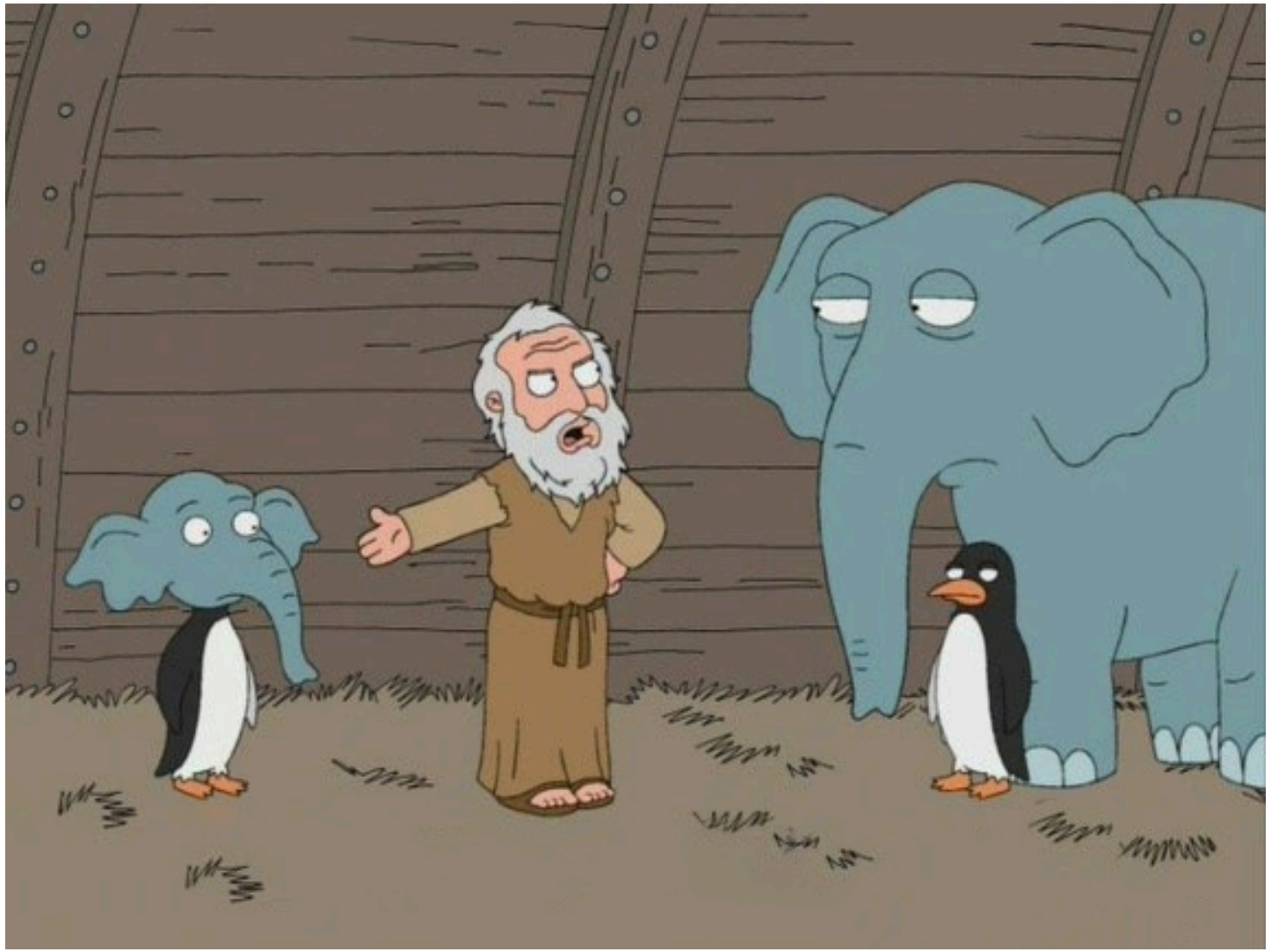

Adding types to Lua

typedlua

- Github
- Last activity - 2020
- Superset of Lua
- Supports Lua<=5.3
- Features
  - type inference
  - generics
  - typing external modules
  - classes implementation
  - nominal x structural typing

Usage

typedlua works as compiler (`*.tl -> *.lua`)

```
echo "x: number = 3.14" > file.tl
tlc file.tl
lua file.lua
```

Basic example

```
local function greet(name: string?): string
    name = name or "Anon"
    return "Hello " .. name
end
print(greet("Alex"))
print(greet())
-- print(greet({})) - compilation error
```

Generated code

```
local function greet(name)
    name = name or "Anon"
    return "Hello " .. name
end
print(greet("Alex"))
print(greet())
```

Type inference

```
local function get_upload_server(
    server: string | {"upload_server": string?}
): (string, string) | (nil, string)
    if type(server) == "string" then
        return server, "specific"
    else
        local server = server.upload_server
        if server then
            return server, "default"
        else
            return nil, "no upload server set"
        end
    end
end
local server, mod_or_err = get_upload_server({})
if not server then
    print("Error: " .. mod_or_err)
else
    print("Using " .. mod_or_err .. " server " .. server)
end
```

Modules

```
local mymath = {}
local RADIANS_PER_DEGREE = 3.14 / 180.0
function mymath.deg(r: number): number
    return r / RADIANS_PER_DEGREE
end
function mymath.rad(d: number): number
    return d * RADIANS_PER_DEGREE
end
mymath.pow = function (x: number, y: number): number
    return x ^ y
end
return mymath
```

```
local mymath = require "mymath"
print(mymath.deg(1))
print(mymath.rad(1))
-- print(mymath.pow(2, "foo")) compilation 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) -> (),
}
```

Typing external modules

```
local love = require "love"

typealias Color = {"r":number, "g":number, "b":number}
typealias Circle = {"x":number, "y":number,
                    "radius":number, "color":Color}

love.window.setTitle("Gray Moon")
local width, height = love.window.getMode()
local gray:Color = { r = 128, g = 128, b = 128 }
local circle:Circle = { x = width / 2, y = height / 2,
                        radius = 10, color = gray, }
function love.update (dt:number)
  if love.keyboard.isDown("escape") then
    love.event.quit()
  end
end
function love.draw ()
  love.graphics.setColor(circle.color.r,
                         circle.color.g, circle.color.b)
  love.graphics.circle("fill", circle.x, circle.y,
                       circle.radius)
end
```

OOP

```
class Circle
    x: number
    y: number
    radius: number

    constructor new(x: number, y: number, radius: number)
        self.x = x
        self.y = y
        self.radius = radius
    end
    method move(x: number, y: number)
        self.x = self.x + x
        self.y = self.y + y
    end
end
```

```
require("circle")
local c1 = class(circle.Cirlce).new(10, 20, 5)
c1:move(50, 50)
```

OOP

```
class Color
    r: number
    g: number
    b: number

    ...
end

class ColoredCircle extends Circle
    color: Color

    constructor new(x: number, y: number, radius: number, color: Color)
        super.new(x, y, radius)
        self.color = color
    end
end
```

OOP

Interfaces

```
local love = require("love")
interface Drawable
    method draw: () => ()
end
class ColoredCircle extends Circle implements Drawable
    method draw()
        love.graphics.setColor(self.color.r, self.color.g, self.color.b)
        love.graphics.ciclr("fill", self.x, self.y, self.radius)
    end
end
```

Nominal vs. Structural

```
class Nominal1
    x: boolean
    contructor new(x: boolean) self.x = x end
end
class Nominal2
    x: boolean
    contructor new(x: boolean) self.x = x end
end
local function get_x_nominal(n: Nominal2): boolean
    return n.x
end
print(get_x_nominal(Nominal.new(false))) -- not ok
```

```
typedef Structural1 = {"x": boolean}
typedef Structural2 = {"x": boolean}
local function get_x_structural(s: Structural2): boolean
    return s.x
end
print(get_x_structural({ x = true })) -- ok
```

Generics

```
class Stack<T>
    contents: {T}
    constructor new() self.contents = {} end
    method push(x: T)
        self.contents[#self.contents + 1] = x
    end
    method pop(): T?
        local top = self.contents[#self.contents]
        self.contents[#self.contents] = nil
        return top
    end
end
local stack = Stack.new<string>()
stack.push("Good evening")
print(stack.pop())
```

Alternatives

Teal

- superset of Lua too
- supports Lua>=5.4
- active (last changes in `master` - 16/11/2024)
- better docs (than typedlua)
- adds more types of types

```
-- an enum: a set of accepted strings
local enum State
   "open"
   "closed"
end
```

```
-- a record: a table with a known set of fields
local record Point
   x: number
   y: number
end
-- an interface: an abstract record type
local interface Character
   sprite: Image
   position: Point
   kind: string
end
-- records can implement interfaces,
-- using a type-identifying `where` clause
local record Spaceship
   is Character
   where self.kind == "spaceship"

   weapon: Weapons
end
-- a record can also declare an array interface,
-- making it double as a record and an array
local record TreeNode<T>
   is {TreeNode<T>}

   item: T
end
local root: TreeNode<number> = {
    item = 1,
    {item = 2},
    {item = 3, {item = 4}}
}
print(root[2][1].item)


local record File
   is userdata

   status: function(): State
   close: function(File): boolean, string
end
```

Problems

- Lua is dynamic language and relies on it in runtime
- Some libraries hard or impossible type checked (ex. `coroutines`)
- Requires compilation

Problems

```
local is_even: (number) -> (boolean)
local is_odd: (number) -> (boolean)
function is_even (n: integer):boolean
    if (n == 0) then
        return true
    else
        return is_odd(n - 1)
    end
end
function is_odd (n: integer):boolean
    if (n == 0) then
        return false
    else
        return is_even(n - 1)
    end
end
print(is_even(8))
print(is_odd(8))
```

Alternatives

Luau

- created and backed by Roblox Inc.
- superset of Lua 5.1 but not compatible with latest Lua
- adds other features as sandboxing and performance
- optional typing

```
type Point = {x: number, y: number}

local p: Point = {x = 1, y = 2}
print(p.x, p.y)
```