Xamarin + Native

Bridge the gap with bindings to native iOS and Android SDK's

Jonathan Dick

Principal Software Engineer - Microsoft

Xamarin.Forms Lead

@redth

Binding Topics

- What & Why?
- iOS
- Android
- Slim Bindings Demo

What & Why?

- C# ➡️ Android / iOS
- Bridge between .NET runtime and Native
- Xamarin is a gaggle of bindings
- P/Invoke, JNI

- Best of both worlds…
- You *want* to have it all
- You *can* have it all

iOS Bindings

Objective C

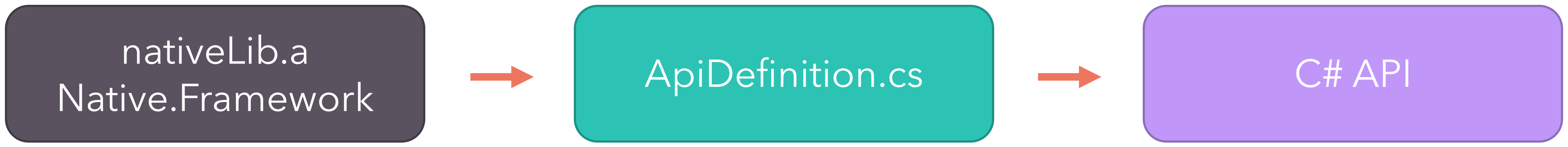
Swift

Obj-C Binding Process

Native Library

Binding Work

End Result



Native Library:

- Frameworks
- CocoaPods
- ‘FAT’ binary

Binding Work:

- Additive
- Objective Sharpie

End Result:

- Generated Code

Native Library

- Framework contains `someLibrary.a`, often FAT
- Use `lipo -info lib.a` check binary

- Creating a FAT binary:
  - Build for all archs:
    ```
    xcodebuild -sdk iphoneos -arch arm64
    xcodebuild -sdk iphonesimulator -arch x86_64
    ```
  - Combine archs:
    ```
    lipo -create -o fat.a /
      build/Release-iphoneos/Lib/libLib.a /
      build/Release-iphonesimulator/Lib/libLib.a
    ```

Objective Sharpie

- macOS Installer: aka.ms/objective-sharpie
- Docs: https://docs.microsoft.com/xamarin/cross-platform/macios/binding/objective-sharpie/

- Command line tool:

```
sharpie bind \
     -framework MapboxWrapper.framework \
     -sdk iphoneos14.2
```

- Generates Binding Code:

```
ApiDefinition.cs
Structs.cs
```

ApiDefinition.cs

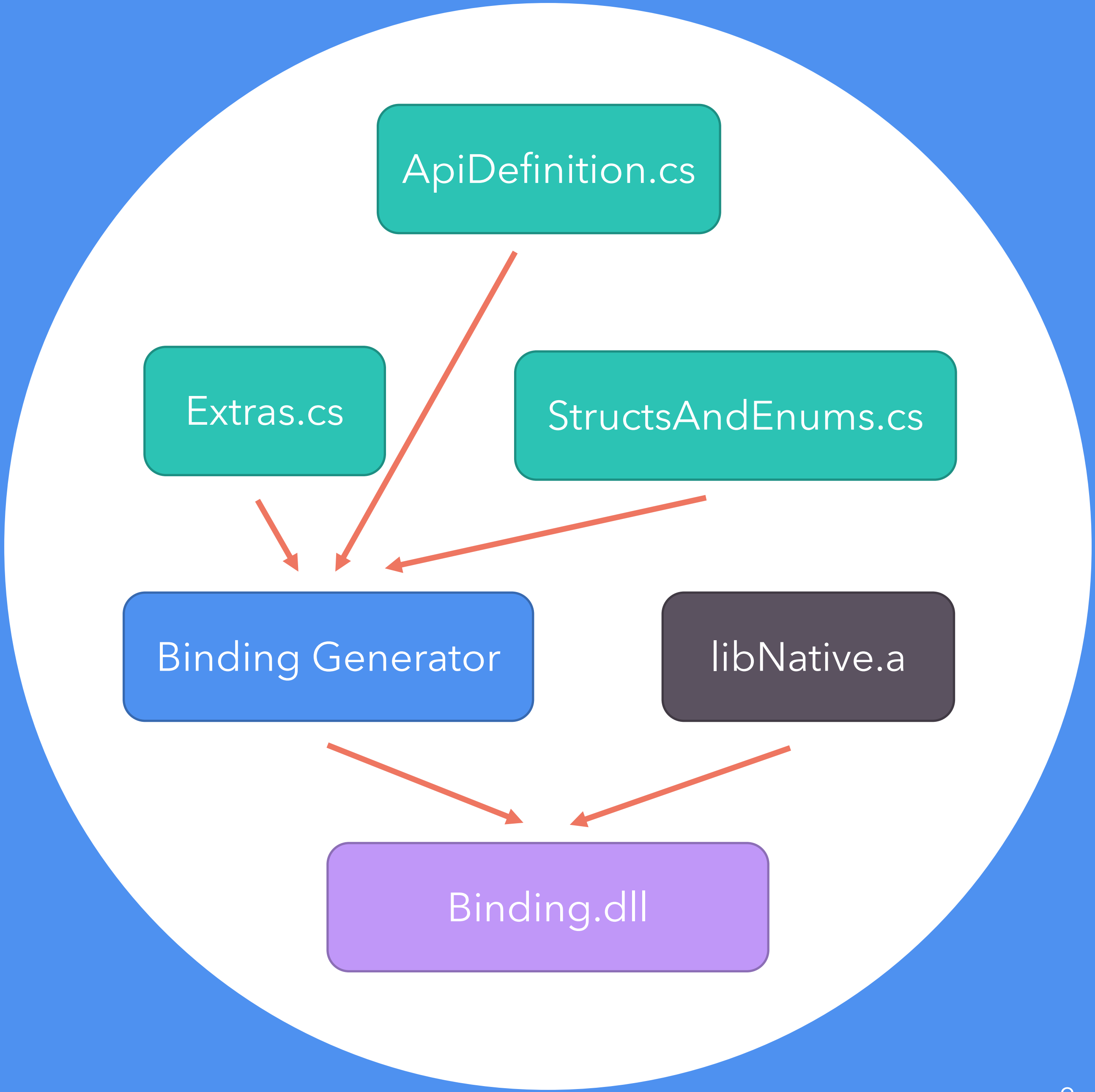
```
[BaseType (typeof (NSObject))]
[Model, Protocol]
interface MyProtocol
{
   [Abstract] [Export ("say:")]
   void Say (string msg);

   [Export ("listen")]
   void Listen ();
}
```

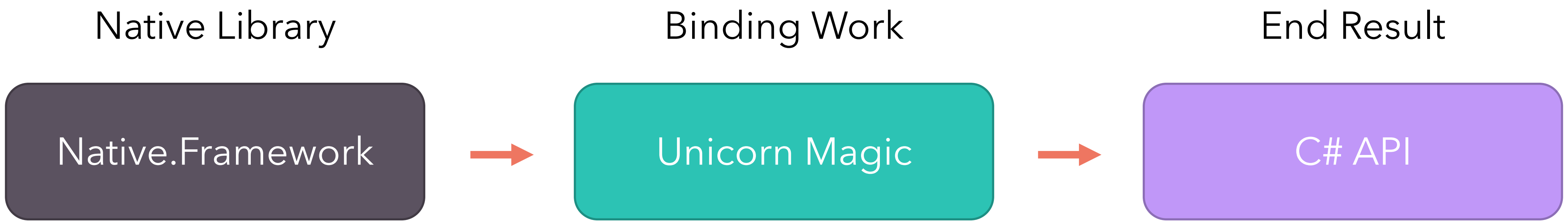
```
interface IMyProtocol {}

[BaseType (typeof(NSObject))]
interface MyTool
{
   [Export ("getProtocol")]
   IMyProtocol GetProtocol ();
}
```

Generate Binding .dll



Swift Binding Process



- CocoaPods

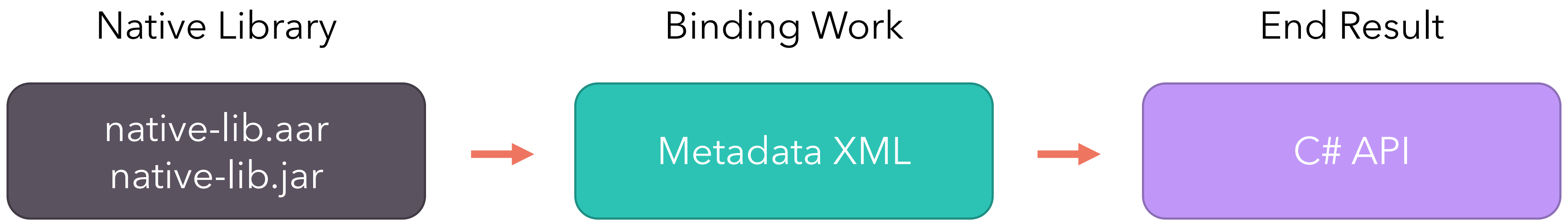
- Binding Tools for Swift (BTFS) https://aka.ms/btfs

- Generated Code

* Swift libraries often contain Obj-C headers

Android Bindings

Android Binding Process



- AAR vs JAR
- Maven

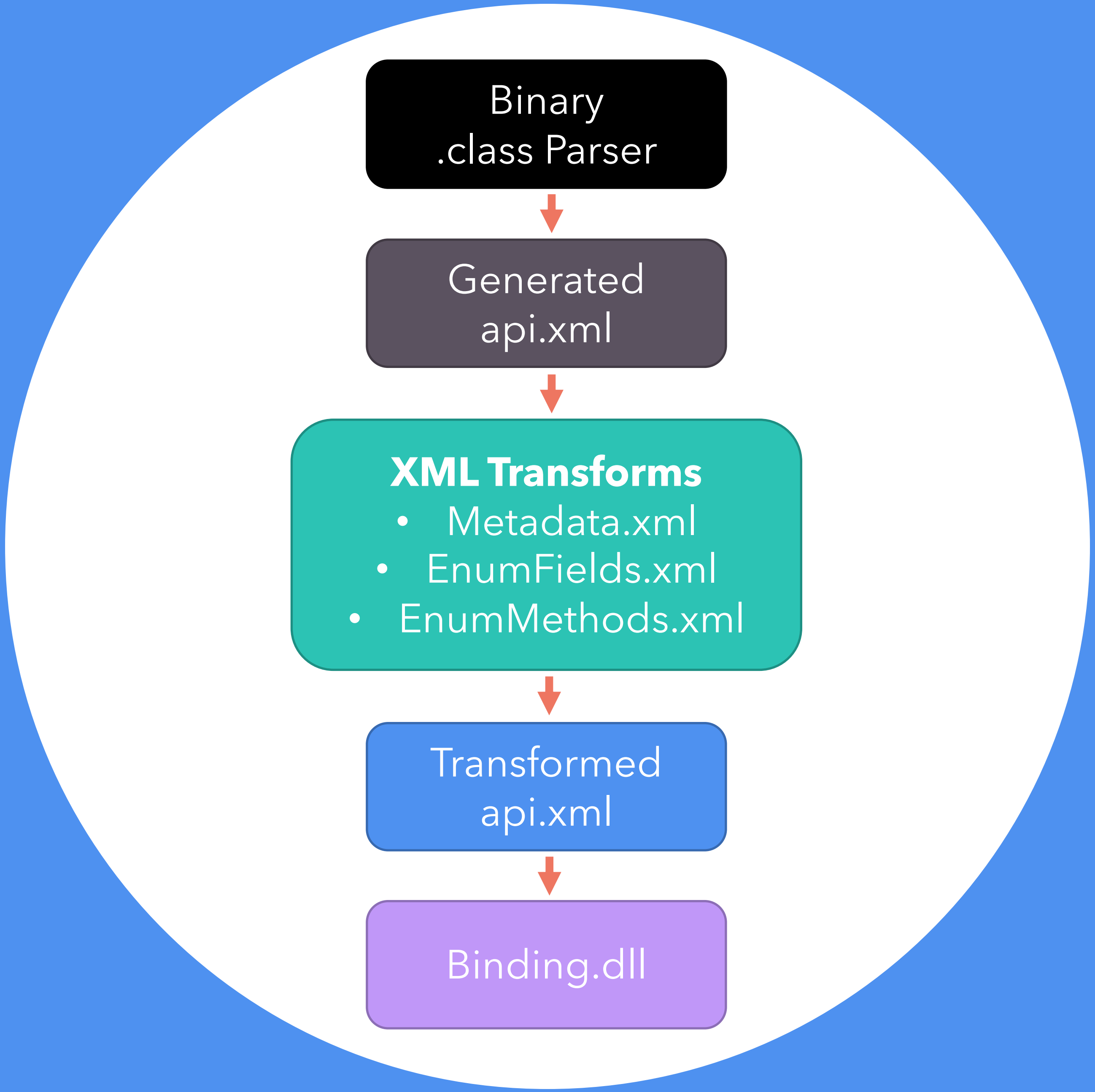
- Subtractive
- XPath Transforms

- Generated Code

Native Library

- .aar's are .jar's for pirates – they have hidden treasure
  - jar(s), resources, manifest, proguard configs, etc.
- Docs: [docs.microsoft.com/xamarin/android/platform/binding-java-library]()
- Maven
  - Maven Central
  - MVN Repository
  - BinTray
  - Google Maven
- Dependencies
  - Manual resolution – walk the tree
  - Xamarin Gradle plugin

Generate Binding .dll



Generated api.xml

```
<api>
   <package name="com.microsoft.device.display" jni-name="com/microsoft/device/display">
     <class abstract="false" deprecated="not deprecated" extends="java.lang.Object"
       extends-generic-aware="java.lang.Object“
       jni-extends="Ljava/lang/Object;" final="true" name="DisplayMask" static="false"
       visibility="public" jni-signature="Lcom/microsoft/device/display/DisplayMask;">
       <method abstract="false" deprecated="not deprecated" final="false" name="fromResourcesRect"
         jni-signature="(Landroid/content/Context;)Lcom/microsoft/device/display/DisplayMask;“
         bridge="false" native="false" return="com.microsoft.device.display.DisplayMask“
         jni-return="Lcom/microsoft/device/display/DisplayMask;“  static="true" synchronized="false"
         synthetic="false" visibility="public">
         <parameter name="context" type="android.content.Context" jni-type="Landroid/content/Context;">
         </parameter>
       </method>
     </class>
   </package>
</api>
```

XML Transforms – Metadata.xml

```
<metadata>
  <!-- Rename the namespace -->
  <attr path="/api/package[@name='com.microsoft.device.display']" name="managedName">Microsoft.Device.Display</attr>

  <!-- Keep these as methods and not properties to match the `getBoundingRectsForRotation(int)` method -->
  <attr  path="/api/package[@name='com.microsoft.device.display']/class[@name='DisplayMask']/method[@name='getBoundingR
ects']" name="propertyName"></attr>
</metadata>
```

Change: `<attr path="{XPATH}" name="{ATTRIBUTE}">{VALUE}</attr>`

{ATTRIBUTE} = managedName, managedType, managedReturn, eventName, argsType, propertyName, obfuscated, visibility

Remove: `<remove-node path="{XPATH}" />`

Add: `<add-node path="{XPATH}"> <!-- {ELEMENTS} --> </add-node>`

* **Optional:** EnumFields.xml & EnumMethods.xml help bind integer constants to delightful C# enums

Why Slim Bindings?

- Complex API's are hard to bind
- Simple API's almost always require no manual intervention
- Native SDK documentation is easier to follow with native tools

How Slim Bindings?

- Create a Simple API
  - Xcode / Swift
  - Android Studio / Java
- Wrapper or Abstraction
- Use known types
  - Strings, numbers …
  - UIKit, Foundation …
  - AndroidX, android.jar …

Slim Bindings
Demo

Questions?

Jonathan Dick

Principal Software Engineer - Microsoft

Xamarin.Forms Lead

@redth

Questions?

Thank you!

Jonathan Dick

Principal Software Engineer - Microsoft

Xamarin.Forms Lead

@redth