Seven More Languages in Seven Weeks

Correl Roush

January 13, 2016

LUA

A powerful, fast, lightweight, embeddable scripting language

- Installing Lua
- Exploring with the REPL
  - Syntax
  - Types
  - Functions

Whitespace doesn't matter

- Lua is *dynamically* typed
- No integers (all numbers are 64-bit floats)
- `nil` is its own type

- Functions are *first-class values*
- Arguments are flexible
- Support arbitrary numbers of arguments
- Support arbitrary numbers of results
- Lua does *tail call optimization*

Lua variables are *global by default*

Exercises


ALLWEIRDPICS.COM

```
book = {
   title  = "Grail Diary",
   author = "Henry Jones",
   pages  = 100
}

book.stars  = 5
book.author = "Henry Jones, Sr."
```

- Lua counts array indices starting at 1

```
medals = {
   "gold",
   "silver",
   "bronze"
}

medals[4] = "lead"
```

```
function table_to_string(t)
   local result = {}

   for k, v in pairs(t) do
      result[#result + 1] = k .. ": " .. v
   end

   return table.concat(result, "\n")
end

greek_numbers = {
   ena  = "one",
   dyo  = "two",
   tria = "three"
}
mt = {
   __tostring = table_to_string
}
setmetatable(greek_numbers, mt)
```

```
> =greek_numbers
ena: one
tria: three
dyo: two
```

```lua
Villain = {
   health = 100,
   new = function(self, name)
      local obj = {
         name   = name,
         health = self.health
      }
      setmetatable(obj, self)
      self.__index = self
      return obj
   end,
   take_hit = function(self)
      self.health = self.health - 10
   end
}
```

```lua
SuperVillain = Villain.new(Villain)

function SuperVillain.take_hit(self)
   -- Haha, armor!
   self.health = self.health - 5
end

SuperVillain:new("Toht")
```

You may be wondering how Lua handles multithreading.
It doesn't.

Example (Generator)

```
function fibonacci()
   local m = 1
   local n = 1

   while true do
      coroutine.yield(m)
      m, n = n, m + n
   end
end

generator = coroutine.create(fibonacci)
succeeded, value = coroutine.resume(generator)
-- value = 1
```

Example: Building a Scheduler

Exercises

Example: Making Music

Exercises

A lot of programmers see the surface of Lua's clean syntax and assume it's just another everyday scripting language. I certainly had that feeling at first glance. But I hope that as you've taken a deeper look at its tables and coroutines, you've enjoyed their beauty and simplicity.

- Approachable
- Portable
- Easily included in other projects

- Batteries not included
- Inefficient string handling
- Quirky

Lua's prototype-based object approach proves that you don't need classes to build a great object system.