HAXE

SHOOTING FOR THE MOON

HAXE LANDS ON LUA

WRITE ONCE, TARGET MANY

iOS HTML NEKO

node JS python

C# php Java

WRITE ONCE, TARGET MANY

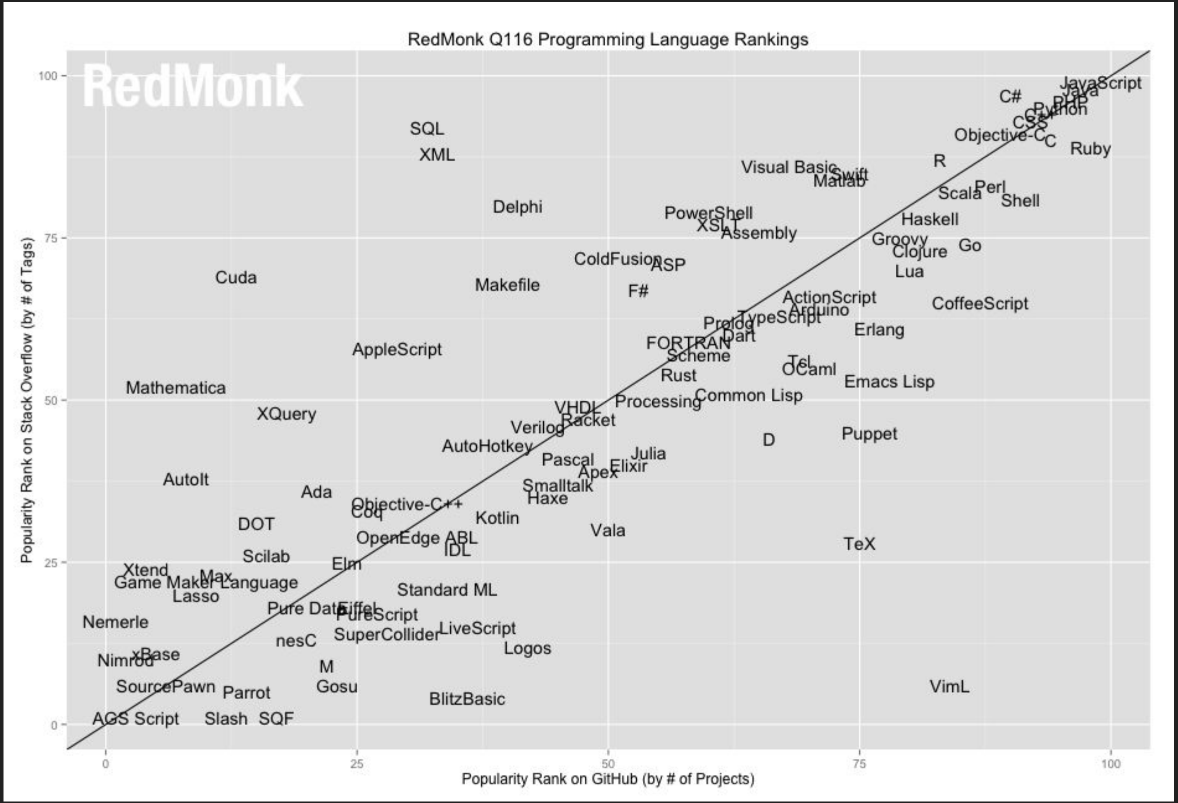
iOS HTML NEKO

node JS™ python™

C# php Java Lua

WHY LUA?

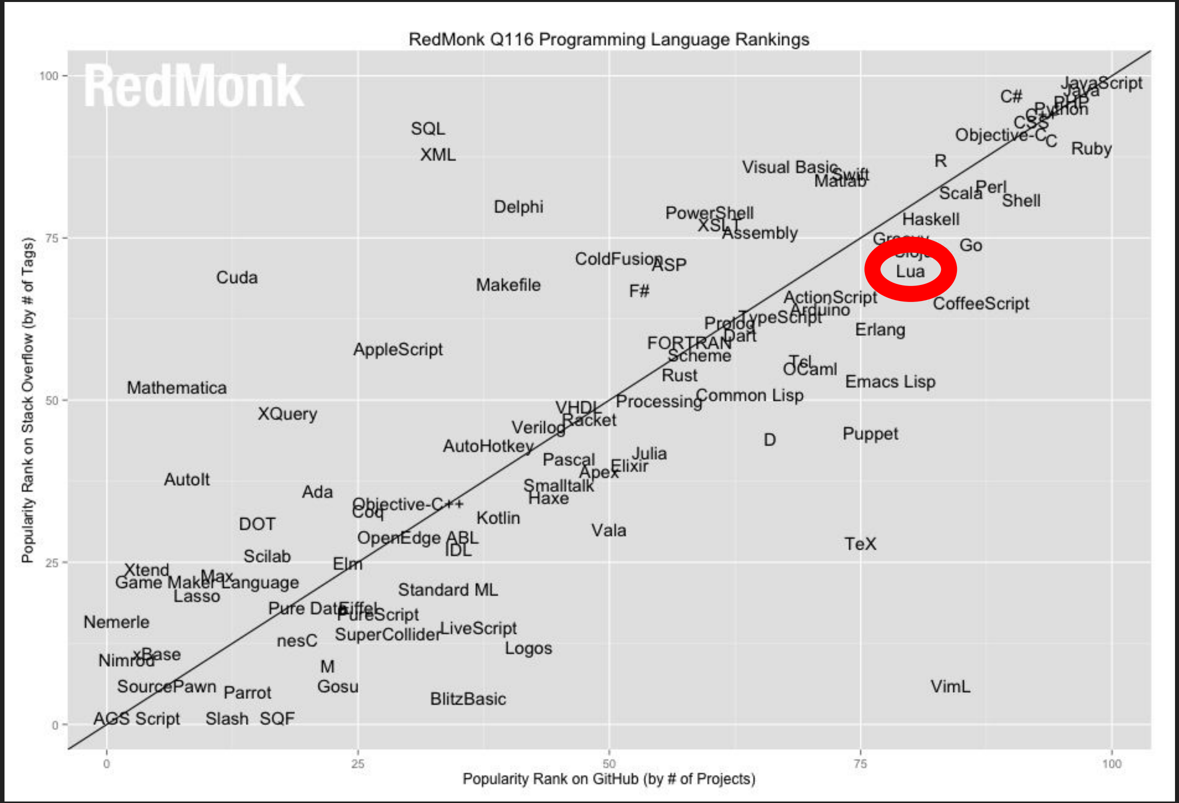
Surprisingly Active



http://redmonk.com/sogrady/2016/02/19/language-rankings-1-16/

WHY LUA?

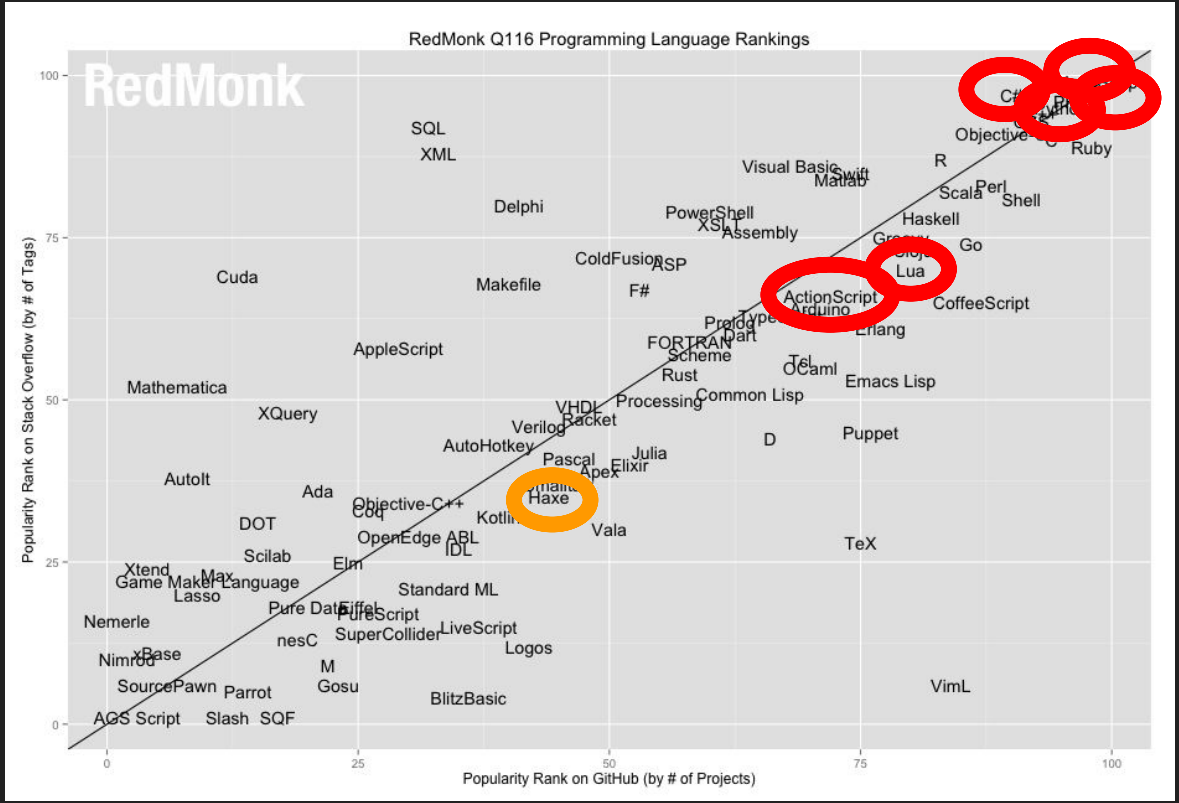
Surprisingly Active



http://redmonk.com/sogrady/2016/02/19/language-rankings-1-16/

WHY LUA?

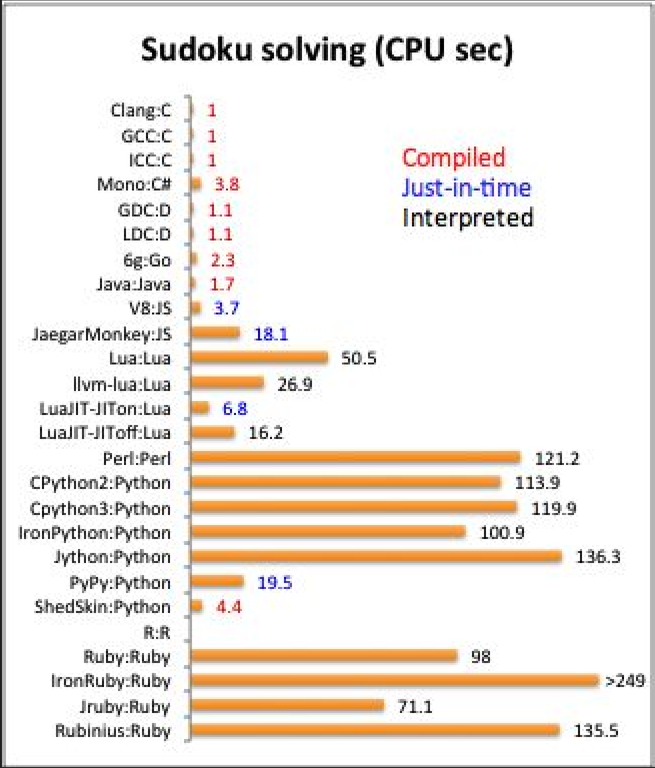
Surprisingly Active

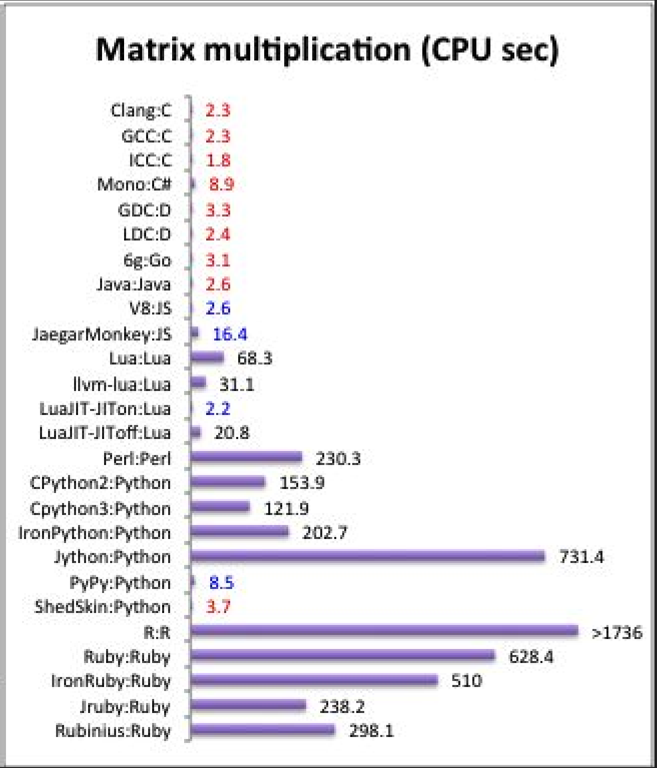


http://redmonk.com/sogrady/2016/02/19/language-rankings-1-16/

WHY LUA?

Raw Speed

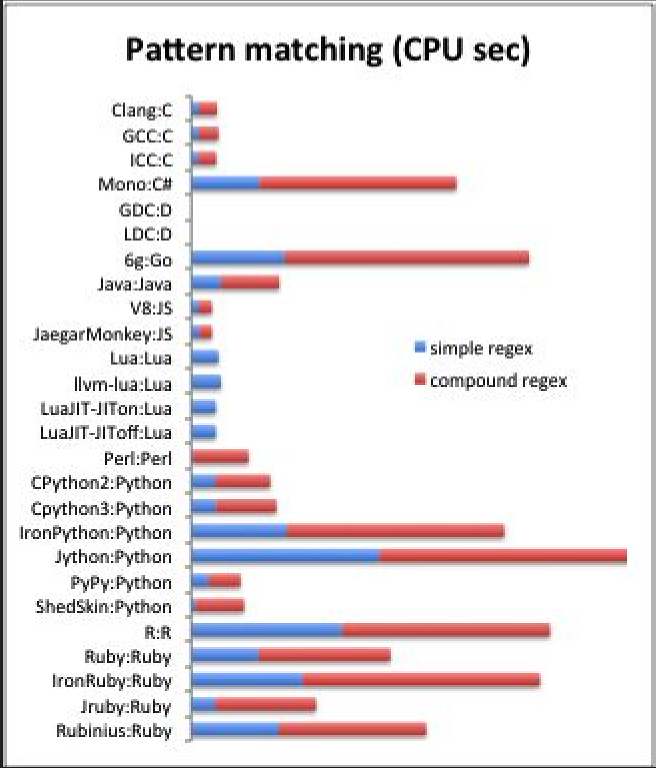


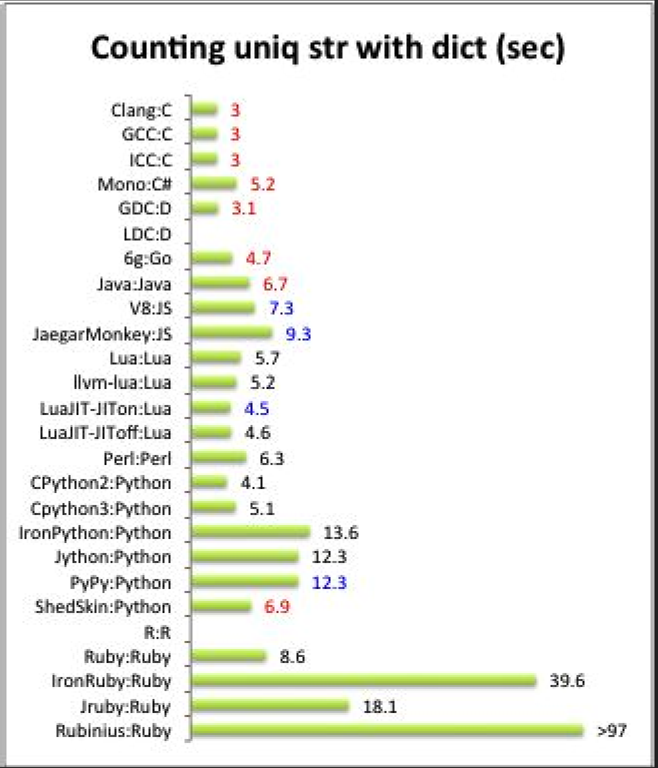


http://attractivechaos.github.io/

WHY LUA?

Raw Speed

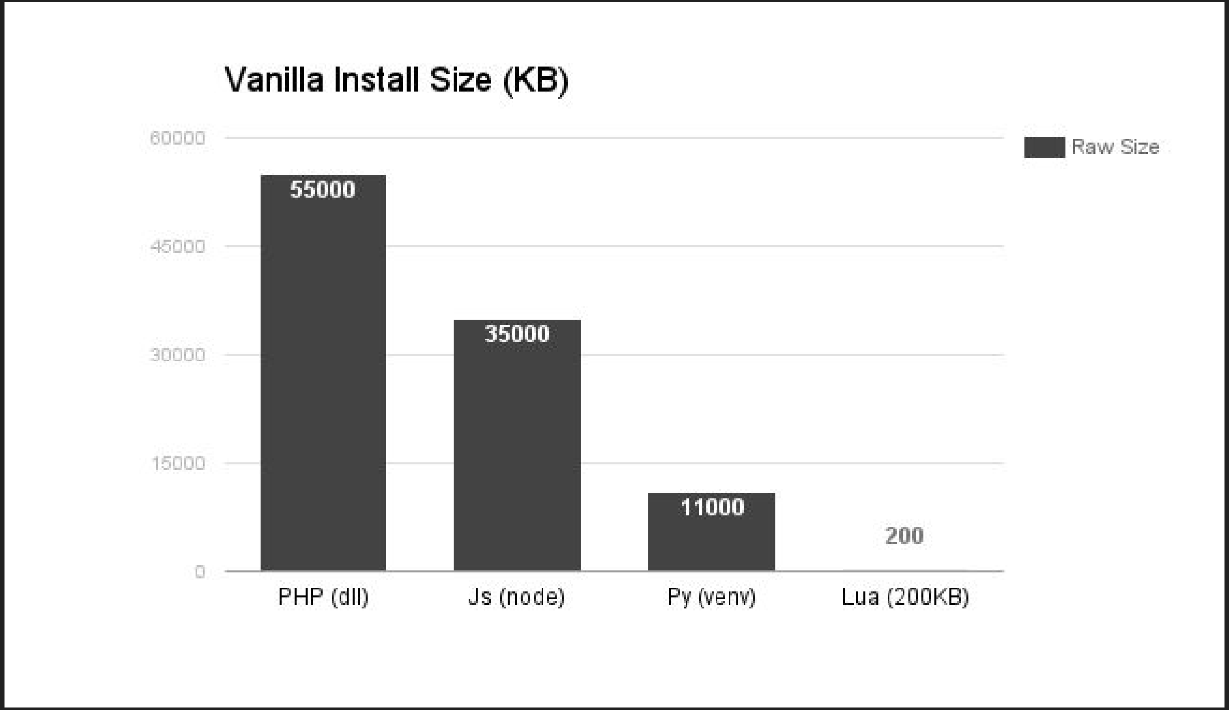




http://attractivechaos.github.io/

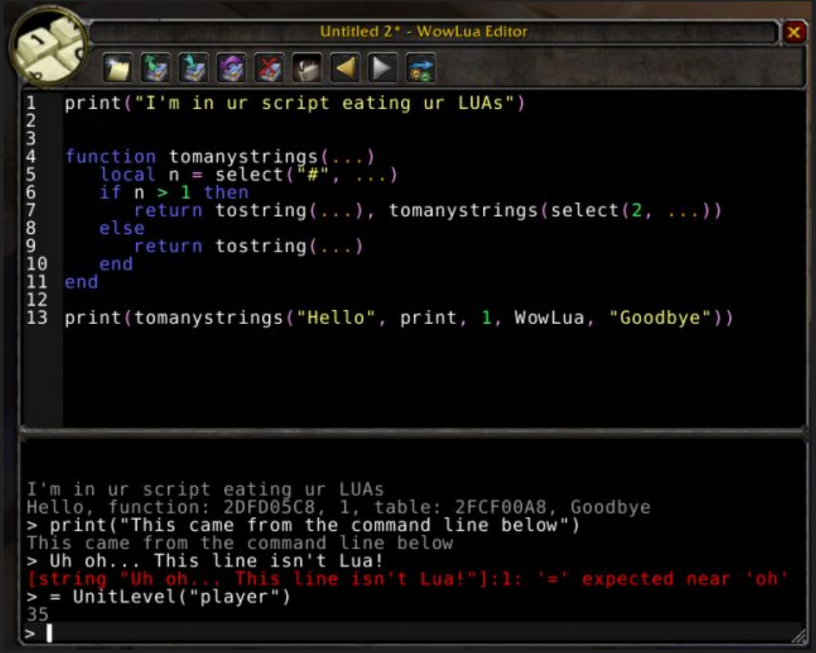
WHY LUA?

Miniscule size



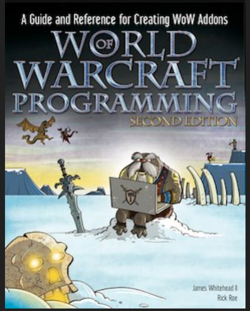
WHY LUA?

Game Scripting



```lua
print("I'm in ur script eating ur LUAs")


function tomanystrings(...)
   local n = select("#", ...)
   if n > 1 then
      return tostring(...), tomanystrings(select(2, ...))
   else
      return tostring(...)
   end
end

print(tomanystrings("Hello", print, 1, WowLua, "Goodbye"))
```

```
I'm in ur script eating ur LUAs
Hello, function: 2DFD05C8, 1, table: 2FCF00A8, Goodbye
> print("This came from the command line below")
This came from the command line below
> Uh oh... This line isn't Lua!
[string "Uh oh... This line isn't Lua!"]:1: '=' expected near 'oh'
> = UnitLevel("player")
35
>
```

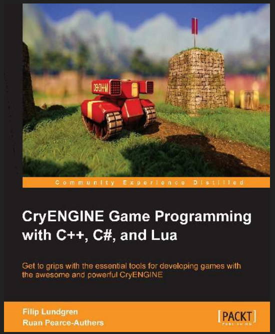


A Guide and Reference for Creating WoW Addons

WORLD OF WARCRAFT PROGRAMMING

SECOND EDITION

James Whitehead II
Rick Roe



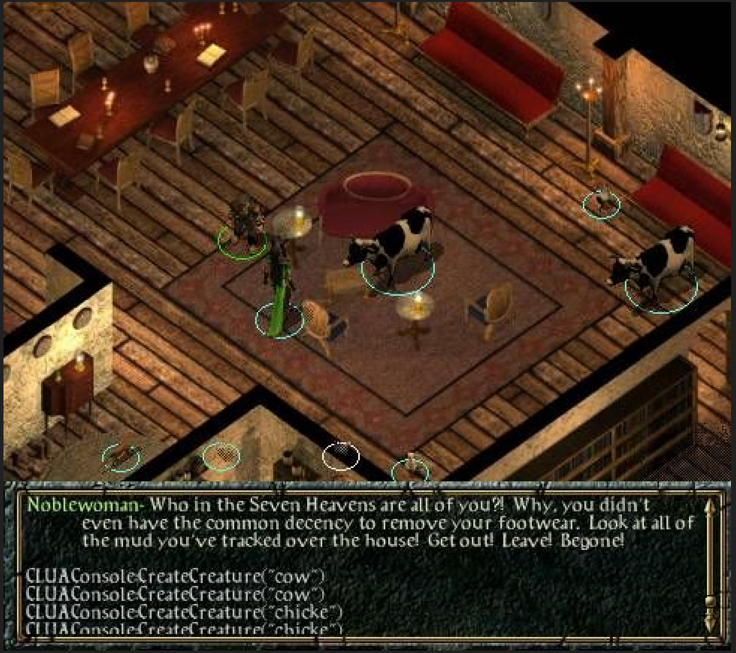
Community Experience Distilled

CryENGINE Game Programming with C++, C#, and Lua

Get to grips with the essential tools for developing games with the awesome and powerful CryENGINE

Filip Lundgren
Ruan Pearce-Authers

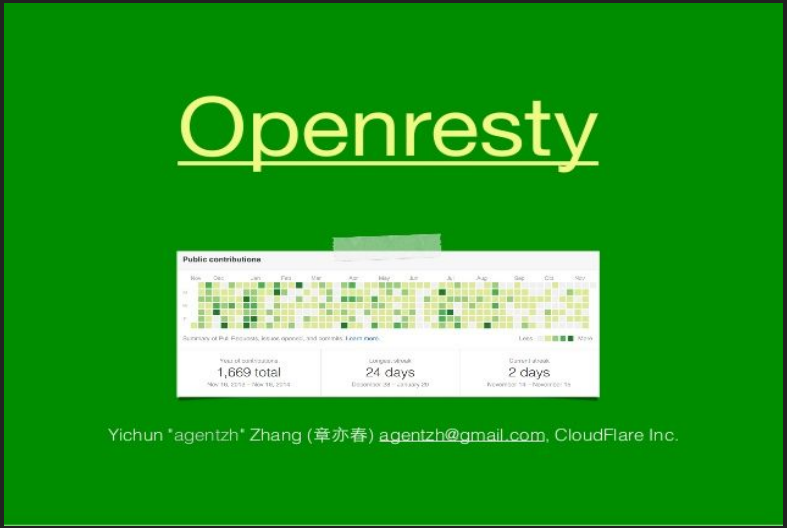
[PACKT]



```
Noblewoman- Who in the Seven Heavens are all of you?! Why, you didn't
        even have the common decency to remove your footwear. Look at all of
        the mud you've tracked over the house! Get out! Leave! Begone!

CLUAConsoleCreateCreature("cow")
CLUAConsoleCreateCreature("cow")
CLUAConsoleCreateCreature("chicke")
CLUAConsoleCreateCreature("chicke")
```

WHY LUA?

Web Development



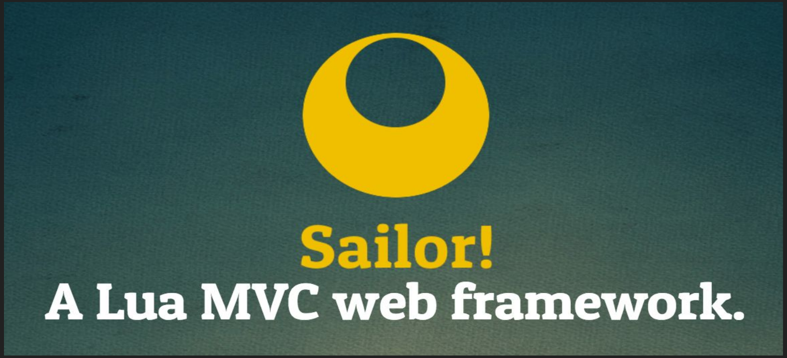
Openresty

1,669 total

24 days

2 days

Yichun "agentzh" Zhang (章亦春) agentzh@gmail.com, CloudFlare Inc.



Sailor!

A Lua MVC web framework.

WHY LUA?

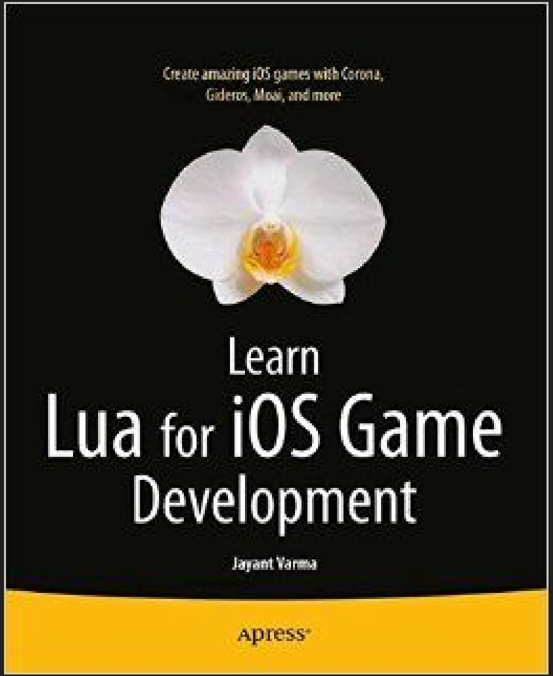
Game Development

DEFOLD

MOAI

GIDEROS MOBILE

LÖVE



Create amazing iOS games with Corona, Gideros, Moai, and more

Learn Lua for iOS Game Development

Jayant Varma

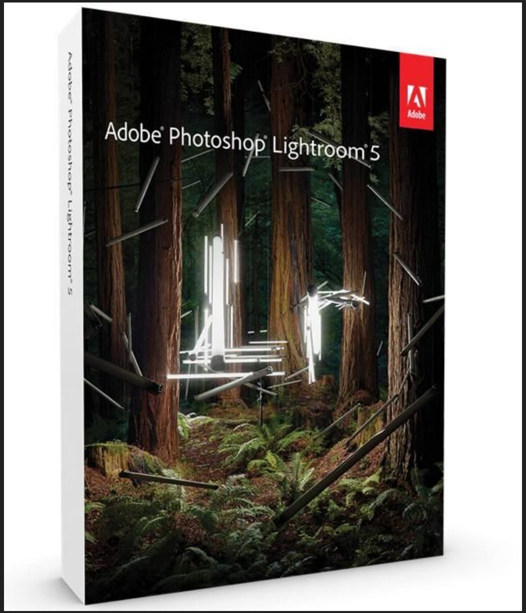
Apress

WHY LUA?

And More!

foldit

Solve Puzzles for Science



torch

A SCIENTIFIC COMPUTING FRAMEWORK FOR LUAJIT

neovim

Related Work

1. Unfinished Lua target by Russel Weir (2008) - Partial support for Lua 5.1 in Haxe 2
2. hx-lua by Matt Tuttle (2012) - Run Lua code inside C++/Neko targets
3. LuaXe by Peyty (2014) - Partial support for Lua 5.1 in Haxe 3 as a custom javascript target*
4. hxpico8 by Vadim Dyachenko (2015) - Run an experimental/limited version of Lua for a virtual console.
5. linc-luajit by RudenkoArts (2016) - @:native bindings for hxcpp/linc
6. A Comparison of Neko and Lua by Nicolas Canasse

* Peyty/Oleg provided much needed support and ideas for this project, thanks!

Lua is Similar to Javascript

- One numeric type
- Strong similarities between metatable and prototype
- First class functions
- Functions are closures
- Hashes take a bracket notation, and can function as arrays
- Functions may accept variable arguments
- *Haxe Lua is based off of earlier Haxe Javascript work*

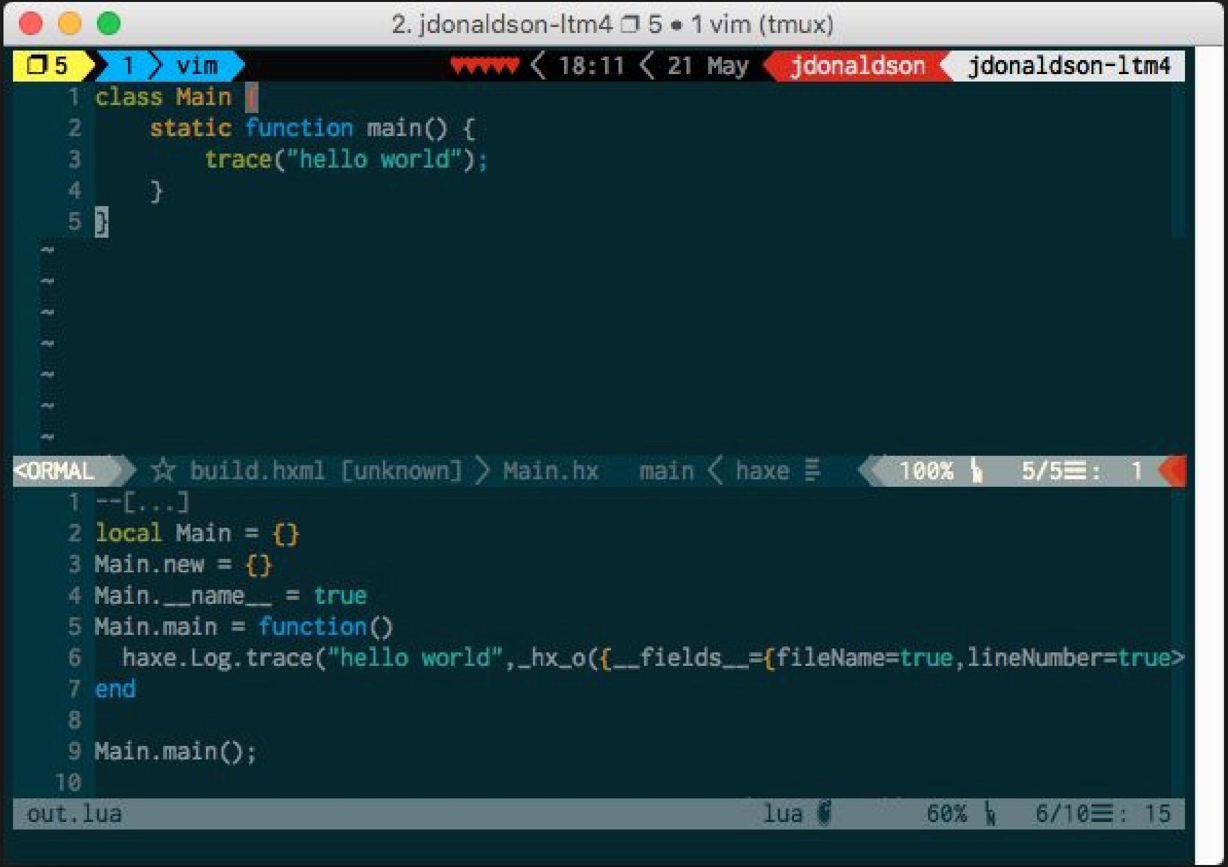
Lua is *very Simple*

- No Integer types (only floats)*
- No boolean operators (special bitops library)*
- No distinction between hashes and arrays
- No distinction between nil and “missing”
- No zero based indexes
- No regular expressions
- No system libraries
- No networking
- No OOP (metatables instead)
- No UTF8

* 5.2 and prior

Hello World

- Simple main()
- Trace == print
- All classes local
- Objects use special _hx_o helper
- __name__ for reflection

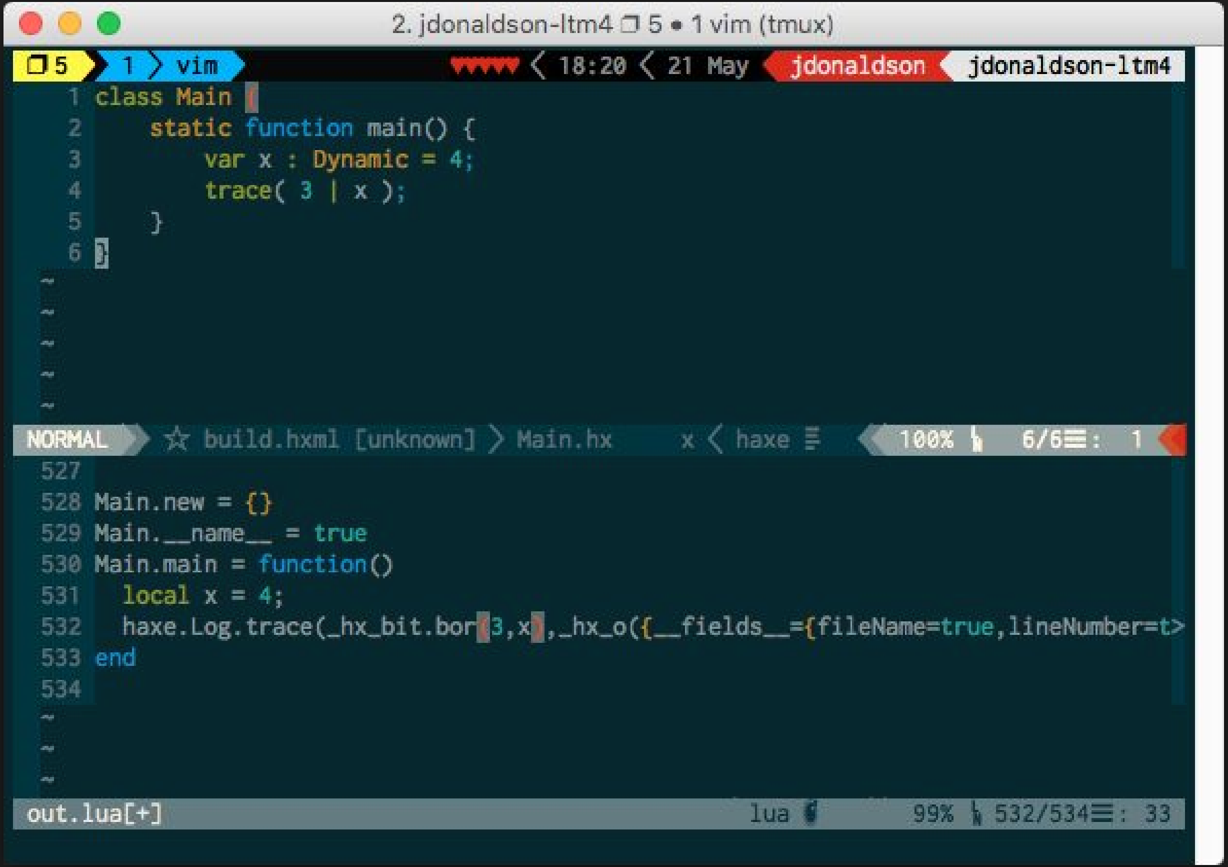


```
class Main {
    static function main() {
        trace("hello world");
    }
}
```

```
--[...]
local Main = {}
Main.new = {}
Main.__name__ = true
Main.main = function()
  haxe.Log.trace("hello world",_hx_o({__fields__={fileName=true,lineNumber=true>
end

Main.main();

```

BitOps

- Bit operators turn into bit methods
- var =~ local



2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)

```
class Main {
    static function main() {
        var x : Dynamic = 4;
        trace( 3 | x );
    }
}
```

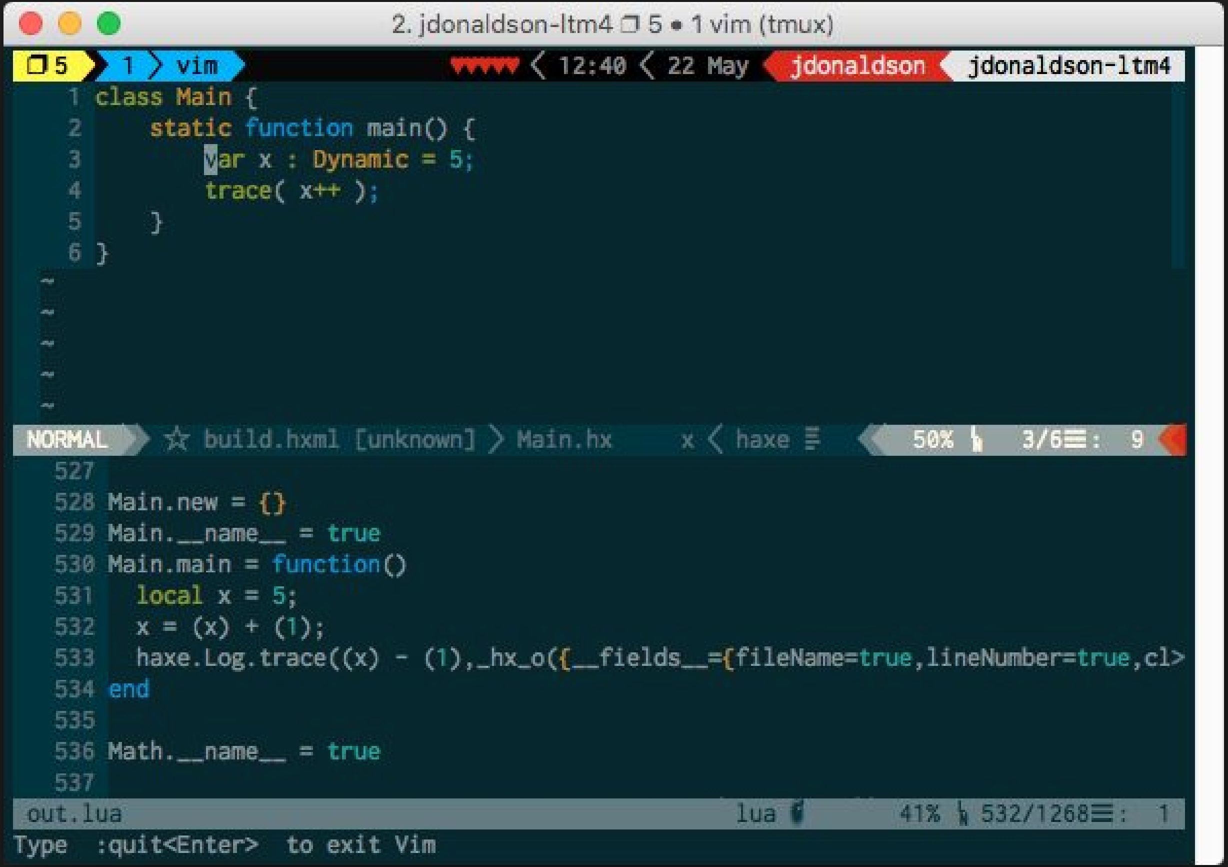
NORMAL ☆ build.hxml [unknown] 〉 Main.hx x 〈 haxe 100% 6/6 : 1

```
527
528 Main.new = {}
529 Main.__name__ = true
530 Main.main = function()
531   local x = 4;
532   haxe.Log.trace(_hx_bit.bor(3,x),_hx_o({__fields__={fileName=true,lineNumber=t>
533 end
534
```

out.lua[+] lua 99% 532/534 : 33

Unops

- Transform unary operators to one or more statements

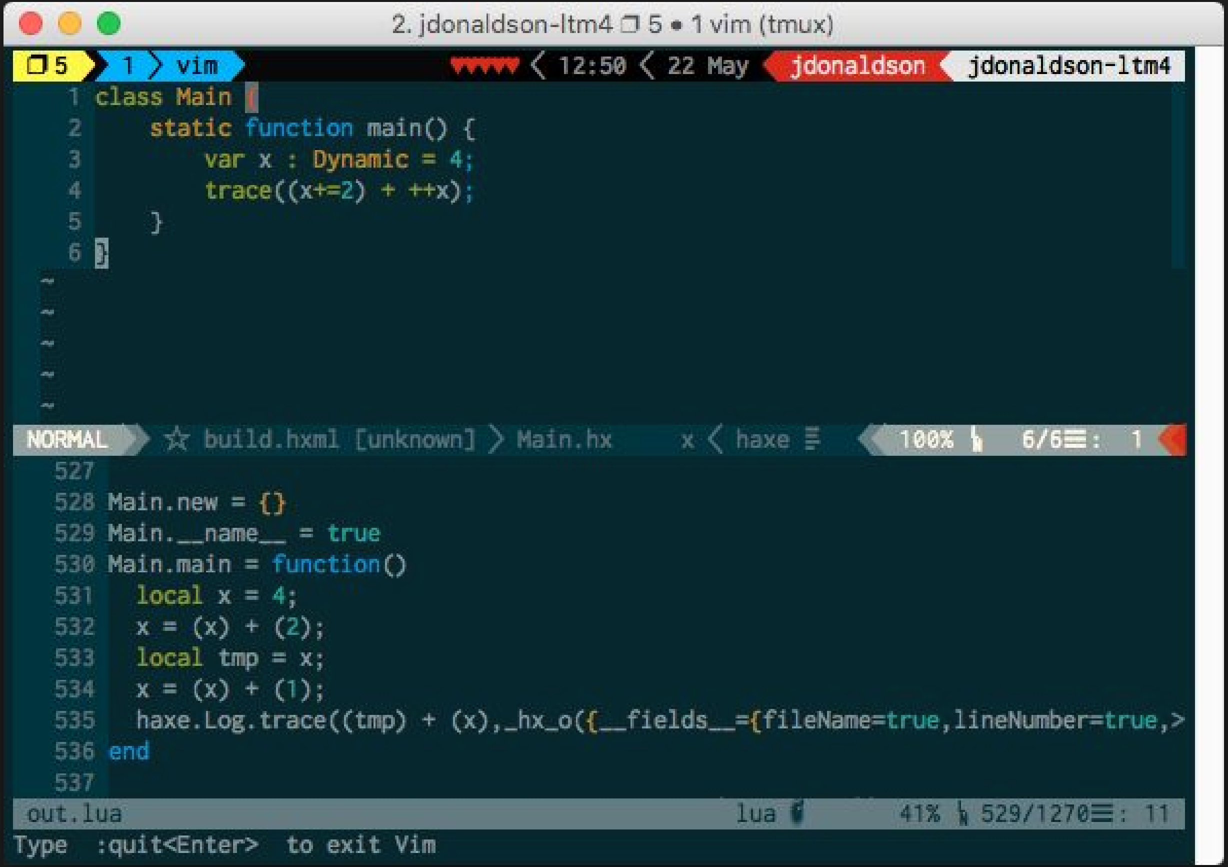


```
class Main {
    static function main() {
        var x : Dynamic = 5;
        trace( x++ );
    }
}
```

```

Main.new = {}
Main.__name__ = true
Main.main = function()
  local x = 5;
  x = (x) + (1);
  haxe.Log.trace((x) - (1),_hx_o({__fields__={fileName=true,lineNumber=true,cl>
end

Math.__name__ = true

```

Unops

- Transform unary operators to one or more statements
- Deconstruct expression


2. jdonaldson-ltm4 ❐ 5 ● 1 vim (tmux)
❐ 5 1 vim ♥♥♥♥♥ 12:50 22 May jdonaldson jdonaldson-ltm4

```
class Main {
    static function main() {
        var x : Dynamic = 4;
        trace((x+=2) + ++x);
    }
}
```

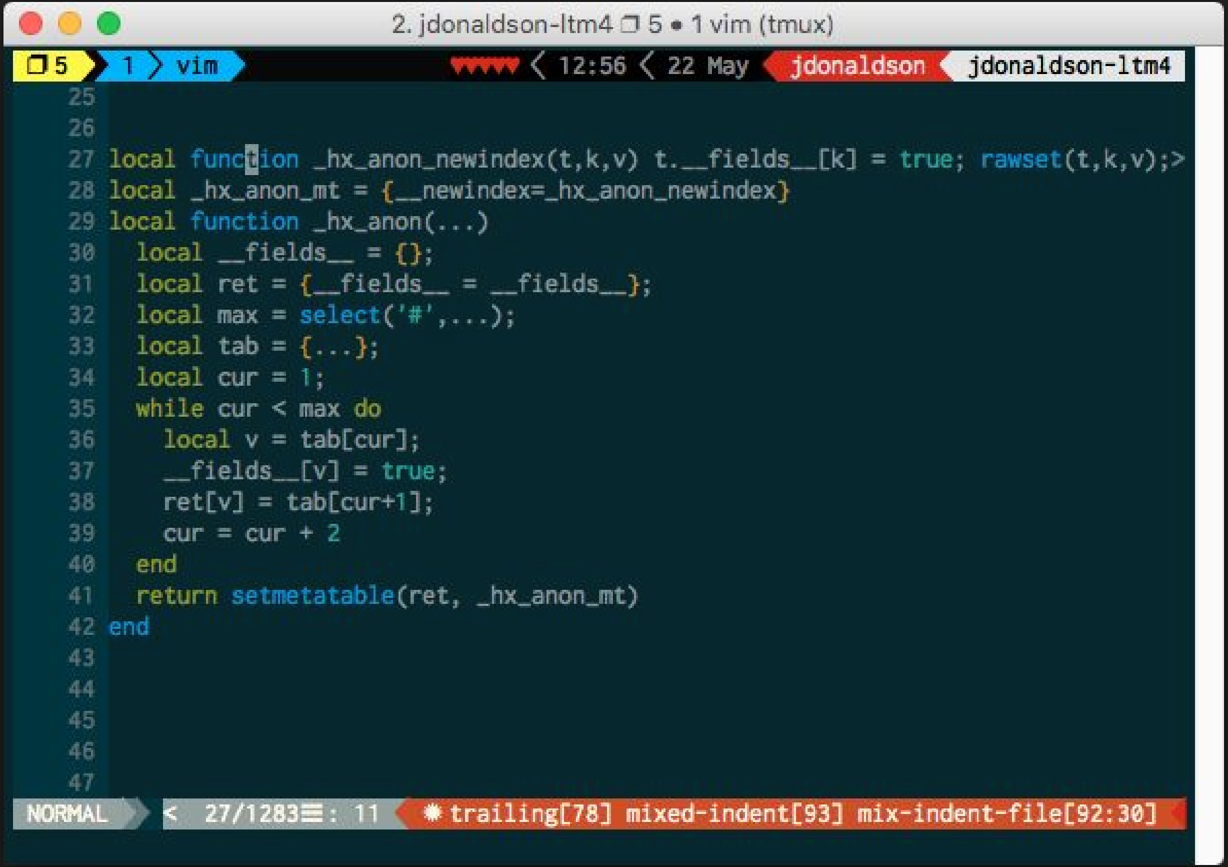
NORMAL ☆ build.hxml [unknown] Main.hx x haxe 100% 6/6 : 1

```

Main.new = {}
Main.__name__ = true
Main.main = function()
  local x = 4;
  x = (x) + (2);
  local tmp = x;
  x = (x) + (1);
  haxe.Log.trace((tmp) + (x),_hx_o({__fields__={fileName=true,lineNumber=true,>
end

```

out.lua lua 41% 529/1270 : 11
Type :quit<Enter> to exit Vim

Anon

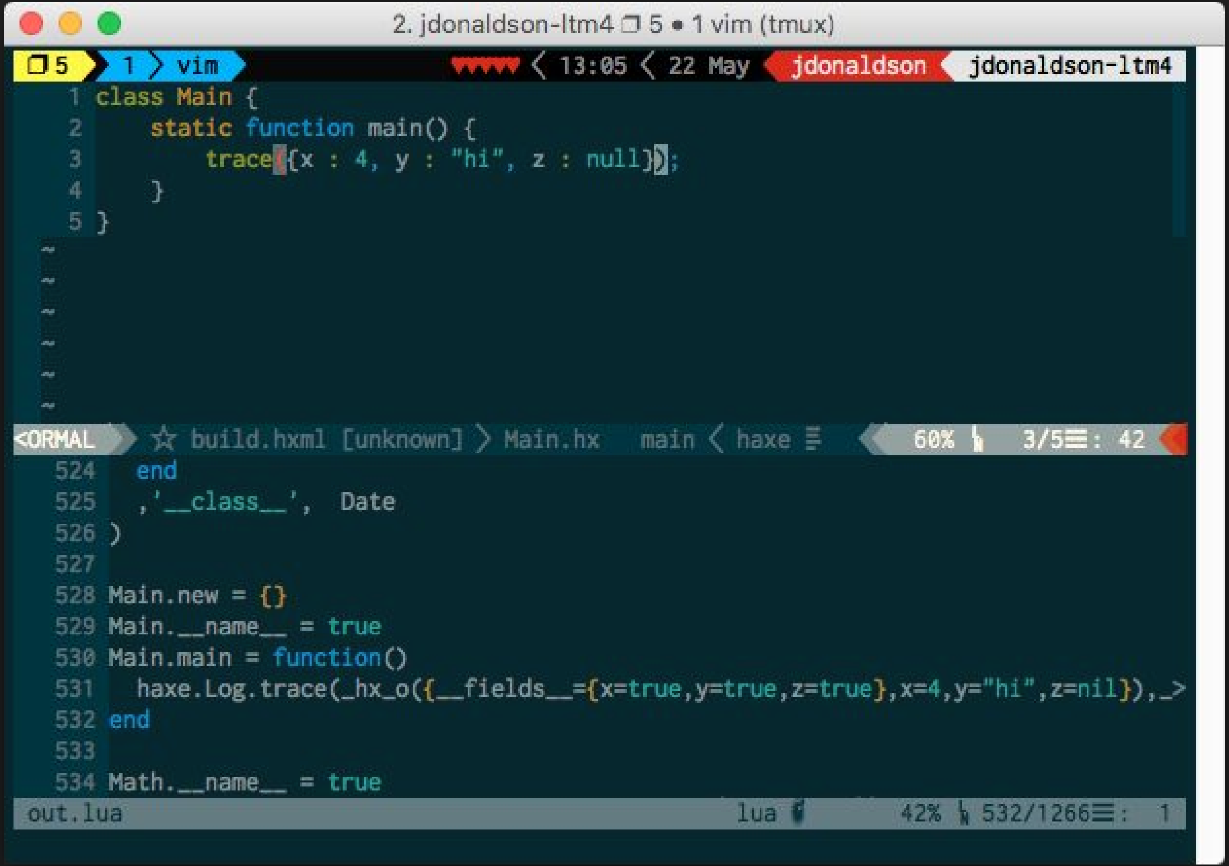
- {x = null}
  x is *not* detectable as empty field.
- Replace default table with new _hx_anon impl



```
2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)
❐ 5  1  vim                ♥♥♥♥♥  12:56  22 May  jdonaldson  jdonaldson-ltm4


local function _hx_anon_newindex(t,k,v) t.__fields__[k] = true; rawset(t,k,v);>
local _hx_anon_mt = {__newindex=_hx_anon_newindex}
local function _hx_anon(...)
  local __fields__ = {};
  local ret = {__fields__ = __fields__};
  local max = select('#',...);
  local tab = {...};
  local cur = 1;
  while cur < max do
    local v = tab[cur];
    __fields__[v] = true;
    ret[v] = tab[cur+1];
    cur = cur + 2
  end
  return setmetatable(ret, _hx_anon_mt)
end





NORMAL  <  27/1283☰ : 11  ✹ trailing[78] mixed-indent[93] mix-indent-file[92:30]
```

Anon

- {x = null}
  x is *not* detectable
  as empty field.
- Replace default
  table with new
  _hx_anon impl
- Field presence is
  stored in a separate
  sub-table



2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)

❐ 5 〉 1 〉 vim   ♥♥♥♥♥ 〈 13:05 〈 22 May jdonaldson jdonaldson-ltm4

```haxe
class Main {
    static function main() {
        trace({x : 4, y : "hi", z : null});
    }
}
```

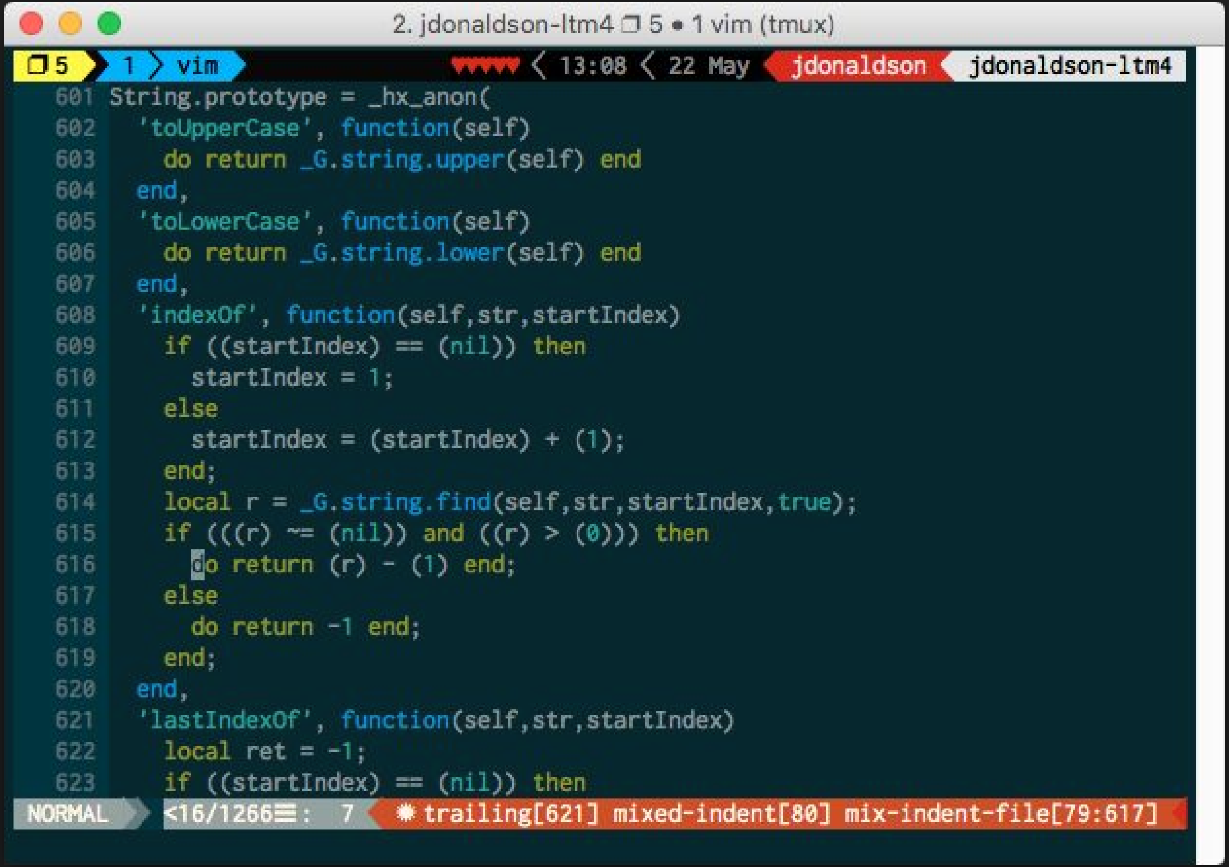
NORMAL 〉 ☆ build.hxml [unknown] 〉 Main.hx   main 〈 haxe   60% 3/5 : 42

```lua
  end
  ,'__class__',  Date
)

Main.new = {}
Main.__name__ = true
Main.main = function()
  haxe.Log.trace(_hx_o({__fields__={x=true,y=true,z=true},x=4,y="hi",z=nil}),_>
end

Math.__name__ = true
```

out.lua   lua   42% 532/1266 : 1

Anon

- {x = null}
  x is *not* detectable as empty field.
- Replace default table with new _hx_anon impl
- Field presence is stored in a separate sub-table
- Used for base in all OOP, Class definitions



2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)

❐ 5 | 1 | vim | ♥♥♥♥♥ | 13:08 | 22 May | jdonaldson | jdonaldson-ltm4

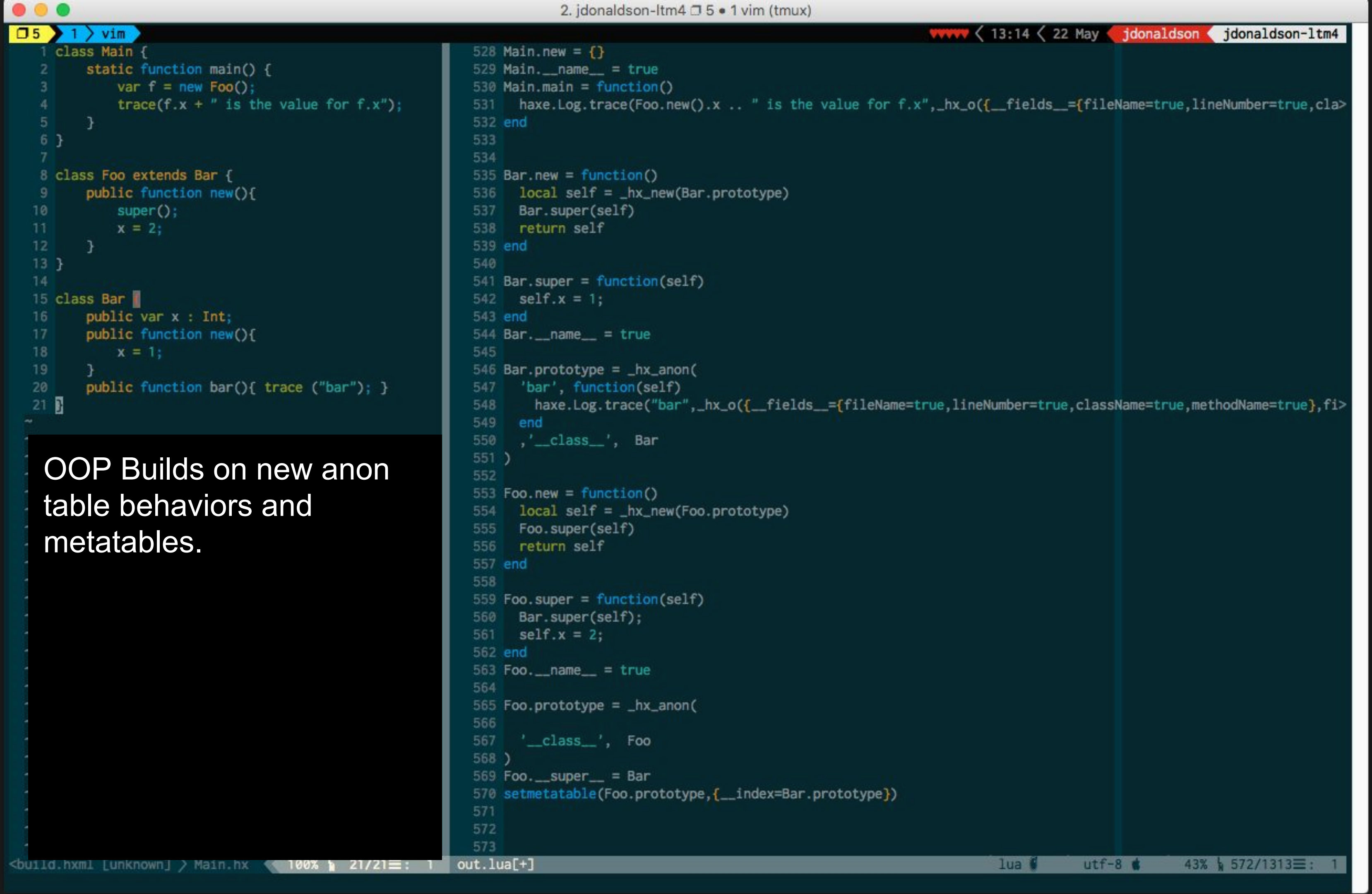
```
String.prototype = _hx_anon(
  'toUpperCase', function(self)
    do return _G.string.upper(self) end
  end,
  'toLowerCase', function(self)
    do return _G.string.lower(self) end
  end,
  'indexOf', function(self,str,startIndex)
    if ((startIndex) == (nil)) then
      startIndex = 1;
    else
      startIndex = (startIndex) + (1);
    end;
    local r = _G.string.find(self,str,startIndex,true);
    if (((r) ~= (nil)) and ((r) > (0))) then
      do return (r) - (1) end;
    else
      do return -1 end;
    end;
  end,
  'lastIndexOf', function(self,str,startIndex)
    local ret = -1;
    if ((startIndex) == (nil)) then
```

```
NORMAL  <16/1266☰:  7  ✹ trailing[621] mixed-indent[80] mix-indent-file[79:617]
```

Quick Overview : Metatables

- Adds special functionality to tables
- **__index** overrides missing value behavior
- **__newindex** overrides new value behavior
- **__concat, __add, __eq, __lt, __gt** : overrides comparison/operator behavior
- **__call** : allow table to be called as a function

```
local x = {}
local f = function() return 4 end;
local mt = { __index = f };
setmetatable(x, f);
print(x.anyfield); -- “4”
```



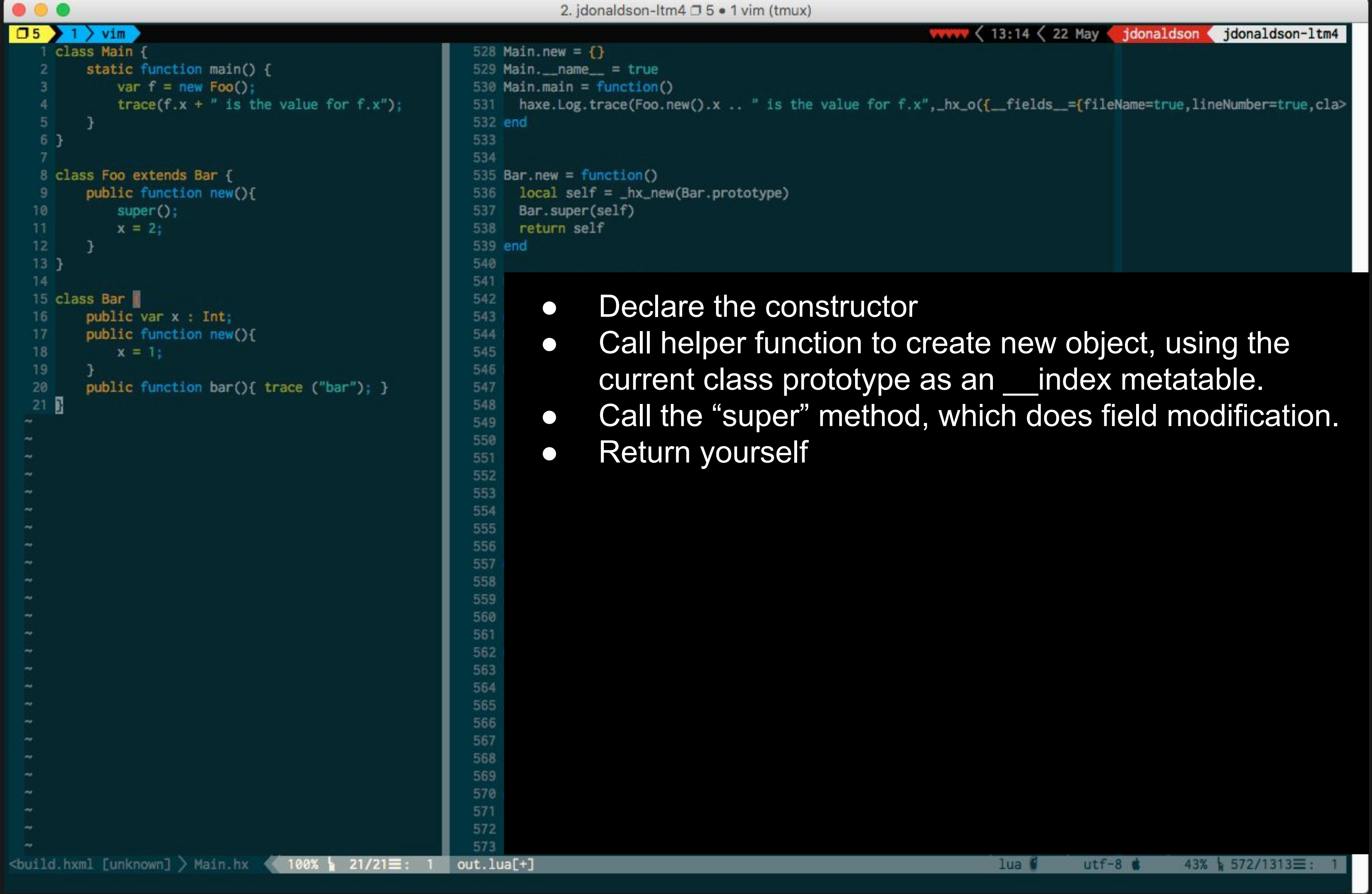
```
2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)
❐ 5  1  vim                                    ♥♥♥♥♥ 〈 13:14 〈 22 May  jdonaldson  jdonaldson-ltm4
```

```haxe
class Main {
    static function main() {
        var f = new Foo();
        trace(f.x + " is the value for f.x");
    }
}

class Foo extends Bar {
    public function new(){
        super();
        x = 2;
    }
}

class Bar {
    public var x : Int;
    public function new(){
        x = 1;
    }
    public function bar(){ trace ("bar"); }
}
```

OOP Builds on new anon table behaviors and metatables.

```lua
Main.new = {}
Main.__name__ = true
Main.main = function()
  haxe.Log.trace(Foo.new().x .. " is the value for f.x",_hx_o({__fields__={fileName=true,lineNumber=true,cla>
end


Bar.new = function()
  local self = _hx_new(Bar.prototype)
  Bar.super(self)
  return self
end

Bar.super = function(self)
  self.x = 1;
end
Bar.__name__ = true

Bar.prototype = _hx_anon(
  'bar', function(self)
    haxe.Log.trace("bar",_hx_o({__fields__={fileName=true,lineNumber=true,className=true,methodName=true},fi>
  end
  ,'__class__',  Bar
)

Foo.new = function()
  local self = _hx_new(Foo.prototype)
  Foo.super(self)
  return self
end

Foo.super = function(self)
  Bar.super(self);
  self.x = 2;
end
Foo.__name__ = true

Foo.prototype = _hx_anon(

  '__class__',  Foo
)
Foo.__super__ = Bar
setmetatable(Foo.prototype,{__index=Bar.prototype})
```

```
<build.hxml [unknown] 〉 Main.hx   100%  21/21☰:  1   out.lua[+]              lua   utf-8   43%  572/1313☰:  1
```



```
2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)
```

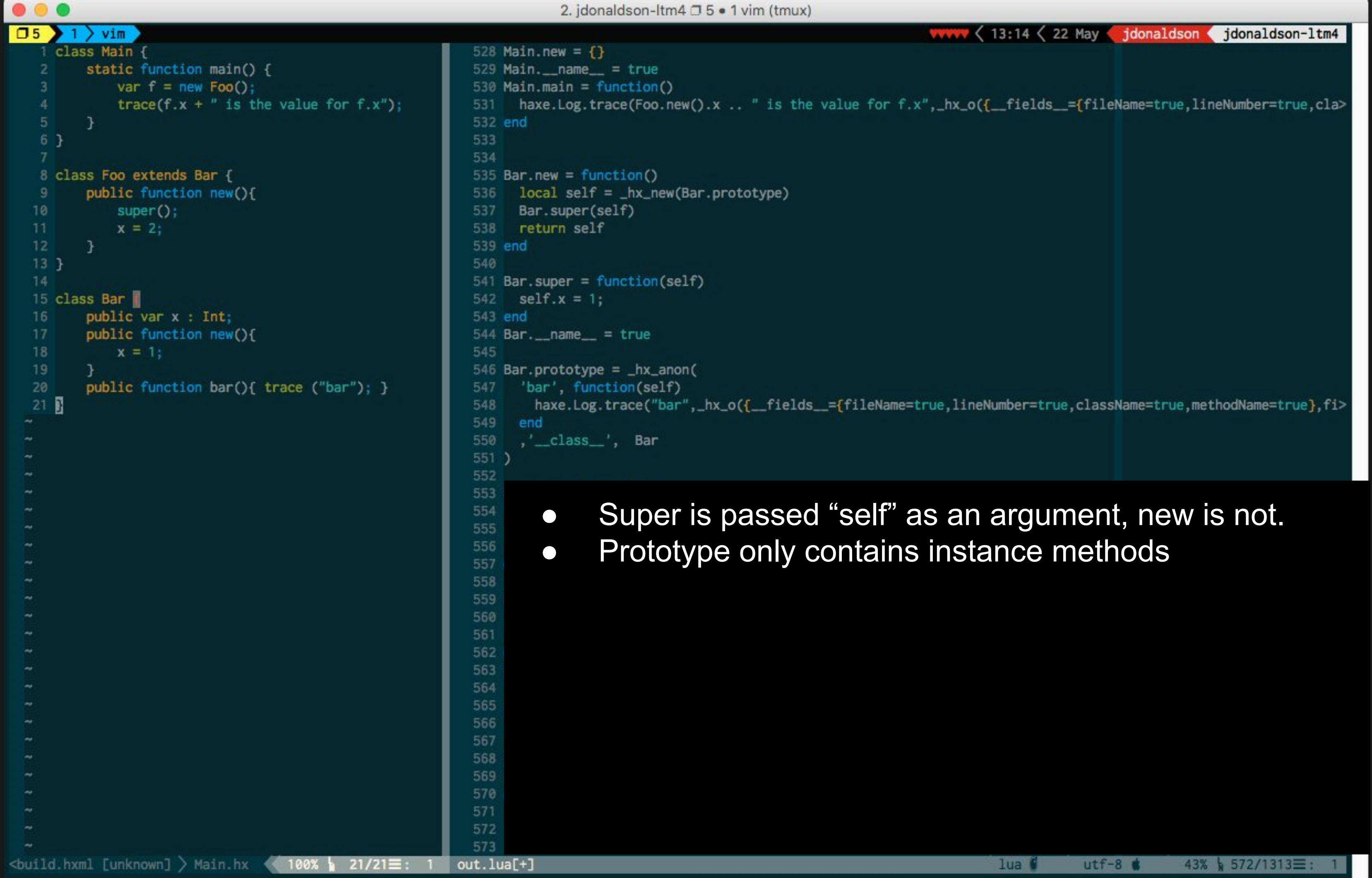
```
❐ 5  1  vim                                    13:14  22 May  jdonaldson  jdonaldson-ltm4
```

```haxe
class Main {
    static function main() {
        var f = new Foo();
        trace(f.x + " is the value for f.x");
    }
}

class Foo extends Bar {
    public function new(){
        super();
        x = 2;
    }
}

class Bar {
    public var x : Int;
    public function new(){
        x = 1;
    }
    public function bar(){ trace ("bar"); }
}
```

```lua
Main.new = {}
Main.__name__ = true
Main.main = function()
  haxe.Log.trace(Foo.new().x .. " is the value for f.x",_hx_o({__fields__={fileName=true,lineNumber=true,cla>
end


Bar.new = function()
  local self = _hx_new(Bar.prototype)
  Bar.super(self)
  return self
end
```

```
<build.hxml [unknown] > Main.hx   100%  21/21☰ :  1   out.lua[+]                  lua   utf-8   43%  572/1313☰ :  1
```

- Declare the constructor
- Call helper function to create new object, using the current class prototype as an __index metatable.
- Call the “super” method, which does field modification.
- Return yourself

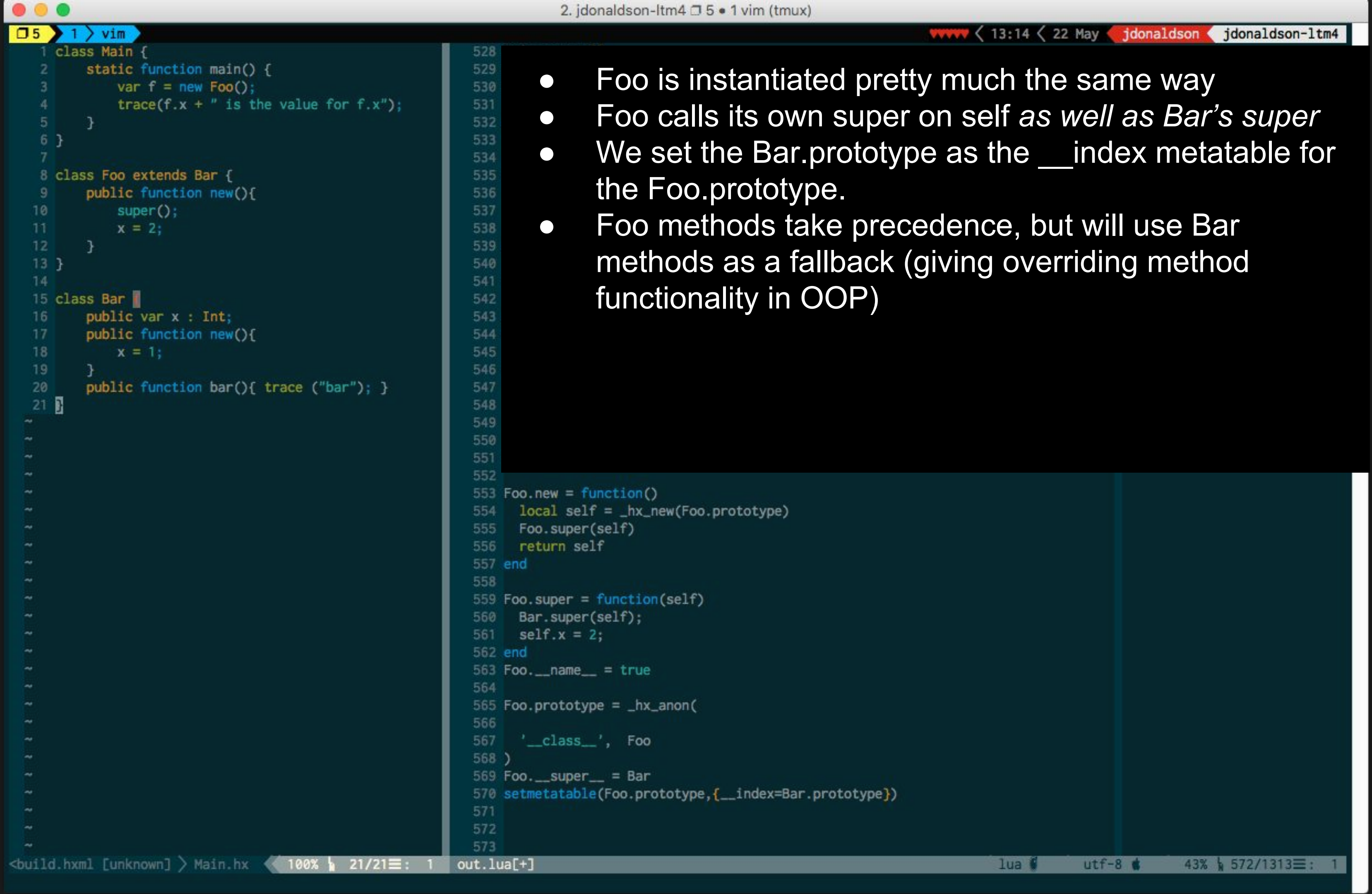


```
class Main {
    static function main() {
        var f = new Foo();
        trace(f.x + " is the value for f.x");
    }
}

class Foo extends Bar {
    public function new(){
        super();
        x = 2;
    }
}

class Bar {
    public var x : Int;
    public function new(){
        x = 1;
    }
    public function bar(){ trace ("bar"); }
}
```

```
Main.new = {}
Main.__name__ = true
Main.main = function()
  haxe.Log.trace(Foo.new().x .. " is the value for f.x",_hx_o({__fields__={fileName=true,lineNumber=true,cla>
end


Bar.new = function()
  local self = _hx_new(Bar.prototype)
  Bar.super(self)
  return self
end

Bar.super = function(self)
  self.x = 1;
end
Bar.__name__ = true

Bar.prototype = _hx_anon(
  'bar', function(self)
    haxe.Log.trace("bar",_hx_o({__fields__={fileName=true,lineNumber=true,className=true,methodName=true},fi>
  end
  ,'__class__',  Bar
)
```

- Super is passed “self” as an argument, new is not.
- Prototype only contains instance methods

- Foo is instantiated pretty much the same way
- Foo calls its own super on self *as well as Bar's super*
- We set the Bar.prototype as the __index metatable for the Foo.prototype.
- Foo methods take precedence, but will use Bar methods as a fallback (giving overriding method functionality in OOP)



```
2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)
❐ 5  1  vim                                   13:14  22 May  jdonaldson  jdonaldson-ltm4
```

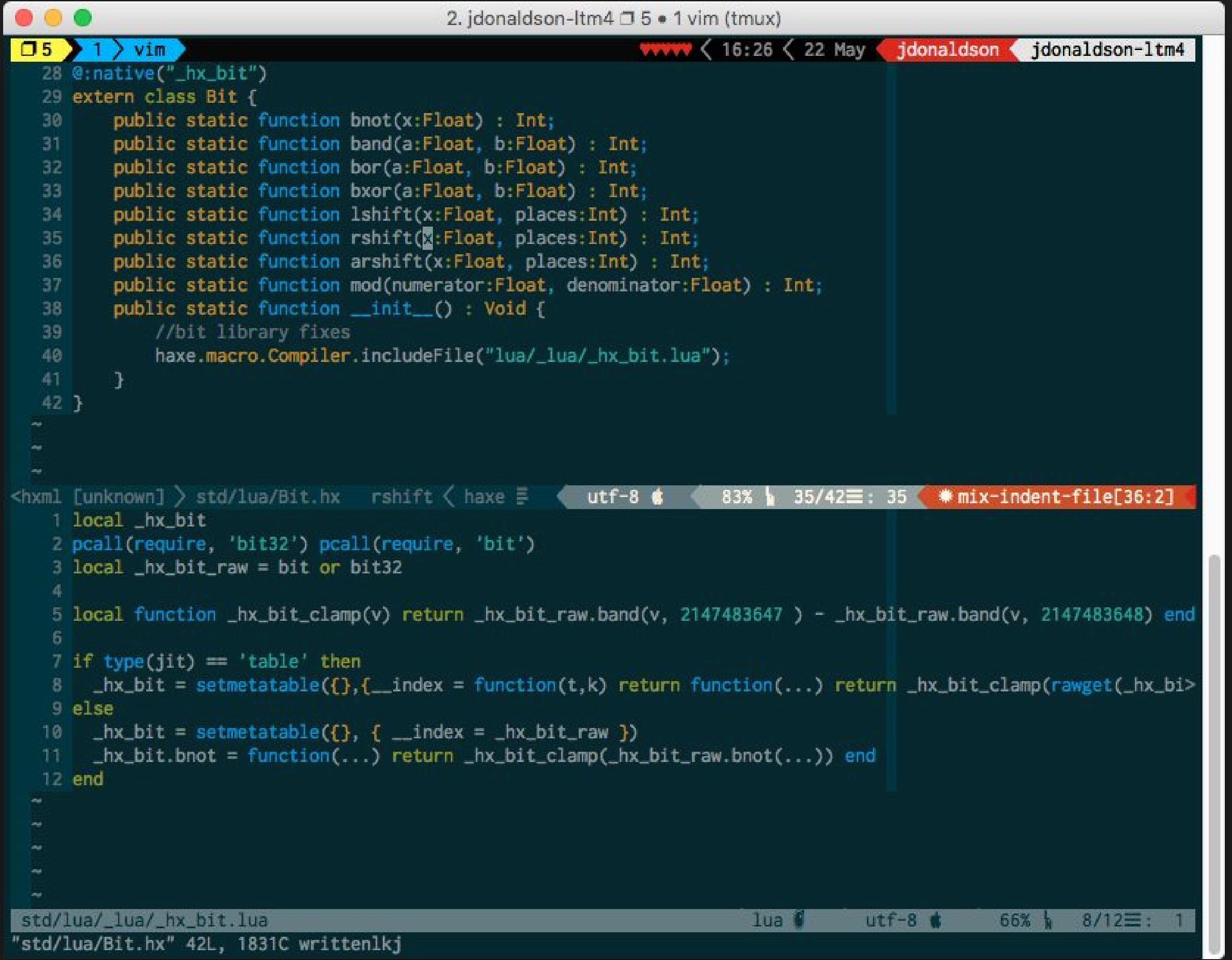
```haxe
class Main {
    static function main() {
        var f = new Foo();
        trace(f.x + " is the value for f.x");
    }
}

class Foo extends Bar {
    public function new(){
        super();
        x = 2;
    }
}

class Bar {
    public var x : Int;
    public function new(){
        x = 1;
    }
    public function bar(){ trace ("bar"); }
}
```

```lua
Foo.new = function()
  local self = _hx_new(Foo.prototype)
  Foo.super(self)
  return self
end

Foo.super = function(self)
  Bar.super(self);
  self.x = 2;
end
Foo.__name__ = true

Foo.prototype = _hx_anon(

  '__class__',  Foo
)
Foo.__super__ = Bar
setmetatable(Foo.prototype,{__index=Bar.prototype})
```

```
<build.hxml [unknown] > Main.hx  100%  21/21 :  1   out.lua[+]      lua   utf-8   43%  572/1313 :  1
```

Extern

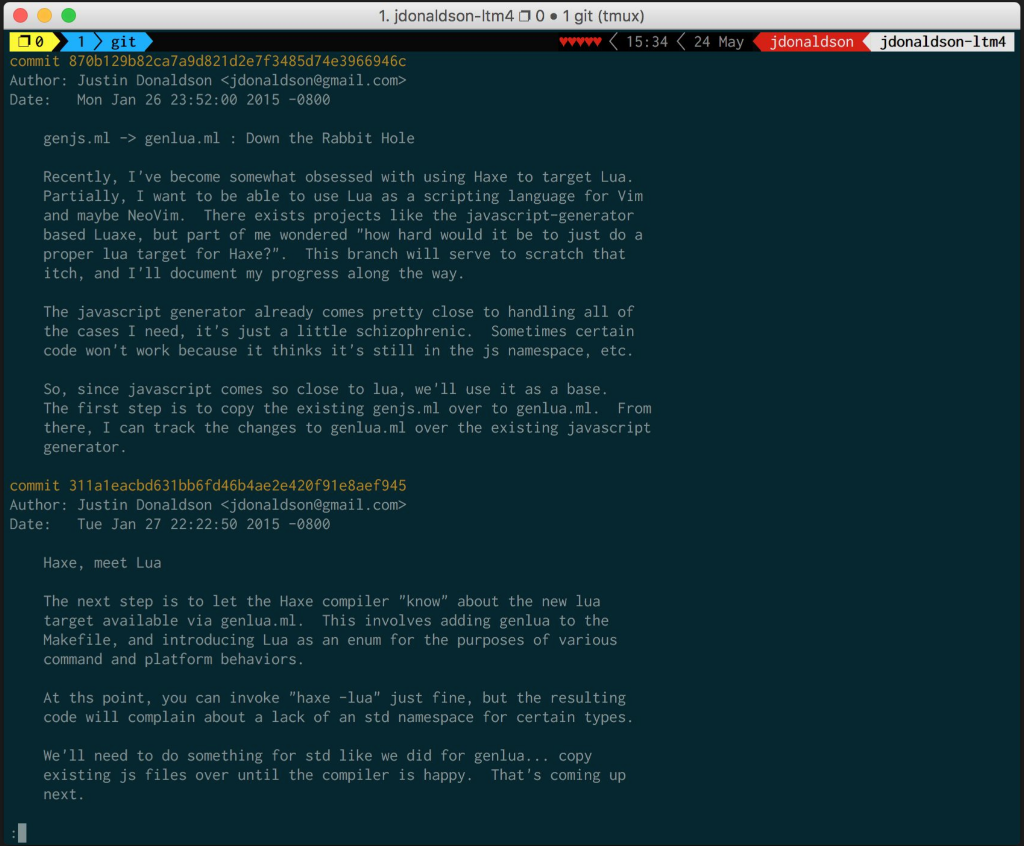
- @:native binds to native interface
- @:expose binds class/method body to global metatable
- @:selfCall allows methods to instead call the module/class
- includeFile adds helper methods in lua



```
2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)
❐ 5  1  vim                                ♥♥♥♥♥  16:26  22 May  jdonaldson  jdonaldson-ltm4
@:native("_hx_bit")
extern class Bit {
    public static function bnot(x:Float) : Int;
    public static function band(a:Float, b:Float) : Int;
    public static function bor(a:Float, b:Float) : Int;
    public static function bxor(a:Float, b:Float) : Int;
    public static function lshift(x:Float, places:Int) : Int;
    public static function rshift(x:Float, places:Int) : Int;
    public static function arshift(x:Float, places:Int) : Int;
    public static function mod(numerator:Float, denominator:Float) : Int;
    public static function __init__() : Void {
        //bit library fixes
        haxe.macro.Compiler.includeFile("lua/_lua/_hx_bit.lua");
    }
}
~
~
~
<hxml [unknown] > std/lua/Bit.hx   rshift < haxe   utf-8   83%  35/42: 35   mix-indent-file[36:2]
local _hx_bit
pcall(require, 'bit32') pcall(require, 'bit')
local _hx_bit_raw = bit or bit32

local function _hx_bit_clamp(v) return _hx_bit_raw.band(v, 2147483647 ) - _hx_bit_raw.band(v, 2147483648) end

if type(jit) == 'table' then
  _hx_bit = setmetatable({},{__index = function(t,k) return function(...) return _hx_bit_clamp(rawget(_hx_bi>
else
  _hx_bit = setmetatable({}, { __index = _hx_bit_raw })
  _hx_bit.bnot = function(...) return _hx_bit_clamp(_hx_bit_raw.bnot(...)) end
end
~
~
~
~
~
std/lua/_lua/_hx_bit.lua                                  lua   utf-8   66%  8/12: 1
"std/lua/Bit.hx" 42L, 1831C writtenlkj
```

History

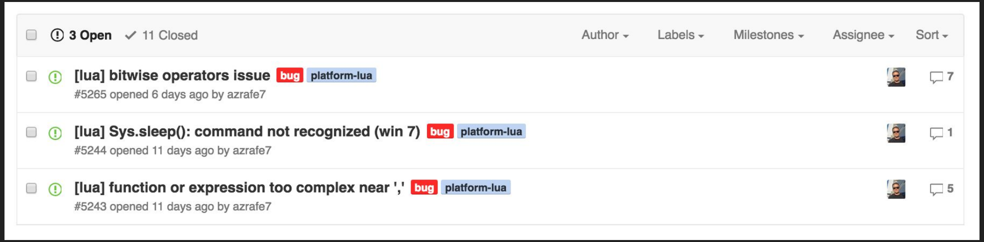
- `git log --reverse --grep Lua`
- Review the lua changes
- “Most” changes get detailed overviews, rationales, observations, concerns, next steps
- Covers development of ~ 1 year


1. jdonaldson-ltm4 ❐ 0 ● 1 git (tmux)
❐ 0
1 git
♥♥♥♥♥ 15:34 24 May jdonaldson jdonaldson-ltm4

```
commit 870b129b82ca7a9d821d2e7f3485d74e3966946c
Author: Justin Donaldson <jdonaldson@gmail.com>
Date:   Mon Jan 26 23:52:00 2015 -0800

    genjs.ml -> genlua.ml : Down the Rabbit Hole

    Recently, I've become somewhat obsessed with using Haxe to target Lua.
    Partially, I want to be able to use Lua as a scripting language for Vim
    and maybe NeoVim.  There exists projects like the javascript-generator
    based Luaxe, but part of me wondered "how hard would it be to just do a
    proper lua target for Haxe?".  This branch will serve to scratch that
    itch, and I'll document my progress along the way.

    The javascript generator already comes pretty close to handling all of
    the cases I need, it's just a little schizophrenic.  Sometimes certain
    code won't work because it thinks it's still in the js namespace, etc.

    So, since javascript comes so close to lua, we'll use it as a base.
    The first step is to copy the existing genjs.ml over to genlua.ml.  From
    there, I can track the changes to genlua.ml over the existing javascript
    generator.

commit 311a1eacbd631bb6fd46b4ae2e420f91e8aef945
Author: Justin Donaldson <jdonaldson@gmail.com>
Date:   Tue Jan 27 22:22:50 2015 -0800

    Haxe, meet Lua

    The next step is to let the Haxe compiler "know" about the new lua
    target available via genlua.ml.  This involves adding genlua to the
    Makefile, and introducing Lua as an enum for the purposes of various
    command and platform behaviors.

    At ths point, you can invoke "haxe -lua" just fine, but the resulting
    code will complain about a lack of an std namespace for certain types.

    We'll need to do something for std like we did for genlua... copy
    existing js files over until the compiler is happy.  That's coming up
    next.

:
```

Still some kinks to work out

- Cannot declare more than 200 local variables in single scope
- Sys api is incomplete
- Null (nil) in string concatenation throws errors
- No first class support for multi-return externs
- Generated Lua could be more clear and compact



3 Open ✓ 11 Closed

Author ▾ Labels ▾ Milestones ▾ Assignee ▾ Sort ▾

[lua] bitwise operators issue bug platform-lua — 7
#5265 opened 6 days ago by azrafe7

[lua] Sys.sleep(): command not recognized (win 7) bug platform-lua — 1
#5244 opened 11 days ago by azrafe7

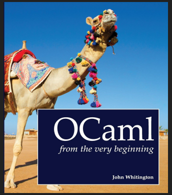
[lua] function or expression too complex near ',' bug platform-lua — 5
#5243 opened 11 days ago by azrafe7

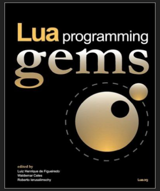
Avoiding Pain And Humiliation

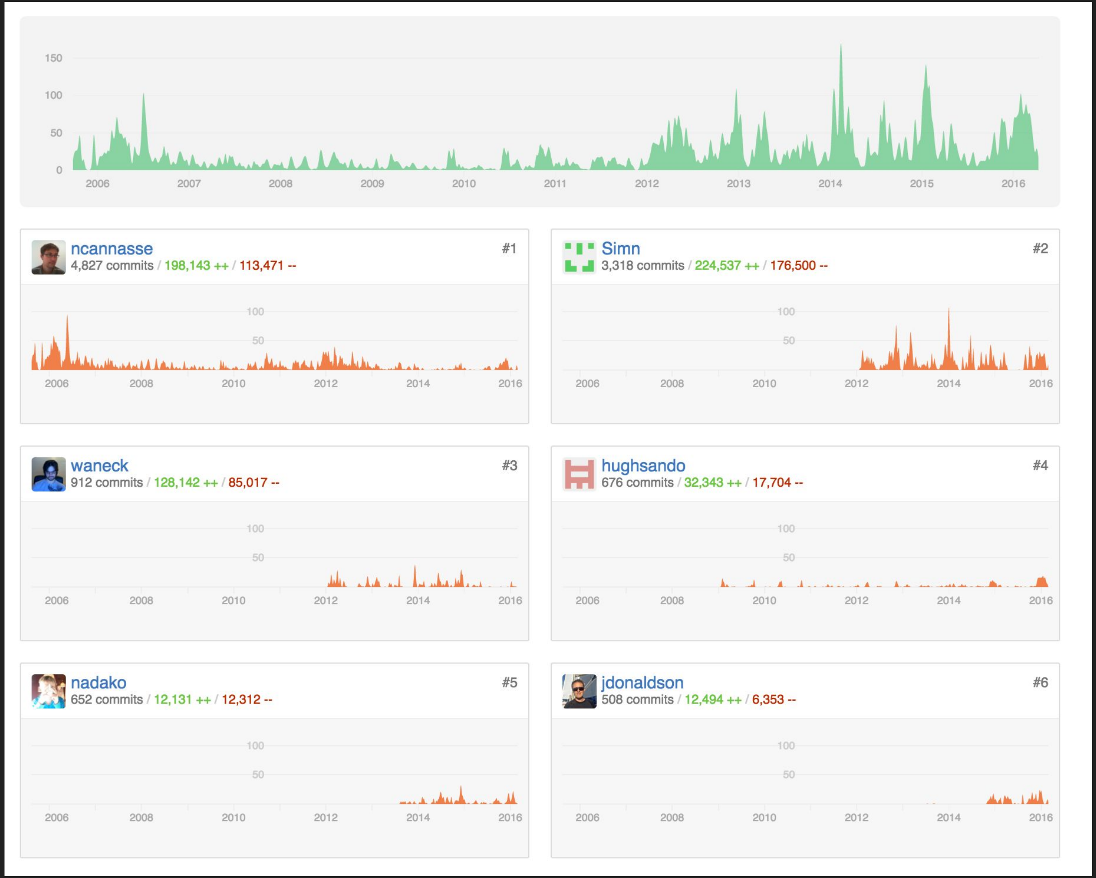
- Don’t use more than 200 local variables (even when workaround is in place).
  - Avoid abstracts/inlines that result in temporary variable creation
- Avoid assigning instance/static methods unnecessarily (e.g. dynamic methods or as fields).
- Avoid using “Lua.arg” or “haxe.extern.Rest” (defeating jit optimizations)
- Use unique variable names in any lua include/__init__ code.

Lessons Learned

- OCaml is not so scary
- Lua has made very few mistakes.
  - It also has very few “batteries” included
- Lua is fragmented... worrisome
  - Lua 5.2, LuaJit, Lua 5.3
- Finding free time with a newborn is extremely easy, then extremely hard.
- NC & Simn are machines







ncannasse #1
4,827 commits / 198,143 ++ / 113,471 --

Simn #2
3,318 commits / 224,537 ++ / 176,500 --

waneck #3
912 commits / 128,142 ++ / 85,017 --

hughsando #4
676 commits / 32,343 ++ / 17,704 --

nadako #5
652 commits / 12,131 ++ / 12,312 --

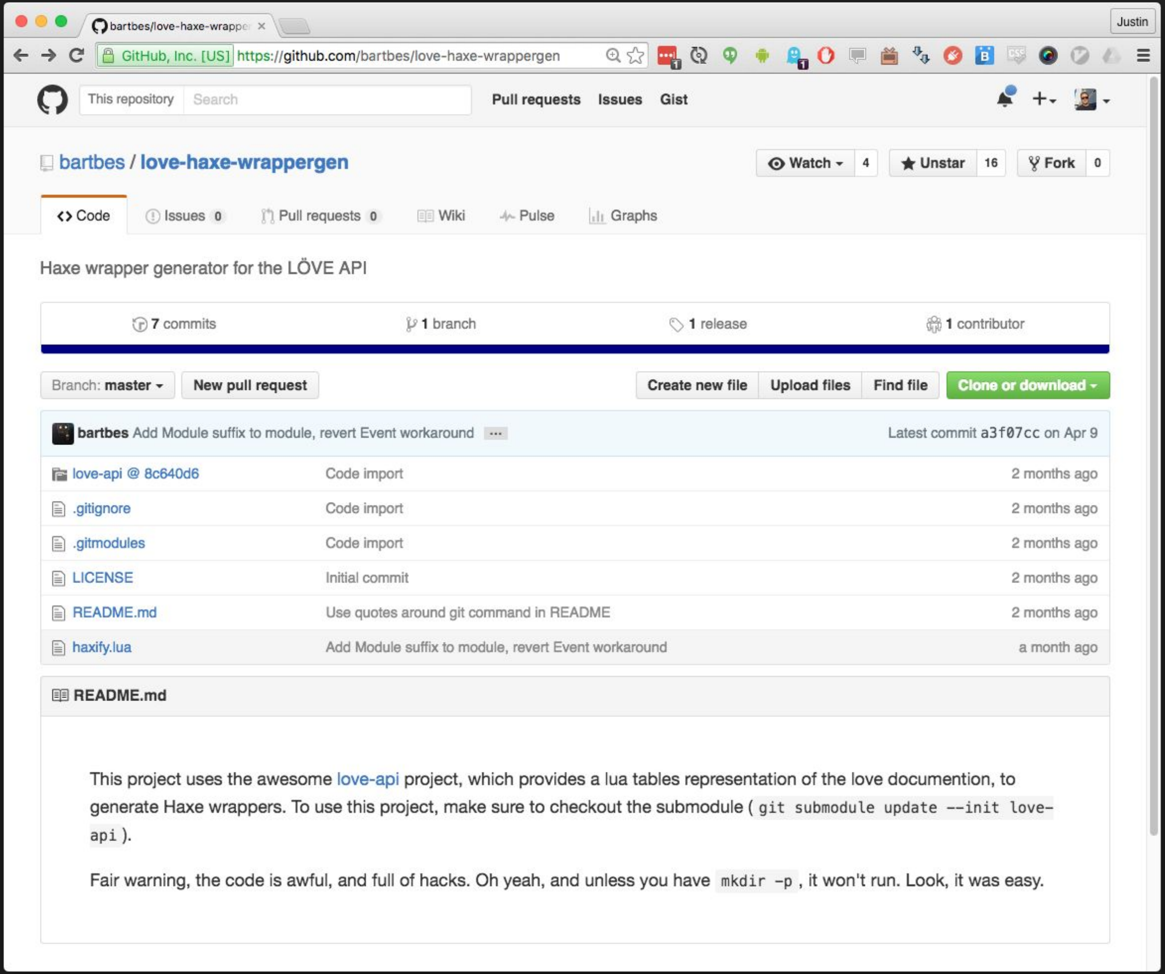
jdonaldson #6
508 commits / 12,494 ++ / 6,353 --

If I were to do it over again...

- Start from scratch instead of copying genjs.ml
- Rewrite expressions rather than emitting Lua code as bare strings
- Better familiarize myself with the utility debug methods in type.ml (s_type_kind, etc)
- Bothered the other core developers more with stupid questions.

Haxe Love

- Love-haxe-wrappergen
- Released ~24 hours after official Haxe Lua announcement



Haxe wrapper generator for the LÖVE API

7 commits | 1 branch | 1 release | 1 contributor

bartbes Add Module suffix to module, revert Event workaround … Latest commit a3f07cc on Apr 9

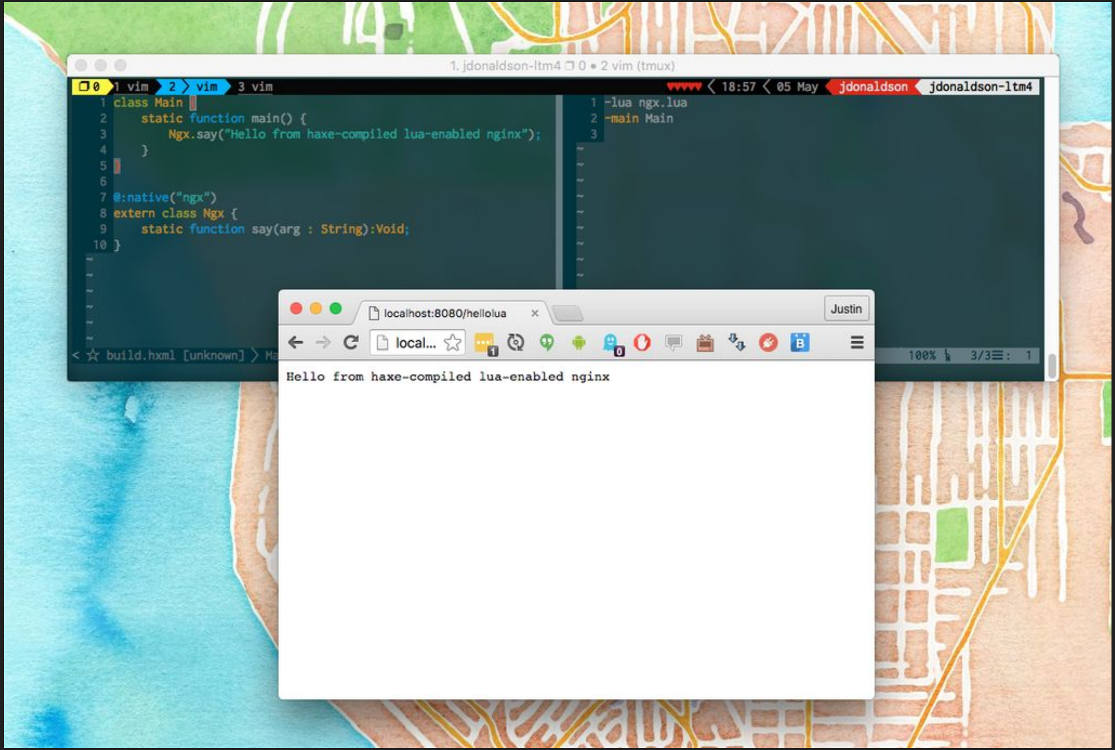
love-api @ 8c640d6	Code import	2 months ago
.gitignore	Code import	2 months ago
.gitmodules	Code import	2 months ago
LICENSE	Initial commit	2 months ago
README.md	Use quotes around git command in README	2 months ago
haxify.lua	Add Module suffix to module, revert Event workaround	a month ago

README.md

This project uses the awesome love-api project, which provides a lua tables representation of the love documention, to generate Haxe wrappers. To use this project, make sure to checkout the submodule (`git submodule update --init love-api`).

Fair warning, the code is awful, and full of hacks. Oh yeah, and unless you have `mkdir -p` , it won't run. Look, it was easy.

Nginhx



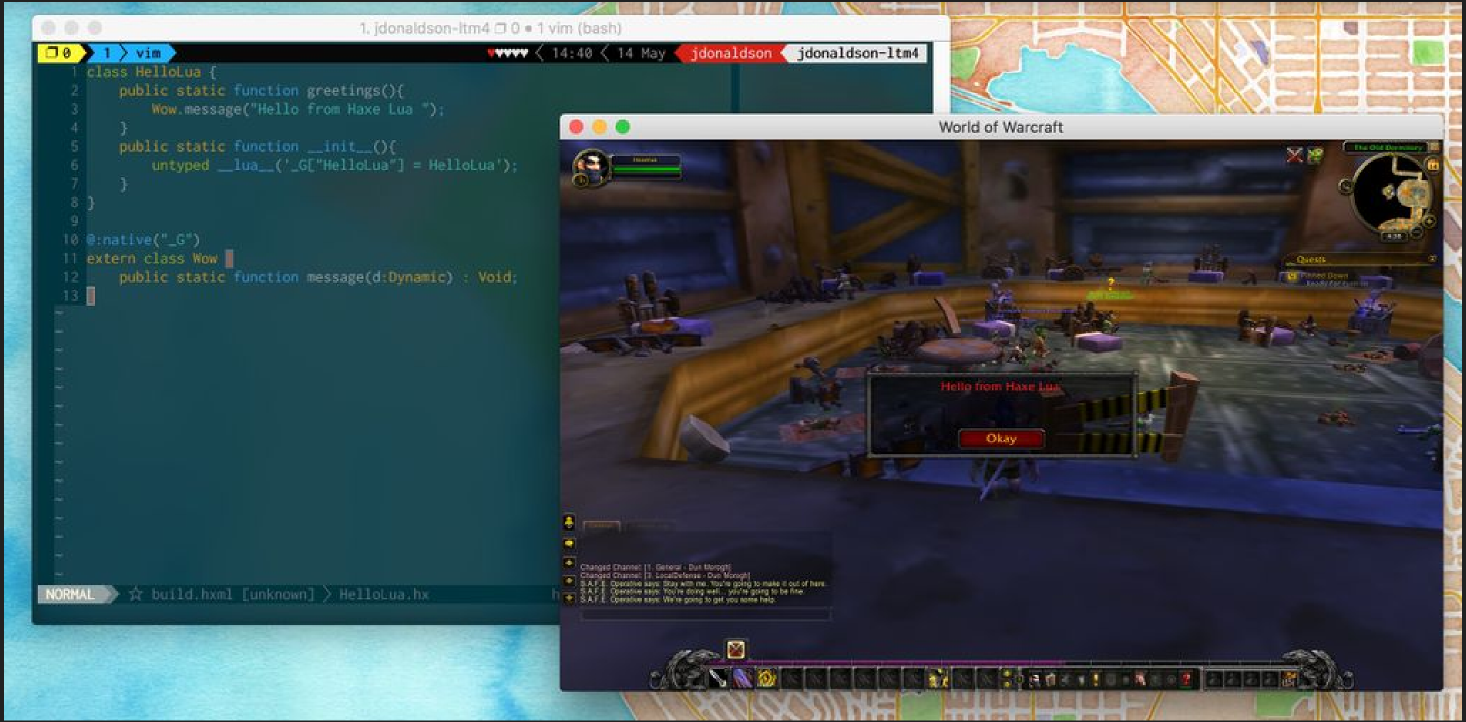
```
class Main {
    static function main() {
        Ngx.say("Hello from haxe-compiled lua-enabled nginx");
    }
}


@:native("ngx")
extern class Ngx {
    static function say(arg : String):Void;
}
```

```
-lua ngx.lua
-main Main
```

```
Hello from haxe-compiled lua-enabled nginx
```

https://github.com/jdonaldson/nginhx

HaxeCraft



1. jdonaldson-ltm4 ❐ 0 • 1 vim (bash)

```
class HelloLua {
    public static function greetings(){
        Wow.message("Hello from Haxe Lua ");
    }
    public static function __init__(){
        untyped __lua__('_G["HelloLua"] = HelloLua');
    }
}

@:native("_G")
extern class Wow {
    public static function message(d:Dynamic) : Void;

```

```
NORMAL  ☆ build.hxml [unknown] 〉 HelloLua.hx
```

World of Warcraft

Hello from Haxe Lua

Okay

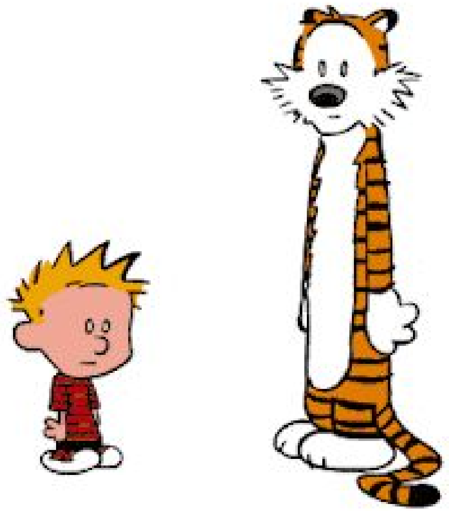
```
Changed Channel: [1. General - Dun Morogh]
Changed Channel: [3. LocalDefense - Dun Morogh]
S.A.F.E. Operative says: Stay with me. You're going to make it out of here.
S.A.F.E. Operative says: You're doing well... you're going to be fine.
S.A.F.E. Operative says: We're going to get you some help.
```

https://github.com/jdonaldson/haxecraft

Recap

1. Lua == speedy, small, and great for sandboxing
2. Lua overlaps a great deal with Haxe community
   (+ a few more niches)
3. Haxe does Lua 5.2, LuaJit 2.0, and Lua 5.3 (wip)
4. Haxe Lua commit messages == Learn how the compiler works
5. HaxeCraft, Nginhx, Haxe Love are some early projects

THE END!
QUESTIONS?



jdonaldson@gmail.com
twitter @omgjjd