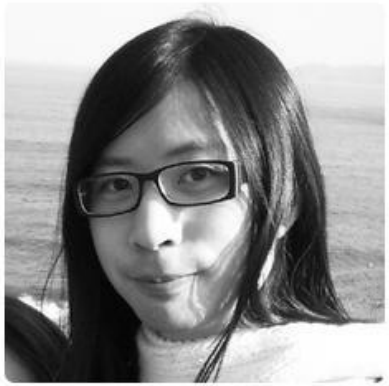
Transcompiling
towards the freedom of programming language and platform choice

HKOSCon 2015

Andy Li

GitHub Search GitHub Explore Features Enterprise Blog Sign up Sign in



Andy Li

andyli

Hong Kong
andy@onthewings.net
http://www.onthewings.net/
Joined on Jul 12, 2009

111 Followers 274 Starre 58 Following

Organizations

Contributions Repositories Public activity Follow

Popular repositories

- jQueryExternForHaxe — Unleash the full power of jQuery in Haxe. 53
- hxOpenFrameworks — Haxe binding to openFrameworks 36
- hxudp — UDP socket for Haxe/C++ 29
- hxLINQ — An implementation ... Haxe 28
- hxColorToolkit — Haxe library for color ... 24

Repositories contributed to

- HaxeFoundation/haxe — Haxe - The Cross-Platform Toolkit 943
- caskroom/homebrew-cask — A CLI workflow for the administration of Mac ... 6,791
- HaxeFoundation/hxcpp — Runtime files for c++ backend for haxe 66
- travis-ci/travis-build — .travis.yml => build.sh converter 107
- HaxeFoundation/haxelib — ... 57

Haxe Foundation

Public contribution

Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May

M W F

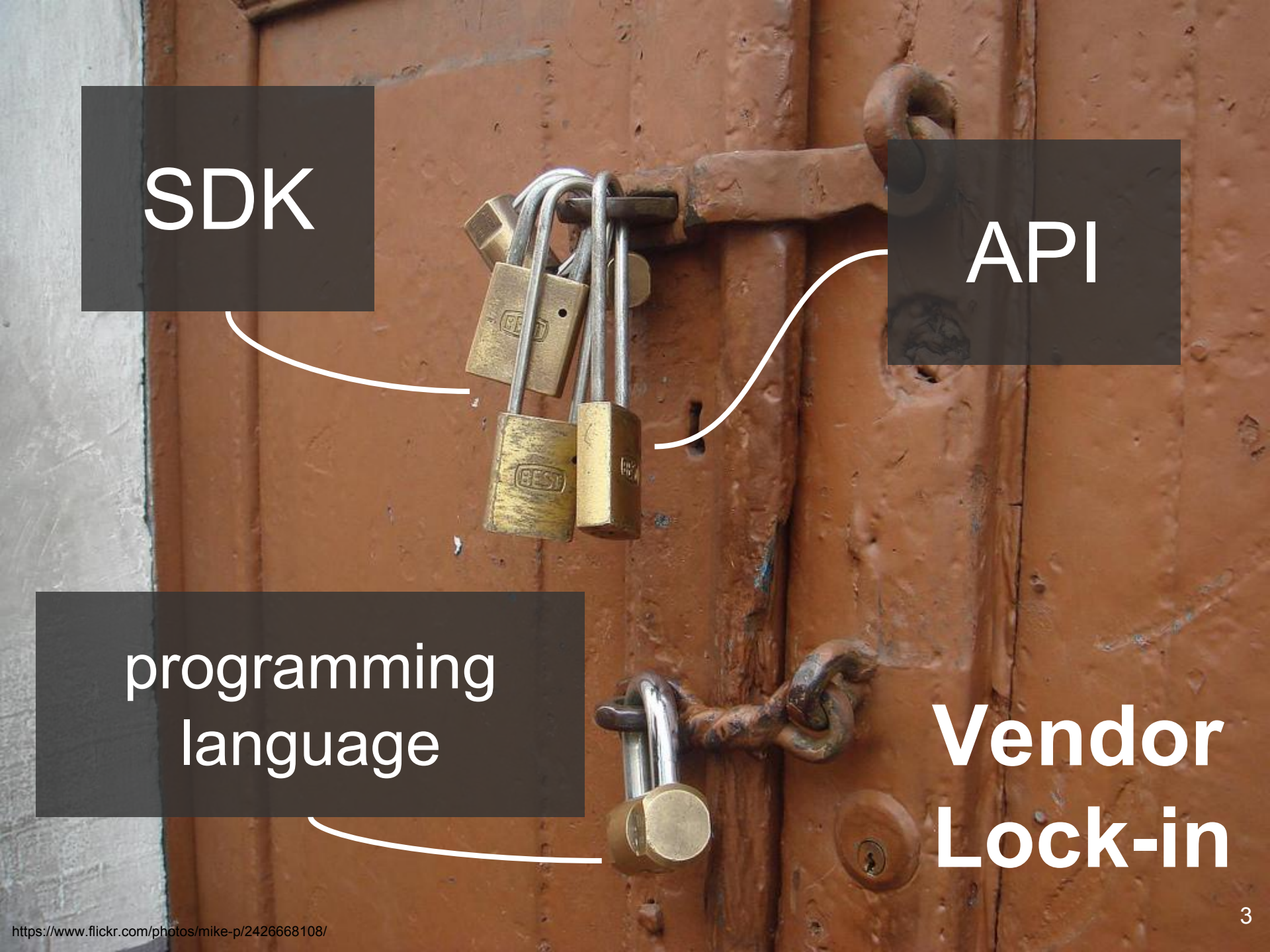
Summary of Pull Requests, issues opened, and commits. Learn more. Less More

Contributions in the last year	Longest streak	Current streak
673 total	11 days	0 days
Jun 11, 2014 – Jun 11, 2015	April 7 – April 17	Last contributed a day ago

Contribution activity Period: 1 week

1 commit

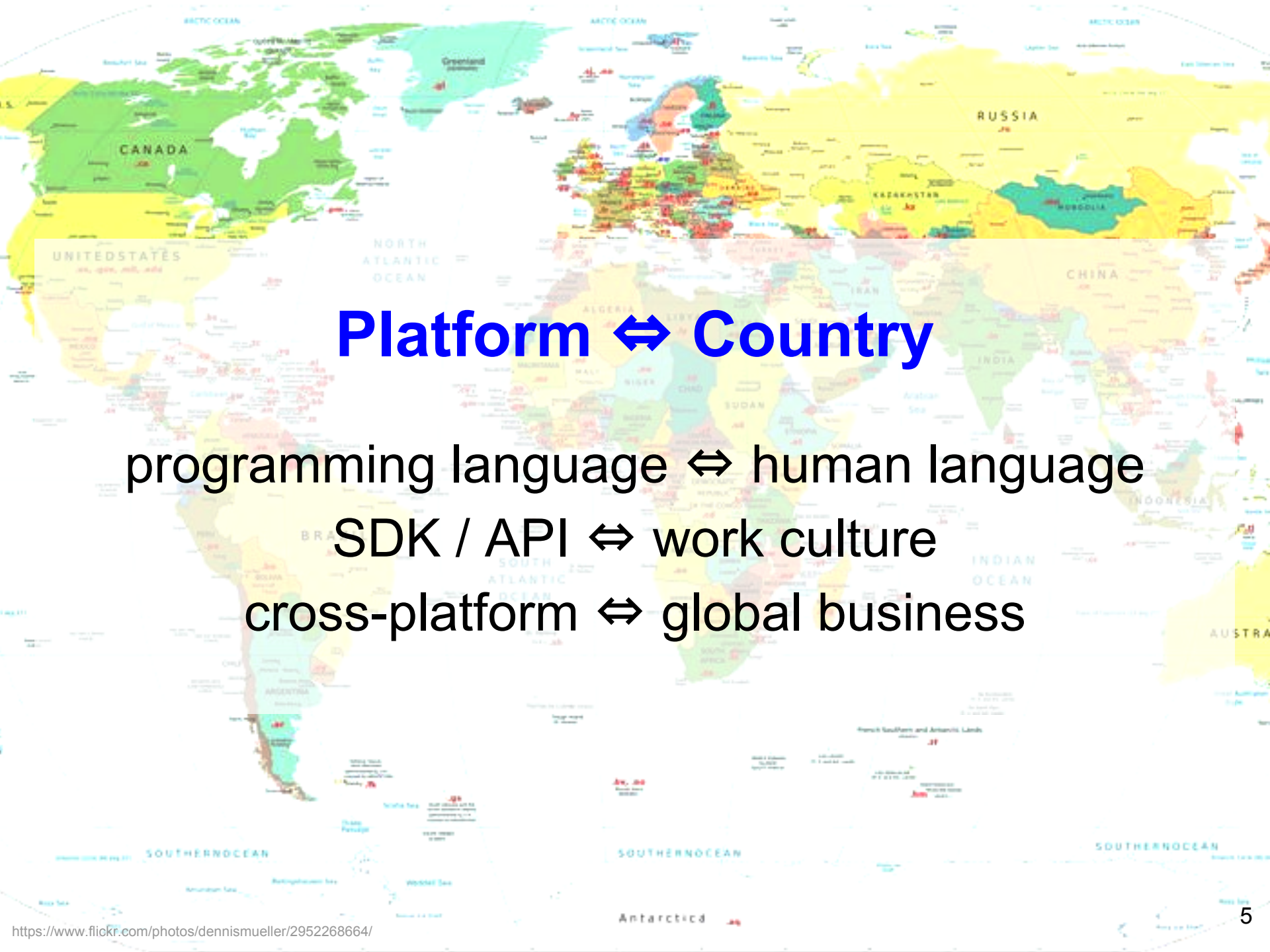
Pushed 1 commit to caskroom/homebrew-cask Jun 10



https://www.flickr.com/photos/mike-p/2426668108/

language lock-in?

- limited “way of thinking”
  - programming paradigm
    e.g. object-oriented vs functional
  - typing system
    e.g. static typing vs dynamic typing
  - syntax
    e.g. Java vs Python vs Lisp vs ...



Platform ⇔ Country

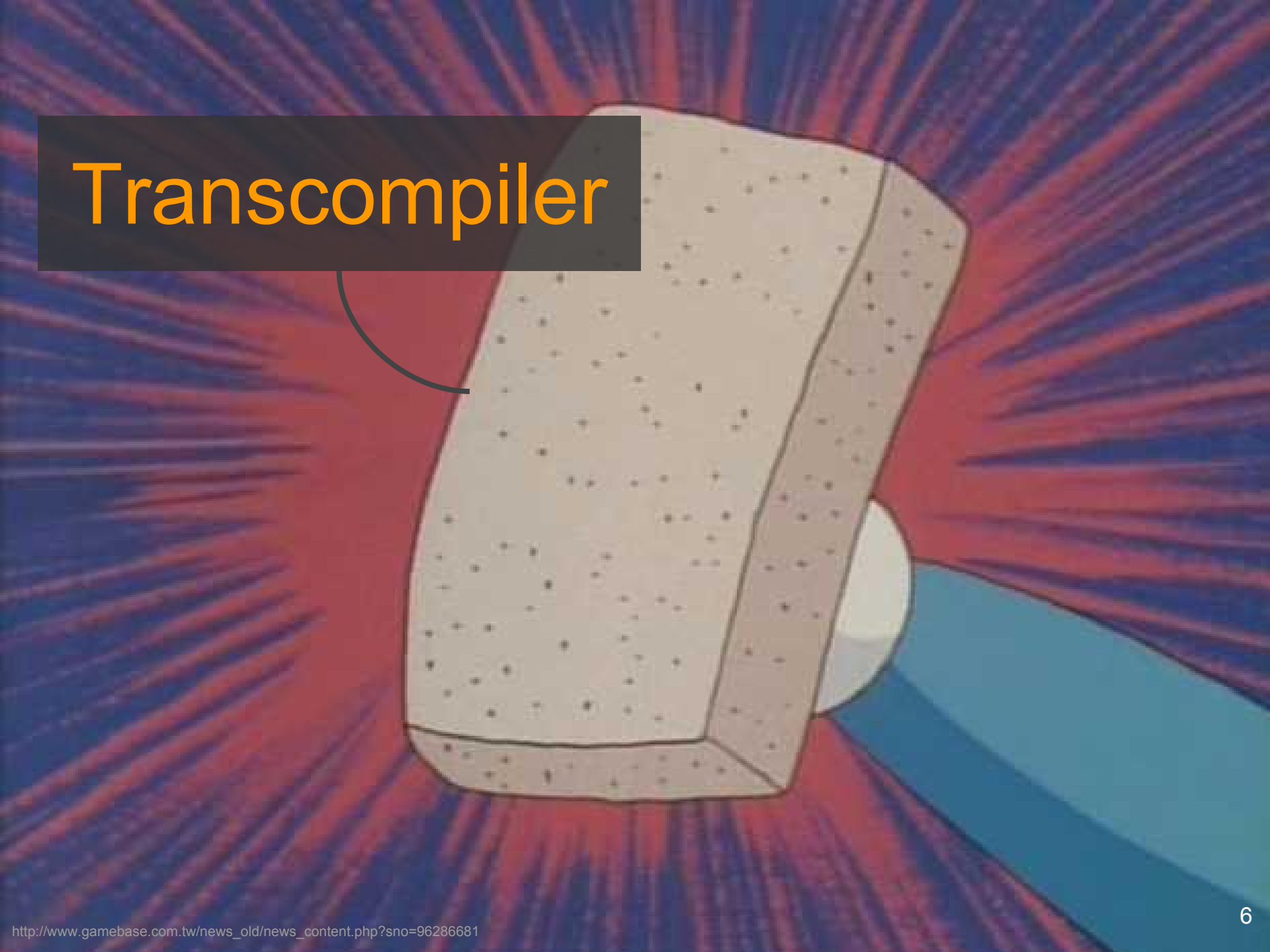
programming language ⇔ human language

SDK / API ⇔ work culture

cross-platform ⇔ global business

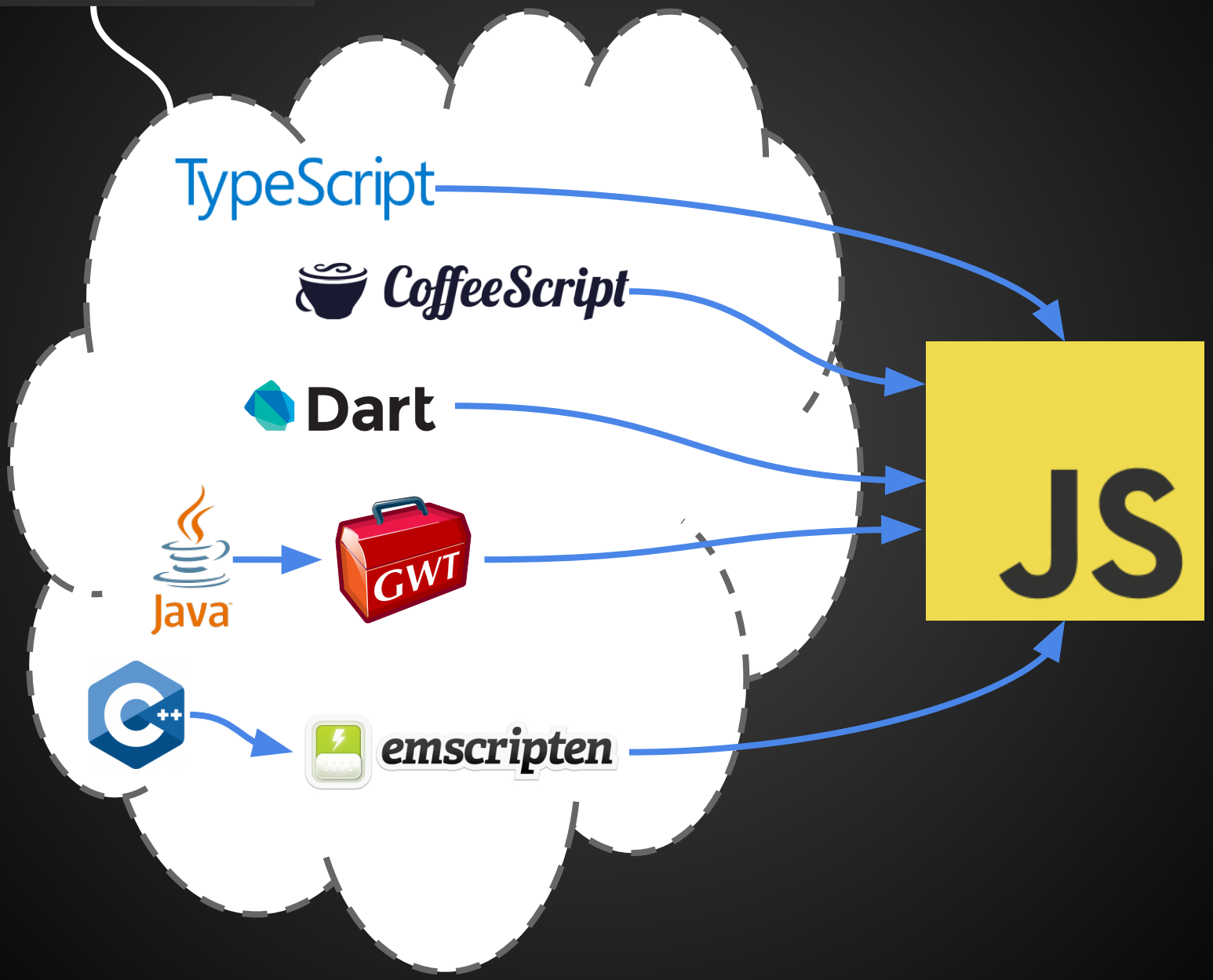
https://www.flickr.com/photos/dennismueller/2952268664/

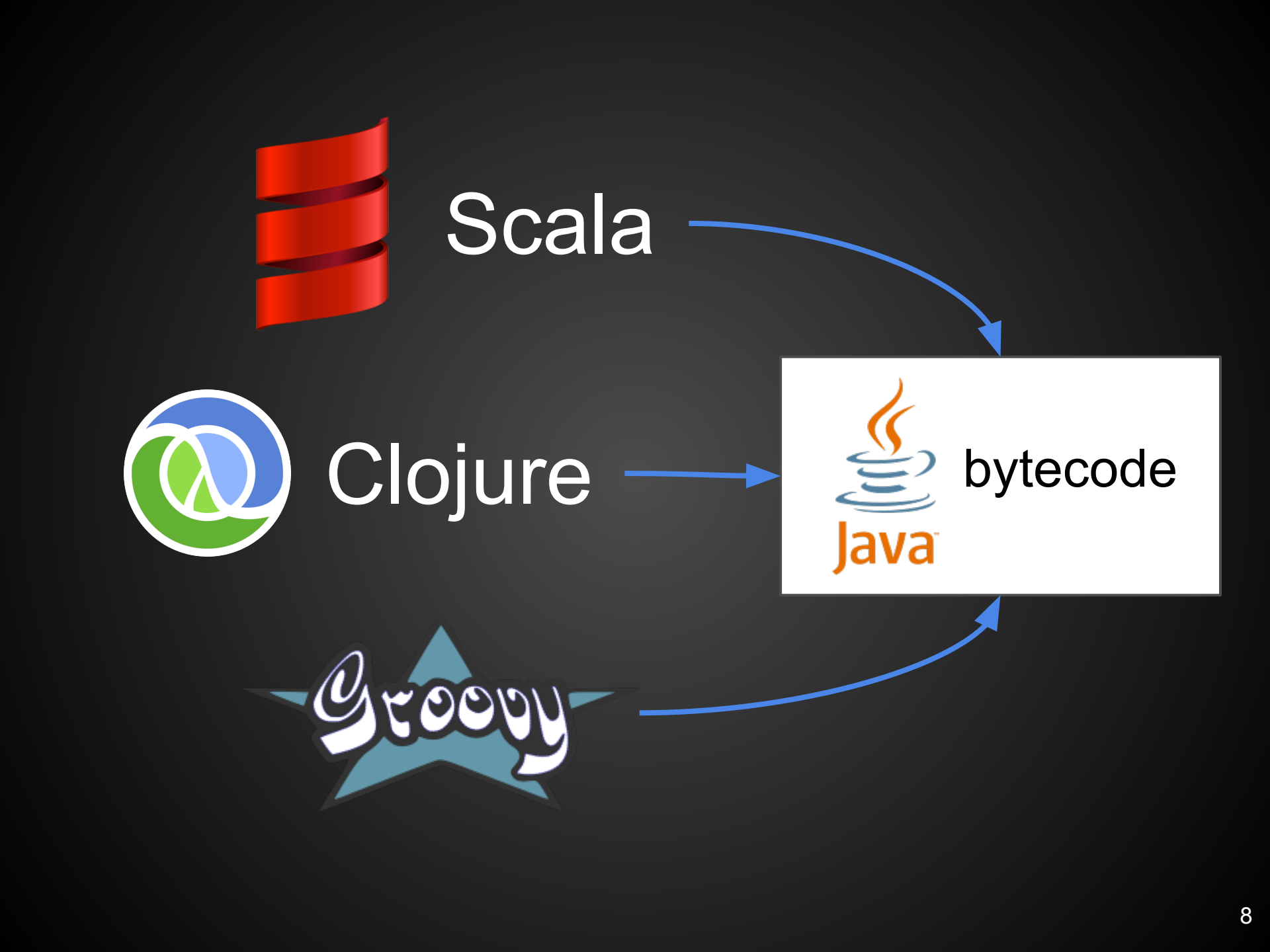
Transcompiler

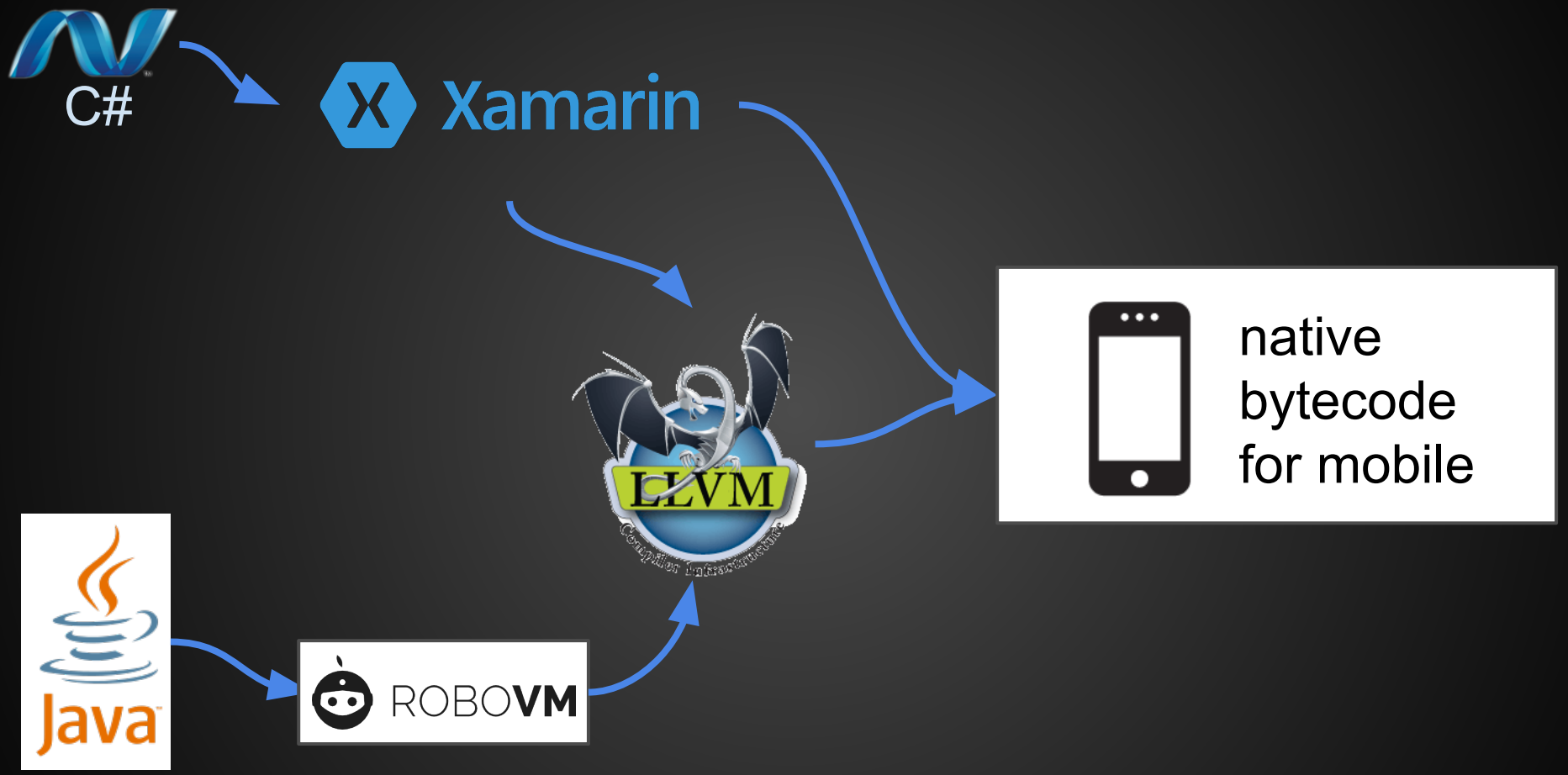


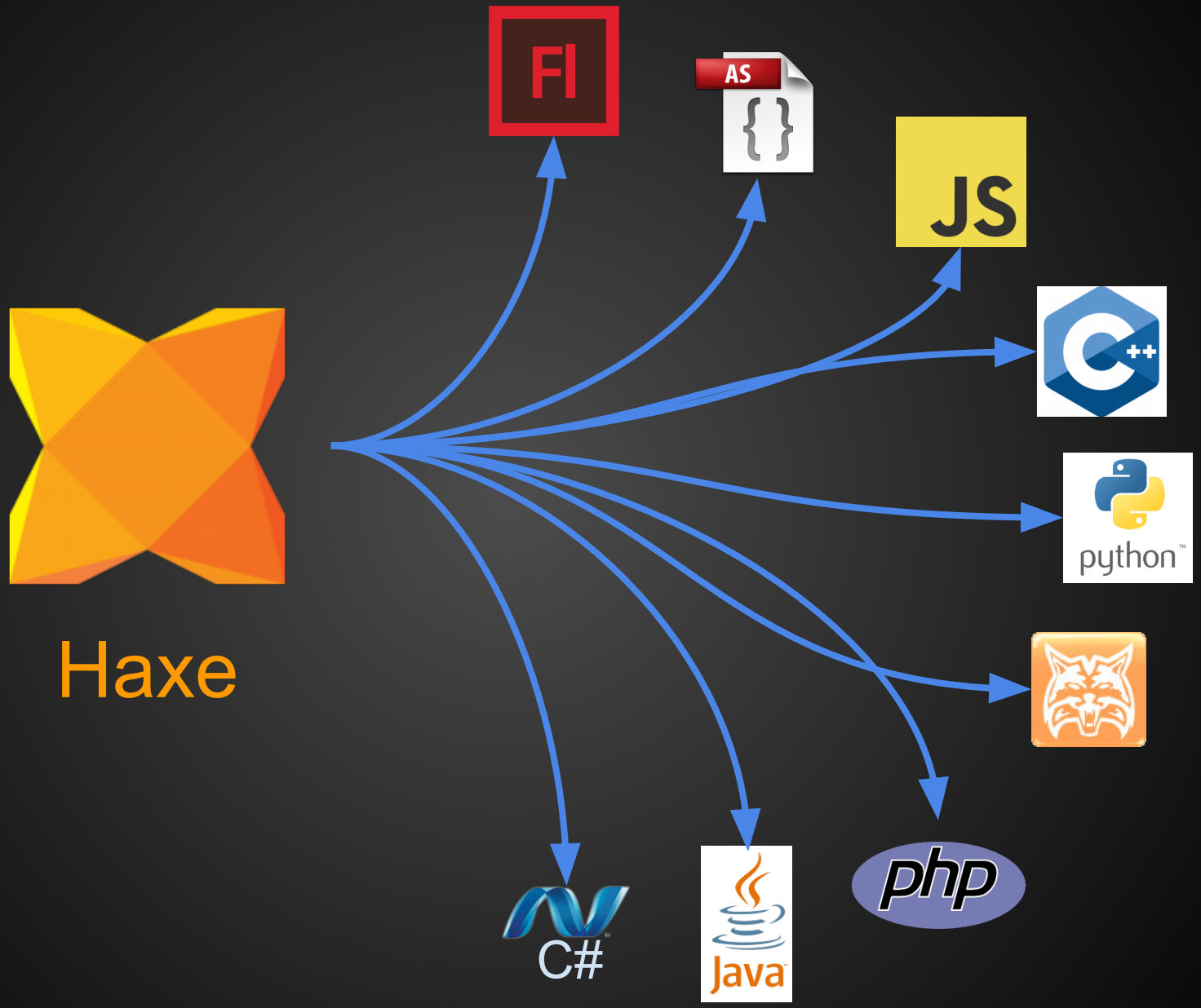
http://www.gamebase.com.tw/news_old/news_content.php?sno=96286681

300+ technologies that can target JS




Scala
Clojure
Groovy
Java
bytecode


C#
Xamarin
LLVM
native
bytecode
for mobile
Java
ROBOVM


Fl
AS
{ }
JS
C++
python
Haxe
C#
Java
php

Transcompiler, yay or nay?

Yay! :D

- an extra layer of abstraction
- reduced lock-in
- better code reuse
  e.g. isomorphic app
- pre/post processing
  e.g. optimizations, embed assets, macros

Nay... :(

- an extra layer of abstraction
- losing benefit of interpreted languages
- compilation takes time
- could be hard to debug
- may have performance overhead
- may not play well with native libs/APIs

A good transcompiler is the one that maximize the Yays and minimize the Nays.

	TypeScript / CoffeeScript	Java	LLVM	Haxe
front-end languages	1	1	> 8	1
back-end targets	JS	JS (via GWT), native (via LLVM)	JS, native	9
optimizations	minimal	output is messy and slow	good	good
language / compiler features	modern language, minimal compiler feature	old-school	depends on front-end	modern and rich
compilation speed	fast	slow	depends on front-end	fast
inter-op with native lib/API	good	okay-ish	okay-ish	pretty good

Haxe features - JS/Java-like syntax

source code: hkoscon/transcompiling/HelloWorld.hx

```
package hkoscon.transcompiling;

class HelloWorld {
 static function main() {
   function greet(speaker, target = "World") {
     trace(speaker.first + " " + speaker.last + ": Hello, " + target + "!");
   }
   var andy = {
     first: "Andy",
     last: "Li"
   };
   greet(andy); //Andy Li: Hello, World!
 }
}
```

Haxe features - JS-like syntax

cmd: haxe -main hkoscon.transcompiling.HelloWorld -js HelloWorld.js

```
(function (console) { "use strict";
var hkoscon_transcompiling_HelloWorld = function() { };
hkoscon_transcompiling_HelloWorld.main = function() {
    var greet = function(speaker,target) {
        if(target == null) target = "World";
        console.log(speaker.first + " " + speaker.last + ": Hello, " + target
+ "!");
    };
    var andy = { first : "Andy", last : "Li"};
    greet(andy);
};
hkoscon_transcompiling_HelloWorld.main();
})(typeof console != "undefined" ? console : {log:function(){}});
```

Haxe features - static typing

source code: Typing.hx

```
class Typing {
 static function main():Void {
   var i = 123;                // same as var i:Int = 123;
   $type(i);                   // Int
   // i = "123";               // error: String should be Int

   var floats = [1.0, 1.1, 1.2];
   $type(floats);              // Array<Float>
   $type(floats[0]);           // Float
   floats[0] = i;
   trace(floats);              // [ 123, 1.1, 1.2 ]
   // floats[0] = "string";    // error: String should be Float
 }
}
```

Haxe features - OOP

source code: Opts.hx

```
class Point {
 public var x:Float;
 public var y:Float;
 public function new(x:Float, y:Float):Void {
   this.x = x;
   this.y = y;
 }
 public function offset(dx:Float = 0, dy:Float = 0):Point {
   return new Point(x + dx, y + dy);
 }
}

class Opts {
 static function main():Void {
   var p = new Point(0, 0);
   var p2 = p.offset(1, 2);
   trace(p2.x); //1
 }
}
```

Haxe features - OOP

cmd: haxe -main Opts -js Opts.js

```
(function (console) { "use strict";
var Point = function(x,y) {
	this.x = x;
	this.y = y;
};
Point.prototype = {
	offset: function(dx,dy) {
		if(dy == null) dy = 0;
		if(dx == null) dx = 0;
		return new Point(this.x + dx,this.y + dy);
	}
};
var Opts = function() { };
Opts.main = function() {
	var p = new Point(0,0);
	var p2 = p.offset(1,2);
	console.log(p2.x);
};
Opts.main();
})(typeof console != "undefined" ? console : {log:function(){}});
```

Haxe features - inlining

source code: Opts.hx

```
class Point {
 public var x:Float;
 public var y:Float;
 inline public function new(x:Float, y:Float):Void {
   this.x = x;
   this.y = y;
 }
 inline public function offset(dx:Float = 0, dy:Float = 0):Point {
   return new Point(x + dx, y + dy);
 }
}

class Opts {
 static function main():Void {
   var p = new Point(0, 0);
   var p2 = p.offset(1, 2);
   trace(p2.x); //1
 }
}
```

Haxe features - inlining

cmd: haxe -main Opts -js Opts.js

```
(function (console) { "use strict";
var Point = function(x,y) {
	this.x = x;
	this.y = y;
};
Point.prototype = {
	offset: function(dx,dy) {
		if(dy == null) dy = 0;
		if(dx == null) dx = 0;
		return new Point(this.x + dx,this.y + dy);
	}
};
var Opts = function() { };
Opts.main = function() {
	var p_x = 0;
	var p_y = 0;
	var p2_x = p_x + 1;
	var p2_y = p_y + 2;
	console.log(p2_x);
};
Opts.main();
})(typeof console != "undefined" ? console : {log:function(){}});
```

previously

```
var p = new Point(0,0);
var p2 = p.offset(1,2);
console.log(p2.x);
```

Haxe features - dead code elimination

cmd: haxe -main Opts -js Opts.js -dce full

```
(function (console) { "use strict";
var Opts = function() { };
Opts.main = function() {
	var p_x = 0;
	var p_y = 0;
	var p2_x = p_x + 1;
	var p2_y = p_y + 2;
	console.log(p2_x);
};
Opts.main();
})(typeof console != "undefined" ? console : {log:function(){}});
```

Haxe features - static analysis

cmd: haxe -main Opts -js Opts.js -dce full -D analyzer

```
(function (console) { "use strict";
var Opts = function() { };
Opts.main = function() {
	var p_x = 0;
	var p2_x = p_x + 1;
	console.log(p2_x);
};
Opts.main();
})(typeof console != "undefined" ? console : {log:function(){}});
```

Haxe features - functional programming

```
using Lambda; // static extension
import haxe.ds.*;

class Functional {
 static function main() {
   // Array comprehension
   var evens:Array<Float> = [for (i in 0...15) if (i % 2 == 0) i];
   trace(evens); // [ 0, 2, 4, 6, 8, 10, 12, 14 ]

   // functional goodies from `using Lambda`
   var maxMultipleOf4 = evens
     .filter(function(i) return i % 4 == 0)
     .fold(function(i, a) return Math.max(i, a), evens[0]);
   trace(maxMultipleOf4); // 12

   // enum (GADT) and pattern matching
   function getAnyHigher(floats:Array<Float>, v:Float):Option<Float> {
     for (f in floats)
       if (f > v)
         return Some(f);
     return None;
   }
   switch (getAnyHigher(evens, 5)) {
     case Some(value):
       // string interpolation (not really FP, but still nice)
       trace('In evens, $value is higher than 5');
     case None:
       trace("No value in evens is higher than 5");
   }
 }
}
```

Haxe features - compile-time macros

```
class Versioned {
 macro static function getGitVersion():haxe.macro.Expr {
   var git = new sys.io.Process("git", ["describe", "--tags"]);
   if (git.exitCode() != 0) {
     throw "`git describe --tags` failed: " + git.stderr.readAll().toString();
   }
   var tag = git.stdout.readLine();
   return macro $v{tag};
 }

 public static var VERSION(default, never):String = getGitVersion();
 static function main() {
   trace(VERSION); // something like "1.0-1-g123b31f"
 }
}
```

output:

```
(function (console) { "use strict";
var Versioned = function() { };
Versioned.main = function() {
    console.log(Versioned.VERSION);
};
Versioned.VERSION = "1.0-1-g123b31f";
Versioned.main();
})(typeof console != "undefined" ? console : {log:function(){}});
```

Summary

Haxe

- 9 compilation targets
  - reduced language/platform lock-in
- modern language and compiler features
  - justified the extra compilation step
- a fast optimizing compiler
  - minimized compilation time and performance overhead

Interested in Haxe?

- Read => http://haxe.org/
- Try => http://try.haxe.org/
- Get =>
  - with an installer: http://haxe.org/download/, or
  - with a package manager:
    - `brew install haxe` on Mac
    - `choco install haxe` on Windows
    - use a ppa on Ubuntu
      (the one from apt-get is super outdated...)

Get in touch!

@andy_li