是函数定义中用到的 WMLScript [WMLScript] 类型。它们既可以表示函数参数的类型，也可以表示返回值的类型，包括：布尔型（ Boolean)、整型（Integer)、浮点型（Float)、字符型 （String）和无效型（Invalid)。

此外，当既可以接受 整数参数又可以接受浮点参数时，参数类型可以用数字（ Number）一词代替，如果支持所有的类型，可以用任意（ Any）一词代替。

7.3.2 数据类型转换

因为 WMLScript是一种弱类型划分语言，如果需要（关于数据类型的转换规则，见[WMLScript])，各种数据类型会自动地进行转换。除非特别声明，否则库函数遵循WMLScript操作符的数据类型转换规则。

7.3.3 错误处理

错误事件的处理方法与 WMLScript语言中使用的方法相同（更详细的内容参见[WMLScript])，即：

- 除非特别声明，否则非法的函数变量会产生一个无效的返回值，这个值不受其他的因素影响。
- 一个函数变量，如果无法把他转换成所需的参数，就会产生一个无效的返回值，这个值不受其他的因素影响。有关数据类型转换的更多信息，见 7.3.2节。
- 利用在每个函数中定义的有关错误返回代码，完成对与函数相关的错误事件的处理。这些错误在函数规范（特例部分）中作了说明。

7.3.4 对整型设备的支持

WMLScript语言可以在不支持浮点运算的设备上运行。虽然 WMLScript标准库中有一些需要浮点支持的运算，但对于那些仅仅支持整型运算的设备，在使用这些函数库时，可以遵循如下的规则：

- 库函数只使用这些变量类型： 布尔型 、整型、字符型和无效型。
- 忽略所有与浮点数据有关的转换规则。
- Lang.float( ) 函数返回假值。
- Lang.parseFloat( ) 函数返回无效型。
- 所有浮点(见第7.5节)库函数都返回无效型。

7.4 Lang函数库

名　称	Lang
库标识符(ID)	0
描　述	这个库包含了一组与WMLScript语言核心内容密切相关的函数。

7.4.1 abs函数

函　数	abs(value)
函数标识符(ID)	0
描　述	返回给定数的绝对值。如果给定的数是整型数，返回一个整型值；如果给定数是浮点型，则返回值也是浮点型。
参　数	*value* = Number
返 回 值	Number或invalid
特　例	—
示　例	var a = -3； var b = Lang.abs(a)； // b = 3

7.4.2 min函数

函　数	min(*value*1,*value*2)
函数标识符(ID)	1
描　述	返回给定的两个数*value*1和*value*2中取值最小的一个 ，返回值和返回类型与选中的数相同。选择方法如下： • 在比较过程中，利用WMLScript的整型和浮点型操作符的数据类型

转换规则，确定最终的数据类型（整型或浮点型）。
• 比较数值并选择较小的一个。
• 如果两个数的值相等，选择第一个数。

参数 *value*1 = Number
*value*2 = Number

返 回 值 Number**或**invalid

特例 **—**

示例

```
var a = -3;
var b = Lang.abs(a);                // c=-3
var d = Lang.min(45, 76.3);         // d = 45 (整型)
var e = Lang.min(45, 45.0);         // e = 45 (整型)
```

7.4.3 max函数

函数 max(*value*1,*value*2)

函数标识符(ID) 2

描述 **返回给定的两个数** value1**和**value2**中取值最大的一个 ，返回值和返回类型与选中的数相同。选择方法如下。**
• 在比较过程中，利用 WMLScript**的整型和浮点型操作符的数据类型转换规则，确定最终的数据类型（整型或浮点型）。**
• 比较数值并选择较大的一个。
• 如果两个数的值相等，选择第一个数。

参数 *value*1 = Number
*value*2 = Number

返 回 值 Number**或**invalid

特例 **—**

示例

```
var a =-3;
var b = Lang.abs(a);
var c = Lang.max (a, b);      // c = 3
var d = Lang. max (45.5, 76); // d = 76 (整数)
var e = Lang. max (45.0, 45); // e = 45.0 (浮点数)
```

7.4.4 parseInt函数

函数 parseInt(*value*)

函数标识符(ID) 3

描述 **返回由该字符串值定义的的整型值。合法整型数的语法结构见**[WMLScript]。WMLScript**规定了十进制整数字符串的检查规则，其分析检查规则如下：**
当第一个非‘ + ’字符，或非‘ - ’字符，亦或非十进制数字字符出现

时，分析检查过程停止。
计算结果是从字符串中分析得到的一个整数值。

参　数	*value* = String
返 回 值	Number或invalid
特　例	当分析出现错误时，返回无效值（invalid）。

示　例	var i = Lang.parseInt(“1234”); // i = 1234
	var j = Lang.parseInt(“100 m/s”); // j = 100

7.4.5 parseFloat函数

函数	parseFloat(*value*)
函数标识符(ID)	4
描述	返回由该字符串定义的浮点型数值。合法浮点型的语法结构见[WMLScript]。WMLScript规定了十进制浮点数字字符串的分析规则，其分析检查规则如下： 当第一个不符合浮点表示方法的字符出现时，分析检查过程停止。 计算结果是从字符串中分析得到的一个浮点数。
参数	*value* =String
返回值	Floating-point或invalid
特例	当分析出现错误时，返回无效值；如果系统不支持浮点运算，也返回无效值。

示例	var a = Lang.parseFloat(“123.7”); // a = 123.7
	var b = Lang.parseFloat(“+7.34e2 Hz”); // b = 7.34e2
	var c = Lang.parseFloat(“70e-2 F “); // c = 70.0e-2
	var d = Lang.parseFloat(“- .1 C “); // d = -0.1
	var e = Lang.parseFloat(“100”); // e = 100.0
	var f = Lang.parseFloat(“Number: 5.5”); // f = invalid
	var g = Lang.parseFloat(“7.3e meters”); // g = invalid
	var h = Lang.parseFloat(“7.3e- m/s”); // h = invalid

7.4.6 isInt函数

函数	isInt(*value*)
函数标识符(ID)	5
描述	如果所给的值*value*可以通过parseInt(*value*) 函数转换成整型，则返回布尔型的true值，否则返回false值。
参数	*value* = Any
返回值	Boolean或invalid
特例	—

实例	var a = Lang.isInt(“-123”); // ture
	var b = Lang.isInt(“123.33”); // ture

```
var c = Lang.isInt(“ string ”);   // false
var d = Lang,isInt(“ #123 ”);     // false
var e = Lang ,isInt(invalid);     // invalid
```

7.4.7 isFloat函数

函数	isFloat(*value*)
函数标识符(ID)	6
描述	**如果所给的值***value***可以通过**parseFloat(*value*) **转换成浮点型，则返回布尔型的**true**值，否则返回**false**值。**
参数	*value* = Any
返回值	Boolean**或**invalid
特例	**如果系统不支持浮点运算，返回**invalid**。**

实例	var a = Lang.isFloat(“ -123 ”); // ture
	var b = Lang.isFloat(“ 123.33 ”); // ture
	var c = Lang.isFloat(“ string ”); // false
	var d = Lang,isFloat(“ #123.33 ”); // false
	var e = Lang.isFloat(invalid); // invalid

7.4.8 maxInt函数

函数	maxInt( )
函数标识符(ID)	7
描述	**返回最大的整数值。**
参数	**—**
返回值	**整数**2147483647
特例	**—**
示例	var a = Lang.maxInt( );

7.4.9 minInt函数

函数	minInt( )
函数标识符(ID)	8
描述	**返回最小的整数值。**
参数	**—**
返回值	**整数-**2147483648
特例	**—**
实例	var a = Lang.minInt( );

7.4.10 float函数

函数	float( )
函数标识符(ID)	9

描述	支持浮点型时，返回true值，否则返回false值。
参数	—
返回值	Boolean
特例	—
示例	var floatSupported = Lang.float( );

7.4.11 exit函数

函数	exit(*value*)
函数标识符(ID)	10
描述	停止WMLScript字节码的翻译，返回由 *value*指定的内容，并把控制权交给WMLScript翻译器的调用程序。在需要停止 WMLScript字节码执行的情况下，用这个函数可以实现正常的退出。
参数	*value* = Any
返回值	无 (此函数停止了翻译过程)
特例	—

实例	Lang.exit(“Value:” + myVal); // 返回一个字符串
	Lang.exit(invalid); // 返回无效型

7.4.12 abort函数

函数	abort(*errorDescription*)
函数标识符(ID)	11
描述	放弃WMLScript字节码的翻译，返回错误描述（errorDescription），并把控制权交回给WMLScript翻译器的调用程序。如果调用函数检测到了严重的错误，不得不停止 WMLScript比特码的执行，可以用这个函数实现非正常退出。如果 errorDescription的类型是无效型，用字符串“invalid”代替errorDescription。
参数	*errorDescription* = String
返回值	无 (此函数停止翻译过程)
特例	—
示例	Lang.abort(“Error:” +errVal); // 错误值是一个字符串

7.4.13 random函数

函数	random(*value*)
函数标识符(ID)	12
描述	返回一个正整数，这个数大于或等于零，但是小于或等于所给的 *value*值。返回值是在上面提到的范围内，随机或伪随机选择的一个近似于均匀分布的随机数。 如果*value*是浮点型的，首先调用函数Float.int( )，计算真正的整数值。
参数	*value* = Integer

返回值 Integer**或**invalid
特例 **如果***value***等于**0, **函数返回**0 。
如果*value***小于**0, **函数返回**invalid。
实例
```
var a = 10;
var b = Lang.random(5.1)*a;          // b = 0.50
var c = Lang.random(“ string ”);     // c = invalid
```

7.4.14 seed函数

函 数 seed(*value*)
函数标识符(ID) 13
描 述 **初始化伪随机数字序列并返回一个空字符串。如果** *value***的值是**0**或者是正整数，使用给定的** *value***值初始化；否则使用一个随机的、与系统有关的数值初始化。**
如果value**是浮点型**, **首先调用函数**Float.int( )**，计算真正的整数。**
参数 *value* = Integer
返回值 String**或**invalid
特例 —
实例
```
Var a = Lang.seed(123);          // a = “ ”
Var b = Lang.random(20);         // b = 0.20
Var c = Lang.seed(“ seed ”);     // c＝invalid (随机函数的初始值没有改变)
```

7.5 Float函数库

名称 Float
库标识符(ID) 1
描述 **这个库包含了一组应用程序经常要用到的典型浮点运算函数。**
这些库函数是可选的，只有当设备支持浮点运算时才能运行（见7.3.4**节）。**
如果不支持浮点运算，这个库中的所有函数都将返回无效型。

7.5.1 int函数

函数 int(value)
函数标识符(ID) 0
描述 **返回给定值**value**的整数部分。如果**value**已经是整数，则返回它本身。**
参数 *value* = Integer
返回值 Integer**或**invalid
特例 —
示例
```
var a = 3.14;
var b = Float.int(a);          // b =3
var c = Float.int(-2.8);       // c =-2
```

7.5.2 floor函数

函数	floor(*value*)
函数标识符(ID)	1
描述	**返回不大于给定值**value**的最大整数。如果**value**已经是整数，则返回它本身。**
参数	*value* = Integer
返 回 值	Integer**或**invalid
特例	**—**

示例	var a = 3.14;
	var b = Float.floor(a); // b = 3
	var c = Float.floor(-2.8); // c = -3

7.5.3 ceil函数

函数	ceil(*value*)
函数标识符(ID)	2
描述	**返回不小于给定值***value***的最小整数。如果***value***已经是整数，则返回它本身。**
参数	*value* = Number
返回值	Integer**或**invalid
特例	**—**

示例	var a = 3.14;
	var b = Float.ceil(a); // b = 4
	var c = Float.ceil(-2.8); // c = -2

7.5.4 pow函数

函数	pow(*value*1,*value*2)
函数标识符(ID)	3
描述	**返回值是一个与具体实现有关的近似值，这个值是以** *value*1**为底、***value*2**为指数的幂。如果***value*1**是负数，***value*2**必须为整数。**
参数	*value*1 = Number *value*2 = Number
返 回 值	Floating-point**或**invalid
特例	**如果***value*1 = = 0**且***value*2 <0 **，返回无效型。** **如果***value*1 < 0**且***value*2**不是整数，返回无效型。**
实例	var a = 3; var b = Float.pow(a,2); // b = 9

7.5.5 round函数

函数	round(*value*)

函数标识符(ID) 4

描述 **返回与给定值**value**最接近的整数。如果** *value***与两个整数的接近程度相同，则取较大的数作为运算结果。如果** *value***值已经是整数，运算结果就等于它本身。**

参数 *value* = Number

返回值 Integer**或**invalid

特例 **—**

示例

```
var a = Float.round(3.5);          // a = 4
var b = Float.round(-3.5);         // b = -3
var c = Float.round(0.5);          // c = 1
var d = Float.round(-0.5);         // d = 0
```

7.5.6 sqrt函数

函数 sqrt(*value*)

函数标识符(ID) 5

描述 **返回给定数** value**平方根的近似值，这个函数的运算结果与具体的应用有关。**

参数 *value* = Floating-point

返回值 Floating-point**或**invalid

特例 **如果***value***为负数，返回无效值 。**

示例

```
var a = 4;
var b = Float.sqrt(a);             // b = 2.0
var c = Float.sqrt(5);             //c= 2.2360679775
```

7.5.7 maxFloat函数

函数 maxFloat( )

函数标识符(ID) 6

描述 **返回** [IEEE754]**标准支持的、单精度浮点格式的最大浮点数。**

参数 **—**

返 回 值 **浮点数**3.40282347E+38

特例 **—**

示例

```
var a = Float.maxFloat( );
```

7.5.8 minFloat函数

函数 minFloat( )

函数标识符(ID) 7

描述 **返回** [IEEE754]**标准支持的、单精度浮点格式的最小非零浮点值。**

参数 **—**

返回值 Floating-point**。比标准的最小单精度浮点数小或者与之相等的数是**

1.17549435E-38

特例 —

示例 var a = Float.minFloat( );

7.6 String函数库

函数 String

库标识符(ID) 2

描述 **这个库包含了一组字符串函数，一个字符串是一系列的Unicode字符构成。在这个序列中，每一个字符都有一个索引，字符串中第一个字符的索引值为0，它的长度定义成在这个序列中字符的个数。**

字符串库的使用者可以自己定义一个特殊的分隔符（ separator），用于分隔字符串中的元素。使用定义的分隔符和元素索引，可以对元素进行存取，字符串中第一个元素的索引是 0。在字符串中，分隔符每出现一次，就将原来的一个元素分成了两个元素（不允许分隔符转义）。

空白字符（whitespace character）是Unicode 2.0中的一个字符代码，这个字符代码小于或等于32（十进制数）。

7.6.1 length函数

函数 length(*string*)

函数标识符(ID) 0

描述 **返回给定字符串string的长度（Unicode字符的个数）。**

参数 *string* = String

返回值 Integer**或**invalid

特例 —

示例

```
var a = “ ABC ”;                    // b = 3
var b = String.length(a);           // c = 0
var c = String.length(“ “);         // d = 3
var d = String.length(342);
```

7.6.2 isEmpty函数

函数 isEmpty(*string*)

函数标识符(ID) 1

描述 **如果字符串 *string*的长度为0，返回布尔型的 true值，否则返回 false值。**

参数 *string* =String

返 回 值 Boolean**或**invalid

特例 —

示例 var a = “ Hello ”;

```
var b = “ ”;
var c = String.isEmpty(a);        // c = false
var d = String.isEmpty(b);        // d = true
var e = String.isEmpty(true);     // e = false
```

7.6.3 charAt函数

函数 charAt(*string*,*index*)

函数标识符(ID) 2

描述 **返回给定字符串*string*中的字符，该字符由索引**index**指出。**
如果*index*是浮点型的，需要先调用函数 Float.int( )**，计算真正的整数索引。**

参数 *string* = String
Index = Number(**返回的字符索引**)

返回值 String**或**invalid

特例 **如果索引值*index*超出了范围**, **返回一个空串（“ ”）。**

示例

```
var a = “ My name is Joe ”;
var b = String.charAt(a, 0);              // b = “ M ”
var c = String.charAt(a, 100);            // c = “ ”
var d = String.charAt(34, 0);             // d = “ 3 ”
var e = String.charAt(a, “ first ”);      // e = invalid
```

7.6.4 subString函数

函数 subString(*string*,*startIndex*,*length*)

函数标识符(ID) 3

描述 **返回一个新的字符串，这个字符串是给定字符串** *string***的子集 ，它以参数*startIndex*开始，长度（字符的个数）等于** length**。如果参数*startIndex*小于**0**，则*startIndex*取**0**值，如果**length**大于字符串中剩余的字符个数，则**length**等于字符串中剩余字符的个数。**
如果*startIndex*或*length*是浮点型的，需要先调用函数 Float.int( )**，计算真正的整数值。**

参数 *string* = String
startIndex = Number**（开始索引）**
length = Number**（新字符串的长度）**

返 回 值 String**或**invalid

特例 **如果*startIndex*大于最后一个索引值，则返回一个空串（“ ”）。**
如果*length*≤0， 也返回一个空串（“ ”）。

示例

```
var a = “ ABCD ”;
var b = String.subString(a, 1, 2);        // b = “ BC ”
var c = String.subString(a, 2, 5);        // c = “ CD ”
```

```
var d = String.subString(1234, 0, 2);    // d = "12"
```

7.6.5 find函数

函数 find(*string*,*subString*)

函数标识符(ID) 4

描述 **返回一个索引值，该索引值是在给定字符串 *string*中，与子串*subString*相匹配的那个字符串的第一个字符的索引。如果没有匹配字符串，返回整数-1。**

所谓匹配就是两个字符串完全相同。在两个字符串中，只有当所有的对应字符相同且排列顺序也相同时不区分字体，才能说这两个字符串是匹配的。

参数 *string* = String

subString = String

返回值 Integer**或者**invalid

特例 **—**

示例

```
var a = "abcde";
var b = String.find(a, "cd");          // b = 2
var c = String.find(34.2, "de");       // c = -1
var d = String.find(a, "qz");          // d = -1
var e = String.find(34, "3");          // e = 0
```

7.6.6 replace函数

函数 replace(*string*,*oldSubString*,*newSubString*)

函数标识符(ID) 5

描述 **返回一个新字符串，在这个新串中，新的子串 *newSubString*替代了字符串*string*中旧的子串*oldSubString*。**

两个字符串匹配就是这两个字符串完全相同。在两个字符串中，只有当所有的对应字符相同且排列顺序也相同时不区分字体，才能说这两个字符串是匹配的。

参数 *string* = String

oldSubString = String

newSubString =String

返回值 String**或**invalid

特例 **—**

示例

```
var a = "Hello Joe. What is up Joe?";
var newName = "Don";
var oldName = "Joe";
var c = String.replace(a, oldname, newname);
                    // c = "Hello Don. What is up Don?";
```

```
var d = String.replace(a, newname, oldname);
// d = “Hello Joe. What is up Joe?”
```

7.6.7 elements函数

函数 elements(*string*,*separator*)

函数标识符(ID) 6

描述 **返回给定的字符串***string***中被分隔符***separator***分隔之后的元素个数。**

参数 *string* = String

separator = String **（分隔符的第一个字符）**

返回值 Integer**或**invalid

特例 **如果分隔符是空字符串，则返回无效型。**

示例

```
var a = “My name is Joe; Age 50;”;
var b = String.elements(a, “ ”);                  // b = 6
var c = String.elements(a, “;”);                  // c = 3
var d = String.elements(“ ”, “;”);                // d = 0
var e = String.elements(“a”, “;”);                // e = 1
var f = String.elements(“;”, “;”);                // f = 2
var g = String.elements(“;; , ;”, “; , ”)         // g = 4  分隔符= ;;
```

7.6.8 elementAt函数

函数 elementAt(*string*,*index*,*separator*)

函数标识符(ID) 7

描述 **搜索第** *index***个元素，并返回找到的元素，其中元素由分隔符** *separator***进行了分隔。如果** *index***小于**0**，返回第一个元素；如果** *index***大于元素的个数，返回最后一个元素；如果字符串是一个空串，返回一个空串。**

如果*index***是浮点型的，需要先调用**Float.int( )**，计算真正的***index***值。**

参数 *string* = String

index = Number(**返回元素的索引**)

separator = String(**分隔符的第一个字符**)

返回值 String**或**invalid

特例 **如果分隔符是空串，则返回**invalid**。**

示例

```
var a = “My name is Joe; Age 50;”;
var b = String.elementAt(a, 0, “ ”);      // b = “My”
var c = String.elementAt(a, 14, “;”);     // c = “ ”
var d = String.elementAt(a, 1, “;”);      // d = “Age 50”
```

7.6.9 removeAt函数

函数 removeAt(*string*,*index*,*separator*)

函数标识符(ID) 8

描述 **返回一个新的字符串，这个字符串是从***string***中删除了***index***指定的元素之后剩余的部分。字符串***string***中的元素被分隔符***separator***进行了分隔。如果***index***小于**0**，删除第一个元素；如果 ***index***大于元素的个数，删除最后一个元素 ；如果字符串***string***是空串，则函数返回一个新的空串。如果***index***是浮点数，需要先调用**Float.int( )**，计算真正的整数值。**

参数 *string* = String
index = Number(**被删除元素的索引**)
separator = String(**分隔符的第一个字符**)

返回值 String**或**invalid

特例 **如果分隔符***separator***是空串，则返回无效型。**

示例
```
var a = “ A A; B C D ”;
var s = “  ”;
var b = String.removeAt(a, 1, s);          // b = “ ABC D ”
var c = String.rermoveAt(a, 0, “ ; ”);     // c = “  B C D ”
var d = String.removedAt(a, 14, “ ; ”);    // d = “ A A ”
```

7.6.10 replaceAt函数

函数 replaceAt(*string*,*element*,*index*,*separator*)

函数标识符(ID) 9

描述 **返回一个经过替换的字符串，其中** element**是替代元素，** *index***指出了被替换元素的位置。如果** *index***小于**0**，替换第一个元素；如果** *index***大于元素的个数，替换最后一个元素；如果** *string***是空串，函数返回一个只包含给定元素***element***的新字符串。**
如果索引index**是浮点数，需要先调用函数**Float.int( )**，计算真正的整数值。**

参数 *string* = String
element = String
index = Number(**被替换元素的索引**)
separator = String(**分隔符的第一个字符**)

返 回 值 String**或**invalid

特例 **如果分隔符***separator***是空串，则返回无效型。**

示例
```
var a = “  B C; E ”;
var s = “  ”;
var b = String.replaceAt(a, “ A ”, 0, s);          // b = “ A C; E ”
var c = String.replaceAt(a, “ F ”, 5, “ ; ”);      // c = “ B C; F ”
```

7.6.11 insertAt函数

函数 insertAt(*string*,*element*,*index*,*separator*)

函数标识符(ID)	10
描述	**返回一个字符串，这个字符串在** *string*的*index*位置上插入了给定的元素*element*。**分隔符***separator*对原字符串*string*进行了分隔。如果** index **小于**0, *index*取值为0；如果*index*大于元素的个数，把元素** *elemen*添加在字符串*string*的结尾；如果*string*是空串，函数返回一个只包含给定元素*element*的新字符串。 **如果索引***index*是浮点数, 需要先调用函数**Float.int( )，计算真正的整数值。
参数	*string* = String(**原始字符串**) *element* = String(**插入的元素**) *index* = Number(**添加元素的索引**) *separator* = String(**分隔符的第一个字符**)
返回值	String**或**invalid
特例	**如果分隔符***separator*是空串，则返回无效型。

示	var a = “ B C; E ”;
	var s = “ ”;
	var b = String.insertAt(a, “ A ”, 0, s); // b = “ A B C; E ”
	var c = String.insertAt(a, “ X ”, 3, s); // c = “ B C; E X ”
	var d = String.insertAt(a, “ D ”, 1, “ ; ”); // d = “ B C;D; E ”
	var e = String.insertAt(a, “ F ”, 5, “ ; ”); // e = “ B C; E;F ”

7.6.12 squeeze函数

函数	squeeze(*string*)
函数标识符(ID)	11
描述	**把字符串***string*中的连续空白符（whitespace）减少到一个, 并返回处理后的字符串。
参数	*string* = String
返回值	String**或**invalid
特例	—

示例	var a = “ Hello ”;
	var b = “ Bye Jon . See you! ”；
	var c = String.squeeze(a); // c = “ Hello ”；
	var d = String.squeeze(b); // d = “ Bye Jon. See you! ”；

7.6.13 trim函数

函数	trim(*string*)
函数标识符(ID)	12
描述	**剪裁字符串***String*首和尾上的空白符, 并返回剪裁后的字符串。
参数	*String* = String

返回值 String**或**invalid
特例 —
示 例
```
var a = “ Hello ”;
var b = “ Bye  Jon . See you! ”;
var c = String.squeeze(a);   // c = “ Hello ”；
var d = String.squeeze(b);   // d = “ Bye  Jon. See  you! ”；
```

7.6.14 compare函数

函数 compare(*string*1,*string*2)
函数标识符(ID) 13
描述 **返回值指出了** *string*1**和***string*2**在字典上的顺序关系。这种关系基于** Unicode**字符的代码关系。如果** *string*1**小于***string*2**，返回值是** -1**；如果***string*1**等于***string*2**，返回**0**；如果***string*1**大于***string*2**，返回**1**。**
参数 *String*1 = String
*String*2 = String
返回值 Integer**或**invalid
特例 —
示 例
```
var a = “ Hello ”;
var b = “ Hello ”;
var c = String.compare(a, b);                    // c = 0
var d = String.compare(“ Bye ”, “ Jon ”);   // d = -1
var e = String.compare(“ Jon ”, “ Bye ”);    // e = 1
```

7.6.15 toString函数

函数 toString(*value*)
函数标识符(ID) 14
描述 **返回给定值***value***的字符串表示。这个函数完成的转换功能与** [WMLScript]**语言支持的转换功能(也就是，从布尔值、整型值和浮点值自动地转换成字符串)相同。但是只有一点不同，这就是无效型返回字符串“** invalid **”。**
参数 *value* = Any
返回值 String
特例 —
示例
```
var  a = String.toString (12);      // a = “ 12 ”
var  b = String.toString (true);    // b = “ true ”
```

7.6.16 format函数

函数 format(*format*, *value*)
函数标识符(ID) 15

描述 利用*format*字符串给出的格式，把给定的值*value*转换成字符串。格式字符串仅包含一个格式说明符，该说明符可以出现在字符串的任何地方。如果出现多个格式说明符，只有第一个(最左边的)说明符被使用，其 他的说明符会被一个空字符串所替代。格式说明符的结构如下：

% [宽度] [. 精度] 类型

宽度项是一个非负的十进制整数，用于控制打印字符的最小个数。如果在输出的值中，字符的个数少于指定的宽度，左边会用空格填充，直至达到最小宽度。宽度项不会减小 *value*值。如果在输出的值中，字符个数大于指定的宽度，或者没有指定宽度， *value*中的所有字符都被打印(取决于精度项)。

精度项是一个非负的十进制整数，前面有一个黑点“ .”，用于设置输出值的精度。输出的格式与精度的类型有关：

d 指定打印数字的最小个数。如果在 *value*中的数字个数小于这个精度值，输出值的左边用零填补；如果数字的个数超过了这个精度，*value*也不会被截短。默认的精度值是 1。如果精度为0，经过转换的值也是0，结果输出一个空字符串。

f 规定小数点后数字的个数。当小数点出现时，它的前面至少要有一个数字。一个给定的数值会被四舍五入，以保证适当的个数。默认的精度是6。当精度是0或者小数点后没有数字时，不打印小数点。

s 规定打印字符的最大个数。在默认的情况下，打印所有的字符。

与宽度项不同，精度项会引起输出值的截短或对浮点数的取舍。

类型项是唯一一个必须的格式变量，它跟在可选项之后。类型字符决定了如何解释给定的值，也就是把给定的值解释成整数，还是浮点数，或者是字符串。支持的类型变量有：

d整数 输出值的格式为[-]dddd，其中dddd是一个或多个十进制数。

f浮点数 输出值的格式为[-]dddd.dddd，其中dddd是一个或多个十进制数。小数点前面的数字个数由数字的大小决定，小数点后面的数字个数由要求的精度决定。

s字符串 打印到字符串的结尾，或者达到所要求的精度。

格式字符串中的百分号(%)可以用双百分号（%%）来表示。

参数 *format* = String

value = Any

返回值 String或invalid

特例 不合法的格式说明符产生无效的返回值

示例

```
var a = 45;
var b = -45
var c = “ now ”;
var d = 1.2345678;
```

```
var  e = String.format (“ e: %6d ”, a);          // e = “ e:  45 ”
var  f = String.format (“ %6d ”, b);             // f = “  -45 ”
var  g = String.format (“ %6.4d ”, a);           // g = “ 0045 ”
var  h = String.format ((“ %6.4d ”, b);          // h= “  -0045 ”
var  i = String.format (“ Do it %s ”, c);        // I= “ Do it now ”
var  j = String.format (“ %3f ”, d);;            // j = “ 1.234567 . ”
var  k = String.format (“ %10.2f%% ”, d);        // k = “   1.23% ”
var  l = String.format (“ %3f %2f. ”, d);        // l = “ 1.234567 . ”
var  m = String.format (“ %.0d ”, 0);            // m = “  ”
var  n = String.format (“ %7d ”, “ Int ”)        // n = invalid
var  o = String.format (“ %s ”, true )           // o = “ true ”
```

7.7 URL函数库

名称 URL

库标识符(ID) 3

描述 **这个库包含了一组用于处理绝对** URL**和相对** URL**的函数。通用** URL(**见**[RFC1808])**的语法格式为：**

<scheme>://<host>:<port>/<path>;<params>?<query>#<fragment>。

即 <**方案**>://<**主机**>:<**端口**>/<**路径**>;<**参数**>?<**查询**>#<**字段**>。

7.7.1 isValid函数

函数 isValid(url)

函数标识符(ID) 0

描述 **如果给定的***url***具有正确的**URL**语法格式，则返回**ture**值；否则返回**false**值。这个函数既支持绝对的** URL**也支持相对的** URL**，相对的** URL**不会被分析成绝对的**URL**地址。**

参 数 *url*=String

返 回 值 Boolean**或**invalid

特例 **—**

示例

```
var a = URL.isValid (“http://w.hst.com/scrip#func()”)       // a = true
var b = URL.isValid(“../common#test( )”);                   // a = true
varc= URL.isVa l i d ( “ e x p e r i m e n t a l ? : / / w w w. h o s t c. c o m / c o n t ”
                                                            ); // c = false
```

7.7.2 getScheme函数

函数 getScheme(url)

函数标识符(ID) 1

描述 **返回给定**url**中使用的方案。这个函数支持绝对和相对** URL**，相对的** URL**不能被分析成绝对的**URL。

参数	*url*=String
返 回 值	String**或**invalid。
特例	**抽取方案时，如果发现**URL**语法结构不合法，则返回一个无效型。**

示例	var a = URL.getScheme (“ http://w.h.com/path#frag ”); // a = “ http ”
	var b = UTL.getScheme (“ w.h.com/path#frag ”); // b = “ ”

7.7.3 getHost函数

函数	getHost(*url*)
函数标识符(ID)	2
描述	**返回给定**url**中指定的主机号。这个函数支持绝对和相对的** URL**，相对**URL**不能被分析成绝对**URL。
参 数	*url*=String
返 回 值	String**或**invalid
特例	**抽取主机号部分时，如果发现** URL**语法结构不合法，返回一个无效型。**

实例	var a = URL.getHost (“ http://w.h.com/path#frag ”); //a = “ w.h.com ”
	var b = URL.getHost (“ path#frag ”); // b = “ ”

7.7.4 getPort函数

函数	getPort(*url*)
函数标识符(ID)	3
描述	**返回给定***url***中指定的端口号。若没有指定端口号，则返回一个空字符串。这个函数支持绝对和相对** URL**，相对**URL**不能被分析成绝对**URL。
参数	*url*=String
返 回 值	String**或**invalid
特例	**在抽取端口号时，如果发现** URL**语法结构不合法，则返回一个无效型。**

示 例	var a = URL.getPort (“ http://w.h.com:80/path#frag ”); // a = “ 80 ”
	var b = URL.getPort (“ http://w.h.com/path#frag ”); // b = “ ”

7.7.5 getPath函数

函数	getPath(*url*)
函数标识符(ID)	4
描述	**返回给定**url**中指定的路径。这个函数支持绝对和相对的** URL**，相对**URL**不能被分析成绝对**URL。
参数	*url*=String
返 回 值	String**或**invalid
特例	**在抽取路径时，如果发现**URL**的语法结构不合法，则返回一个无效型。**

示例 a = URL.get Path (“ http://w.h.com/home/sub/comp#frag ”);
// a = “ /home/sub/comp ”
b = URL.getPath(“ ../home/sub/comp#frag ”);
// b = “ ../home/sub/comp ”

7.7.6 getParameters函数

函数 getParameters(url)
函数标识符(ID) 5
描述 **返回给定*url*中使用的参数。如果没有指定参数，则返回一个空字符串。**
这个函数支持绝对和相对URL ，相对URL不能被分析成绝对的URL。
参数 *url*=String
返 回 值 String**或**invalid
特 例 **在抽取参数时，如果发现 URL的语法结构不合法，则返回一个无效型。**
实例 a = URL.getParameters (“ http://w.h.com/script; 3; 2?x=1&y=3 ”);
// a = “ 3;2 ”
b = URL.getParameters (“ ../script;3;2?x =1&y = 3 ”);
// b = “ 3;2 ”

7.7.7 getQuery函数

函数 getQuery(*url*)
函数标识符(ID) 6
描述 **返回给定**url**中的查询部分。若没有指定查询部分，则返回一个空字符串。这个函数支持绝对和相对 URL，相对URL不能被分析成绝对的URL。**
参数 *url*=String
返 回 值 String**或**invalid
特 例 **在抽取查询部分时，如果发现 URL的语法结构不合法，则返回一个无效型。**
示例 a = URL.getParameters (“ http://w.h.com/home;3;2?x = 1&y =3 ”);
// a = “ x = 1&y = 3 ”

7.7.8 getFragment函数

函数 GetFragment(*url*)
函数标识符(ID) 7
描述 **返回给定*url*中的字段（**fragment**）。若没有指定字段，则返回一个空字符串。这个函数支持绝对和相对 URL，相对URL不能被分析成绝对的URL。**

参数	*url*=String
返 回 值	String**或**invalid
特例	**在抽取字段时，如果发现** URL**的语法结构不合法，则返回一个无效型。**
实例	var a = URL.getFragment (“ http://w.h.com/cont#frag ”); // a = “ frag ”

7.7.9 getBase函数

函数	getBase( )
函数标识符(ID)	8
描述	**返回当前**WMLScript**编译单元的绝对**URL(**没有字段**)。
参数	—
返 回 值`	String
特例	—
示 例	var a = URL.getBase( ); //Result: “ http://www.host.com/test.scr ”

7.7.10 getReferer函数

函数:	getReferer( )
函数标识符(ID)	9
描述	**返回最小的相对** URL(**相对于当前编译单元中的基本** URL，**见**7.7.9**节)，这个**URL**是相对于调用当前编译单元的资源。本地函数的调用不改变引用索引。如果当前的编译 单元没有引用索引，则返回一个空字符串。**
参数	—
返 回 值	String
特例	—

示例	var base = URL.getBase( );
	// base = “ http://www.host.com/current.scr ”
	var referer = URL.getReferer( );
	// referer = “ app.wml ”

7.7.11 resolve函数

函数	resolve(*baseUrl*, *embeddedUrl*)
函数标识符(ID)	10
描述	**根据**[RFC1808]**定义的规则，从给定的** *baseUrl***和***embeddedUrl***中返回一个绝对**URL。**如果***embeddedUrl***已经是一个绝对** URL，**函数不做任何修改地直接返回这个** URL。
参数	*baseUrl* =String*embeddedUrl* = String

返 回 值	String**或**invalid
特例	**如果分析后发现**URL**语法结构不合法，则返回一个无效型。**
示例	var a = URL.resolve (“ http://foo.com/ ”, ” foo.vcf ”); // a = “ http://foo.com/foo.vcf ”

7.7.12 escape函数

函数	escape(*string*)
函数标识符(ID)	11
描述	**这个函数计算生成字符串** *string***的一个新版本。在这个版本中，一个十六进制的转义序列取代了** *string***中某些特殊字符，文件** [RFC1783] **给出了这些特殊字符定义。对于在** URL**中有特定含义的字符，不进行计算。** **根据**[RFC1783]**的规定，对于** Unicode**字符集中编码等于或者小于** 0xFF**的特殊字符，使用双数字格式的转义序列** %xx**。** **这个函数支持绝对和相对** URL **，相对**URL**不能被分析成绝对的** URL**。**
参数	*string* =String
返 回 值	String**或**invalid
特例	**如果***string***中包含非**US-ASCII**字符集中的字符，返回一个无效型。** **对一个给定的**URL**进行转义时，如果出现无效的** URL**语法，则返回一个无效型。**
示例	var a = URL.escape (“ http://w.h.com/dck?x = \u00ef#crd ”); // a = “ http://w.h.com/dck?x = %ef#crd ”

7.7.13 unescape函数

函数	unescape(*string*)
函数标识符(ID)	12
描述	**这个转义恢复函数计算生成** *string***的一个新版本。在这个版本中，由转义函数(见**7.7.12**节)引入每一个转义序列都被它所代表的字符取代。这个函数支持绝对和相对** URL**，相对**URL**不能被分析成绝对的** URL**。**
参数	*string* = String
返 回 值	String**或**invalid
特例	**如果***string***中包含非**US-ASCII**字符集中的字符，返回一个无效型。** **在对一个给定的**URL**恢复转义时，如果出现无效的** URL**语法，则返回一个无效型。**
示例	var a = URL.unescape (“ http://w.h.com/dck?x = %31%32#crd ”); // a = “ http://w.h.com/dck?x = 12#crd ”

7.7.14 escapeString函数

函数	escapeString(*string*)
函数标识符(ID)	13
描述	这个函数计算生成*string*的一个新版本。在这个版本中，[RFC1738]规定的特殊字符（非安全字符、保留字符和不可打印字符）被一个十六进制的转义序列取代，给定的字符串可以这样转义；这个函数可以不进行URL分析。根据[RFC1783]的规定，对于Unicode字符集中编码等于或者小于 0xFF的特殊字符，使用双数字格式的转义序列%xx。
参数	*string* = String
返 回 值	String或invalid
特例	如果*string*中包含非US-ASCII字符集中的字符，返回一个无效型。
示 例	var a = URL.escapeString (“ http://w.h.com.dck?x =\u00ef#crd ”); // a = “ http%3a%2f%2fw.h.com%2fdck%3fx%3d%ef%23crd ”

7.7.15 unescapeString函数

函数	unescapeString(*string*)
函数标识符(ID)	14
描述	这个恢复转义函数计算生成 *string*的一个新版本。在这个版本中，由URL.escapeString( ) 函数(见7.7.14节)引入每一种转义序列，被它表示的字符取代。给定的字符串可以这样恢复转义；这个函数不进行URL分析。
参数	*string* = String
返 回 值	String或invalid
特例	如果字符串包含不属于US-ASCII字符集的字符，返回一个无效型。

示例	var a = “ http%3a%2f%2fw.h.com%2fdck%3fx%3d12%23crd ”;
	var b = URL.unescapeString(a);
	// b = “ http://w.h.com/dck?x =12#crd ”

7.7.16 loadString函数

函数	loadString(*url*, *contentType)*
函数标识符(ID)	15
描述	返回由绝对*url*和*content type*指定的内容。*content type*只能规定一种内容类型，不允许使用通配符。 这个函数的功能如下： • 装入由给定的content type和url指定的内容，装载内容所需的其 他属性由用户代理的默认设置规定。 • 如果装载成功并且返回的内容类型与 *content type*给定的类型相匹

配，那么这个内容被转换成Unicode字符串，并被返回。

· 如果装载失败，或者返回的内容类型是错误的，那么返回一个与采用方案有关的特定错误代码。

参数	*url* = String*contentType* = String
返 回 值	String、Integer或者invalid
特例	在装载失败的情况下，返回一个整型错误代码，这个错误代码与使用的URL方案有关。如果使用 HTTP[RFC2068]或者WSP(见[WAE])方案，则返回HTTP错误代码。 如果*content type*规定了几个内容类型或者包含了通配符，则返回无效型。
示例	var myUrl = “ http://www.host.com/vcards/myaddr.vcf ”; myCard = URL.loadString (myUrl, “ text/x-vcard ”);

7.8 WMLBrowser函数库

名称	WMLBrowser
库标识符(ID)	4
描述	这个库包含了一组函数， WMLScript利用这些函数存取各种 WML浏览器变量和属性。如果系统不支持 WML浏览器，那么下面的所有函数都返回无效型。

7.8.1 getVar函数

函数	getVar(*name*)
函数标识符(ID)	0
描述	返回当前浏览器上下文中由 *name*给定的变量值。如果给定的变量不存在，则返回一个空字符串。
参数	*name* = String
返 回 值	String或invalid
特例	—
示例	var a = WMLBrowser.getVar (“ name ”); // a = “ Jon ” or whatever value the variable has

7.8.2 setVar函数

函数	setVar(*name*, *value*)
函数标识符(ID)	1
描述	在当前的浏览器上下文中，如果能把由 *name*给定的变量成功地设置成给定的值*value*，则返回true值；否则返回false值。
参数	*name* =String *value* = String
返 回 值	Boolean或invalid

特例 —

示例 var a = WMLBrowser. serVar (“ name ”, Mary); // a = true

7.8.3 go函数

函数 go(*url*, *vars*)

函数标识符(ID) 2

描述 规定由给定的*url*和变量映射指定的内容。这个函数与 WML中(更多的信息，参见[WML])的GO任务有相同的语义，当 WMLScript翻译器将控制权返回给 WML浏览器之后，这个内容被装入。如果给定的 *url*是一个空字符串 string(“ ”)，不装载任何内容，函数返回一个空字符串。

函数*go*( )和*prev*( )（见7.8.4节）可以互相代替。在把控制权返回给WML浏览器之前，可以多次调用这两个函数 ，然而，只有最后一次调用的设置保持有效。特殊情况下，如果最后一次调用 *go*( )或者*prev*( )，将URL设置成了一个空字符串(“ ”)，那么所有的请求都被取消。

参数 *url* = String*vars* =String

返 回 值 String或invalid。

特例 —

示 例
```
var card = “ http://www.host.com/loc/app.dck#start ”;
var vars = “ x = 3&y = 2 ”;
WMLBrowser.go (card, vars);
```

7.8.4 prev函数

函数 prev(vars)

函数标识符(ID) 3

描述 通过给定变量映射，通知 WML浏览器返回到先前的卡页。这个函数在与WML中(更多的信息，参见 [WML])的PREV任务有相同的语义。在WMLScript翻译器将控制权返回给 WML浏览器之后，装入加载先前的WML卡页。这个函数返回一个空字符串。

函数*go*( )和*prev*( )（见7.8.3节）可以互相替换。在把控制权返回给WML浏览器之前，可以多次调用这两个函数 ，然而，只有最后一次调用的设置保持有效。特殊情况下，如果最后一次调用 *go*（)或者*prev*( )，将URL设置成了一个空字符串(“ ”)，那么所有的请求都被取消。

参数 *vars* = String

返 回 值 String或invalid

特例 —

示例 WMLBrowser,prev (“ price= ” + currentPrice);

7.8.5 newContext函数

函数	newContext( )
函数标识符(ID)	4
描述	**清除当前WML浏览器的上下文，并返回一个空字符串。这个函数与WML中（更多的信息，参见[WML]）的NEWCONTEXT属性有相同的语义。**
参数	—
返 回 值	String**或**invalid
特例	—
示例	WMLBrowser.newContext ( );

7.8.6 getCurrentCard函数

函数	getCurrentCard( )
函数标识符(ID)	5
描述	**返回最小相对URL（相对于当前的编译单元基准,与如何接入当前起点有关的信息，参见7.7.9节），这个URL规定了WML浏览器（更多的信息，参见[WML]）正在处理的卡页(如果有的话)。如果WML页面中包括的卡片与当前编辑单元中使用的基准不相同，该函数返回一个绝对URL。**
参数	—
返 回 值	String**或**invalid
特例	**在没有当前卡页的情况下，返回无效型。**
示例	var a = WMLBrowser.getCurrentCard( ); // a = “deck#input”

7.8.7 refresh函数

函数	refresh( )
函数标识符(ID)	6
描述	**强迫WML浏览器更新它的上下文，并返回一个空字符串。这个函数与WML中（更多的信息，参见[WML]）的REFRESH任务有相同的语义。结果是，上下文的更新引起用户接口的更新。**
参数	—
返 回 值	String**或**invalid
特例	—
示例	WMLBrowse.setVar (“name”, “Zorro”); WMLBrowser.refresh( );

7.9 Dialogs函数库

名称	Dialogs

函数库标识符(ID) 5
描述 **这个库包含一组典型的用户接口函数。**

7.9.1 prompt函数

函数 prompt(*message*,*defaultInput*)
函数标识符(ID) 0
描述 **显示给定的消息***message*，**提示用户输入。默认输入参数***defaultInput* **包含了用户输入的初始内容。返回用户输入。**
参数 *message* = String
defaultInput = String
返 回 值 String**型或**invalid
特例 —
示 例
```
var a = “09-555 3456”;
var b = Dialogs.prompt (“Phone number: “, a);
```

7.9.2 confirm函数

函数 confirm(*message, ok, cancel*)
函数标识符(ID) 1
描述 **显示给定的消息***message***及两个应答：***ok***和***cancel*，**等待用户选择其中一个应答，***ok***返回**true**值，***cancel***返回**false**值。**
参数 *message* = String
ok = String **（文本、空字符串产生与具体实现相关的文本**）
cancel = String **（文本、空字符串产生与具体实现相关的文本**）
返 回 值 Boolean**或**invalid
特例 —
示例
```
function onAbort( ) {
return Dialogs.confirm(“Are you sure?”, “Yes”, “Well...”);
                                                    };
```

7.9.3 alert函数

函数 alert(*message*)
函数标识符(ID) 2
描述 **向用户显示给定的消息***message*，**等待用户确认，并返回一个空字符串。**
参数 *message* = String
返 回 值 String**或**invalid
特例 —
示例
```
function testValue (textElement) {
  If (String.length (textElement) > 8 ) {
```

```
        Dialogs.alert ( " Enter name < 8 chars! " );
    };
};
```

7.10 术语定义

本规范中使用的术语和习惯用法如下：

字节码(Bytecode) 一种内容的编码。在这里，内容是指一系列典型的操作码（也就是指令），这些操作码用在目标硬件（或虚拟机器）之中。

客户端（Client） 客户端是向服务器发出连接请求的设备或应用程序。

内容（Content） 内容是源服务器生成或存储的数据（或事件）。典型的是在响应用户请求时，内容由用户代理显示或解释。

内容编码（Content Encoding） 当被用做动词时，内容编码指的是把数据对象从一种格式转换为另外一种格式的行为。通常，目标格式需要的物理空间比原格式要少，更易于处理或存储，和/或被加密。当被用做名词时，内容编码指的是一种特殊的格式或编码的标准或处理。

内容格式（Content Format） 内容的实际表示。

设备（Device） 一个网络实体，能够发送和接收信息包的，并且有一个唯一的地址。在一个给定的上下文或跨越多重上下文，一个设备既可作为客户端，也可作为服务器。例如，一个设备作为其他服务器的客户端时可充当其他客户端的服务器。

Java脚本（JavaScript） Java脚本是一种实际的标准语言，用于向HTML文档添加动态行为，它是ECMA脚本（ECMAScript）的起源技术之一。

源服务器（Origin server） 它作为一种服务器，是给定资源（或称内容）存储或将被生成的地方，通常被看作是Web服务器或HTTP服务器。

资源（Resource） 它是一个可以被URL识别的网络数据对象或服务。资源可以用多种表述格式所表达（例如，多种语言、数据格式、数据块尺寸和分辨率）或以其他方式进行变化。

服务器（Server） 它是一种被动地等待一个或多个客户端连接请求的设备（或应用程序）。服务器可以接收或拒绝来自客户端的连接请求。

用户（User） 用户是一个通过用户代理观看、聆听或使用资源的人。

用户代理（User Agent） 用户代理是对无线标记语言 WML、无线标记语言脚本(WMLScript)、无线电话应用接口（WTAI）或其他资源进行解释的软件或设备。它可能是文本浏览器、语音浏览器、搜索引擎等。

Web服务器（Web Server） 用做HTTP服务器的网络主机。

无线标记语言（WML） 无线标记语言是一种超文本标记语言，用于表示在窄带设备（如：电话）中传输的信息。

无线标记语言脚本（ WMLScript） 它是一种脚本语言，用于移动设备编程，是JavaScript 脚本语言的一个扩展子集。

7.11 缩略语

本规范采用了下列缩略语：

API	Application Programming Interface	应用编程接口
ECMA	European Computer Manufacturer Association	欧洲计算机制造商协会
HTTP	Hypertext Transfer Protocol[RFC2068]	超文本传输协议
LSB	Least significant Bits	最低有效位
MSB	Most Significant Bits	最高有效位
RFC	Request For Comments	请求注释
UI	User Interface	用户接口
URL	Uniform Resource Locator[RFC1738]	统一资源定位器
W3C	World Wide Web Consortium	万维网联盟
WWW	World Wide Web	万维网
WSP	Wireless Session Protocol	无线会话协议
WTP	Wireless Transport Protocol	无线事务协议
WAP	Wireless Application Protocol	无线应用协议
WAE	Wireless Application Environment	无线应用开发环境
WTA	Wireless Telephony Application	无线电话应用
WTAI	Wireless Telephony Application Interface	无线电话应用接口
WBMP	Wireless BitMaP	无线位图

7.12 参考标准

[ECMA262] Standard ECMA-262: "ECMAScript Language Specification", ECMA, June 1997

[IEEE754] ANSI/IEEE Std 754-1985: "IEEE Standard for Binary Floating-Point Arithmetic", Institute of Electrical and Electronics Engineers, New York, 1985

[RFC1738] "Uniform Resource Locators (URL)",T. Berners-Lee, et al., December 1994
URL: ftp://ftp.isi.edu/in-notes/rfc1738.txt

[RFC1808] "Relative Uniform Resource Locators",R. Fielding, June 1995
URL: ftp://ftp.isi.edu/in-notes/rfc1808.txt

[RFC2119] "Key Words for Use in RFCs to Indicate Requirement Levels",S. Bradner, March 1997
URL: ftp://ftp.isi.edu/in-notes/rfc2119.tx

[UNICODE] "The Unicode Standard: Version 2.0",The Unicode Consortium, Addison-Wesley Developers Press, 1996
URL: http://www.unicode.org/

[WAP] "Wireless Application Protocol Architecture Specification",WAP Forum, 30-April-1998
URL: http://www.wapforum.org/

[WML] "Wireless Markup Language Specification",WAP Forum, 30-April-1998
URL: http://www.wapforum.org/

[WMLScript] "WMLScript Language Specification", WAP Forum, 30-April-1998
URL: http://www.wapforum.org/

7.13 参考资料

[JavaScript] "JavaScript: The Definitive Guide", David Flanagan. O'R eilly & Associates, Inc. 1997

[RFC2068] "Hypertext Transfer Protocol—HTTP/1.1", R. Fielding, et al., January 1997
URL: ftp://ftp.isi.edu/in-notes/rfc2068.txt

[WAE] "Wireless Application Environment Specification", WAP Forum, 30-April-1998
URL: http://www.wapforum.org/

[WSP] "Wireless Session Protocol", WAP Forum, 1998
URL: http://www.wapforum.org/

[XML] "Extensible Markup Language (XML), W3C Proposed Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al., February 10, 1998
URL: http://www.w3.org/TR/REC-xml

7.14 函数库小结

1. 函数库及库标识符(ID)

表7-1总结了函数库及库标识符。

表7-1 函数库及库标识符

函数库名称	库标识符(ID)	PAGE
Lang函数库	0	7
Float函数库	1	17
String函数库	2	22
URL函数库	3	38
WMLBrowser函数库	4	50
Dialogs函数库	5	56

2. 库及库函数

表7-2～表7-7总结了LANG库及函数标识符。

表7-2 LANG库及函数标识符

LANG库	函数标识符(ID)	LANG库	函数标识符(ID)
abs函数	0	maxInt函数	7
min函数	1	minInt函数	8
max函数	2	float 函数	9
parseInt函数	3	exit 函数	10
parseFloat函数	4	abort函数	11
isInt函数	5	random函数	12
isFloat函数	6	seed函数	13

表7-3 FLOAT库及函数标识符

FLOAT库	函数标识符(ID)	FLOAT库	函数标识符(ID)
int函数	0	round函数	4
floor函数	1	sqrt函数	5
ceil函数	2	maxFloat函数	6
pow函数	3	minFloat函数	7

表7-4 STRING库及函数标识符

STRING库	函数标识符(ID)	STRING库	函数标识符(ID)
length函数	0	removeAt函数	8
isEmpty函数	1	replaceAt函数	9
charAt函数	2	insertAt函数	10
subString函数	3	squeeze函数	11
find函数	4	trim函数	12
replace函数	5	compare函数	13
elements函数	6	toString函数	14
elementAt函数	7	format函数	15

表7-5 URL库及函数标识符

URL库	函数标识符(ID)	URL库	函数标识符(ID)
isValid函数	0	getBase函数	8
getScheme函数	1	getReferer函数	9
getHost函数	2	resolve函数	10
getPort函数	3	escape函数	11
getPath函数	4	unescape函数	12
getParameters函数	5	escapeString函数	13
getQuery函数	6	unescapeString函数	14
getFragment函数	7	loadString函数	15

表7-6 WMLBROWSER 库及函数标识符

WMLBROWSER库	函数标识符(ID)	WMLBROWSER库	函数标识符(ID)
getVar函数	0	newContext函数	4
setVar函数	1	getCurrentCard函数	5
go函数	2	refresh函数	6
prev函数	3		

表7-7 DIALOGS 库及函数标识符

DIALOGS库	函数标识符(ID)
prompt函数	0
confirm函数	1
alert函数	2