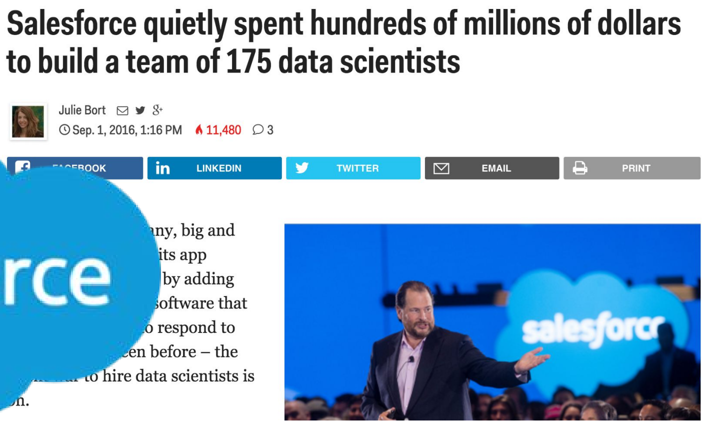
HAXE

SHOOTING FOR THE MOON

HAXE LANDS ON LUA

WRITE ONCE, TARGET MANY

Your humble presenter



Salesforce quietly spent hundreds of millions of dollars to build a team of 175 data scientists

Julie Bort
Sep. 1, 2016, 1:16 PM 11,480 3

FACEBOOK LINKEDIN TWITTER EMAIL PRINT

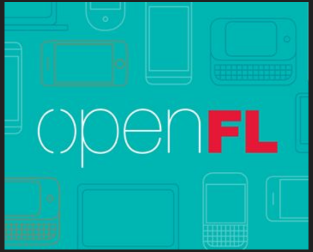
ny, big and
its app
by adding
software that
o respond to
een before – the
to hire data scientists is
n.

salesforce

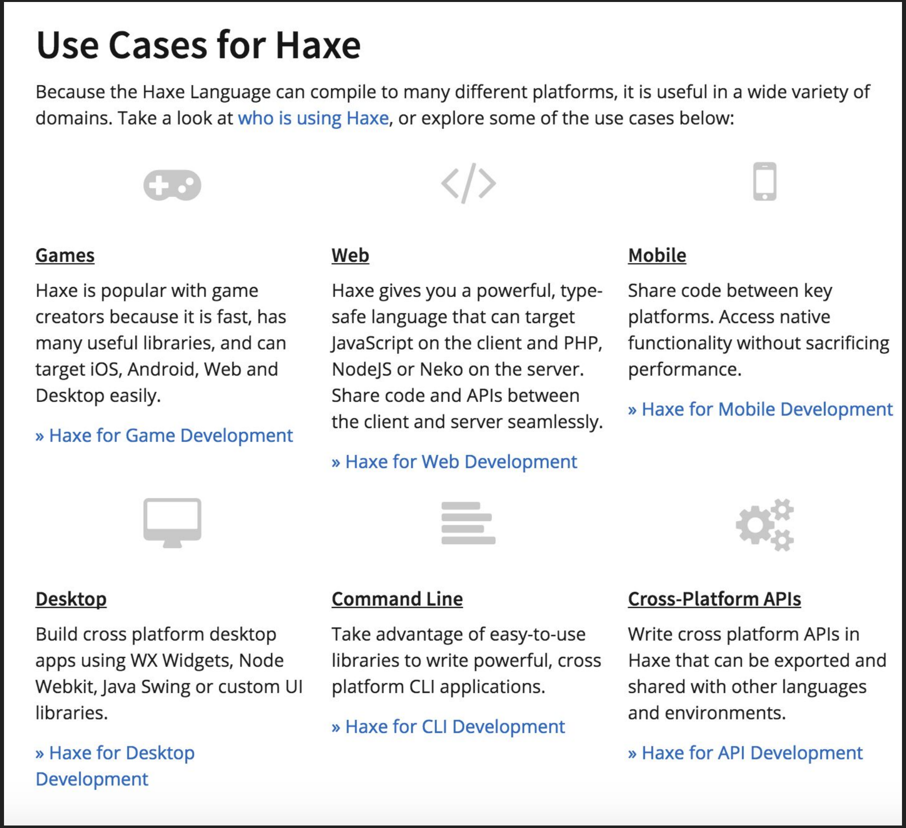
WWX 2016
WORLD WIDE HAXE CONFERENCE
27-28-29-30 MAY ◦ PARIS FRANCE
BY SILEXLABS.ORG

mozilla
HAXE FOUNDATION
SECRET MEDIA
DH DOCLER HOLDING
Keymetrics
gandi.net

Why Haxe?



Kha
Ultra-portable, high performance, open source multimedia framework.



Use Cases for Haxe

Because the Haxe Language can compile to many different platforms, it is useful in a wide variety of domains. Take a look at who is using Haxe, or explore some of the use cases below:

Games

Haxe is popular with game creators because it is fast, has many useful libraries, and can target iOS, Android, Web and Desktop easily.

» Haxe for Game Development

Web

Haxe gives you a powerful, type-safe language that can target JavaScript on the client and PHP, NodeJS or Neko on the server. Share code and APIs between the client and server seamlessly.

» Haxe for Web Development

Mobile

Share code between key platforms. Access native functionality without sacrificing performance.

» Haxe for Mobile Development

Desktop

Build cross platform desktop apps using WX Widgets, Node Webkit, Java Swing or custom UI libraries.

» Haxe for Desktop Development

Command Line

Take advantage of easy-to-use libraries to write powerful, cross platform CLI applications.

» Haxe for CLI Development

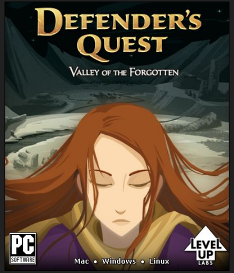
Cross-Platform APIs

Write cross platform APIs in Haxe that can be exported and shared with other languages and environments.

» Haxe for API Development

http://haxe.org/use-cases/

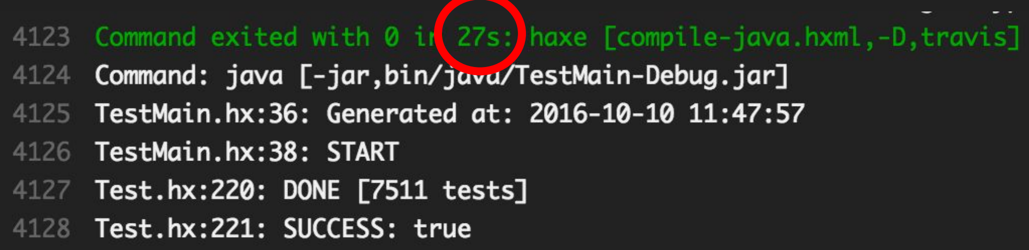
L. Pope



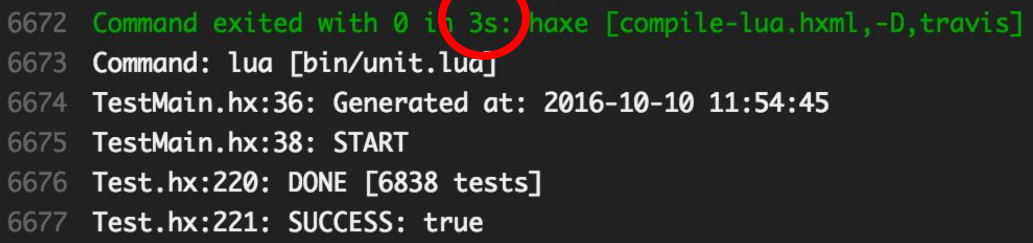
L. Doucet

N. Canasse

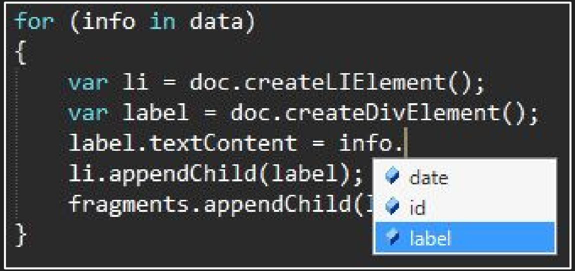
Why Haxe?



```
Command exited with 0 in 27s: haxe [compile-java.hxml,-D,travis]
Command: java [-jar,bin/java/TestMain-Debug.jar]
TestMain.hx:36: Generated at: 2016-10-10 11:47:57
TestMain.hx:38: START
Test.hx:220: DONE [7511 tests]
Test.hx:221: SUCCESS: true
```

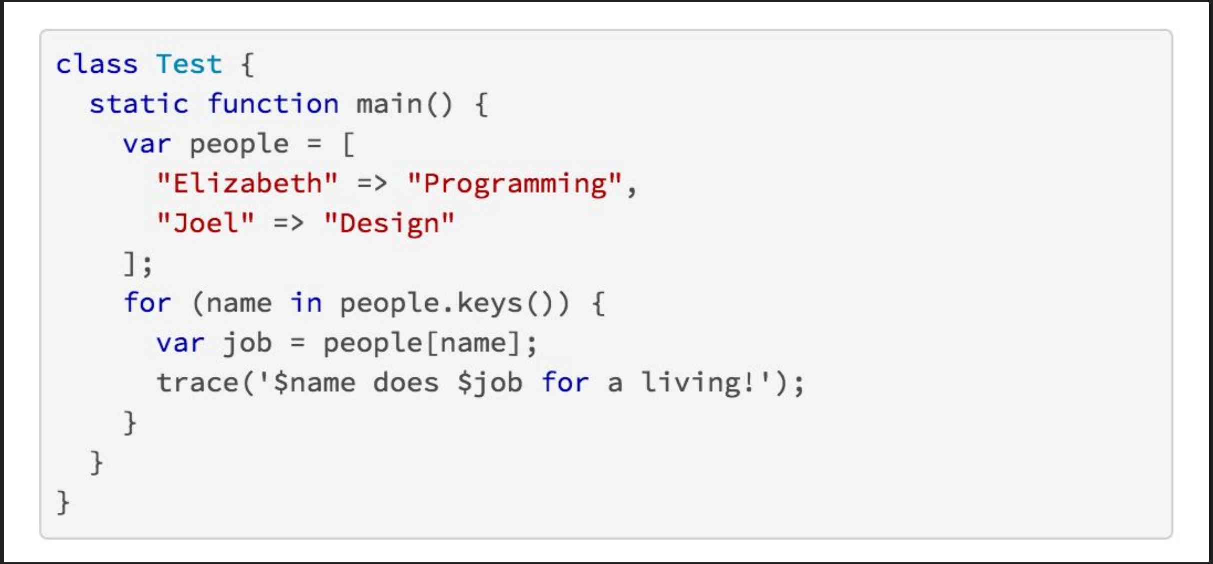


```
Command exited with 0 in 3s: haxe [compile-lua.hxml,-D,travis]
Command: lua [bin/unit.lua]
TestMain.hx:36: Generated at: 2016-10-10 11:54:45
TestMain.hx:38: START
Test.hx:220: DONE [6838 tests]
Test.hx:221: SUCCESS: true
```



```
for (info in data)
{
    var li = doc.createLIElement();
    var label = doc.createDivElement();
    label.textContent = info.
    li.appendChild(label);
    fragments.appendChild(l
}
```

A Taste of Haxe



```haxe
class Test {
  static function main() {
    var people = [
      "Elizabeth" => "Programming",
      "Joel" => "Design"
    ];
    for (name in people.keys()) {
      var job = people[name];
      trace('$name does $job for a living!');
    }
  }
}
```

```
$> haxe -main Test -lua out.lua
```

Haxe Features

- Abstract Types
- Anonymous Types
- Array Comprehension
- Classes, Interfaces, and Inheritance
- Conditional Compilation
- (Generalized) Algebraic Data Types
- Inlined Calls
- Iterators
- Local functions and closures
- Metadata
- Static Extensions
- String Interpolation
- Partial function application
- Pattern matching
- Properties
- Type parameters, constraints, variance
- Reflection
- AST macros
- Static Analysis
  - Const propagation
  - Copy propagation
  - Local dead code elimination
  - Fusion
  - Purity Inference

https://haxe.org/documentation/introduction/language-features.html

Why Haxe and Lua?

1. Why not?
2. LuaJit (Nginx, Torch, etc.)
3. Scripting for editors (Neovim, vim)
4. Scripting for games (WoW, Factorio)
5. Community match (game + webdev)
6. Boredom/Hubris

torch

A SCIENTIFIC COMPUTING FRAMEWORK FOR LUAJIT

NGINX

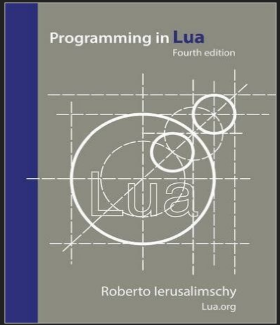
redis

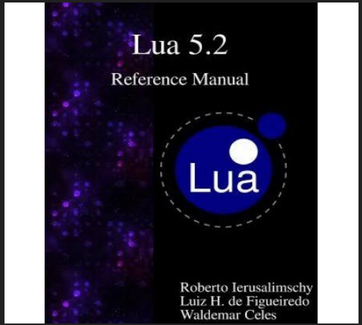
Lua

Which Lua?

1. Lua 5.1
2. Lua 5.2
3. LuaJit 2.0
4. LuaJit 2.1
5. Lua 5.3*

LuaJIT





* Partial support backwards compatibility flags

NEW! Announcement!

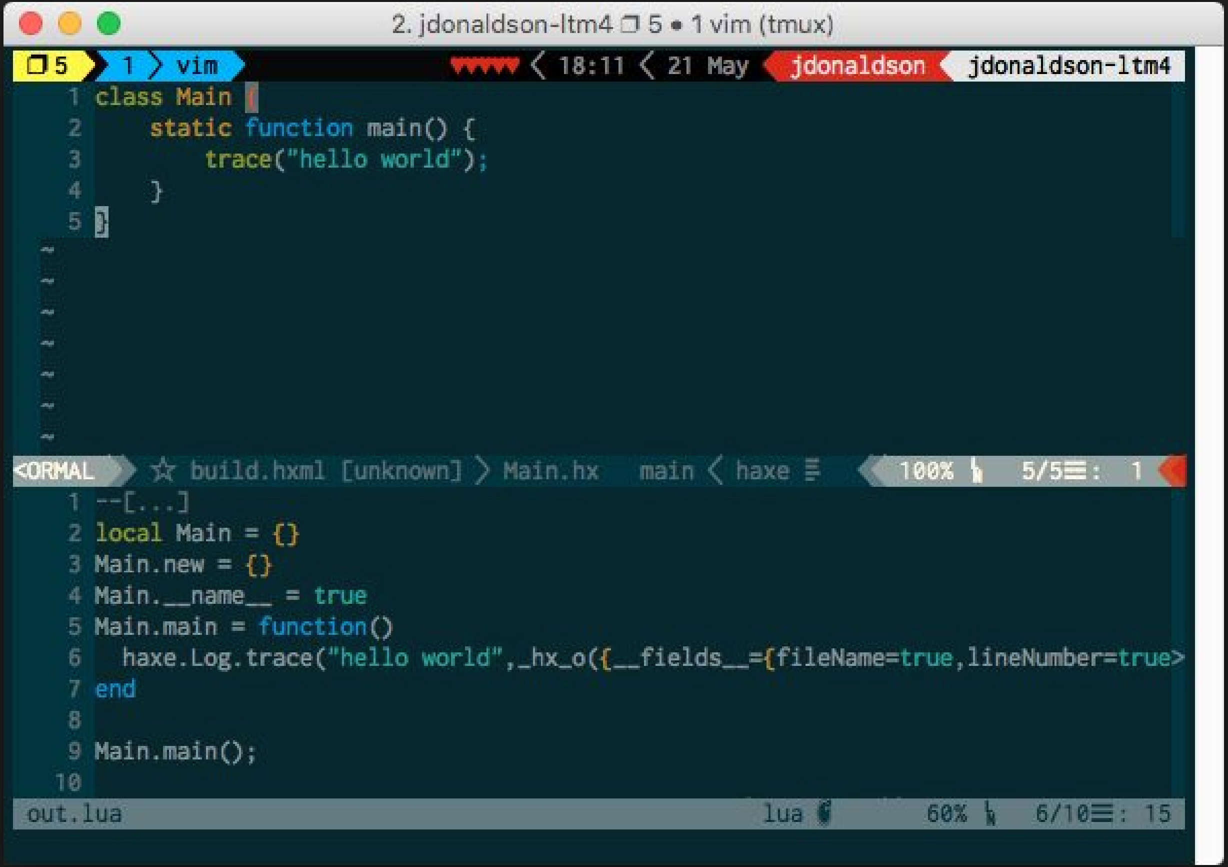
Related Work

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!

Hello World

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



2. jdonaldson-ltm4 ❐ 5 ● 1 vim (tmux)

❐ 5 〉 1 〉 vim ♥♥♥♥♥ 〈 18:11 〈 21 May 〈 jdonaldson 〈 jdonaldson-ltm4

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

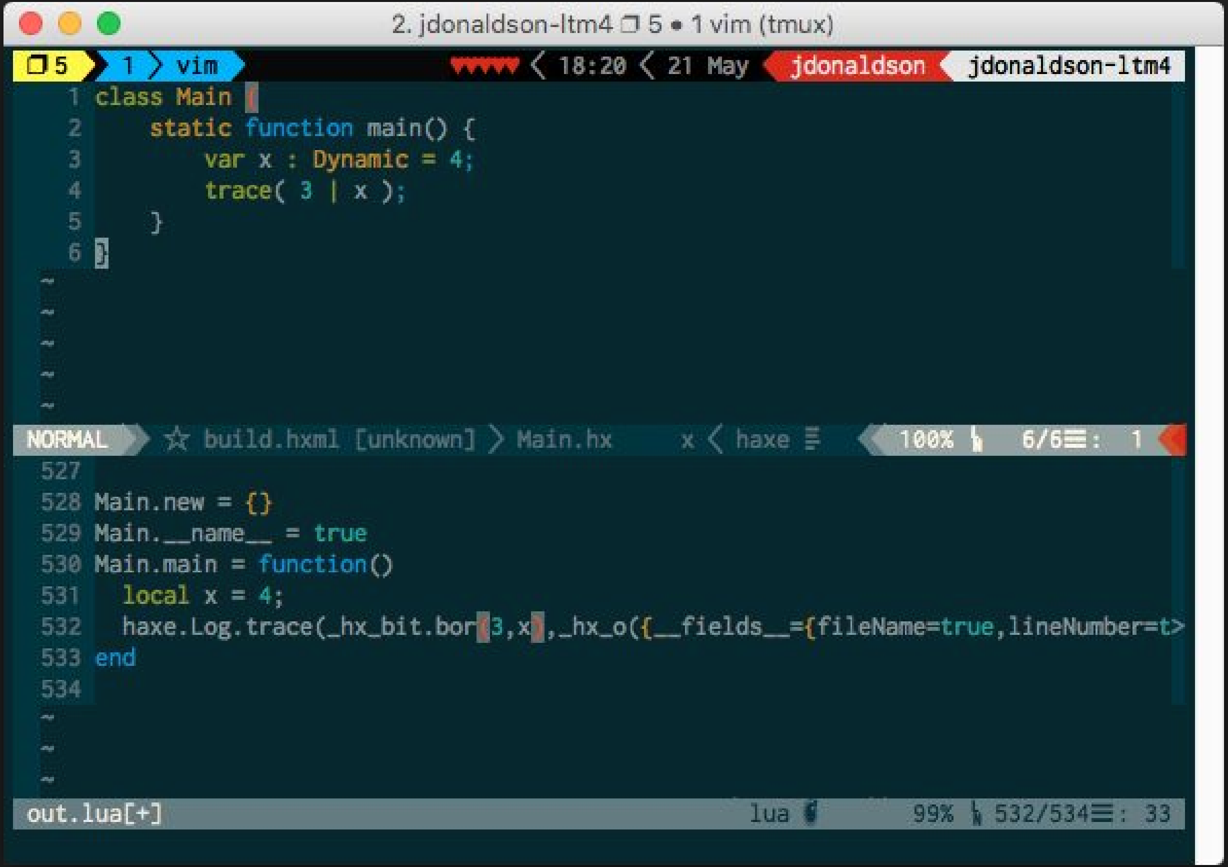
<ORMAL 〉 ☆ build.hxml [unknown] 〉 Main.hx   main 〈 haxe   100%   5/5 : 1

```
--[...]
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();
```

out.lua   lua   60%   6/10 : 15

BitOps

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

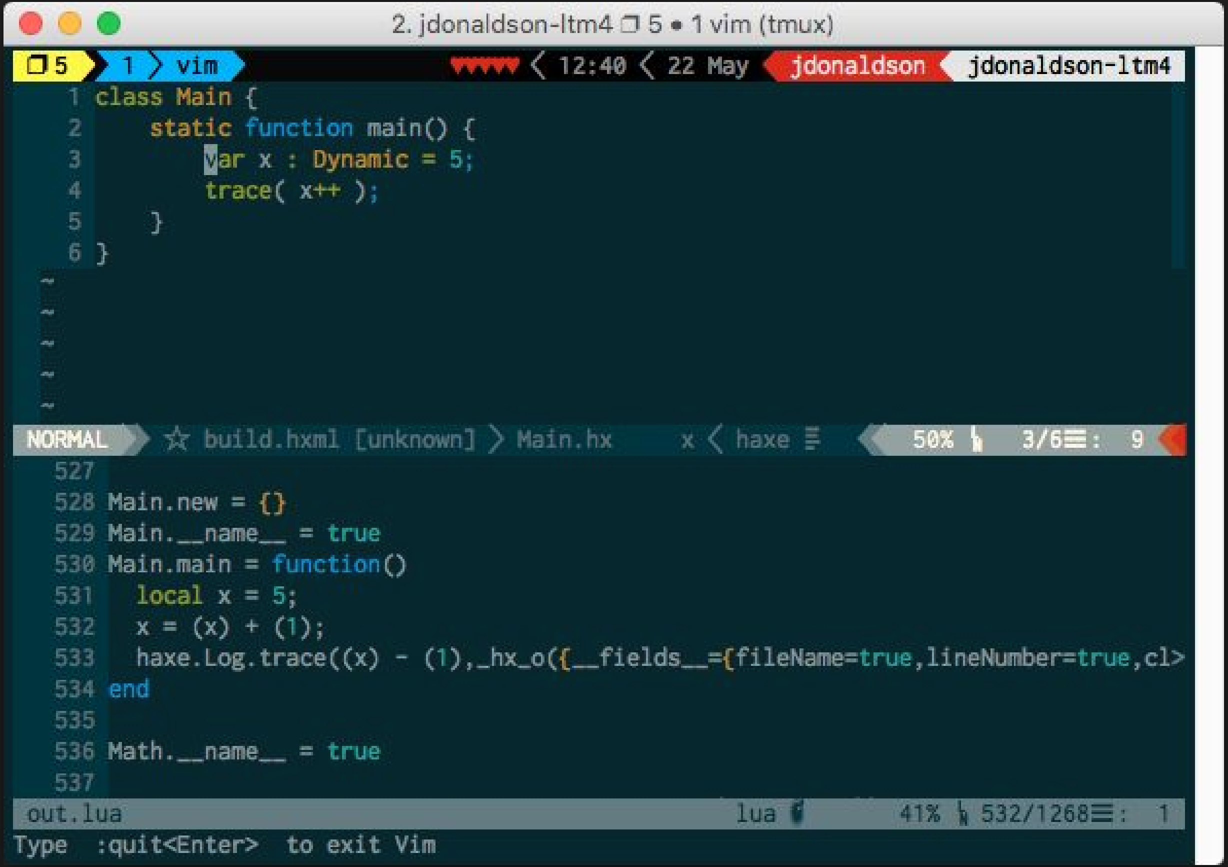


2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)

```
❐ 5 〉 1 〉 vim                              ♥♥♥♥♥ 〈 18:20 〈 21 May  jdonaldson  jdonaldson-ltm4
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

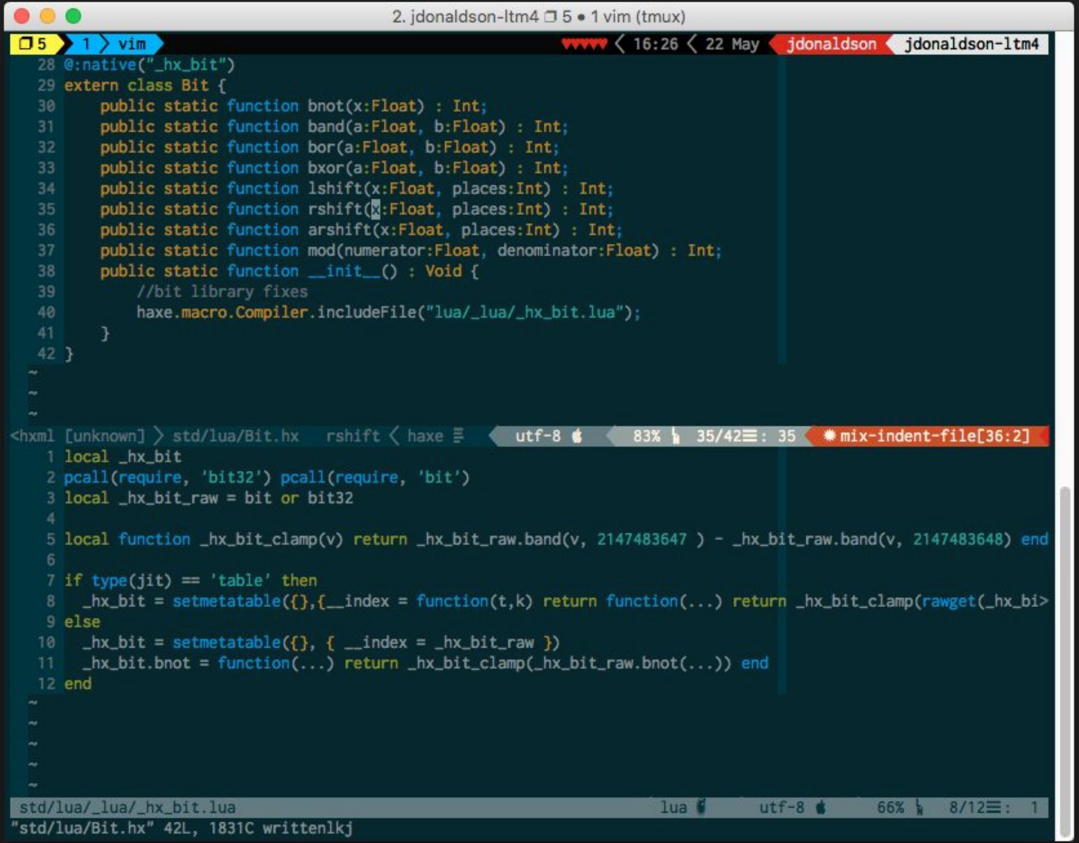


2. jdonaldson-ltm4 ❐ 5 • 1 vim (tmux)

```
❐ 5  1  vim                    ♥♥♥♥♥ 12:40  22 May  jdonaldson  jdonaldson-ltm4
class Main {
    static function main() {
        var x : Dynamic = 5;
        trace( x++ );
    }
}
~
~
~
~
~
NORMAL  ☆ build.hxml [unknown]  Main.hx      x  haxe      50%   3/6☰:  9

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

out.lua                                                lua      41%  532/1268☰:  1
Type  :quit<Enter>  to exit Vim
```

Extern

- @:native binds to native or non-conformingly named interface
- @:expose binds class/method body to global metatable
- @:selfCall allows methods to call the module/class name as a function
- 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]
```

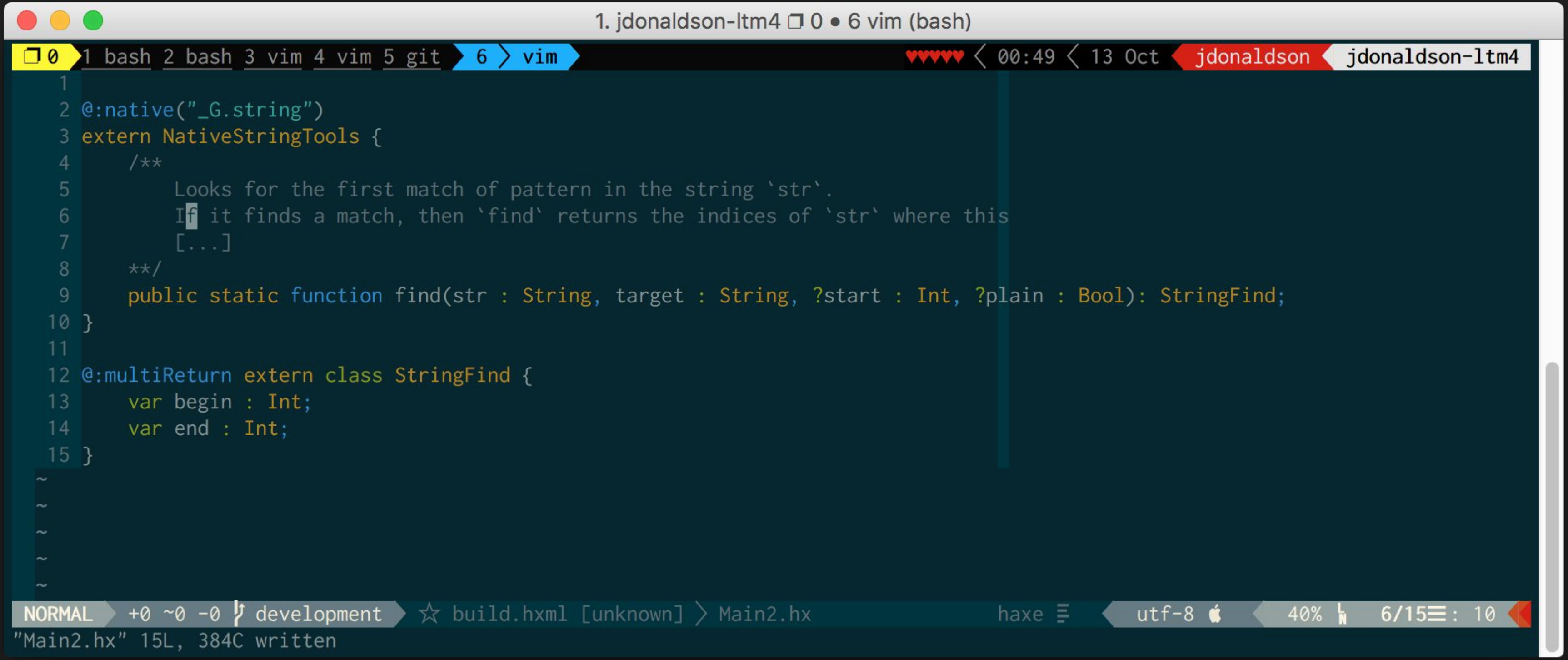
```
local _hx_bit
pcall(require, 'bit32') pcall(require, 'bit')
local _hx_bit_raw = bit or bit32

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
```

Extern

NEW!!! Announcement!

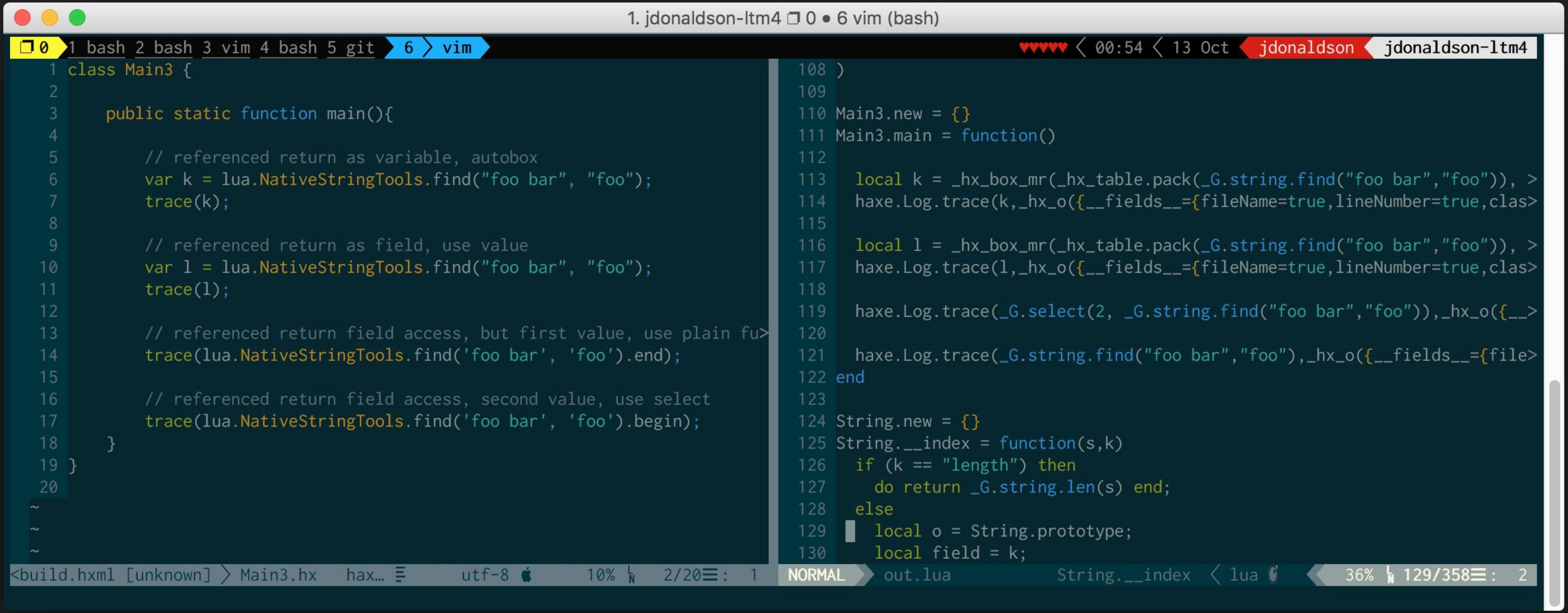
- @:multiReturn allows specification of extern-only classes that represent multiple returns



1. jdonaldson-ltm4 ❐ 0 ● 6 vim (bash)
❐ 0 1 bash 2 bash 3 vim 4 vim 5 git 6 vim
♥♥♥♥♥ 00:49 13 Oct jdonaldson jdonaldson-ltm4

```

@:native("_G.string")
extern NativeStringTools {
    /**
        Looks for the first match of pattern in the string `str`.
        If it finds a match, then `find` returns the indices of `str` where this
        [...]
    **/
    public static function find(str : String, target : String, ?start : Int, ?plain : Bool): StringFind;
}

@:multiReturn extern class StringFind {
    var begin : Int;
    var end : Int;
}
~
~
~
~
~
NORMAL  +0 ~0 -0  development  ☆ build.hxml [unknown] 〉Main2.hx        haxe   utf-8   40%   6/15☰ : 10
"Main2.hx" 15L, 384C written
```

NEW!!! Announcement!



1. jdonaldson-ltm4 ❐ 0 ● 6 vim (bash)

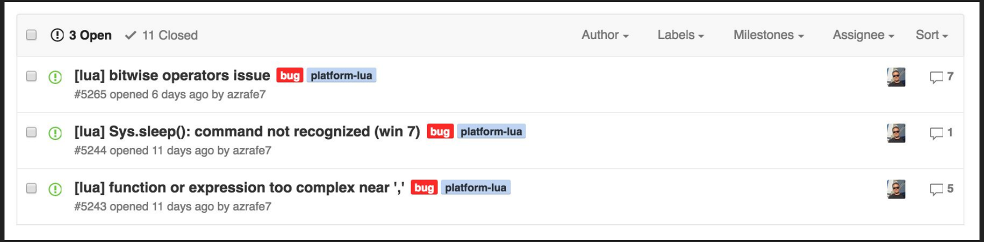
```
❐ 0  1 bash  2 bash  3 vim  4 bash  5 git  6  vim          ♥♥♥♥♥  00:54  13 Oct  jdonaldson  jdonaldson-ltm4
```

```
class Main3 {

    public static function main(){

        // referenced return as variable, autobox
        var k = lua.NativeStringTools.find("foo bar", "foo");
        trace(k);

        // referenced return as field, use value
        var l = lua.NativeStringTools.find("foo bar", "foo");
        trace(l);

        // referenced return field access, but first value, use plain fu>
        trace(lua.NativeStringTools.find('foo bar', 'foo').end);

        // referenced return field access, second value, use select
        trace(lua.NativeStringTools.find('foo bar', 'foo').begin);
    }
}

~
~
~
<build.hxml [unknown] 〉 Main3.hx    hax…    utf-8     10%   2/20☰:  1
```

```
)

Main3.new = {}
Main3.main = function()

  local k = _hx_box_mr(_hx_table.pack(_G.string.find("foo bar","foo")), >
  haxe.Log.trace(k,_hx_o({__fields__={fileName=true,lineNumber=true,clas>

  local l = _hx_box_mr(_hx_table.pack(_G.string.find("foo bar","foo")), >
  haxe.Log.trace(l,_hx_o({__fields__={fileName=true,lineNumber=true,clas>

  haxe.Log.trace(_G.select(2, _G.string.find("foo bar","foo")),_hx_o({__>

  haxe.Log.trace(_G.string.find("foo bar","foo"),_hx_o({__fields__={file>
end

String.new = {}
String.__index = function(s,k)
  if (k == "length") then
    do return _G.string.len(s) end;
  else
    local o = String.prototype;
    local field = k;
NORMAL 〉 out.lua          String.__index 〈 lua     36%  129/358☰:  2
```

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



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

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

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

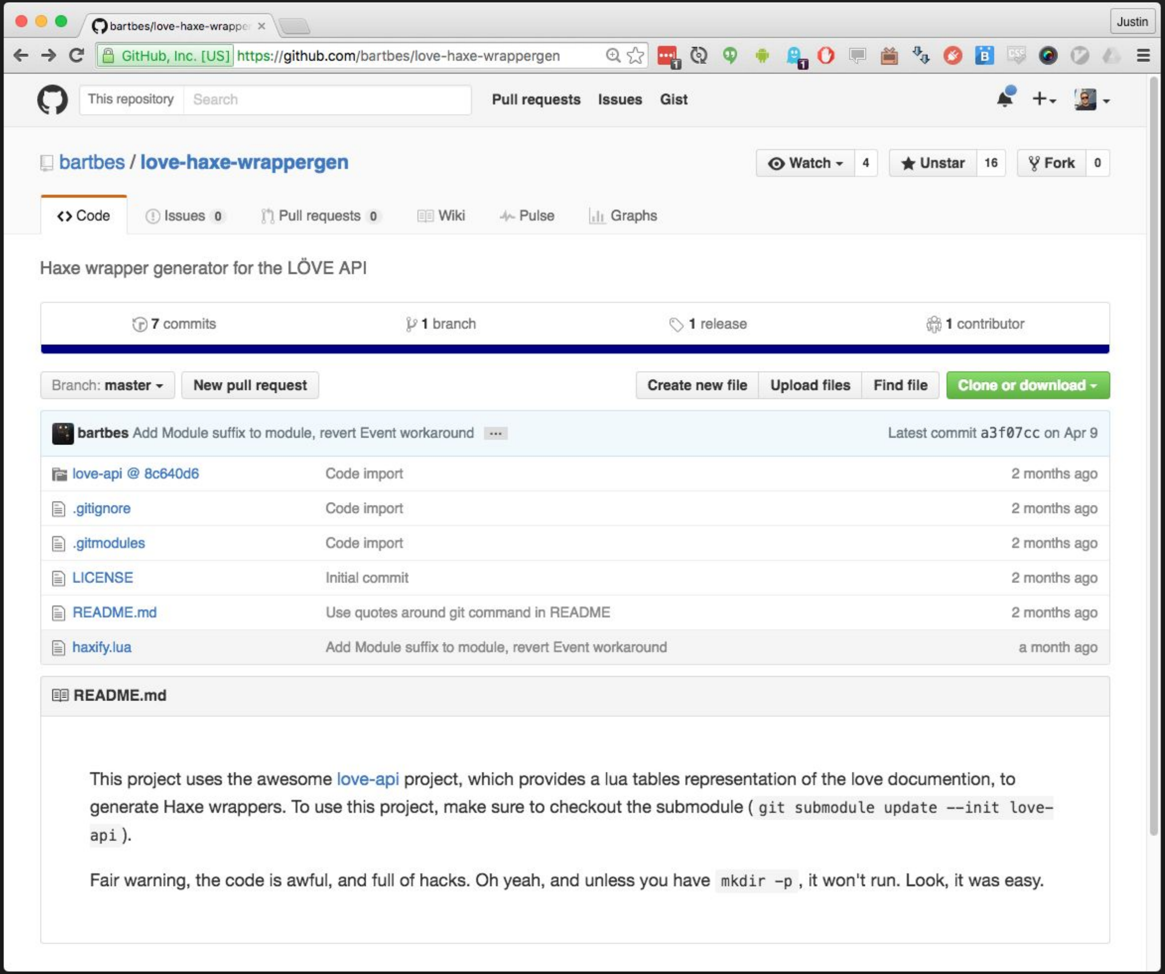
* Progress on libuv/luv backend

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.

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

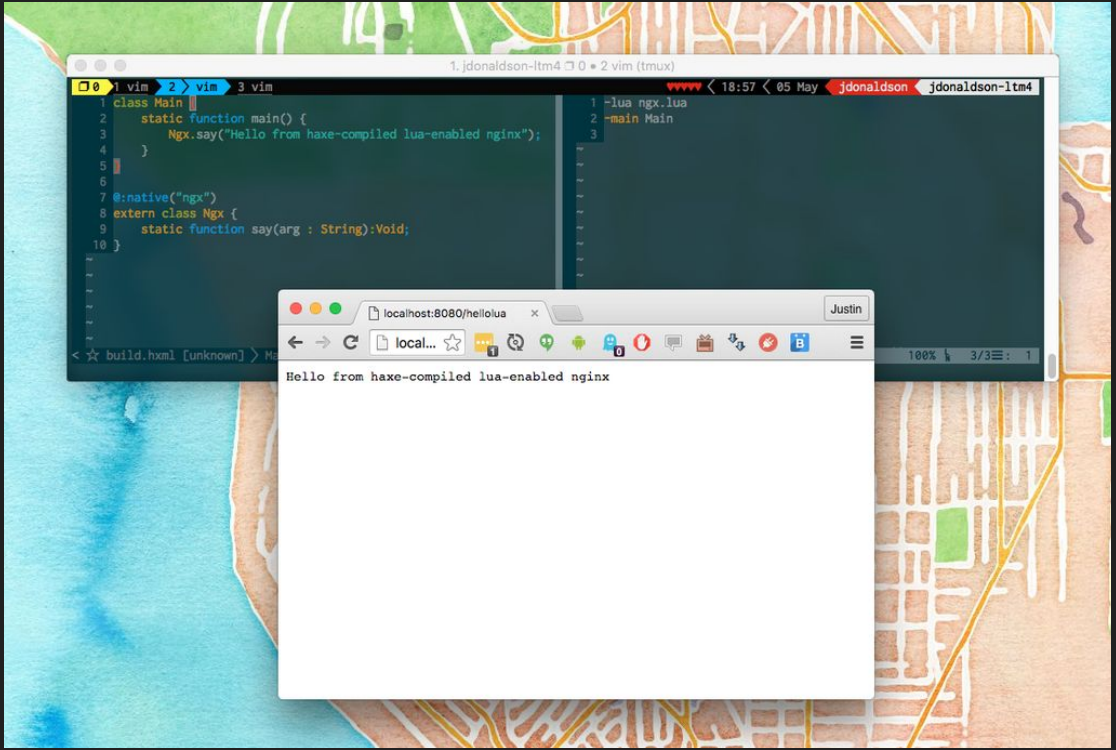
love-api @ 8c640d6	Code import	2 months ago
.gitignore	Code import	2 months ago
.gitmodules	Code import	2 months ago
LICENSE	Initial commit	2 months ago
README.md	Use quotes around git command in README	2 months ago
haxify.lua	Add Module suffix to module, revert Event workaround	a month ago

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



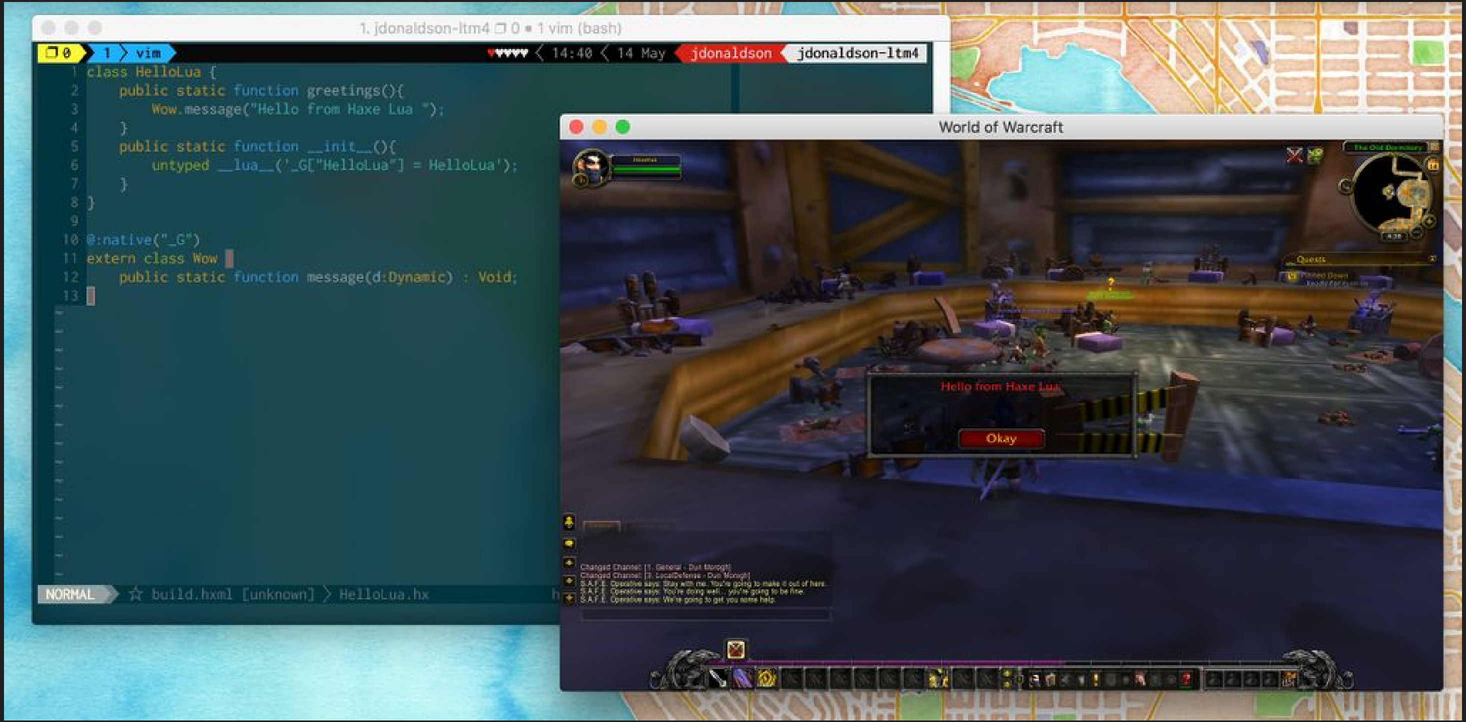
```
class Main {
    static function main() {
        Ngx.say("Hello from haxe-compiled lua-enabled nginx");
    }
}


@: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



```
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
```

```
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

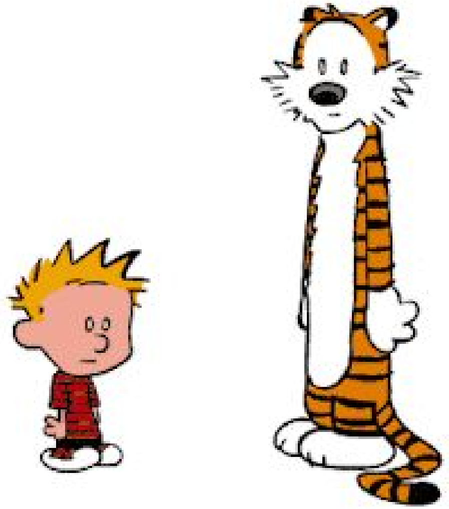
How to get started

1. Haxe manual : https://haxe.org/manual/introduction.html
2. Haxe cookbook : http://code.haxe.org/
3. Haxe mailing list : https://groups.google.com/forum/#!forum/haxelang
4. Haxe discord group : https://discord.gg/znfNW
5. Haxe IRC : (freenode #haxe) http://webchat.freenode.net/?channels=haxe
6. Haxe Twitter : #haxe https://twitter.com/search?q=haxe&src=typd
7. Haxe Github : https://github.com/HaxeFoundation/haxe

Recap/Conclusion

- Haxe and Lua communities are similar : creative, independent, mindful
    - (even though languages are different)
- Haxe as a language is very “standard”
    - Ecmascript based, multi paradigm language
- Haxe provides a way to leverage an existing language ecosystem, while expanding towards other targets/platforms.
    - You don’t leave the Lua community by joining the Haxe community
- Haxe avoids impedance mismatch by supporting target specific extern features (e.g. @:multiReturn)
- Haxe provides convenient and powerful static typing features on dynamic languages
    - I’ve learned more from Haxe than any other programming language community

THE END!
QUESTIONS?



jdonaldson@gmail.com

twitter @omgjjd