

Lua

Presentation for the Chicago GLUG 5/5/07

Tristan Sloughter (sloutri@iit.edu)

May 5, 2007

"When I am working on a problem, I never think about beauty. I think only of how to solve the problem. But when I have finished, if the solution is not beautiful, I know it is wrong." – Richard Buckminster Fuller

The Classics

- Dynamically Typed
- String, Math, I/O, etc Libraries
- REPL

The Important Stuff

- Size – 150K
- First class functions
- Tables and MetaTables – the only data structure, mold to your needs
- Coroutines
- C Interface

What's it Good For?

- Any script
- Interface with C code, used by a lot of video games (WoW, Far Cry, Ion WM, Adobe Photoshop Lightroom, nmap)
- Portable, only need ANSI C
- Simple and Beautiful scripts!

Functions and Variables

- Local and Global Variables
- First class functions
- Lexical Scope
- Variables number of arguments, ...
- Multiple return values

Multiple Return Values Example

```
s,e = string.find("hello Lua users", "Lua")
```

Results

```
s -> 7
```

```
e -> 9
```

Another Multiple Return Values Example

```
a,b = unpack{10, 20, 30}
```

Results

```
a -> 10
```

```
b -> 20
```

Functions

Closures

```
function counter()  
  local i = 0  
  return  
    function ()  
      i = i+1  
      return i  
    end  
end
```

Use

```
c1 = count ()  
c1 () -> 1  
c2 = count ()  
c2 () -> 1  
c1 () -> 2
```

- Does Everything
- Namespaces and OOP created through tables
- All other data structures created with tables
- Even the environment is a table

Table, Array, List

```
a = {x=10, y=20}
a.w = 30
```

```
seq= {"First", "Second", "Third"}
print(seq[1])           -> "First"
```

```
list=nil
for line in io.line() do
  list = {next=list, value=line}
end
```

Setting the Index

```
> optable = {'+'="plus", ['-']="minus"}  
> print(optable['-'])  
minus
```

```
> optable = {'+'="plus", ["x"]="minus"}  
> print(optable.x)  
minus
```

Overwrite/Store Functions

```
> math.sin = print  
> math.sin("hello")  
hello
```

Example

```
> t = {10, 20, 30}
> for i,v in ipairs(t) do
    print(i, v)
end
```

```
1      10
2      20
3      30
```

```
> t = {S={"AB"}, A={"CB","a"}}
> t.s="s"
> for k,v in pairs(t) do
    print(k, v)
end
```

```
A      table: 0x62cc60
S      table: 0x62cc10
s      s
```

MetaTables

- Reset certain builtins
- `__add` is `+`
- `__eq` is `==`
- etc.
- `__index` creates a function that is called each time you access a tables field
- `__newindex` is called each time you create a new field in a table
- Change them for `_G` (The Global Environment)

Example

```
Set = {}

Set.new (list)
  local set = {}
  setmetatable(set, mt)
  for _, v in ipairs(list)
    do set [v] = true end
  return set
end

mt.__add = Set.union

s1 = Set.new{10, 20, 30, 50}
s2 = Set.new{30, 1}

Set.print (s1 + s2) ->{1, 10, 20, 30, 50}
```

Example

```
Account = {balance=0}

function Account:new(o)
    o = o or {}
    setmetatable(o, self)
    self.__index = self
    return o
end

function Account:withdraw(v)
    self.balance = self.balance - v
end

a = Account:new{balance = 10}
a:withdraw(5)
print(a.balance)
```

- Lua - <http://www.lua.org/>
- Programming Lua -
<http://www.amazon.com/exec/obidos/ASIN/8590379825/lua-home-20>
- These Slides - <https://tristan.diginux.net/>