Lua 数据类型 Lua 循环

Lua 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。

Lua 变量有三种类型：全局变量、局部变量、表中的域。

Lua 中的变量全是全局变量，那怕是语句块或是函数里，除非用 local 显式声明为局部变量。

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 nil。

```
-- test.lua 文件脚本
a = 5               -- 全局变量
local b = 5         -- 局部变量

function joke()
    c = 5           -- 全局变量
    local d = 6     -- 局部变量
end

joke()
print(c,d)          --> 5 nil

do
    local a = 6     -- 局部变量
    b = 6           -- 对局部变量重新赋值
    print(a,b);     --> 6 6
end

print(a,b)      --> 5 6
```

执行以上实例输出结果为：

```
$ lua test.lua
5    nil
6    6
5    6
```

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。

```
a = "hello" .. "world"
t.n = t.n + 1
```

Lua可以对多个变量同时赋值，变量列表和值列表的各个元素用逗号分开，赋值语句右边的值会依次赋给左边的变量。

```
a, b = 10, 2*x       <-->       a=10; b=2*x
```

遇到赋值语句Lua会先计算右边所有的值然后再执行赋值操作，所以我们可以这样进行交换变量的值：

```
x, y = y, x                     -- swap 'x' for 'y'
a[i], a[j] = a[j], a[i]         -- swap 'a[i]' for 'a[j]'
```

当变量个数和值的个数不一致时，Lua会一直以变量个数为基础采取以下策略：

```
a. 变量个数 > 值的个数             按变量个数补足nil
b. 变量个数 < 值的个数             多余的值会被忽略
```

例如：

```
a, b, c = 0, 1
print(a,b,c)             --> 0   1   nil

a, b = a+1, b+1, b+2     -- value of b+2 is ignored
print(a,b)               --> 1   2

a, b, c = 0
print(a,b,c)             --> 0   nil   nil
```

上面最后一个例子是一个常见的错误情况，注意：如果要对多个变量赋值必须依次对每个变量赋值。

```
a, b, c = 0, 0, 0
print(a,b,c)             --> 0   0   0
```

多值赋值经常用来交换变量，或将函数调用返回给变量：

```
a, b = f()
```

f()返回两个值，第一个赋给a，第二个赋给b。

应该尽可能的使用局部变量，有两个好处：

- 1. 避免命名冲突。
- 2. 访问局部变量的速度比全局变量更快。

索引

对 table 的索引使用方括号 []。Lua 也提供了 . 操作。

```
t[i]
t.i                 -- 当索引为字符串类型时的一种简化写法
gettable_event(t,i) -- 采用索引访问本质上是一个类似这样的函数调用
```

例如：

```
> site = {}
> site["key"] = "www.w3cschool.cc"
> print(site["key"])
www.w3cschool.cc
> print(site.key)
www.w3cschool.cc
```

← Lua 数据类型 Lua 循环 →