NativeScript

A New & Better Way to Build Cross-Platform Native Apps

John Bristowe
Principal Developer Advocate
Telerik

April 21, 2015

nativescript.org

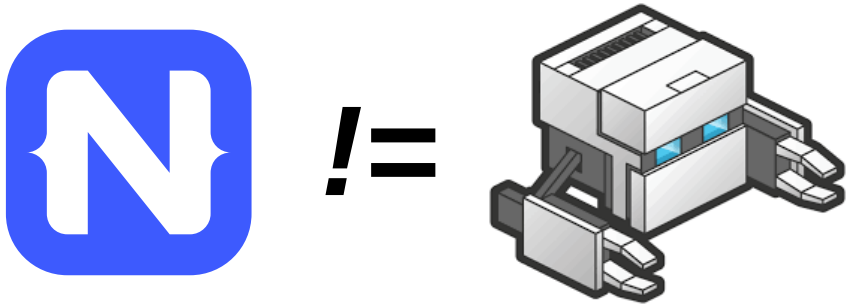
NativeScript Timeline and Roadmap

- 0.9
  - Public Beta
  - March 5th, 2015
- 1.0
  - Go-live license
  - Windows Phone support
  - May 2015

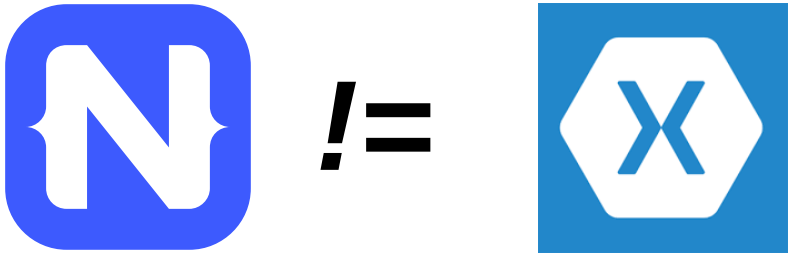
What is NativeScript?

A runtime for building and running **native** iOS, Android, and Windows Phone apps with a single, JavaScript code base

PROGRESS SPARK



No DOM



No cross-compilation

PROGRESS SPARK

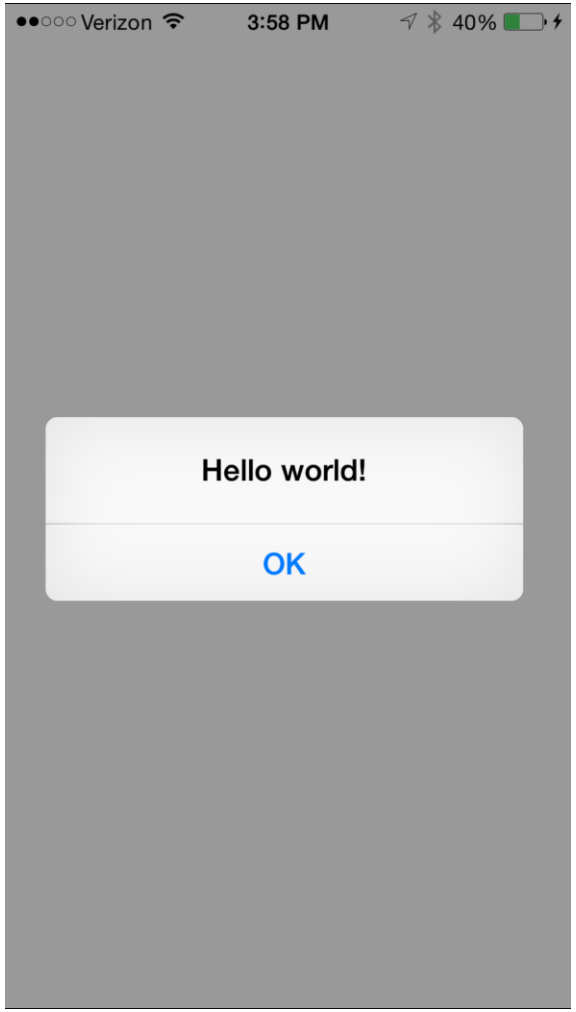
NativeScript Android Example

```
var time = new android.text.format.Time();
time.set(1, 0, 2015);
console.log(time.format("%D"));
```

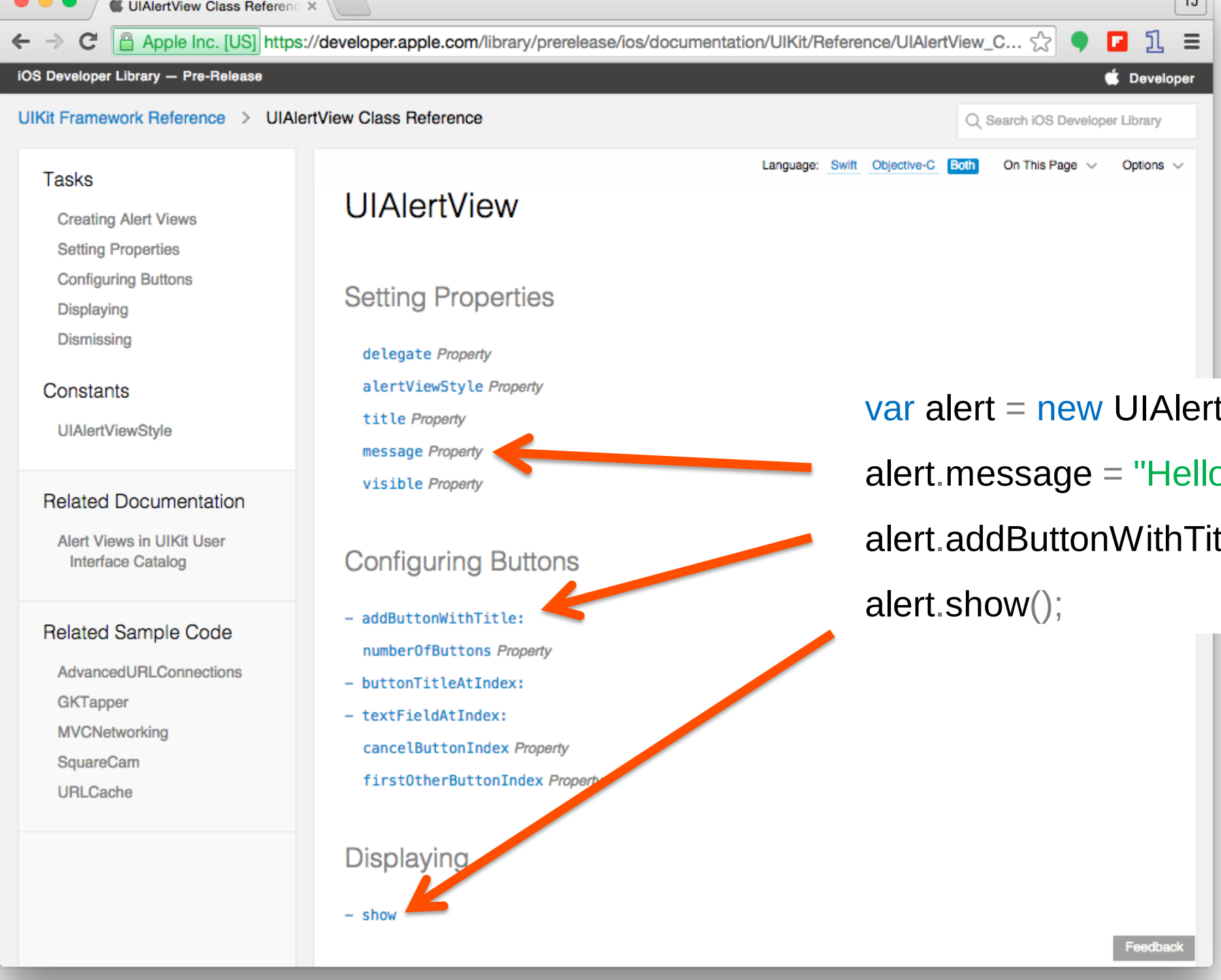
"01/01/15"

NativeScript iOS Example

```
var alert = new UIAlertView();
alert.message = "Hello world!";
alert.addButtonWithTitle("OK");
alert.show();
```



PROGRESS SPARK




```
var alert = new UIAlertView();
alert.message = "Hello world!";
alert.addButtonWithTitle("OK");
alert.show();
```

APJ
PROGRESS SPARK

How does this work?

NativeScript and JavaScript VMs

- NativeScript runs JavaScript on a JavaScript VM
  - JavaScriptCore on iOS
  - V8 on Android
  - JavaScriptCore on Windows

```
var time = new android.text.format.Time();
time.set(1, 0, 2015);
console.log(time.format("%D"));
```

Runs on V8

```
var alert = new UIAlertView();
alert.message = "Hello world!";
alert.addButtonWithTitle("OK");
alert.show();
```

Runs on JavaScriptCore

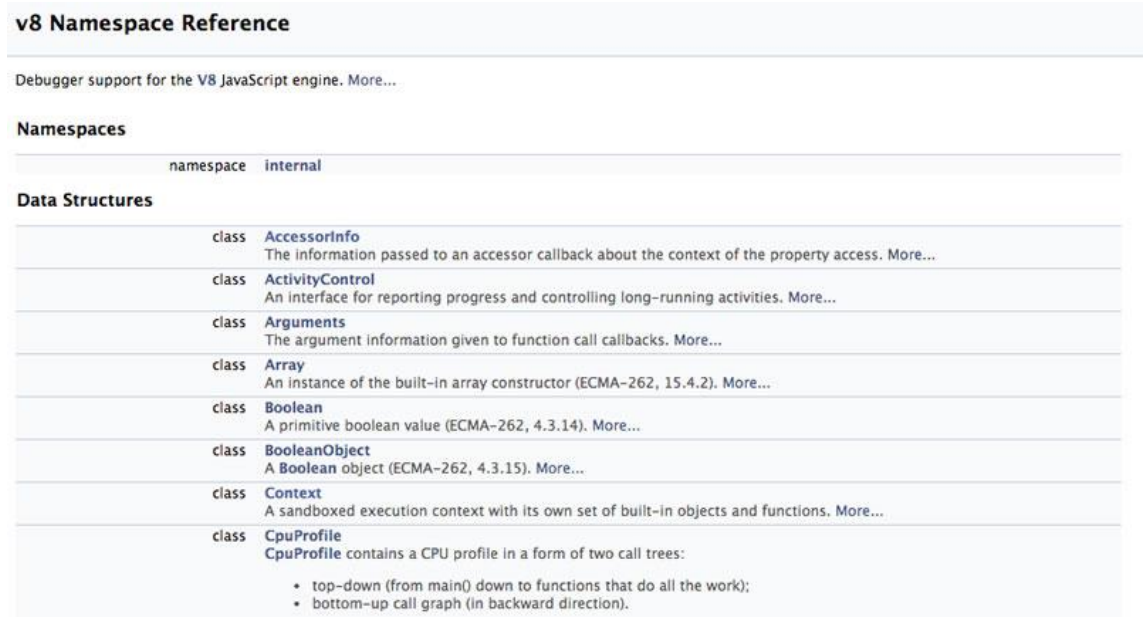
APJ
PROGRESS SPARK

Gathering Native APIs

- NativeScript uses **reflection** to build a list of available APIs for each platform
- For optimal performance, this metadata is pre-generated and injected into the app package at build-time

Injecting Native APIs

- V8/JavaScriptCore have APIs to inject global variables



v8 Namespace Reference

Debugger support for the V8 JavaScript engine. More...

Namespaces

namespace	internal

Data Structures

class	**AccessorInfo** The information passed to an accessor callback about the context of the property access. More...
class	**ActivityControl** An interface for reporting progress and controlling long-running activities. More...
class	**Arguments** The argument information given to function call callbacks. More...
class	**Array** An instance of the built-in array constructor (ECMA-262, 15.4.2). More...
class	**Boolean** A primitive boolean value (ECMA-262, 4.3.14). More...
class	**BooleanObject** A **Boolean** object (ECMA-262, 4.3.15). More...
class	**Context** A sandboxed execution context with its own set of built-in objects and functions. More...
class	**CpuProfile** **CpuProfile** contains a CPU profile in a form of two call trees: • top-down (from main() down to functions that do all the work); • bottom-up call graph (in backward direction).

APJ
PROGRESS SPARK

Invoking Native APIs

```
var time = new android.text.format.Time();
```

- V8/JavaScriptCore have C++ callbacks for JavaScript function calls and property accesses
- The NativeScript runtime uses those callbacks to translate JavaScript calls into native calls
- On iOS, you can directly call Objective-C APIs from C++ code
- On Android, NativeScript uses Android's JNI (Java Native Interface) to make the bridge from C++ to Java

Invoking Native APIs

```
var time = new android.text.format.Time();
```

1. The V8 function callback runs
2. The NativeScript runtime uses its metadata to know that Time() means it needs to instantiate an android.text.format.Time object
3. The NativeScript runtime uses the JNI to instantiate an android.text.format.Time object and keeps a reference to it
4. The NativeScript runtime returns a JavaScript object that proxies the Java Time object
5. Control returns to JavaScript where the proxy object gets stored as a local time variable

So, do you only write native code?

No.

TNS modules

- NativeScript-provided modules that provide cross-platform functionality
- There are dozens of them and they’re easy to write yourself
- TNS modules follow Node module’s conventions (CommonJS)

TNS File Module

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

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

```
NSFileManager.defaultManager();
fileManager.createFileAtPathContentsAttributes(path);
```

APJ PROGRESS SPARK

HTTP module example

```
var http = require("http");
http.getJSON("https://api.myservice.com")
    .then(function(result) {
      // result is JSON object
    });
```

Custom TNS modules

```
// device.ios.js
module.exports = {
    version: UIDevice.currentDevice().systemVersion
}

// device.android.js
module.exports = {
    version: android.os.Build.VERSION.RELEASE
}
```

PROGRESS SPARK

Using the custom device module

```
var device = require( "./device" );
console.log( device.version );
```

Community modules

- https://github.com/alejonext/NativeNumber
  - Someone created this 7 hours after the NativeScript public release.

“But how do I turn this into an app?”

Two Ways to Use NativeScript

1. Telerik Platform
2. NativeScript CLI

NativeScript CLI

- Free and open source
- https://github.com/nativescript/nativescript-cli

- System Requirements: https://github.com/nativescript/nativescript-cli#system-requirements
- Android: JDK, Apache Ant, Android SDK
- iOS: Xcode, Xcode CLI tools, iOS SDK

Starting a New Project with the NativeScript CLI

```
$ npm install -g nativescript
```

```
$tns create hello-world
$cd hello-world
```

Running on iOS

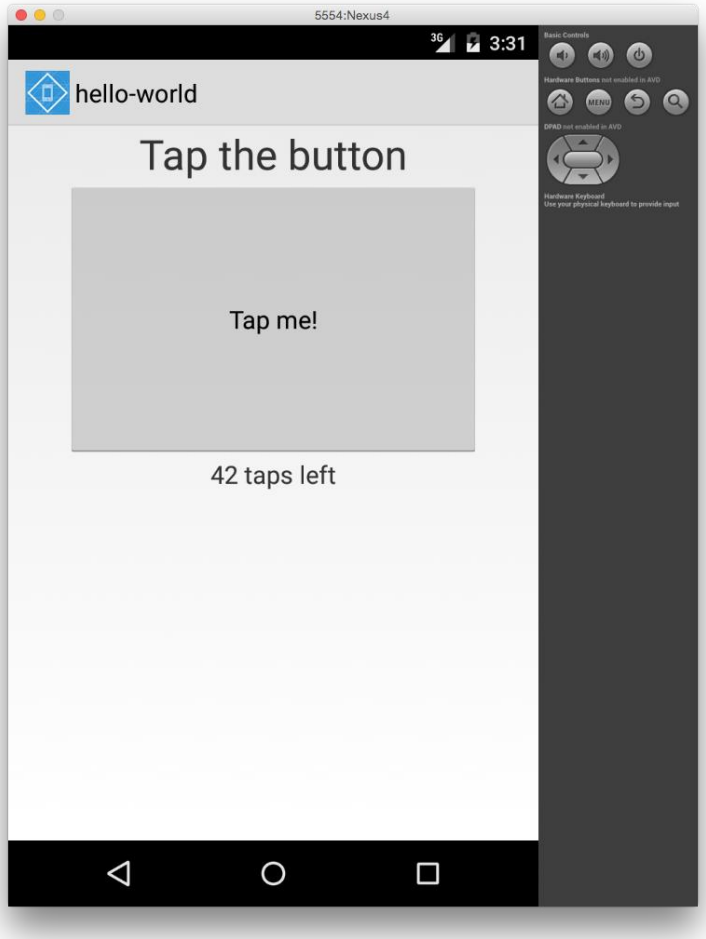
$ tns platform add ios

$ tns run ios --emulator

Running on Android

$ tns platform add android

$ tns run android --emulator



PROGRESS SPARK

Directory Structure

```
.
└── name-of-app
    ├── app
    │   ├── app
    │   │   ├── app.css          <-- app styling
    │   │   ├── app.js           <-- app starting point
    │   │   ├── main-page.css
    │   │   ├── main-page.js
    │   │   ├── main-page.xml
    │   │   └── node_modules     <-- npm modules
    │   │       └── ...
    │   ├── App_Resources        <-- icons, splash screens, config files
    │   │   └── ...
    │   └── tns_modules          <-- NativeScript modules
    │       └── ...
    └── platforms
        ├── android
        └── ios
```

APJ PROGRESS SPARK

app.js

```
var application = require( "application" );
application.mainModule = "main-page";
application.start();
```

Pages

- XML markup structure
- Elements (e.g. `<Page>, <Label>`) are TNS modules

```
<Page>
    <Label text="hello world" />
</Page>
```

PROGRESS SPARK

Custom XML Components

Example: Code-Only Custom Component

This sample `main-page.xml` uses two custom components defined in separate declarations in the `xml-declaration` directory. The custom controls are wrapped horizontally.

XML

```
<Page
    xmlns:customControls="app/xml-declaration/mymodule"
    xmlns:customOtherControls="app/xml-declaration/mymodulewithxml">
  <WrapLayout>
    <customControls:MyControl />
    <customOtherControls:MyControl />
  </WrapLayout>
</Page>
```

http://docs.nativescript.org/ui-with-xml#custom-components

Data binding

```
<Page loaded="load">
    <Label text="{{ message }}" />
</Page>
```

```
exports.load = function( args ) {
    args.object.bindingContext = { message: "hello world" };
}
```

Data binding improved

```
var observableModule = require( "data/observable" );

exports.load = function( args ) {
    var data = new observableModule.Observable();
    data.set( "message" , "hello world" );
    args.object.bindingContext = data;
}
```

PROGRESS SPARK

CSS

```
Label {
    color: red;
    font-size: 20;
    width: 200;
    margin: 20;
}
```

http://docs.nativescript.org/styling#supported-properties

Supported Properties

This is the list of the properties that can be set in CSS or through the style property of each View:

CSS Property	JavaScript Property	Description
color	color	Sets a solid-color value to the matched view's foreground.
background-color	backgroundColor	Sets a solid-color value to the matched view's background.
font-size	fontSize	Sets the font size of the matched view (only supports

Contribute! (nativescript.org/contribute)

Contributing to NativeScript

Thank you for your interest in contributing to the NativeScript project!

Anyone wishing to contribute to the NativeScript project MUST read & sign the NativeScript Contribution License Agreement. The NativeScript team cannot accept pull requests from users who have not signed the CLA first.

NativeScript is a complex framework, involving cross-platform modules, a Command-Line Interface and platform-specific runtimes. Each of these follows a specific technology, therefore the contribution instructions are different for each.

Please, visit these repositories for detailed contribution guidelines:
Cross-Platform modules
Command-Line Interface
Android-Runtime
iOS-Runtime

PROGRESS SPARK

Follow NativeScript

- @nativescript
- https://nativescript.org/blog

PROGRESS SPARK

demo

A Lap Around NativeScript

Q&A

Your Feedback Matters

Best Tweets

2 Winners get a GoPro Hero 4 Camera worth USD 399 each!

#APJSPARK

Take 10 Surveys

2 Winners get a Microsoft Band worth USD 199 each!

Take 10 surveys and stand a chance in the lucky draw!

bit.do/apjspark

APJ PROGRESS SPARK


APJ
PROGRESS SPARK