Introduction to HaXe

For Flash and JavaScript Developers

David Peek @DavidPeek
Dominic De Lorenzo @misprintt
Dean Burge @DeanBurge

massive

What is haXe?

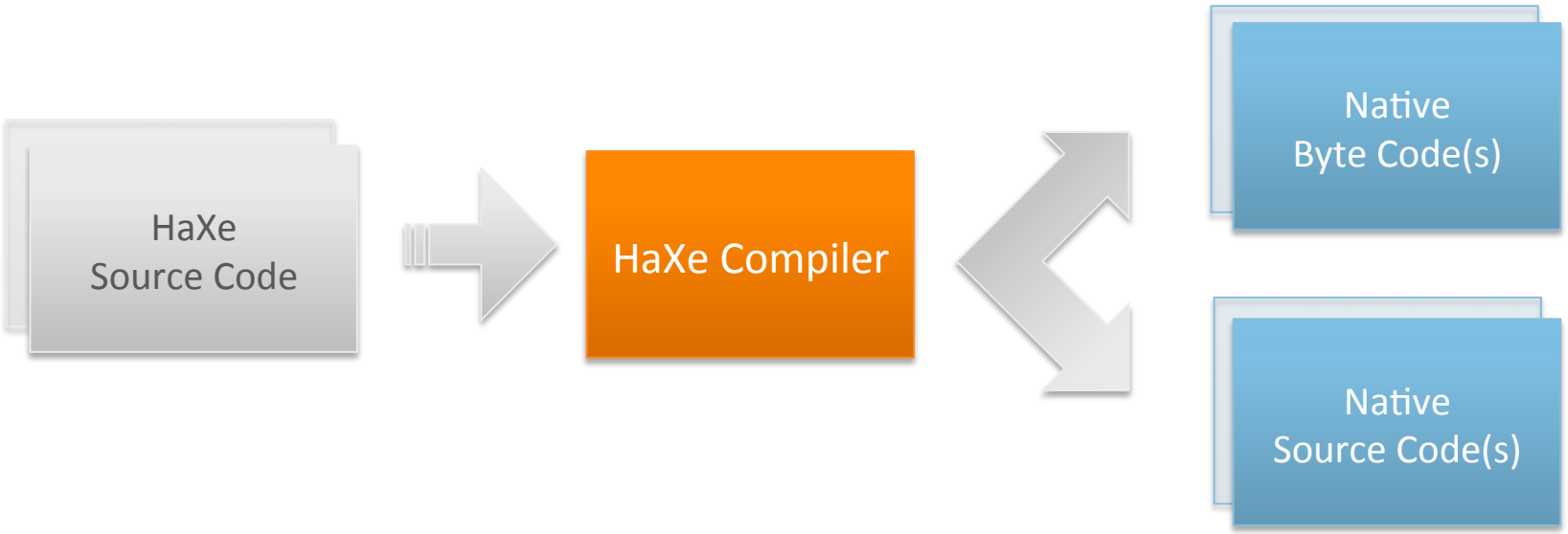
- Multi-platform language
- Open source (www.haxe.org)
- Community driven
- Version 2.08 (around since 2005)
- Single syntax for client-side, server-side & command line programming
  - » ECMAScript based syntax is easy for ActionScript and JavaScript developers to work with
- Strictly typed (compile time checking) across all targets
  - » Even for non compiled languages like JavaScript and PHP
- Lightning fast compilation
  - » Even for large applications with multiple target/device outputs
- Excellent runtime performance on target platforms

What platforms can HaXe target?

- Client side
  - Flash (AS2/3, AIR, Stage3D, etc)
  - JavaScript (HTML, Canvas, WebGL)
  - C++ (desktop, mobile platforms like Android and iOS)
- Server side
  - PHP
  - Neko
  - NodeJS
- Command Line
  - Neko
  - C++
- HaXe Compiler Macros!
  - Write HaXe code that executes at compile time against your application
  - Ideal for advanced code generation/customisation/optimisation
- Java and C#(.NET) Coming 'Soon'

How HaXe works

- The HaXe compiler translates source code into native source and/or bytecode
  - » Source code like ActionScript, JavaScript, C++, PHP
  - » Byte code like Flash, NekoVM

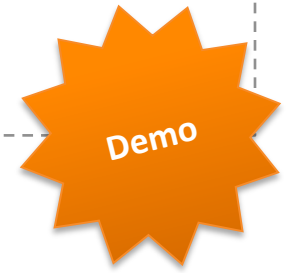


Hello World

HelloWorld.hx

```
class HelloWorld
{
    static function main()
    {
        return new HelloWorld();
    }

    public function new()
    {
        trace("Hello World!");
    }
}
```




```
haxe -x HelloWorld
```

Hello [ActionScript] World

```
haxe -main HelloWorld -swf HelloWorld.swf
```

Demo

Hello [JavaScript] World

```
haxe -main HelloWorld -js HelloWorld.js
```

index.html

```
<html>
<head>
    <title>Hello World</title>

</head>
<body>
<div id="haxe:trace"></div>
<script type="text/javascript" src="HelloWorld.js"></script>
</body>
</html>
```

Hello [Multi-Platform] World

build.hxml

```
# js
-main HelloWorld
-js HelloWorld.js

--next
# swf 9
-main HelloWorld
-swf HelloWorld.swf

--next
# swf 8
-main HelloWorld
-swf8 HelloWorld_8.swf

--next
# php
-main HelloWorld
-php build/php
-cmd echo ”----- PHP-----”
-cmd php build/php/index.php

--next
# neko
-main HelloWorld
-neko build/HelloWorld.n
-cmd echo ”----- NEKO-----”
-cmd neko build/HelloWorld.n

--next
# c++
-main HelloWorld
-cpp build/cpp
-cmd echo
-cmd echo ”----- CPP-----”
-cmd build/cpp/HelloWorld
```


```
haxe build.hxml
```

What HaXe has to offer

- **Single Language and syntax** across client, server and command line
- **Standard cross platform libraries** work across all client and server targets (like Date, Xml, Math, Http, etc)
- **Access full target platform APIs** (JavaScript, Flash, etc)
- **Interface with existing platform libraries** (e.g. SWCs, Away3D, JQuery, Raphael, etc)
- **Conditional compilation** flags for targeting specific platforms (#if js …. #elseif flash … #else)
- **3rd Party HaXe libraries** via HaxeLib (great open source community)

HaXe makes targeting new platforms is just looking up additional APIs, rather than having to learn additional languages

What HaXe has to offer ActionScript Developers?

- **Additional language features** like enums, generics, typedefs
- **Less verbose** strict typing (var s~~:String~~ = “foo”)
- Shorter compile times
- Write workflow tools in the same language (e.g. command line/server)
- Integrate with assets created from the Flash IDE
- Integrate with existing SWC libraries
- Output code to swf, ActionScript or swcs
- Target AVM1 and AVM2 with the same language

What HaXe has to offer JavaScript Developers?

- **Strict typing** (and compile time checking) of JavaScript code
- True **Object Oriented Programming** (vs. Prototype Based Programming)
- Advanced language features that allow for **beautiful, readable code**
- Generated JavaScript is fast but not always pretty… developers can use **macros** to easily extend or replace the default JS generator (using haXe of course!)

Cross Platform APIs

```
var string = "foo”;
var date = new Date();

var array = ["a", "b", "c"];

for(s in array) //less verbose loops
{
    trace(s);
}

var xml = Xml.parse("<foo id="bar" />");

var o:Dynamic = {};//equivalent to Object or *

var http:haxe.Http = new haxe.Http("http://www.example.org/
foo.bar");
```

Language Features

```
//smarter type inference
var foo = "bar";
foo = 2;//throws error typeof foo is String

//advanced compiler typing
var array:Array<String> = ["a", "b", "c"]; //proper generics
var handler:Int -> String -> Void; //typed function properties -
e.g. handler = function(i:Int, s:String):Void {};

//inlining constants for faster performance
inline static var FOO = "bar";

//enums rather than static strings and ints
enum Color{ red; green blue;}

//typedefs for compile time typing of dynamic objects
typedef Point = {x:Int; y:Int;}

//'using' mixin to inline references to external methods and APIs
using Reflect;
instance.fields();
```

Platform Specific APIs

Example.hx

```
#if js
import js.Dom;
#else if flash
import flash.display.Sprite;
#end

class Example
{
    public function new()
    {
        #if js
            js.Lib.alert("foo");
        #elseif flash
            flash.Lib.trace("foo");
        #else
            trace("foo");
        #end
    }
}
```

Leveraging 3rd Party APIs

- Use **Extern Classes** to define functions in external native libraries
- Haxe compiler can use native flash swcs, generate externs for typed API
- Many popular JS libraries have haXe externs already and JavaScript devs benefit from compile time type checking on those external JS APIs.
  - » e.g. jQuery, raphael, nodeJS, google-js

Foo.hx

```
extern class Foo
{
    public function new(id:String):Void;
    public function bar():Void;
    public var visible(default, null):Bool
}
```

Using HaxeLib Libraries

```
haxelib install jeash        //JavaScript/Canvas library
haxelib install nme          //C++ SDL Library
haxelib install hxcpp        //Neko-to-C++ library
```

build.hxml

```
#flash
-main Example
-swf9 Example.swf

--next

# js
--remap flash:jeash
-main Example
-lib jeash
-js Example.js
```

```
--next

# cpp
--remap flash:nme
-main Example
-lib nme
-lib hxcpp
-cpp example-cpp


-cmd example-cpp/Example
```

Multi Platform UI

Example.hx

```
import flash.display.Shape;
import flash.display.Sprite;
import flash.Lib;

class Example extends Example
{
    public static function main()
    {
        var example = new Example ();
        Lib.current.addChild(example);
    }

    public function new()
    {
        super();
        var circle:Shape = new Shape( );
        circle.graphics.beginFill( 0x7A9AA9 , 1 );
        circle.graphics.drawCircle( 0 , 0 , 40 );
        addChild(circle);
    }
```

Cross Platform Considerations

- Libraries like Jeash and NME make it simple to target multiple platforms using a single API
- But sometimes the flash API is overkill, and each library implements different subsets of it
- Your application will still require plenty of conditional compilation and package remapping
- The larger the project the more complex it is to manage cross platform implementations

Massive Open Source HaXe Libraries

We contribute a number of high quality, cross platform libraries to the HaXe community including:

- MUnit (metadata driven unit testing framework/runner similar to JUnit and FlexUnit)
- MCover (code coverage analysis)
- RobotHaxe (HaXe port of RobotLegs)
- Hxsignal (HaXe port of Robert Penner's AS3Signals)
- hamcrest-haxe (Haxe port of Hamcrest)
- More at https://github.com/massiveinteractive

Next Steps

- Download HaXe from http://haxe.org
- Browse through http://haxe.org/haxelib
- Subscribe to the HaXe mailing list (http://lists.motion-twin.com/mailman/listinfo/haxe)
- Enable your IDEs (http://haxe.org/com/ide)
- Follow people on twitter (@ncannasse @skial, @BrunoFonzi)
- Check out http://github.com/massiveinteractive
- Start coding!

Download this presentation and examples from http://ui.massive.com.au/talks/

What we are up to!

- Massivision UI Framework (http://massivision.com)
- Sublime Text Plugins (auto-completion, error checking)

Thank You.

www.massiveinteractive.com

David Peek @DavidPeek
Dominic De Lorenzo @misprintt
Dean Burge @DeanBurge