To type or not to type Lua

Lua Workshop 2017

Andre Murbach Maidl

Typed Lua

Optional type annotations

```
 1 local function greet (greeting:string?)
 2
 3
 4   greeting = greeting or "Hello"
 5
 6
 7   return greeting .. " Lua Workshop 2017!"
 8 end
 9
10 print(greet("Good Afternoon")) -- ok
11 print(greet())                 -- ok
12 print(greet({}))               -- not ok
```

Local type inference

```
 1 local function greet (greeting:string?):
 2   string
 3   -- greeting: string | nil
 4   greeting = greeting or "Hello"
 5
 6   -- greeting: string
 7   return greeting .. " Lua Workshop 2017!"
 8 end
 9
10 print(greet("Good Afternoon")) -- ok
11 print(greet())                 -- ok
12 print(greet({}))               -- not ok
```

Generated code

```
1 local function greet (greeting)
2   greeting = greeting or "Hello"
3   return greeting .. " Lua Workshop 2017!"
4 end
5 print(greet("Good Afternoon"))
6 print(greet())
```

Limitations on forward declarations

```
 1 local is_even, is_odd
 2 function is_even (n: integer):boolean -- err
 3   if (n == 0) then
 4     return true
 5   else
 6     return is_odd(n – 1)                -- err
 7   end
 8 end
 9 function is_odd (n: integer):boolean
10   if (n == 0) then
11     return false
12   else
13     return is_even(n - 1)
14   end
15 end
16 print(is_even(8))
17 print(is_odd(8))
```

Lua-like overloading

```
 1 local function get_upload_server (
 2   server: string | { "upload_server":string? }):
 3   (string, string) | (nil, string)
 4   if type(server) == "string" then
 5     return server, "specific"
 6   else
 7     local server = server.upload_server
 8     if server then
 9       return server, "default"
10     else
11       return nil, "no upload server set"
12     end
13   end
14 end
15
16 local server, mod_or_err = get_upload_server({})
17 if not server then
18   print("ERROR: " .. mod_or_err)
19 else
20   print("using " .. mod_or_err .. " server " .. server)
21 end
```

Table is the only data structure

```
 1 local array:{string} =
 2   { "typedlua", "at", "lua", "workshop" }
 3
 4 local map:{string:integer} =
 5   { moscow = 2014, san_francisco = 2017 }
 6
 7 local record:{"d":integer, "m":integer,
 8   "y":integer} = { d = 10, m = 6, y = 2021 }
 9
10 local record_with_array:{"z":integer,
11                                 string} =
12   { z = 2049, "K", "Joe" }
13
14 local va:string? = array[2]
15 local vm:integer? = map.san_francisco
16 local vr:integer = record.y
```

Nilable values (recalling Hugo's talk)

```
1 function sum_list (xs:{integer}):integer
2   local sum = 0
3   for i = 1, #xs do
4     sum = sum + xs[i] --> integer | nil
5   end
6   return sum
7 end
8
9 print(sum_list({[1] = 1,[2] = 2,[4] = 3}))
```

Records and refinement of tables

```
 1 local typealias Color =
 2   { "r": number, "g": number, "b": number }
 3
 4 local typealias Circle =
 5   { "x": number, "y": number,
 6     "radius": number, "color": Color }
 7
 8 local gray:Color =
 9   { r = 128, g = 128, b = 128, a = 255 }
10
11 local circle = {}
12 circle.x = 10
13 circle.y = 20
14 circle.radius = 5
15 circle.color = gray
16
17 -- circle has type Circle
```

Famous idiom, but polemic feature

```
 1 local typealias T = {"i":integer, "s":string,
 2   "f":(integer) -> (integer)}
 3
 4 local function get_s (t:T):string
 5   t.y = 20.0                                  -- not ok
 6   return t.s
 7 end
 8
 9 local open = { i = 1 }
10 open.s = "foo"
11 open.f = function (x:integer):integer return x + 1 end
12                                               -- open:  T
13
14 local closed1 = open                          -- closed1: T
15 closed1.x = 10.0                              -- not ok
16 open.x = 10.0                                 -- ok
17 closed1.s = closed1.s:reverse()
18
19 print(string.format("%q", get_s(open))) -- prints "oof"
20
21 local closed2:T = open
22 closed2.y = 20.0                              -- not ok
23 open.y = 20.0                                 -- ok
```

Defining modules

```
 1 local mymath = {}
 2
 3 local RADIANS_PER_DEGREE = 3.14 / 180.0
 4
 5 function mymath.deg (r:number):number
 6   return r / RADIANS_PER_DEGREE
 7 end
 8
 9 function mymath.rad (d:number):number
10   return d * RADIANS_PER_DEGREE
11 end
12
13 mymath.pow = function (x:number,
14                         y:number):number
15   return x ^ y
16 end
17
18 return mymath
```

Using modules

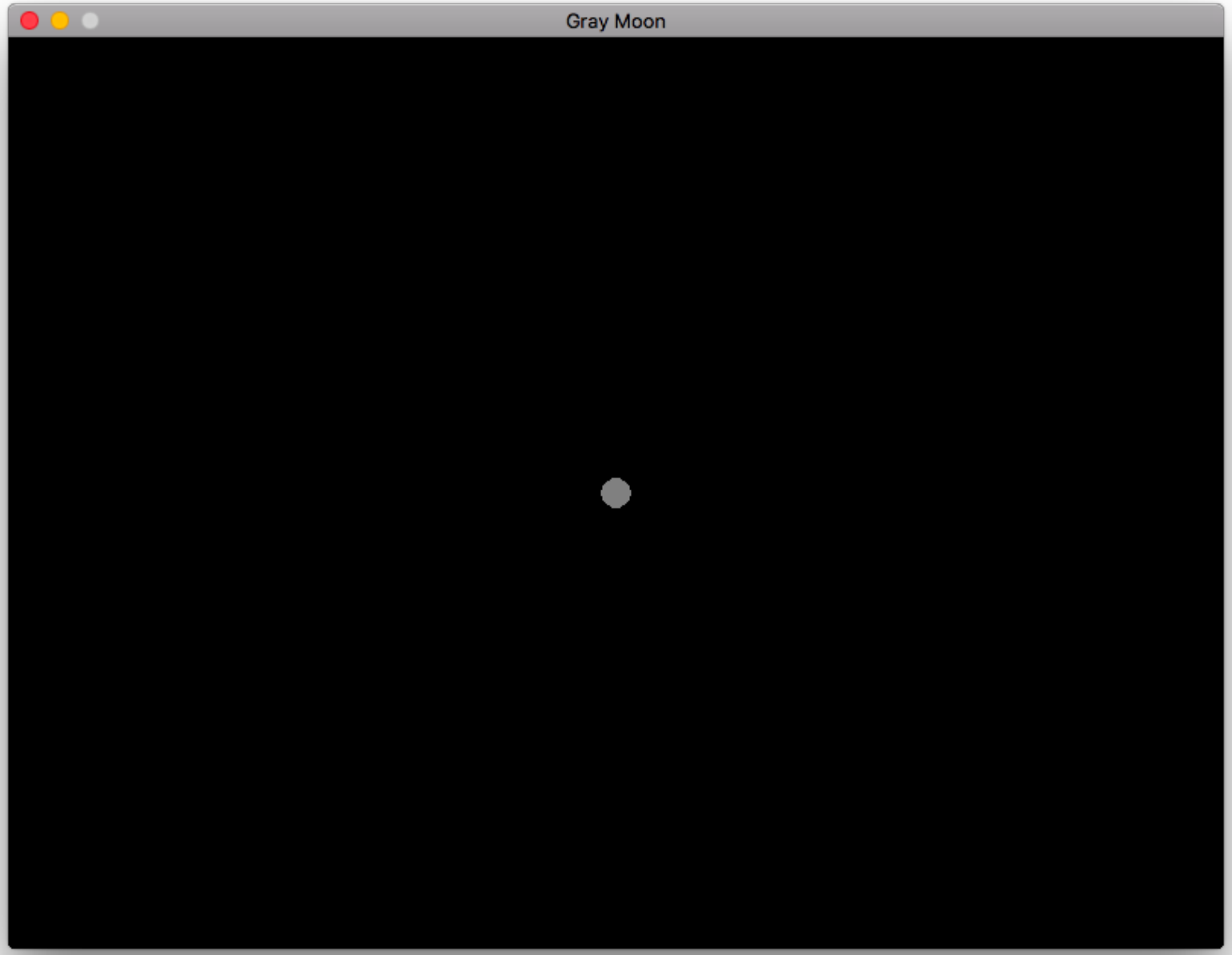
```
 1 local mymath = require "mymath"
 2 --[[
 3   mymath:{
 4     "deg":(number) -> (number),
 5     "rad":(number) -> (number),
 6     "pow":(number, number) -> (number)
 7   }
 8 ]]
 9
10 print(mymath.deg(1))          -- 57.324840764331
11 print(mymath.rad(1))          -- 0.017444444444
12 print(mymath.pow(2, "foo"))-- error
```

Typing external modules

```
draw : () -> ()
update : (number) -> ()

event : { "quit" : () -> () }

graphics : {
  "circle" : (string, number, number, number) -> (),
  "setColor" : (number, number, number, number?) -> (),
}

keyboard : { "isDown" : (string) -> (boolean) }

typealias flags = { "fullscreen":boolean,
  "fullscreentype":string, "vsync":boolean,
  "msaa":number, "resizeable":boolean,
  "borderless":boolean, "centered":boolean,
  "display":number, "minwidth":number,
  "minheight":number, "highdpi":boolean,
  "refreshrate":number, "x":number, "y":number }

window : { "getMode": () -> (number, number, flags),
           "setTitle": (string) -> (),
}
```

Using external modules

```
 1 local love = require "love"
 2
 3 typealias Color = {"r":number, "g":number, "b":number}
 4 typealias Circle = {"x":number, "y":number,
 5                     "radius":number, "color":Color}
 6
 7 love.window.setTitle("Gray Moon")
 8 local width, height = love.window.getMode()
 9 local gray:Color = { r = 128, g = 128, b = 128 }
10 local circle:Circle = { x = width / 2, y = height / 2,
11                          radius = 10, color = gray, }
12 function love.update (dt:number)
13   if love.keyboard.isDown("escape") then
14     love.event.quit()
15   end
16 end
17 function love.draw ()
18   love.graphics.setColor(circle.color.r,
19                          circle.color.g, circle.color.b)
20   love.graphics.circle("fill", circle.x, circle.y,
21                        circle.radius)
22 end
```


Gray Moon

OO Support

```
 1 class Circle
 2   x: number
 3   y: number
 4   radius: number
 5
 6   constructor new (x:number, y:number, radius:number)
 7     self.x = x
 8     self.y = y
 9     self.radius = radius
10   end
11
12   method move (x:number, y:number)
13     self.x = self.x + x
14     self.y = self.y + y
15   end
16
17   method getPosition ():(number, number)
18     return self.x, self.y
19   end
20 end
```

Using objects

```
 1 require("circle")
 2
 3 local c1 = class(circle.Circle).new(10, 20, 5)
 4 c1:move(50, 50)
 5 print(c1:getPosition()) -- 60 70
 6
 7 local c2 = class(circle.Circle).new(100, 200, 10)
 8 print(c2.radius)         -- 10
 9 c2.radius = 5
10 print(c2.radius)         -- 5
```

Inheritance

```
21 class Color
22   r: number
23   g: number
24   b: number
25
26   constructor new (r: number, g: number, b:number)
27     self.r = r
28     self.g = g
29     self.b = b
30   end
31 end
32 class ColoredCircle extends Circle
33   color: Color
34
35   constructor new (x: number, y: number,
36                    radius: number, color: Color)
37     super.new(x, y, radius)
38     self.color = color
39   end
40   method getColor ():(Color)
41     return self.color
42   end
43 end
```

Using inherited objects

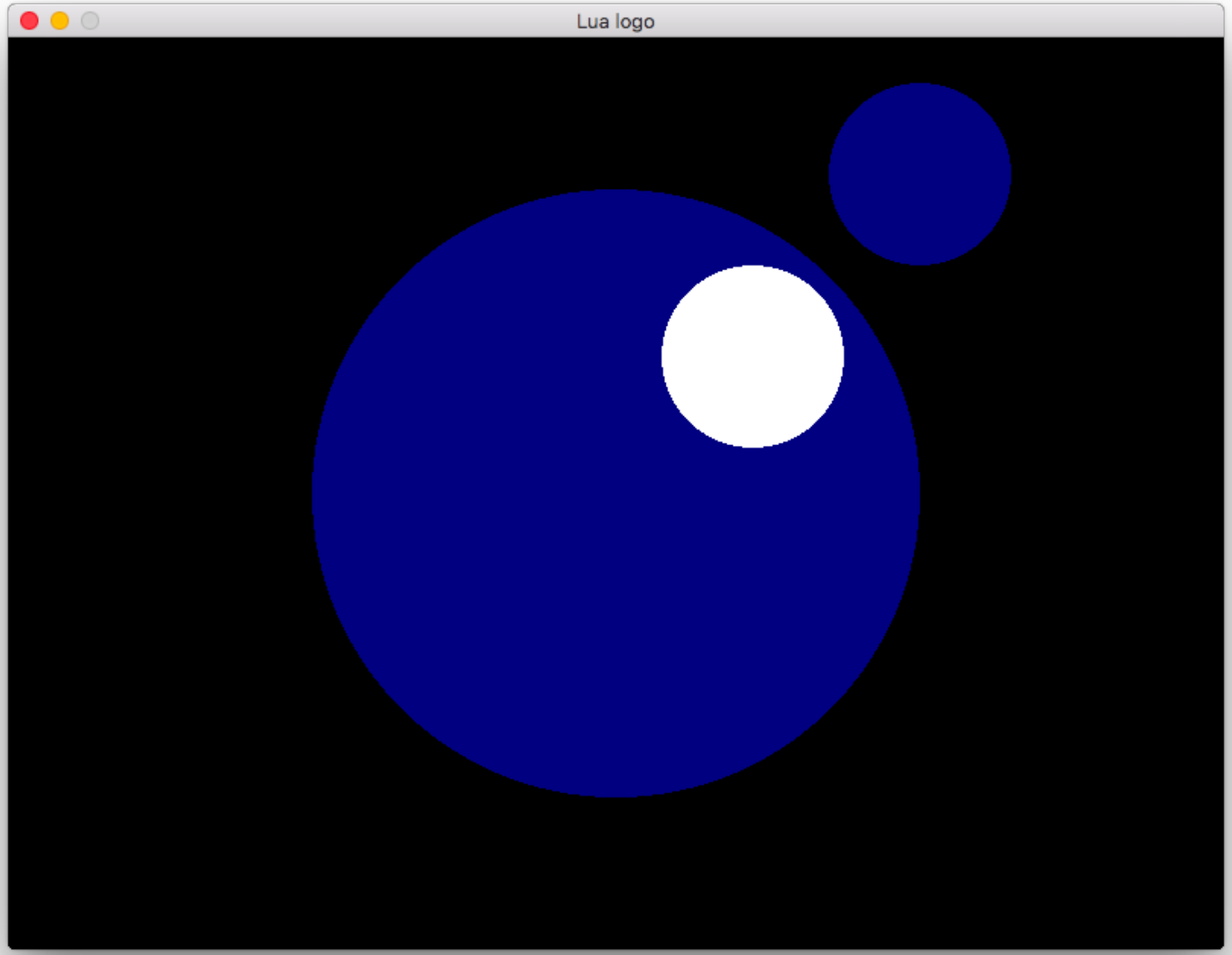
```
 1 require("circle")
 2
 3 local gray = class(circle.Color).new(128, 128, 128)
 4 local cb = class(circle.ColoredCircle).new(10, 20, 5,
 5                                            gray)
 6
 7 local c = cb:getColor()
 8 print(cb.color.r, cb.color.g, cb.color.b)-- 128 128 128
 9 c.r = 255
10 print(cb.color.r, cb.color.g, cb.color.b)-- 255 128 128
11
12 function cb:move (x:number, y:number)
13   print("exit")
14   os.exit(1)
15 end
16
17 cb:move(50, 50) -- exits here
18 os.exit(0)
```

Interfaces

```
 1 local love = require "love"
 2
 3 interface Drawable
 4   method draw: () => ()
 5 end
 6
   ...

38 class ColoredCircle extends Circle implements Drawable

   ...

49   method draw ()
50     love.graphics.setColor(self.color.r,
51       self.color.g, self.color.b)
52     love.graphics.circle("fill", self.x, self.y,
53       self.radius)
54   end
55 end
```

Back to the löve example

```
 1 local love = require "love"
 2 require "lua_logo.circle"
 3
 4 love.window.setTitle("Lua logo")
 5 local width, height = love.window.getMode()
 6
 7 local blue = class(lua_logo.circle.Color).new(0, 0, 128)
 8 local white = class(lua_logo.circle.Color).new(255, 255, 255)
 9 local earth = class(lua_logo.circle.ColoredCircle).new(width / 2,
10   height / 2, 200, blue)
11 local hole = class(lua_logo.circle.ColoredCircle).new(width / 2 + 90,
12   height / 2 - 90, 60, white)
13 local moon = class(lua_logo.circle.ColoredCircle).new(width – 200,
14   90, 60, blue)
15
16 function love.update (dt:number)
17   if love.keyboard.isDown("escape") then
18     love.event.quit()
19   end
20 end
21
22 function love.draw ()
23   earth:draw()
24   hole:draw()
25   moon:draw()
26 end
```


Lua logo

Nominal x Structural

```
 1 class Nominal1
 2   x: boolean
 3   constructor new (x:boolean) self.x = x end
 4 end
 5 typedef Structural1 = { "x": boolean }
 6
 7 class Nominal2
 8   x: boolean
 9   constructor new (x:boolean) self.x = x end
10 end
11 typedef Structural2 = { "x": boolean }
12
13 local function get_x_n (n:Nominal2):boolean
14   return n.x
15 end
16 local function get_x_s (s:Structural2):boolean
17   return s.x
18 end
19
20 print(get_x_n(Nominal1.new(false)))  -- not ok
21 print(get_x_s({ x = true }))         -- ok
```

Generics

```
 1 class Stack<T>
 2   contents: {T}
 3   constructor new ()
 4     self.contents = {}
 5   end
 6   method push (x:T)
 7     self.contents[#self.contents + 1] = x
 8   end
 9   method pop ():T?
10     local top = self.contents[#self.contents]
11     self.contents[#self.contents] = nil
12     return top
13   end
14 end
15 local stack1 = Stack.new<string>()
16 local stack2 = Stack.new<integer>()
17 stack1:push("Goodbye Lua Workshop")
18 stack2:push(2017)
19 print(tostring(stack1:pop())
20       .. " " ..
21       tostring(stack2:pop()))
```

IDE Support

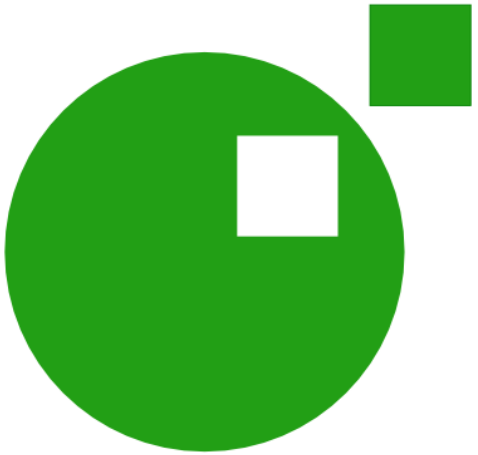
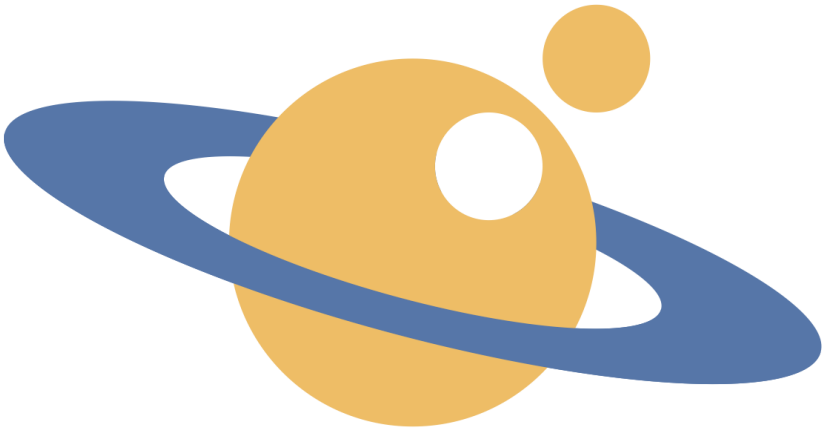
- Language Server Protocol
  - A JSON-RPC API for integrating IDE features.
- Typed Lua language server
  - Implements the API with Typed Lua type checker.
  - Adds error recovery to Typed Lua.
  - Improves AST handling through Visitor Patterns.
- Enabled features
  - Find all references, renaming variables, goto definition, code completion, linting, etc.

Academic feeling

- Typed Lua
  - PhD Thesis
  - https://github.com/andremm/typedlua
- Typed Lua + OO Support
  - GSoC 2016
  - https://github.com/kevinclancy/typedlua
- Typed Lua + IDE Support
  - GSoC 2017
  - https://gitlab.com/martanne/typedlua/tree/visitor

Lua community wants fast code

- Typing Lua semantics is challenging.
  - Coroutines, operator overloading, etc.
- Typed Lua was not designed for optimizations.
  - It is optionally typed instead of statically typed.
- Titan goes for it and Typed Lua goes together.
  - Not all Typed Lua, but the experiences we had.

Thank you!

- Questions?
- For more information:
  - http://www.typedlua.org
  - andremm@gmail.com