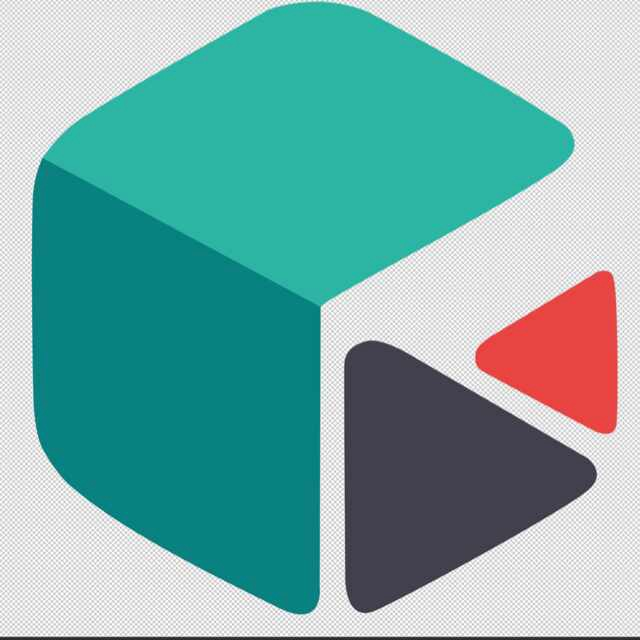
****

**gaiascript 教程**

gaiascript 是一种轻量小巧的脚本语言，用C，C++，C#，Java等语言开发，其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。

gaiascript 由坤仪金科于2017年1月开发完成。

**设计目的**

其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。

**gaiascript 特性**

* **轻量级**: 它用标准C语言和C++，C#，Java等语言开发，仅使用底层代码，编译后仅仅一百余K，可以很方便的嵌入别的程序里。
* **可扩展**: gaiascript提供了非常易于使用的扩展接口和机制：由宿主语言(通常是C或C++)提供这些功能，gaiascript可以使用它们，就像是本来就内置的功能一样。
* **其它特性**:
  + 支持面向过程(procedure-oriented)编程和函数式编程(functional programming)；
  + 自动内存管理；可以实现数组，哈希表，集合；
  + 跨语言跨平台；包含C++，C#，Java等多个语言版本，可在Windows，Linux，IOS，Android等多个平台使用。

兼容股票公式，可用于金融数学计算。

**gaiascript 应用场景**

* 游戏开发
* 独立应用脚本
* Web 应用脚本
* 金融数学运算

**第一个 gaiascript程序**

接下来我们使用 gaiascript来输出"Hello World!"

**实例(gaiascript 1.0)**

out(‘Hello World!’);

运行后，会在屏幕上显示 Hello, world!。

**gaiascript基本语法**

gaiascript 学习起来非常简单，我们可以创建第一个 gaiascript 程序！

**注释**

**多行注释**

{

多行注释

多行注释

}

**标示符**

gaiascript 表示符用于定义一个变量，函数获取其他用户定义的项。标示符以一个字母 A 到 Z 或 a 到 z 或下划线 \_ 开头后加上0个或多个字母，下划线，数字（0到9）。。

gaiascript 不允许使用特殊字符如 @, $, 和 % 来定义标示符。 Gaiascript不是一个区分大小写的编程语言。因此在 gaiascript 中 W3c 与 w3c 是两个相同的标示符。以下列出了一些正确的标示符：

mohd zara abc move\_name a\_123

myname50 \_temp j a23b9 retVal

**关键词**

以下列出了 gaiascript 的保留关键字。保留关键字不能作为常量或变量或其他用户自定义标示符：

|  |  |  |  |
| --- | --- | --- | --- |
| and | break | dowhile | else |
| elseif | var | false | for |
| function | if | in | local |
| none | not | or | repeat |
| return | set | true | until |
| while | else |  |  |

**全局变量**

在默认情况下，变量总是认为是全局的。

全局变量不需要声明，在最外层给一个变量赋值后即创建了这个全局变量，访问一个没有初始化的全局变量也不会出错，只不过得到的结果是：0。

out(b);

-->0

var b = 10;

out(b);

-->10

如果你想删除一个全局变量，只需要调用delete方法。

delete(b);

out(b); --> 0

**gaiascript 数据类型**

gaiascript不是动态类型语言，变量要类型定义。 值可以存储在变量中，作为参数传递或结果返回。

gaiascript中有3个基本类型分别为：number，string，function

|  |  |
| --- | --- |
| **数据类型** | **描述** |
| number | 表示双精度类型的实浮点数 |
| string | 字符串由一对双引号或单引号来表示 |
| function | 由 C++，Java，C# 或 gaiascript 编写的函数 |

**number（数字）**

gaiascript 默认只有一种 number 类型 -- double（双精度）类型，以下几种写法都被看作是 number 类型：

var a = 2;

var a = 2.2;

var a = 0.2;

**string（字符串）**

字符串由一对单引号来表示。

var a = ‘string1’;

var b = ‘string2’;

在对一个数字字符串上进行算术操作时，gaiascript 会尝试将这个数字字符串转成一个数字:

var a =‘1’, b = ‘1’

out(a + b);

-->2

**function（函数）**

gaiascript所有的返回类型都是number。

function factorial1(n)

{

if(n = 0)

{

return 1;

}

else

{

return n \* factorial1(n - 1);

}

}

out(factorial1(5));

-->120

**gaiascript 变量**

变量在使用前，必须在代码中进行声明，即创建该变量。

编译程序执行代码之前编译器需要知道如何给语句变量开辟存储区，用于存储变量的值。

gaiascript 变量有二种类型：全局变量、局部变量。

gaiascript 中的变量全是局部变量，除非在最外层声明为全局变量。

局部变量的作用域为从声明位置开始到所在语句块结束。

变量的默认值均为 0。

{全局变量}

var a = 1;

{局部变量}

function joke()

{

var a = 1;

}

gaiascript 可以连续定义多个变量。

var a = 1, b = 2, c = 3, d = 4;

**赋值语句**

赋值是改变一个变量的值和改变表域的最基本的方法。

set a = 2;

gaiascript 可以连续为多个变量赋值。

set a = 1, b = 2, c = 3, d = 4;

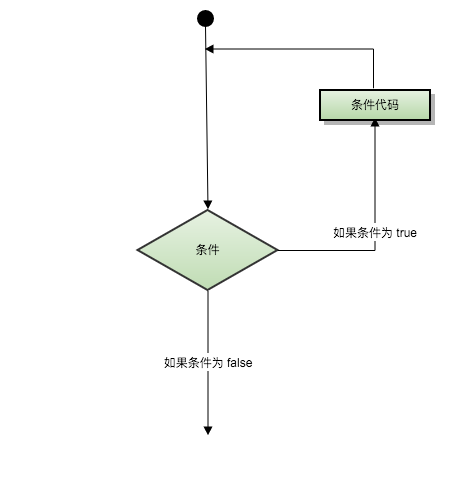
**gaiascript 循环**

很多情况下我们需要做一些有规律性的重复操作，因此在程序中就需要重复执行某些语句。

一组被重复执行的语句称之为循环体，能否继续重复，决定循环的终止条件。

循环结构是在一定条件下反复执行某段程序的流程结构，被反复执行的程序被称为循环体。

循环语句是由循环体及循环的终止条件两部分组成的。



gaiascript语言提供了以下几种循环处理方式：

|  |  |
| --- | --- |
| **循环类型** | **描述** |
| [while 循环](http://www.runoob.com/lua/lua-while-loop.html) | 在条件为 true 时，让程序重复地执行某些语句。执行语句前会先检查条件是否为 true。 |
| [for 循环](http://www.runoob.com/lua/lua-for-loop.html) | 重复执行指定语句，重复次数可在 for 语句中控制。 |
| [dotimes](http://www.runoob.com/lua/lua-repeat-until-loop.html) 循环 | 重复执行循环，直到达到指定次数为止 |
| [循环嵌套](http://www.runoob.com/lua/lua-nested-loops.html) | 可以在循环内嵌套一个或多个循环语句（while、for） |

**循环控制语句**

循环控制语句用于控制程序的流程， 以实现程序的各种结构方式。

gaiascript 支持以下循环控制语句：

|  |  |
| --- | --- |
| **控制语句** | **描述** |
| [break() 语句](http://www.runoob.com/lua/lua-break-statement.html) | 退出当前循环或语句，并开始脚本执行紧接着的语句。 |
| continue() 语句 | 退出当前语句，继续循环 |

**while循环**

样式1，目前只能嵌入function中使用

while(条件)

{

执行语句

}

例：

var i = 0;

while(i < 1000)

{

set i = i + 1;

}

样式2

while(条件，执行语句);

例：

var i = 0:

while(i < 1000, set i = i + 1);

**for循环**

样式1，目前只能嵌入function中使用

for(起始值，目标值，增减数值)

{

执行语句

}

例：

var i = 0;

for(0, 1000, 1)

{

set i = i + 1;

}

样式2

for(起始值，目标值，增减数值，执行语句);

例：

var i = 0;

for(0, 1000, 1, set i = i + 1);

**dotimes循环**

样式1，目前只能嵌入function中使用

dotimes(循环次数)

{

执行语句

}

例:

var i = 0;

dotimes(1000)

{

set i = i + 1;

}

样式2

dotimes(循环次数，执行语句);

例：

var i = 0;

dotimes(1000, set i = i + 1);

**break方法**

break()，用于跳出循环

例：

var i = 0;

while(1)

{

if(i > 1000)

{

break;

}

set i = i + 1;

}

**continue方法**

continue，用于继续循环

var i = 0;

while(1)

{

set i = i + 1;

if(i < 1000)

{

continue;

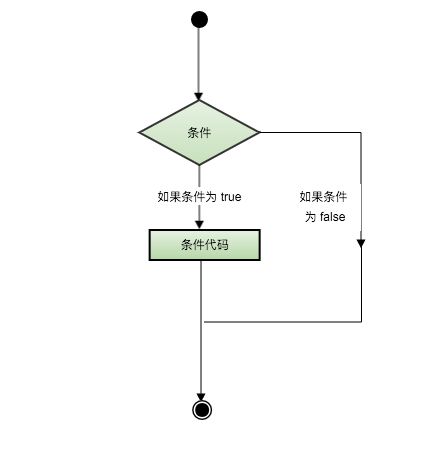
}

}

**gaiascript 流程控制**

gaiascript 编程语言流程控制语句通过程序设定一个或多个条件语句来设定。在条件为 1 时执行指定程序代码，在条件为 0 时执行其他指定代码。

以下是典型的流程控制流程图：



控制结构的条件表达式结果可以是任何值，gaiascript认为0为假，1为真。

要注意的是gaiascript中 1 为 true：

if(1)

{

out("1 为 true");

}

-->1 为 true

gaiascript 提供了以下控制结构语句：

|  |  |
| --- | --- |
| **语句** | **描述** |
| [if 语句](http://www.runoob.com/lua/if-statement-in-lua.html) | **if 语句** 由一个布尔表达式作为条件判断，其后紧跟其他语句组成。 |
| [if...else 语句](http://www.runoob.com/lua/if-else-statement-in-lua.html) | **if 语句** 可以与 **else 语句**搭配使用, 在 if 条件表达式为 false 时执行 else 语句代码。 |
| [if 嵌套语句](http://www.runoob.com/lua/nested-if-statements-in-lua.html) | 你可以在**if** 或 **else if**中使用一个或多个 **if** 或 **else if** 语句 。 |

样式1，目前只能嵌入function中使用。

if(条件)

{

执行语句

}

else if(条件)

{

执行语句

}

else

{

执行语句

}

例:

var a = 1, b = 1;

if(a > 0)

{

set b = 100;

}

else if(a == 0)

{

set b = 200;

}

else

{

set b = 300;

}

样式2

if(条件，当结果为真时的执行语句，当结果为假时的执行语句);

例：

if(a > 1, set b = 100, set b = 200);

**gaiascript 函数**

在gaiascript中，函数是对语句和表达式进行抽象的主要方法。既可以用来处理一些特殊的工作，也可以用来计算一些值。

gaiascript 提供了许多的内建函数，你可以很方便的在程序中调用它们，如out()函数可以将传入的参数打印在控制台上。

gaiascript 函数主要有两种用途：

* 1.完成指定的任务，这种情况下函数作为调用语句使用；

2.计算并返回值，这种情况下函数作为赋值语句的表达式使用。

gaiascript函数只能返回number

* **实例**
* 以下实例定义了函数 **max()**，参数为 num1, num2，用于比较两值的大小，并返回最大值：
* {函数返回两个值的最大值}
* function maxnum(num1, num2)
* {

var result = 0;

* if (num1 > num2)
* {
* set result = num1;
* }
* else
* {
* set result = num2;
* }
* return result;
* }
* {调用函数}
* out(‘两值比较最大值为 ‘, maxnum(10,4));

out(‘两值比较最大值为 ‘, maxnum(5,6));

-->10

-->7

**可变参数**

gaiascript函数可以接受可变数目的参数，在函数参数列表中使用[ref] 表示函数有可变的参数。

function test([ref]a,[ref]b)

{

set a = 1;

set b = 1;

}

var a = 0, b = 0;

test(a, b);

out(a, ‘,’, b);

-->1,1

**return方法**

return，用于中断语句，返回结果

例:

function test(a, b)

{

return a + b;

}

**控制台交互**

1)out输出方法

out(语句1，语句2，语句3...);

例如：

out('Hello', ' ', 'world');

-->Hello world

2)in输入方法

var a = '';

in(a);

**gaiascript 运算符**

运算符是一个特殊的符号，用于告诉解释器执行特定的数学或逻辑运算。gaiascript提供了以下几种运算符类型：

* 算术运算符
* 关系运算符
* 逻辑运算符

其他运算符

**算术运算符**

下表列出了 gaiascript 语言中的常用算术运算符，设定 A 的值为10，B 的值为 20：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **实例** |
| + | 加法 | A + B 输出结果 30 |
| - | 减法 | A - B 输出结果 -10 |
| \* | 乘法 | A \* B 输出结果 200 |
| / | 除法 | B / A w输出结果 2 |
| % | 取余 | B % A 输出结果 0 |
| ^ | 乘幂 | A^2 输出结果 100 |
| - | 负号 | -A 输出结果v -10 |

**实例**

我们可以通过以下实例来更加透彻的理解算术运算符的应用：

var a = 21；

var b = 10;

var c = a + b;

out(‘Line 1 - c 的值为 ‘, c );

set c = a – b;

out (‘Line 2 - c 的值为 ‘, c );

set c = a \* b;

out (‘Line 3 - c 的值为 ‘, c );

set c = a / b;

out (‘Line 4 - c 的值为 ‘, c );

set c = a % b;

out (‘Line 5 - c 的值为 ‘, c );

set c = a^2;

out (‘Line 6 - c 的值为 ‘, c );

set c = -a;

out (‘Line 7 - c 的值为 ‘, c );

以上程序执行结果为：

Line 1 - c 的值为 31

Line 2 - c 的值为 11

Line 3 - c 的值为 210

Line 4 - c 的值为 2.1

Line 5 - c 的值为 1

Line 6 - c 的值为 441

Line 7 - c 的值为 -21

**关系运算符**

下表列出了 gaiascript 语言中的常用关系运算符，设定 A 的值为10，B 的值为 20：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **实例** |
| = | 等于，检测两个值是否相等，相等返回 true，否则返回 false | (A = B) 为 false。 |
| != | 不等于，检测两个值是否相等，相等返回 false，否则返回 true< | (A != B) 为 true。 |
| > | 大于，如果左边的值大于右边的值，返回 true，否则返回 false | (A > B) 为 false。 |
| < | 小于，如果左边的值大于右边的值，返回 false，否则返回 true | (A < B) 为 true。 |
| >= | 大于等于，如果左边的值大于等于右边的值，返回 true，否则返回 false | (A >= B) 返回 false。 |
| <= | 小于等于， 如果左边的值小于等于右边的值，返回 true，否则返回 false | (A <= B) 返回 true。 |

**实例**

我们可以通过以下实例来更加透彻的理解关系运算符的应用：

var a = 21;

var b = 10;

if( a == b )

{

out(‘Line 1 - a 等于 b’ );

}

else

{

out (‘Line 1 - a 不等于 b’ );

}

if( a != b )

{

out (‘Line 2 - a 不等于 b’ );

}

else

{

out (‘Line 2 - a 等于 b’ );

}

if ( a < b )

{

out (‘Line 3 - a 小于 b’ );

}

else

{

out (‘Line 3 - a 大于等于 b’ );

}

if ( a > b )

{

out (‘Line 4 - a 大于 b’ );

}

else

{

out (‘Line 5 - a 小于等于 b’ );

}

{修改 a 和 b 的值}

set a = 5;

set b = 20;

if ( a <= b )

{

out (‘Line 5 - a 小于等于 b’ );

}

if ( b >= a )

{

out (‘Line 6 - b 大于等于 a’ );

}

以上程序执行结果为：

Line 1 - a 不等于 b

Line 2 - a 不等于 b

Line 3 - a 大于等于 b

Line 4 - a 大于 b

Line 5 - a 小于等于 b

Line 6 - b 大于等于 a

**逻辑运算符**

下表列出了 gaiascript 语言中的常用逻辑运算符，设定 A 的值为 true，B 的值为 false：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **实例** |
| && | 逻辑与操作符。 若 A 为 false，则返回 A，否则返回 B。 | (A && B) 为 false。 |
| || | 逻辑或操作符。 若 A 为 true，则返回 A，否则返回 B。 | (A || B) 为 true。 |

**实例**

我们可以通过以下实例来更加透彻的理解逻辑运算符的应用：

var a = 1;

var b = 1;

if ( a && b )

{

out(‘a and b - 条件为 true’ );

}

if ( a || b )

{

out(‘a or b - 条件为 true’ );

}

out("---------分割线---------" )

{修改 a 和 b 的值}

set a = 0;

set b = 1;

if ( a && b )

{

out("a and b - 条件为 true" );

}

else

{

out(‘a and b - 条件为 false’ );

}

以上程序执行结果为：

a and b - 条件为 true

a or b - 条件为 true

---------分割线---------

a and b - 条件为 false

**运算符优先级**

从高到低的顺序：

\* / %

+ -

< > <= >= != ==

&&

||

**字符串**

1)定义

var a = 'Hello world';

out(a);

-->Hello world

2)连接字符串

var a = 'Hello ';

var b = 'world';

var c = '';

set c = str.contact(a, b);

out(c);

-->Hello world

3)查找字符

var a = 'Hello world';

var b = str.find(a, 'o');

out(b);

-->4

var c = str.findlast(a, 'o');

out(c);

-->4

4)获取字符串的长度

var a = 'Hello world';

out(str.length(a));

-->11

5)截取字符串

var a = 'Hello world';

var b = '';

set b = str.substr(a, 6);

out(b);

-->world

set b = str.substr(a, 6, 2);

out(b);

-->wo

6)替换字符串

var a = 'Hello world';

var b = '';

set b = str.replace(a, 'o', 'i');

out(b);

-->Helli wirld

**集合**

1)定义

var a = list;

2)添加元素

var a = list;

list.add(a, 1, 2, 3, 4, 5);

3)获取元素

var a = list;

var b = 0;

list.get(b, a, 0);

4)插入元素

var a = list;

list.insert(a, 2, 1);

5)移除元素

var a = list;

list.remove(a, 2);

6)获取集合的长度

var a = list;

var b = list.size(a);

7)清除元素

var a = list;

list.clear(a);

**哈希表**

1)定义

var a = map;

2)添加元素

var a = map;

map.set(a, 'A', 1);

map.set(a, 'B', 1);

3)获取元素

var a = map;

var b = 0;

map.get(b, a, 'A');

4)移除元素

var a = map;

map.remove(a, 'A');

5)获取集合的长度

var a = map;

var b = map.size(a);

6)清除元素

var a = map;

map.clear(a);

7)是否包含元素

var a = map;

map.containskey(a, 'A');