Tutorial MOBILESoft 2016: Opportunities and pitfalls when using cross-platform tools for mobile app development.

Ruben Smeets  
ES&S  
KU Leuven, Belgium

Michiel Willocx  
MSEC iMinds-Distrinet  
KU Leuven, Belgium
About us

MSEC – Mobile and Secure

- **Research line 1**: designing secure mobile applications
- **Research line 2**: inspecting system level security & privacy

MSEC Website: https://iiw.kuleuven.be/onderzoek/msec
About us
ES&S – Embedded Systems & Security

Research focus: Internet Of Things

Configurable
- Logic Block
- Interconnection Resources
- I/O Cell

Hardware

Networks

Security
- Security
- Power/Energy
- Delay
- Resources

Heterogeneous
CrossMoS

Cost-efficient development of advanced, cross-platform mobile applications

- IWT/VLAIO TETRA project
- Project of 2 years
- In collaboration with app developers, small companies and SMEs
- Researchers:
  - Michiel Willocx & Ruben Smeets
- Project Website:
  - https://www.msec.be/crossmos/
Table of contents

• TUTORIAL TODAY:
  o PART 1:
    • Introduction: What are cross-platform tools and why should I use them?
    • Classification of cross-platform tools
    • Cross-platform tool selection criteria
  o PART 2: The native JavaScript framework landscape (Ruben)
    • Why Native JavaScript?
    • Possible candidates
    • Comparison of three popular tools
  o PART 3: Discussion

• TOMORROW: Tutorial Session
  o Web-based Hybrid Mobile Apps: State of the Practice and Research Opportunities (Ivano Malavolta)
Introduction
Introduction

• Mobile platforms
**Introductie**

- Mobile platform sales

### Worldwide Smartphone Sales to End Users by Operating System in 4Q15 (Thousands of Units)

<table>
<thead>
<tr>
<th>Operating System</th>
<th>4Q15 Units</th>
<th>4Q15 Market Share (%)</th>
<th>4Q14 Units</th>
<th>4Q14 Market Share (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>325,394.4</td>
<td>80.7</td>
<td>279,057.5</td>
<td>76.0</td>
</tr>
<tr>
<td>iOS</td>
<td>71,525.9</td>
<td>17.7</td>
<td>74,831.7</td>
<td>20.4</td>
</tr>
<tr>
<td>Windows</td>
<td>4,395.0</td>
<td>1.1</td>
<td>10,424.5</td>
<td>2.8</td>
</tr>
<tr>
<td>Blackberry</td>
<td>906.9</td>
<td>0.2</td>
<td>1,733.9</td>
<td>0.5</td>
</tr>
<tr>
<td>Others</td>
<td>887.3</td>
<td>0.2</td>
<td>1,286.9</td>
<td>0.4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>403,109.4</strong></td>
<td><strong>100.0</strong></td>
<td><strong>367,334.4</strong></td>
<td><strong>100.0</strong></td>
</tr>
</tbody>
</table>

Source: Gartner (February 2016)
Native Development
Native development: Android

Java + XML
Native development: Android

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/propview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="top"
    android:layout_margin="16dp"
    android:orientation="vertical">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:text="@string/search_description" />

    <EditText
        android:id="@+id/search"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:inputType="text"
        android:imeOptions="actionSearch" />

    <Button
        android:id="@+id/do_search"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:layout_margin="@attr/activity_horizontal_margin"
        android:text="@string/do_search_text" />

</LinearLayout>
```

Use the form below to search for houses to buy. You can search by place-name, postcode, or click 'My location', to search in your current location!
Native development: iOS

- Objective-C
- Swift

[Diagram of tools and icons]
Native development: iOS

- Storyboards
Native development: Windows Phone

Visual Studio

C# + XAML
Native development: Windows Phone

```xml
<TextBlock Text="PropertyCross"
FontSize="{StaticResource PhoneFontSizeExtraLarge}"/>
<TextBlock Text="Use the form below to search for houses to buy. You can search by place-name, postcode, or click 'My location', to search in your current location!"
Grid.Row="1"
TextWrapping="Wrap"
Margin="0,30,0,0"/>

<!-- search form -->
<StackPanel Orientation="Horizontal"
Grid.Row="2"
Margin="0,10,0,0">
<TextBox Text="{Binding SearchText, Mode=TwoWay}"
KeyDown="TextBox_KeyDown"
Width="200"
IsEnabled="{Binding IsLoading, Converter={StaticResource BooleanNotConverter}}"/>
<Button Content="Go"
IsEnabled="{Binding IsLoading, Converter={StaticResource BooleanNotConverter}}"
Command="{Binding SearchCommand}"
/>
<Button IsEnabled="{Binding IsLoading, Converter={StaticResource BooleanNotConverter}}"
Command="{Binding SearchMyLocationCommand}"
Content="My location"/>
</StackPanel>

<!-- loading indicator -->
<Grid Height="8" Grid.Row="3">
<ProgressBar Indeterminate="True"
Visibility="{Binding IsLoading, Converter={StaticResource BoolToVisibilityConverter}}"/>
</Grid>

Recent Searches

<table>
<thead>
<tr>
<th>Place</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bristol</td>
<td>314</td>
</tr>
<tr>
<td>BS1</td>
<td>33</td>
</tr>
</tbody>
</table>
```
Overview native development

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Objective – C Swift</th>
<th>Java</th>
<th>.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>Story boards</td>
<td>XML-files</td>
<td>.XAML-files</td>
</tr>
<tr>
<td>IDE</td>
<td>Xcode</td>
<td>Android Studio</td>
<td>Visual Studio</td>
</tr>
</tbody>
</table>
# Problems native development

<table>
<thead>
<tr>
<th></th>
<th>Apple</th>
<th>Android</th>
<th>.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programming Language</strong></td>
<td>Objective – C Swift</td>
<td>Java</td>
<td>.NET</td>
</tr>
<tr>
<td><strong>User Interface</strong></td>
<td>Story boards</td>
<td>XML-files</td>
<td>.XAML-files</td>
</tr>
<tr>
<td><strong>IDE</strong></td>
<td>Xcode</td>
<td>Android Studio</td>
<td>Visual Studio</td>
</tr>
</tbody>
</table>

+ different development techniques
+ different application lifecycles
Problems native development

- Development time
- Time to release updates and to fix bugs
- Development cost
- Necessary programming skills
Solution?

Cross-Platform Tools
### Cross-Platform Tools (CPTs)

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Objective-C</th>
<th>Java</th>
<th>.NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swift</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Story boards</th>
<th>XML-files</th>
<th>.XAML-files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1 user interface</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDE</th>
<th>Xcode</th>
<th>Android Studio</th>
<th>Visual Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

→ support all platforms with one (partially) shared code base
Examples

Xamarin  snowkit  Marmalade

Qt  OpenFL  appcelerator

>100  PhoneGap  Corona SDK

AppGyver  Tfactr

Sencha  ICENIUM

ENYO  ADOBE AIR

unity  ionic  jQuery mobile
Classification of Cross-Platform-Tools
Classification of CPT’s

- Web Apps (JavaScript Frameworks)
- Web-to-native Wrappers
- Runtimes
- Source code translators
- App Factories
Classification of CPT’s

- Web Apps (JavaScript Frameworks)
- Web-to-native Wrappers
- Runtimes
- Source code translators
- App Factories
App Factories

- Drag and drop app design
- Automatic code generation
- Little to no self-written code
- Used for writing simple applications (e.g. RSS feed reader)
App Factories

- No programming skills required
- Often ability to develop in cloud

- Limited UI capabilities
- Limited overall possibilities
App Factories: Examples

Available in-app functions

Common Functions
- Websites
- MyBlog
- News
- Photos
- YouTube
- Videos
- Contacts
- Directions
- Calendar
- HTML Page
- Forms
- Docs
- LiveChat

Social Feeds
- News & Blogs
- Photos & Videos

Appearance
- Backgrounds
- Header
- Icons

My Very First App

Video Help

MyBlog
Include RSS feed from your favourite blog, or just about any RSS feed. You will need to find the URL (address) for the feed itself, not just the homepage. Look for this symbol or the word 'RSS' or 'feed' on the source site.

Specs / Info
- Icon Title: MyBlog
- Feed URL: RSS feed link

Live Preview
App Saved 32 secs ago
App Factories: Examples
Classification of CPT’s

- Web Apps (JavaScript Frameworks)
- Web-to-native Wrappers
- Runtimes
- Source code translators
- App Factories
Web Apps

➤ Mobile Websites

- Accessed in standard mobile browser (Chrome, Safari, …)
- Optimized for mobile device screen sizes
Web Apps: JavaScript Frameworks

UI Components

• Layout
  • Optimization, scaling and formatting for mobile screen sizes
  • Native-looking skins (not always available)
• Optimization for touch functionality
Web Apps: JavaScript Frameworks

Other Components

Assistance in:
• DOM manipulation
• Utility functions (e.g. array manipulations, access Web resource)
• Event handling (e.g. on click, gestures)
Web Apps: JavaScript Frameworks

Use of design patterns

- MVC (e.g. AngularJS)
- MVVM (e.g. KnockoutJS)
- ...
Web Apps

- No platform-specific code
- Easy to develop
- Easy to update
- Easy to distribute (URL)
- Lots of support and different frameworks available

- Internet access always needed
- Responsiveness (partly) depends on Internet connection
- Not a real, stand-alone application
- Limited access to device features
- Often no native look and feel
- Depends on browser capabilities
JavaScript Frameworks: Examples

- ionic
- ANGULARJS by Google
- jQuery mobile
- Sencha
- PhoneJS
- Knockout
- REQUiRE.JS

KU LEUVEN
Web Apps: Sencha Touch

• Ability to use native skins
Web Apps: Ionic

- Based on AngularJS by Google

→ MVC Design Pattern
Web Apps: Ionic

• Focusses on look & feel and UI interaction

(Recently also native skins)

<table>
<thead>
<tr>
<th>Beers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbey Roade Belgian Ale</td>
</tr>
<tr>
<td>By Appalachian Brewing Company</td>
</tr>
<tr>
<td>Abbey Triple</td>
</tr>
<tr>
<td>By Stoudt's Brewery</td>
</tr>
<tr>
<td>Abijah Rowe IPA</td>
</tr>
<tr>
<td>By The Cambridge House</td>
</tr>
<tr>
<td>Abita Amber</td>
</tr>
<tr>
<td>By Abita Brewing Company</td>
</tr>
<tr>
<td>Abita Golden</td>
</tr>
<tr>
<td>By Abita Brewing Company</td>
</tr>
<tr>
<td>Abita Jockamo IPA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beers</th>
<th>Breweries</th>
<th>Nearby</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Abbey Triple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stoudt's Brewery</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belgian and French Ale</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belgian-Style Tripel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Triple is a strong, full-bodied Belgian abbey-style ale. The authentic Belgian yeast strain used in fermentation contributes to a rich array of spicy, phenolic, and fruit-like flavors and noticeable alcoholic warmth. This unfiltered ale has an</td>
</tr>
</tbody>
</table>
Classification of CPT’s

- Web Apps (JavaScript Frameworks)
- Web-to-native Wrappers
- Runtimes
- Source code translators
- App Factories
Web-To-Native Wrappers

- Web Apps, packaged as a native, stand-alone application
- Web code is displayed in a chromeless webview
- Wider range of native API calls compared to normal Web browser
Web-To-Native Wrappers

- Allow Web developers to make mobile applications
- Convert existing Web services to mobile applications
- Stand-alone application
- More available device features than Web apps

- Poorer UX compared to native
- Often no native look and feel
- Performance overhead
Web-To-Native Wrapper: PhoneGap

- Acquired by Adobe in 2011
- Supported OS:
  - Android
  - iOS
  - Windows Phone
  - BlackBerry
  - ...

- Alternatives for PhoneGap as web-to-native wrapper?
  ➔ discontinued, never used, bankrupt,…
PhoneGap Explained
PhoneGap Explained: Packaging

Build options:
- Local ➔ PhoneGap CLI
- Cloud ➔ PhoneGap Build

Web App

Native Application
Examples of PhoneGap applications
Examples of PhoneGap applications

Official Wikipedia App for Android, iOS and Playbook. Wikipedia is the free encyclopedia containing more than 20 million articles in 280 languages, and is the most comprehensive and widely used reference work humans have ever compiled.

Features:
- Save article to read later or offline
- Search articles nearby
- Share articles using Android "Share" function
- Read article in a different language
- Full screen search

"Wikipedia has to be everywhere, and Adobe PhoneGap helps us get it there," says Tomasz Finc, director of mobile and special projects at Wikimedia. "Within its first three weeks of release, the app became the number one search result for Wikipedia in the Android marketplace, and now has more than 3.6 million total user installs and 5.3 million active device installs (as of Sept 2012). As we push our products out to multiple mobile platforms, the PhoneGap development process becomes easier and easier. If we're spending less and less time on each platform then we're doing something right."
For more information on this topic…

KEEP CALM
AND
COME BACK TOMORROW
Classification of CPT’s

• Web Apps (JavaScript Frameworks)
• Web-to-native Wrappers
• **Runtimes**
• Source code translators
• App Factories
Runtimes

- Cross-platform compatibility layer
- Shields app form underlying differences between platforms
- Different strategies:
  - Interpreted at runtime
  - Compiled in advance (source code translators)
Runtimes

- Good overall user experience
- Less reliant on native webview component/JavaScript engine
- Application developers can choose Runtime based on programming language

- Often, platform specific code is needed
- Runtimes introduce significant overhead
- Learning curve is often quite steep
Runtimes: examples

- Adobe AIR
- titanium
- Xamarin
- unity
- Corona
Titanium explained

- Written in JavaScript
- No cross-compilation
- JavaScript code evaluated at runtime
- Titanium API maps JavaScript code on Native API (1:1)
## Difference between PhoneGap and Titanium

<table>
<thead>
<tr>
<th>PhoneGap</th>
<th>titanium™</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses JavaScript</td>
<td>Uses JavaScript</td>
</tr>
<tr>
<td>WebView</td>
<td>Runtime</td>
</tr>
<tr>
<td>Renders HTML pages in chrome-less browser</td>
<td>Interprets JavaScript code and maps on Native API</td>
</tr>
<tr>
<td>Developer writes Web app</td>
<td>Developer writes “native” application using JavaScript</td>
</tr>
</tbody>
</table>
Classification of CPT’s

- Web Apps (JavaScript Frameworks)
- Web-to-native Wrappers
- Runtimes
- **Source code translators**
- App Factories
Source Code Translator

- Different strategies:
  - Translate to native source
  - Translate to intermediary language
  - Translate to low level machine code
- Often used in combination with Runtime
Source Code Translator

- Good user experience and performance
- Application developers can choose tool based on programming language
- Generate 100% native applications

- Often, platform specific code is needed
- Learning curve is often quite steep
- High complexity, supporting new APIs is time consuming, extending the framework is not trivial
Source Code Translator: examples

Xamarin

Qt

NEOMADES
Source Code Translator: NEOMAD
Cross-platform technology

1) Based on web technology

Web Apps

Web-To-Native Wrappers

2) Not based on Web technology

Runtimes

- Application Code
- Runtime Environment
- Operating System

Source Code Translator
Runtimes & Source code translators

4 possibilities

1)

Source Code → Executable Code

Executable Code → Operating System

2)

Source Code

Platform specific source code

Executable Code → Operating System

3)

Source code

Runtime Environment

Operating System

4)

Source code

Intermediary language

Runtime Environment

Operating System
Combination Source Code Translator and Runtime: Xamarin

- Uses Runtime
- Code written in C#
- Supported platforms:
  - Android
  - iOS
  - Windows Phone
  - (OS X)
  - ...
- Recently acquired by Microsoft
Xamarin explained: development

Shared logic (business logic, domain models, view models)

Portable .NET Framework
Xamarin explained: Xamarin.Forms

Windows Tablets + Ultrabooks (touch enabled)
Windows Phones
Android Phones and Tablets
iPhone and iPads

UI (Xaml)
Xamarin.Forms
Stylus support
Near Field Communication
Physical
SD cards
Fingerprint sensor
Airplay

IOS UI (Cocoa Touch)

Shared logic (business logic, domain models, view models)

Portable .NET Framework
<table>
<thead>
<tr>
<th>Views for Presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
</tr>
<tr>
<td>Image</td>
</tr>
<tr>
<td>BoxView</td>
</tr>
<tr>
<td>WebView</td>
</tr>
<tr>
<td>Map</td>
</tr>
<tr>
<td>Views that Initiate Commands</td>
</tr>
<tr>
<td>Button</td>
</tr>
<tr>
<td>SearchBar</td>
</tr>
<tr>
<td>Views for Common Data Types</td>
</tr>
<tr>
<td>Slider (double)</td>
</tr>
<tr>
<td>Stepper (double)</td>
</tr>
<tr>
<td>Switch (bool)</td>
</tr>
<tr>
<td>DatePicker</td>
</tr>
<tr>
<td>TimePicker</td>
</tr>
<tr>
<td>Views for Editing Text</td>
</tr>
<tr>
<td>Entry (single line)</td>
</tr>
<tr>
<td>Editor (multiple lines)</td>
</tr>
<tr>
<td>Views to Indicate Activity</td>
</tr>
<tr>
<td>ActivityIndicator</td>
</tr>
<tr>
<td>verwenden</td>
</tr>
</tbody>
</table>

**KU LEUVEN**
Xamarin: Android vs iOS

- Source translated to Intermediary Language (IL)
- Just-In-Time (JIT) compilation

- Source translated to executable binary code
- Ahead-Of-Time (AOT) compilation
Runtimes & Source code translators

4 possibilities

1) Source Code → Executable Code → Operating System

2) Source Code → Platform specific source code → Executable Code → Operating System

3) Source code → Runtime Environment → Operating System

4) Source code → Intermediary language → Runtime Environment → Operating System
Unity

- Used in many gaming applications
- Specialised in rendering 3D and animated images
- Runtime (with translation to an intermediary language)
Runtimes & Source code translators

4 possibilities

1) Source Code → Executable Code → Operating System

2) Source Code → Platform specific source code → Executable Code → Operating System

3) Source code → Runtime Environment → Operating System

4) Source code → Intermediary language → Runtime Environment → Operating System
Runtimes & Source code translators

4 possibilities

1) Source Code -> Executable Code
   Executable Code -> Operating System

2) Source Code
   Platform specific source code -> Executable Code
   Executable Code -> Operating System

3) Source code
   Runtime Environment
   Operating System

4) Source code
   Intermediary language
   Runtime Environment
   Operating System
Cross-Platform Tool Selection Criteria
Cross-Platform Tool Selection Criteria

Performance
- Development infrastructure
- Supported platforms
- UI capabilities
- Programming Language
- Access to device features
- Security
## Supported platforms

<table>
<thead>
<tr>
<th>Technology</th>
<th>Tool</th>
<th>Android</th>
<th>iOS</th>
<th>WP</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Framework + PhoneGap</td>
<td>ALL JavaScript Frameworks</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Runtime</td>
<td>Titanium</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NativeScript</td>
<td>✔️</td>
<td>✔️</td>
<td>ALPHA</td>
</tr>
<tr>
<td></td>
<td>ReactNative</td>
<td>✔️</td>
<td>✔️</td>
<td>ALPHA</td>
</tr>
<tr>
<td>Source code Translator</td>
<td>Eqela</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>NeoMAD</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Source Code Translator + Runtime</td>
<td>Xamarin</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Qt</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Adobe Air</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>Unity</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>App Factory</td>
<td>AppMakr</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>ViziApps</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
## Development infrastructure: Programming languages

<table>
<thead>
<tr>
<th>Technology</th>
<th>Tool</th>
<th>Programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Framework + PhoneGap</td>
<td>ALL JavaScript Frameworks</td>
<td>JavaScript, HTML, CSS</td>
</tr>
<tr>
<td>Runtime</td>
<td>Titanium</td>
<td>JavaScript</td>
</tr>
<tr>
<td></td>
<td>NativeScript</td>
<td>JavaScript</td>
</tr>
<tr>
<td></td>
<td>ReactNative</td>
<td>JavaScript</td>
</tr>
<tr>
<td>Source code Translator</td>
<td>NeoMAD</td>
<td>Java</td>
</tr>
<tr>
<td>Source Code Translator + Runtime</td>
<td>Xamarin</td>
<td>C#</td>
</tr>
<tr>
<td></td>
<td>Qt</td>
<td>C++/QML</td>
</tr>
<tr>
<td></td>
<td>Adobe Air</td>
<td>ActionScript</td>
</tr>
<tr>
<td></td>
<td>Unity</td>
<td>C#, UnityScript (JavaScript)</td>
</tr>
<tr>
<td>App Factory</td>
<td>AppMakr</td>
<td>Drag &amp; Drop</td>
</tr>
<tr>
<td></td>
<td>ViziApps</td>
<td>Drag &amp; Drop</td>
</tr>
</tbody>
</table>
## Development infrastructure: Programming environment

<table>
<thead>
<tr>
<th>Technology</th>
<th>Tool</th>
<th>Programming environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Framework + PhoneGap</td>
<td>ALL JavaScript Frameworks</td>
<td>Any text editor / web IDE</td>
</tr>
<tr>
<td>Runtime</td>
<td>Titanium</td>
<td>Titanium IDE</td>
</tr>
<tr>
<td></td>
<td>NativeScript</td>
<td>Appbuilder</td>
</tr>
<tr>
<td></td>
<td>ReactNative</td>
<td>Text editor, Nuclide, Deco</td>
</tr>
<tr>
<td>Source code Translator</td>
<td>NeoMAD</td>
<td>NeoMAD IDE (based on eclipse)</td>
</tr>
<tr>
<td>Source Code Translator + Runtime</td>
<td>Xamarin</td>
<td>Xamarin Studio</td>
</tr>
<tr>
<td></td>
<td>Qt</td>
<td>QT creator</td>
</tr>
<tr>
<td></td>
<td>Adobe Air</td>
<td>Adobe Flash Builder</td>
</tr>
<tr>
<td></td>
<td>Unity</td>
<td>Visual Studio (+plugin)</td>
</tr>
<tr>
<td>App Factory</td>
<td>AppMakr</td>
<td>Cloud development tool</td>
</tr>
<tr>
<td></td>
<td>ViziApps</td>
<td>Cloud development tool</td>
</tr>
</tbody>
</table>
## Development infrastructure: License cost

<table>
<thead>
<tr>
<th>Technology</th>
<th>Tool</th>
<th>Programming environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Framework + PhoneGap</td>
<td>ALL JavaScript Frameworks</td>
<td>PhoneGap: always free FREE: Ionic, jQuery Mobile, … PAID: Sencha Touch $4475+/year (5 devs), …</td>
</tr>
<tr>
<td>Runtime</td>
<td>Titanium</td>
<td>$39/month (1 dev)</td>
</tr>
<tr>
<td></td>
<td>NativeScript</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>ReactNative</td>
<td>Free</td>
</tr>
<tr>
<td>Source code Translator</td>
<td>NeoMAD</td>
<td>$999/year (1 dev)</td>
</tr>
<tr>
<td>Source Code Translator + Runtime</td>
<td>Xamarin</td>
<td>Free community license</td>
</tr>
<tr>
<td></td>
<td>Qt</td>
<td>$3540/year (1dev)</td>
</tr>
<tr>
<td></td>
<td>Adobe Air</td>
<td>Free</td>
</tr>
<tr>
<td></td>
<td>Unity</td>
<td>$75/month (1dev)</td>
</tr>
<tr>
<td>App Factory</td>
<td>AppMakr</td>
<td>$1/month</td>
</tr>
<tr>
<td></td>
<td>ViziApps</td>
<td>$33/month</td>
</tr>
</tbody>
</table>

Based on lowest prices, additional support/tools introduce extra costs
UI Capabilities
UI Capabilities: Web Apps and Web-To-Native Wrappers

- Easy UI development
  - CSS templates
  - JavaScript Frameworks (e.g. Ionic)
- Tons of CSS and JavaScript Frameworks freely available
- Good looking applications with little to no effort
- Some JavaScript Frameworks offer native skins (e.g. Sencha Touch)
Example Sencha Touch

PropertyCross

Use the form below to search for houses to buy. You can search by place-name, postcode, or click 'My location', to search in your current location!

LOCATION
BS5
Go My Location

Previous Searches
BS5 56
Bristol 275

20 of 56 matches
£160,000 Victoria Place, Redfield, Bristol, Avon, BS5
£184,950 Thorne Rd, Greenbank, Bristol, Avon, BS5
£145,950 Devlin Road, Easton, Bristol, Avon, BS5
£179,950 Goldham Rd, St. George, Bristol, Avon, BS5
£184,950 Northcote Rd, St. George, Bristol, Avon, BS5
£135,000 Tudor Road, Easton, Bristol, Avon, BS5

Load more...
Results for BS5, showing 20 of 56 properties

No Service 18:04

PropertyCross

Use the form below to search for houses to buy. You can search by place-name, postcode, or click 'My location', to search in your current location!

Location

Go My Location

Previous Searches
BS2 53
BS1 31
51.45,-2.59 65

20 of 31 matches
£183,950 Canons Way, Harbourside, BS1 - Modern
£164,000 Hotwell Road, BS8
£229,950 Horizon, Broad Weir, Bristol, BS1
£169,950 St. Michaels Hill, Flat, Bristol, BS2
£175,000 Rowanham Mead, Bristol, BS8

Load more...
Results for bs1, showing 20 of 31 properties
UI Capabilities: Runtimes and Source Code Translators

- Sometimes, platform specific code is needed for the UI (e.g. Xamarin)
- Often access to native UI *components* (e.g. Xamarin, native javascript frameworks)
- Some provide advanced graphical support (e.g. Unity, Qt)
  - Game Enigines
  - 2D and 3D acceleration
  - …
Example Unity
Device feature access & platform specific code

- **No platform specific code required**
  - Plugins for "all" device features
    - PhoneGap, Titanium, Natives

- **Xamarin.Mobile mobile has some cross-platform device features**
  - "Learn once, write everywhere"
  - All Native APIs are accessible

- **Platform Specific code for everything**
  - Regular Xamarin
  - Native Apps
Performance

• See presentation: *Comparing performance parameters of mobile app development strategies*
Performance

Cross-platform tools of the same category show similar behavior

The performance penalty resulting from the use of cross-platform tools is generally acceptable

Page rendering: JavaScript frameworks vs Runtimes, speed vs Native UI components

The performance of a cross-platform application strongly depends on the targeted platform
Cross-platform tools of the same category show similar behavior.
The performance penalty resulting from the use of cross-platform tools is generally acceptable.
Page rendering: JavaScript frameworks vs Runtimes, speed vs Native UI components

### JavaScript Frameworks
- Webview renders HTML pages
- Some JavaScript frameworks have faster response times than native apps
- Sometimes native skins
- No real, native UI components

### Runtimes
- Creates UI View elements
- Makes use of Native UI components
- Additional overhead introduced
The performance of a cross-platform application strongly depends on the targeted platform.

**Xamarin: Same tool, different strategy**

- Source translated to Intermediate Language (IL)
- Just-In-Time (JIT) compilation
- Source translated to executable binary code
- Ahead-Of-Time (AOT) compilation
Security concerns in cross-platform apps

- **Possible introduced software vulnerabilities by**
  - Runtimes
  - Translation of code
  - Mapping of code on native APIs
  - Extra software layers

- **Significant part of the code base in the application becomes third party**
  - Developer has little to no control over this.
Case Study: PhoneGap
Plugins and Security
Hot code update plugins

- Allow PhoneGap applications to be updated without the app store

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Version</th>
<th>Author</th>
<th>License</th>
<th>Downloads Last Month</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>cordova-hot-code-push-plugin</td>
<td>v1.2.5</td>
<td>nikdem</td>
<td>MIT</td>
<td>509</td>
<td>47 days ago</td>
</tr>
<tr>
<td>cordova-plugin-code-push</td>
<td>v1.5.1-beta</td>
<td>vsmobile</td>
<td>Licensed</td>
<td>393</td>
<td>19 days ago</td>
</tr>
<tr>
<td>meteor-cordova-update-plugin</td>
<td>v0.0.2</td>
<td>luisherranz</td>
<td>MIT</td>
<td>80</td>
<td>255 days ago</td>
</tr>
<tr>
<td>cordova-plugin-dynamic-update</td>
<td>v0.2.1</td>
<td>leecrosley</td>
<td>MIT</td>
<td>38</td>
<td>199 days ago</td>
</tr>
</tbody>
</table>
Hot code update plugins

<Location outside of the APK>
Meteor-cordova-update-plugin
Meteor Todo App
Man-In-The-Middle Attack

```
iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j DNAT --to 192.168.1.100:8080
```
Man-In-The-Middle Attack
Man-In-The-Middle Attack
<table>
<thead>
<tr>
<th>#</th>
<th>Host</th>
<th>Method</th>
<th>URL</th>
<th>Params</th>
<th>Edited</th>
<th>Status</th>
<th>Length</th>
<th>MIME type</th>
<th>Extension</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>POST</td>
<td>/sockjs/439/734kv08o/xhr_send</td>
<td>✔️</td>
<td></td>
<td>204</td>
<td>291</td>
<td>text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>POST</td>
<td>/sockjs/439/734kv08o/xhr</td>
<td></td>
<td></td>
<td>200</td>
<td>21045</td>
<td>JSON</td>
<td>json</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>GET</td>
<td>/_cordova/manifest.json</td>
<td></td>
<td></td>
<td>200</td>
<td>2036</td>
<td>JSON</td>
<td>json</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>POST</td>
<td>/sockjs/439/734kv08o/xhr</td>
<td></td>
<td></td>
<td>200</td>
<td>520</td>
<td>JSON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>POST</td>
<td>/sockjs/439/734kv08o/xhr_send</td>
<td>✔️</td>
<td></td>
<td>204</td>
<td>291</td>
<td>text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>GET</td>
<td>/_cordova/img/logo-todos.svg</td>
<td>✔️</td>
<td></td>
<td>200</td>
<td>4349</td>
<td>XML</td>
<td>svg</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>GET</td>
<td>/_cordova/icon/todos.svg?e2Mu...</td>
<td>✔️</td>
<td></td>
<td>200</td>
<td>17696</td>
<td>XML</td>
<td>svg</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>GET</td>
<td>/_cordova/3a332b1e4e39ee844b...</td>
<td>✔️</td>
<td></td>
<td>200</td>
<td>427520</td>
<td>script</td>
<td>js</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>GET</td>
<td>/_cordova/index.html?qfHyTzkJa...</td>
<td>✔️</td>
<td></td>
<td>200</td>
<td>2491</td>
<td>HTML</td>
<td>html</td>
<td>Todos – All your todo...</td>
</tr>
<tr>
<td>20</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>POST</td>
<td>/sockjs/439/734kv08o/xhr</td>
<td></td>
<td></td>
<td>200</td>
<td>964</td>
<td>JSON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>POST</td>
<td>/sockjs/439/734kv08o/xhr</td>
<td></td>
<td></td>
<td>200</td>
<td>339</td>
<td>JSON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><a href="http://todos.meteor.com">http://todos.meteor.com</a></td>
<td>GET</td>
<td>/sockjs/info?cb=5mbk88fiul</td>
<td>✔️</td>
<td></td>
<td>200</td>
<td>387</td>
<td>JSON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td><a href="http://fonts.gstatic.com">http://fonts.gstatic.com</a></td>
<td>GET</td>
<td>/s/opensans/v13/cjZKeOu8rm4kE...</td>
<td>✔️</td>
<td></td>
<td>200</td>
<td>10752</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Actual content, loaded into the application

Replaced the HTML content of the response
Result
Okay, but what is the point?
Actual content, loaded into the application
Result

App launches as always, nothing special?
Pictures from the device
How to protect against this?
The impact of bugs, patches and updates
Agenda

• Hybrid Approaches
• Runtime Based CPT Candidates
• Philosophy of Titanium / React Native / NativeScript
• Why choose what when? And how to use?
  o Architecture
  o Developer Experience
  o Future goals
• Conclusions
Hybrid Approaches

**WebView Based**

- WebView Container
  - Web Code
    - HTML
    - CSS
    - JS
  - Device APIs

**Runtime Based**

- JavaScript Runtime
  - JavaScript
  - Native UI + Device APIs
Hybrid Approaches
WebView Based

Web Code

UIImage frameworks & libs

DOM - based  |  WebGL - based

ionic

React

Famo.us

Architectural frameworks

Combined frameworks

Pure Hybrid Apps

Tools

PhoneGap

Properties

- Single WebView
- Content & navigation in HTML5
- Thin native wrapper

Mixed hybrid Apps

Pattern

Tools

Properties

Blended

Appcelerator

- Multiple WebViews
- Native navigation
- E.g.: Apple Store

Mullet

- WebViews for later stages of user flow
- E.g.: Walmart

Fallback

- WebViews for little used or frequently changing content
- E.g.: Instagram
Hybrid Approaches
Runtime Based

WebView Based

WebView Container

Web Code

HTML
CSS
JS

Device APIs

Runtime Based

JavaScript Runtime

JavaScript

Native UI
+ Device APIs
Hybrid Approaches
Runtime Based – The “WHY”…

- Consistent with platform
- Fast and responsive
- Complex gestures and smooth animations

---

- No knowledge & code sharing
- Different technology stacks
- Slow iteration speeds*
- Hard to scale

*N: Android Instant Run
Hybrid Approaches
Runtime Based – The “WHY”…

- HTML / CSS / JavaScript
- Same code and technologies
- Frameworks provide scaling
- F5 / ⌘+R

- Very hard to provide smooth experiences
- Not designed for complex interactions
- Feel out of place with the platform
- WebView fragmentation
- Performance
Hybrid Approaches
Runtime Based – The Solution??

Web advantages
- JSX / XML / CSS / JavaScript
- Same code and technologies
- Frameworks provide scaling
- Live-reload

Native advantages
- Consistent with platform
- Fast and responsive UI
- Complex gestures and smooth 60fps animations
Hybrid Approaches
Runtime Based – The “HOW”

Composition
- Native side
- JavaScript side
- JS-to-native bridge

Key concepts
- Proxy objects
- Asynchronous calls
## Runtime Based CPT Candidates

<table>
<thead>
<tr>
<th></th>
<th>Titanium</th>
<th>React Native</th>
<th>Nativescript</th>
<th>Fusetools</th>
<th>TabrisJS</th>
<th>Smartface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td>V5.2.2</td>
<td>V0.25.1</td>
<td>V2.0</td>
<td>V0.12.4</td>
<td>V1.7</td>
<td>v4.5.0</td>
</tr>
<tr>
<td><strong>Platforms</strong></td>
<td>Android 4.0.x – 6.0.x iOS 7.1.x – 9.2.x WP8.1-UWP</td>
<td>Android 4.1.x – 6.0.x iOS 7.0.x – 9.2.x UWP (alpha)</td>
<td>Android 4.2.x – 6.0.x iOS 7.1.x – 9.2.x UWP (alpha)</td>
<td>Android 4.2.x – 6.0.x iOS 7.1.x – 9.2.x UWP (alpha)</td>
<td>Android 3.7.x - 5.x iOS 6.x - 8.x</td>
<td>Android 4.2.x – 6.0.x iOS 7.1.x – 9.2.x</td>
</tr>
<tr>
<td><strong>Popularity</strong></td>
<td>11658</td>
<td>556</td>
<td>2300</td>
<td>5871</td>
<td>634</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>23564</td>
<td>72340</td>
<td>2300</td>
<td>16028</td>
<td>252</td>
<td>1105</td>
</tr>
<tr>
<td></td>
<td>2063</td>
<td>30206</td>
<td>6393</td>
<td>/</td>
<td>384</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>2303</td>
<td>3439</td>
<td>259</td>
<td>1119</td>
<td>2</td>
<td>283</td>
</tr>
</tbody>
</table>

*Popularity numbers are checked on 04/2016*
Agenda

- Hybrid Approaches
- Runtime Based CPT Candidates
- Philosophy of Titanium / React Native / NativeScript
- Why choose what when? And how to use?
  - Architecture
  - Developer Experience
  - Future goals
- Conclusions
Philosophy of Titanium/React Native/NativeScript

“Write once, adapt everywhere”

“Learn once, write anywhere”

“Write once, run anywhere”

UI ≠ shared
Logic = shared

Up to 70,80,90% Code share

Up to 85% Code share
Facebook’s Ads manager

UI = shared
Logic = shared

Possibility of writing platform specific UI
Why choose what when? And how to use?
Architecture

- Application structure
- Runtime Architecture
- Used technologies
- Modularity
Comparing Ti vs RN vs \{N\}
Architecture – Application Structure

<table>
<thead>
<tr>
<th>Design Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>• MVC (using Alloy)</td>
</tr>
</tbody>
</table>

| Additional concepts     |
Comparing Ti vs RN vs {N}

Architecture – MVC pattern

Model: uses BackboneJS models and collections

View: represent a page in your application

Controller: each view can have a controller
Comparing Ti vs RN vs {N}

Architecture – Application Structure

1. Definitive application structure
2. Platform separation on folder level

```
/app/
|-- assets/ (images, etc.)
|   |-- controllers/ (controllers for the views)
|     |   |-- android/
|     |     |   |-- index.js
|     |     |   |   |-- index.js
|     |     |-- models/ (models)
|     |     |-- views/ (views)
|     |-- ios/
|     |   |-- index.xml
|-- index.xml
|-- styles/ (view styling)
|-- themes/ (customizes assets and styles)
|-- widgets/ (app-like directory structure for each widget)
|-- migrations/ (database migration)
|-- lib/ (JavaScript libraries)
|-- specs/ (same as lib/ but for development only)
|-- i18n/ (Language strings)
|-- platform/ (platform resources)
|   |-- alloy.jmk (build config)
|   |-- alloy.js (init file for pereconfig)
|   |-- config.json (project config)
```

Resources/
i18n/
Comparing Ti vs RN vs {N}

Architecture – Application Structure

<table>
<thead>
<tr>
<th>Design Pattern</th>
<th>Additional concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>• MVC (using Alloy)</td>
<td>• Flux → Redux (unidirectional data flow)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional concepts</td>
<td></td>
</tr>
<tr>
<td>• React Components</td>
<td></td>
</tr>
</tbody>
</table>
Comparing Ti vs RN vs {N} Architecture – React Components

One way data flow inside components

Components can either get immutable data (via props) or management their own state (via state)
Comparing Ti vs RN vs {N} Architecture – Flux pattern

**One way data flow inside the application**

*Action creators* are helper methods, collected into a library, that create an action from method parameters, assign it a *type* and provide it to the dispatcher.

Every action is sent to all stores via the *callbacks* the stores register with the dispatcher.

After stores update themselves in response to an action, they emit a *change* event.

Special views called *controller-views*, listen for *change* events, retrieve the new data from the stores and provide the new data to the entire tree of their child views.

[https://facebook.github.io/flux/docs/overview.html](https://facebook.github.io/flux/docs/overview.html)
Comparing Ti vs RN vs \{N\}

Architecture – Application Structure

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>App/</td>
</tr>
<tr>
<td>2</td>
<td>- actions/ (flux actions)</td>
</tr>
<tr>
<td>3</td>
<td>- components/ (react components)</td>
</tr>
<tr>
<td>4</td>
<td>- BigButton.ios.js</td>
</tr>
<tr>
<td>5</td>
<td>- BigButton.android.js</td>
</tr>
<tr>
<td>6</td>
<td>- dispatchers/ (flux dispatchers)</td>
</tr>
<tr>
<td>7</td>
<td>- stores/ (flux stores)</td>
</tr>
<tr>
<td>8</td>
<td>- utils/ (utilities)</td>
</tr>
<tr>
<td>9</td>
<td>- constants/ (global constants)</td>
</tr>
<tr>
<td>10</td>
<td>- mixins/ (share common functionality inside components)</td>
</tr>
<tr>
<td>11</td>
<td>ios/ (ios resources, builds, config, etc...)</td>
</tr>
<tr>
<td>12</td>
<td>- ...</td>
</tr>
<tr>
<td>13</td>
<td>android/ (android resources, builds, config, etc...)</td>
</tr>
<tr>
<td>14</td>
<td>- ...</td>
</tr>
<tr>
<td>15</td>
<td>node_modules/ (external libraries)</td>
</tr>
<tr>
<td>16</td>
<td>index.ios.js (root component ios)</td>
</tr>
<tr>
<td>17</td>
<td>index.android.js (root component android)</td>
</tr>
<tr>
<td>18</td>
<td>package.json (project configuration)</td>
</tr>
</tbody>
</table>

1. User defined application structure (Scaffolding available)
2. Platform separation on file level
Comparing Ti vs RN vs {N}
Architecture – Application Structure

<table>
<thead>
<tr>
<th>Design Pattern</th>
<th>Additional concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>• MVC (using Alloy)</td>
<td>• React Components</td>
</tr>
<tr>
<td>• Flux → <strong>Redux</strong></td>
<td></td>
</tr>
<tr>
<td>(unidirectional data flow)</td>
<td></td>
</tr>
<tr>
<td>• MVC or <strong>MVVM</strong></td>
<td></td>
</tr>
</tbody>
</table>

**KU LEUVEN**
Comparing Ti vs RN vs {N} Architecture – MVVM pattern

- Two-way data binding
  - Observables

https://msdn.microsoft.com
Comparing Ti vs RN vs {N} Architecture – Application Structure

1. User defined application structure (Scaffolding available)
2. Platform separation on file level

```
Hello-World
- app/
  - App_Resources/  (Assets, manifests, plist, ...)
    - Android/
    - ios/
    - fonts/       (icon fonts)
    - shared/      (services, utility, ...)
    - ...
  - views/       (views)
    - main/
      - main-page.css  (view related styling)
      - main-page.js    (compiled from typescript)
      - main-page.ts    (code behind view)
      - main-page.xml   (page mark-up)
      - main-page.ios.xml (platform specific mark-up)
      - ...
    - widgets/      (custom widgets)
      - ...
    - app.css       (global styling, theming)
    - app.js        (starting point, app lifecycle)
    - package.json  (NativeScript configuration)
    - node_modules/ (linked libs)
    - platforms/    (platform specific files, build, ...)
    - package.json  _app's configuration_
```
### Comparing Ti vs RN vs \{N\} Architecture – Application Structure Rating

<table>
<thead>
<tr>
<th></th>
<th>MVC</th>
<th>Flux $\rightarrow$ Redux</th>
<th>MVC or MVVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry Level</strong></td>
<td><img src="emoji" alt="Smiley" /></td>
<td><img src="emoji" alt="Neutral" /></td>
<td><img src="emoji" alt="Smiley" /></td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td><img src="emoji" alt="Sad" /></td>
<td><img src="emoji" alt="Smiley" /></td>
<td><img src="emoji" alt="Neutral" /></td>
</tr>
<tr>
<td><strong>Testability</strong></td>
<td><img src="emoji" alt="Neutral" /></td>
<td><img src="emoji" alt="Smiley" /></td>
<td><img src="emoji" alt="Smiley" /></td>
</tr>
</tbody>
</table>

**Rating**

![Sad](emoji) ![Neutral](emoji) ![Sad](emoji) ![Neutral](emoji) ![Smiley](emoji) ![Smiley](emoji)
## Comparing Ti vs RN vs {N}

**Architectural – Used Technologies**

### Programming
- JavaScript (ES5)
- XML mark-up
- TSS styling (CSS-like styling)

### Layout system
- Relative to parent view
Comparing Ti vs RN vs {N}

Architecture – Used Technologies Examples

views/index.xml

```xml
<Alloy>
  <Window class="container">
    <Label id="label" onclick="doClick">
      Hello, World
    </Label>
  </Window>
</Alloy>
```

styles/index.tss

```json
"container": {
  backgroundColor: "white"
}

"Label": {
  width: Ti.UI.SIZE,
  height: Ti.UI.SIZE,
  color: "#000"
}
```

controllers/index.js

```javascript
function doClick(e) {
  alert($label.text);
}

$.index.open();
```

- Per view styling
- Global styling in “themes”
## Comparing Ti vs RN vs {N}

### Architecture – Used Technologies

<table>
<thead>
<tr>
<th>Programming</th>
<th>Layout system</th>
</tr>
</thead>
<tbody>
<tr>
<td>• JavaScript (ES5)</td>
<td>• Relative to parent view</td>
</tr>
<tr>
<td>• XML mark-up</td>
<td></td>
</tr>
<tr>
<td>• TSS styling (CSS-like styling)</td>
<td></td>
</tr>
<tr>
<td>• JavaScript (ES6+ES7 using Babel transpiler)</td>
<td></td>
</tr>
<tr>
<td>• JSX (XML like mark-up language)</td>
<td></td>
</tr>
<tr>
<td>• JavaScript “inline” styles</td>
<td></td>
</tr>
<tr>
<td>• Flow (static type checker for JavaScript)</td>
<td></td>
</tr>
<tr>
<td>• Flexbox</td>
<td></td>
</tr>
</tbody>
</table>
Comparing Ti vs RN vs {N}
Architecture – Used Technologies Examples

index.ios.js

```javascript
class HelloWorldApp extends React.Component {
    renderText: function() {
        return (
            <Text style={styles.baseText}>
                Hello World
            </Text>
        );
    },
}
```

```
var styles = React.StyleSheet.create({
  baseText: {
    color: 'black',
    backgroundColor: 'white',
  }
});
```

- Everything is a React component
- Mixing XML-like mark-up inside JavaScript
- Inline styles → referenced inside the same file
Comparing Ti vs RN vs {N}

Architecture – Used Technologies

<table>
<thead>
<tr>
<th>Programming</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• JavaScript (ES5)</td>
<td>• JavaScript (ES6+ES7 using Babel transpiler)</td>
<td>• JavaScript (ES5) or Typescript (ES6+ES7)</td>
</tr>
<tr>
<td>• XML mark-up</td>
<td>• JSX (XML like mark-up language)</td>
<td>• XML mark-up</td>
</tr>
<tr>
<td>• TSS styling (CSS-like styling)</td>
<td>• JavaScript “inline” styles</td>
<td>• CSS styling</td>
</tr>
<tr>
<td></td>
<td>• Flow (static type checker for JavaScript)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layout system</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Relative to parent view</td>
<td>• Flexbox</td>
<td>• Native layout sys.</td>
</tr>
</tbody>
</table>

- JavaScript: A language for writing code that runs on the client side of the web.
- XML mark-up: A way to structure documents that can be processed by a parser.
- TSS styling: Similar to CSS styling, but with JavaScript.
- JSX: A syntax extension to JavaScript that allows for writing HTML-like code.
- Flow: A static type checker for JavaScript.
- Native layout sys.: A layout system that is native to the platform it runs on.

KU LEUVEN
Comparing Ti vs RN vs {N} Architecture – Used Technologies Examples

View/home/home.xml

```xml
1. `<Page navigatingTo="onNavigatingTo">`
2. `  <StackLayout>`
3. `  | <Label text="Tap the button" class="title"/>`
4. `  | <Button text="TAP" tap="{{ onTap }}"/>`
5. `  | <Label text="{{ message }}" class="message"/>`
6. `</StackLayout>`
7. `</Page>`
```

View/home/home.js

```javascript
1. `var viewModel = require("./main-view-model");`
2. `function onNavigatingTo(args) {
3. `  var page = args.object;
4. `  page.bindingContext = viewModel.createViewModel();
5. `}
6. `exports.onNavigatingTo = onNavigatingTo;`
```

View/home/home.css

```css
1. `.title {
2. `  font-size: 30;
3. `  horizontal-align: center;
4. `  margin: 20;
5. `}
6. `button {
7. `  font-size: 42;
8. `  horizontal-align: center;
9. `  color: red
10. `}
```

- Per view styling
- Global styling in “app.css”
Comparing Ti vs RN vs {N}
Architecture – Used Technologies Rating

<table>
<thead>
<tr>
<th></th>
<th>ES5, XML, TSS</th>
<th>ES6/7, JSX, JS-styles</th>
<th>TypeScript, XML, CSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web compliance</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Strong typed entry level</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Layout</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
</tbody>
</table>

*Rating*
## Comparing Ti vs RN vs {N} Architecture – Runtime Architecture

<table>
<thead>
<tr>
<th>Properties</th>
<th>2 Threads</th>
<th>3 Threads</th>
<th>1 Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 2 Threads</td>
<td>• Main UI thread</td>
<td>• Shadow thread</td>
<td>• Main UI thread</td>
</tr>
<tr>
<td>• Main UI thread</td>
<td>• JS thread</td>
<td>• Main UI thread</td>
<td>• Asynchronous</td>
</tr>
<tr>
<td>• Asynchronous</td>
<td>• Serializable (iOS)</td>
<td>• JS thread</td>
<td>• Direct native API access</td>
</tr>
<tr>
<td>• Spawn native threads for work offloading</td>
<td>• Serializable</td>
<td>• Asynchronous</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Batched native calls</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Serializable</td>
<td></td>
</tr>
</tbody>
</table>

Note: every bridge has a type conversion mechanism
Comparing Ti vs RN vs {N}
Architecture – Runtime Architecture Rating

<table>
<thead>
<tr>
<th></th>
<th>Titanium Runtime</th>
<th>React Native Runtime</th>
<th>NativeScript Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge performance</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Size overhead</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Flexibility</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>Used runtime iOS, Android</td>
<td>🍍</td>
<td>🐘 Rhino</td>
<td>🍍</td>
</tr>
</tbody>
</table>

Rating 🙁 😞 😞 😞 😊 😊 😊
Comparing Ti vs RN vs {N} Architecture – Modularity

<table>
<thead>
<tr>
<th>Options</th>
<th>Titanium modules</th>
<th>React Native modules</th>
<th>NativeScript modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Titanium modules</td>
<td>• Built-in (100+)</td>
<td>• Build-in (60+)</td>
<td>• Build-in (54)</td>
</tr>
<tr>
<td>• giTrio community modules and widgets (1351)</td>
<td>• gitTrio community modules and widgets (1351) + (874)</td>
<td>• JS Coach (1000+)</td>
<td>• Community (198)</td>
</tr>
<tr>
<td>• Built-in (100+)</td>
<td>• Module integration</td>
<td>• Cordova plugin integration</td>
<td>• Cordova plugin integration</td>
</tr>
<tr>
<td>• NPM JS libraries for Titanium</td>
<td>• NPM JS libraries (no DOM reliance)</td>
<td>• CocoaPods JAR integration</td>
<td>• CocoaPods/ Java JAR integration</td>
</tr>
<tr>
<td>• NPM JS libraries through titaniumifier (1351)</td>
<td>• NPM JS libraries (no DOM reliance)</td>
<td>• NPM JS libraries (no DOM reliance)</td>
<td>• NPM JS libraries (no DOM reliance)</td>
</tr>
<tr>
<td>• NPM JS libraries through titaniumifier (1351)</td>
<td>• NPM JS libraries (no DOM reliance)</td>
<td>• NPM JS libraries (no DOM reliance)</td>
<td>• NPM JS libraries (no DOM reliance)</td>
</tr>
</tbody>
</table>

Numbers are checked on 04/2016
# Comparing Ti vs RN vs \{N\} Architecture – Modularity Rating

<table>
<thead>
<tr>
<th></th>
<th>Ti Modules</th>
<th>React Native components, etc…</th>
<th>NS modules, etc…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module utilization ease</td>
<td>🙁</td>
<td>🙂</td>
<td>🙂</td>
</tr>
<tr>
<td>Module Development complexity</td>
<td>🙁</td>
<td>😞</td>
<td>🍀</td>
</tr>
<tr>
<td>3rd party library integration</td>
<td>😞</td>
<td>🙂</td>
<td>🙂</td>
</tr>
</tbody>
</table>
Comparing Ti vs RN vs {N}
Architecture – Modularity

```javascript
var fileModule = require( "file-system" );
new fileModule.File( path );
```

At runtime executed on the JS engine

```javascript
new java.io.File( path );
```

```javascript
NSFileManager.defaultManager();
fileManager.createFileAtPathContentsAttributes( path );
```
Comparing Ti vs RN vs {N} Architecture – Modularity

NativeScript Modules (JavaScript)
- UI Abstraction
- File System Abstraction
- Device sensors
- Local Storage
- Push Notifications
- Launchers, Choosers
- Localization, Globalizations
- Other...

Runtime
- Native UI Components
- JavaScript Running on VM
- User Interaction Handled by Native platform
- Hardware Access Handled by Native platform

Native iOS application
Native Android application
Native WP application

How NativeScript Works - Telerik
Agenda

- Hybrid Approaches
- Runtime Based CPT Candidates
- Philosophy of Titanium / React Native / NativeScript
- Why choose what when? And how to use?
  - Architecture
  - **Developer Experience**
  - Future goals
- Conclusions
Developer Experience

Available tools
Documentation
Support
Contribution ease
Comparing Ti vs RN vs \{N\}
Developer Experience – Tools

<table>
<thead>
<tr>
<th>IDE</th>
<th>Build/Deploy/Debug</th>
<th>OverTheAir JS updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Appcelerator Studio</td>
<td>• Any IDE*</td>
<td>• Possibility</td>
</tr>
<tr>
<td>(based on Eclipse), (paid)</td>
<td>• Nuclide Atom (free)</td>
<td>• Siphon, Codepush,..</td>
</tr>
<tr>
<td></td>
<td>• Deco IDE (paid?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Visual studio code extension (free)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Any IDE</td>
<td>• Possibility</td>
</tr>
<tr>
<td></td>
<td>• Visual studio code extension (free)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Telerik Platform (paid)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CLI tool</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• LiveView</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Unit test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Etc..</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CLI tool</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Live reload</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Performance debug</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Etc..</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• CLI tool</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Live sync</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Node-inspector debug</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Etc..</td>
<td></td>
</tr>
</tbody>
</table>
Comparing Ti vs RN vs {N}
Developer Experience – Hot-reload IDE

Deco IDE
## Comparing Ti vs RN vs {N}

### Developer Experience – Community

<table>
<thead>
<tr>
<th>Support</th>
<th>Official Docs</th>
<th>Reactiflux (channel)</th>
<th>Official Docs</th>
<th>Slack channel</th>
<th>Product Pains</th>
<th>Github issues</th>
<th>Product Pains</th>
<th>Github (51)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Official Docs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Slack channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• JIRA Tickets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Stack Overflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Developer Blogs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Paid support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Official Blog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Twitter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Github (160)</th>
<th>Product Pains</th>
<th>Github (704)</th>
<th>Product Pains</th>
<th>NativeScript Ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Github (160)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• JIRA Tickets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Numbers are checked on 04/2016**
Comparing Ti vs RN vs \{N\}
Developer Experience – Tools/Support Rating

<table>
<thead>
<tr>
<th></th>
<th>Ti</th>
<th>RN</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Tools</td>
<td>🍀</td>
<td>🍀</td>
<td>🍀</td>
</tr>
<tr>
<td>Live Cycle Support</td>
<td>🍀</td>
<td>🌧️</td>
<td>🍀</td>
</tr>
<tr>
<td>Community</td>
<td>🌧️</td>
<td>🍀</td>
<td>🍀</td>
</tr>
<tr>
<td>Support</td>
<td>🍀</td>
<td>🍀</td>
<td>🍀</td>
</tr>
<tr>
<td>Maturity</td>
<td>🍀</td>
<td>🌧️</td>
<td>🍀</td>
</tr>
</tbody>
</table>

*Rating*
Future Goals

"The Web is for audience reach and native apps are for rich experiences. Both are strategic. Both are valuable. So when it comes to mobile, it’s not Web vs. Native. It’s both.”
Comparing Ti vs React Native vs {N} Future Goals – The Horizontal Platform

React

React Native

iOS  Android  …  Web

Desktop, UWP
Comparing Ti vs RN vs {N}
Future Goals – NativeScript + Angular 2

Building native mobile apps with Angular 2.0 and NativeScript - Sebastian Witalec
Comparing Ti vs RN vs \{N\} 
Future Goals – NativeScript + Angular 2

```typescript
@Component({
  selector: "checkbox",
  templateUrl: "checkbox.html"
});

<switch>

<input type="checkbox">
```
Conclusion

React Native brings React further than the browser

Titanium Appcelerator is still a viable and stable runtime based solution

NativeScript is easiest to start with and has 0-day support for new features
End of part 2
Discussion
Appendix
Comparing Ti vs RN vs `{N}`
Developer Experience – Over-the-air Updates

3.3.2 An Application may not download or install executable code. Interpreted code may only be used in an Application if all scripts, code and interpreters are packaged in the Application and not downloaded. The only exception to the foregoing is scripts and code downloaded and run by Apple’s built-in WebKit framework, provided that such scripts and code do not change the primary purpose of the Application by providing features or functionality that are inconsistent with the intended and advertised purpose of the Application as submitted to the App Store.
Comparing **Ti** vs **RN** vs **{N}** Architecture – Runtime Architecture

Various Proxies:
- Proxy
- Module
- ViewProxy
- View
Comparing Ti vs RN vs {N}
Architecture – Runtime Architecture

Native

1. Event (touch, timer, networks, etc.)
2. Collect data and notify JS
3. Serialized payload
4. Process event
5. Call 0 - ∞ Native methods
6. Serialized response
7. Process commands
8. Update UI (if needed)

React Native Bridge

JavaScriptCore

Under the hood of React Native - Martin Konicek
Comparing Ti vs RN vs {N}
Architecture – Runtime Architecture Ex.

UI Event Queue

1 Touch Event

RN Worker Event Queue(s)

3 Dispatch View Updates

Image decode, Disk I/O, Layout, Etc.

JS Event Queue

2 Handle Event → bridge → Runs JS

Under the hood of React Native - Martin Konicek
Appendix

Developer Experience – Tools LiveSync

LiveSync both ios and Android
Comparing Ti vs RN vs {N} Architecture – Runtime Architecture

Native

Native APIs

Call Dispatcher

Type-conversion marshalling service

Meta-data Bindings

App Code (JS ES5 & Typescript)

Calls JavaScript functions

Calls native API using JavaScript

NativeScript runtime

JavaScript VM

How NativeScript Works - Telerik
Comparing Ti vs RN vs {N} Architecture – Runtime Architecture Ex.

V8 JavaScript Engine

var file = new java.io.File(path);

File = FileProxy

Native Android

java.lang.String

java.io.File()

java.io.File()