Appcelerator Titanium

In Two Days!

About me

- My name is Craig Marvelley
- I work at Box UK
- I mainly develop in Objective-C, PHP and JavaScript
- I’m on Twitter: @craigmarvelley

What is Titanium?

- Framework for building cross-platform applications, created by Appcelerator
- Two main products: Titanium Mobile and Titanium Desktop
- The framework supports multiple languages: HTML, JavaScript, CSS, PHP, Ruby and Python

Titanium Mobile

- Able to compile iOS and Android apps from JavaScript
- Blackberry support in closed beta
- Provides a layer of abstraction for each platform
- Supports common features, and some platform-specific ones
- Modules add additional functionality

What is iOS?

- A mobile OS Software Development Kit (SDK) from Apple written in Objective-C
- Allows developers to create apps for the iPhone and iPad devices
- Developers must sign up to the Apple developer programme to produce apps
- Apps are reviewed by Apple before release to general public

What is Android?

- A mobile OS SDK from Google written in Java
- Comprised of ‘stock’ Android and plugins from third parties
- Works on multiple devices, from phones to tablets
- Open source, free to develop for
- No app review system

Titanium: The Good

- It’s really easy to put applications together
- The platforms share common paradigms which suit Titanium’s approach
- Appcelerator’s IDE, Titanium Studio, helps development immensely
- The community is quite strong, and is growing
- It’s open source, so anyone can contribute, and it’s free to use
- Highly extensible: modules allow any missing functionality to be added

Titanium: The Not-So-Good

- Bugs are quite common
- Official support is only given to paid members
- Unofficial support is hit and miss
- Extended documentation is currently lacking
- API documentation is often inaccurate or out of date

The Titanium Ecosystem

- The developer center (documentation, API)
- The JIRA issue tracker
- The Appcelerator Github page
- The Kitchen Sink app
- Titanium Studio
- ‘Forging Titanium’ developer tutorials

So what are we going to learn?

Training Topics

- Setup checklist
- Anatomy of a Titanium project
- Application metadata
- Getting to know the API
- Views and Windows
- Modular applications and components

Training Topics

- Displaying content with Labels, TextViews, ImageViews, WebViews and Buttons
- Positioning your Views
- TabGroups
- Navigation within your app
- Displaying data with TableViews

Training Topics

- Using the Internet: the HTTP Client
- Translating your app with Localisation
- Mapping: Using MapViews and Geolocation
- Putting it all together: Building a Yahoo Client
- Deploying to App Stores

Setup Checklist

- Android apps can be developed on any OS with the Android SDK installed
- iOS apps can only be developed on Apple machines running OS X with XCode and the iOS SDK installed
- iOS Developers must be a member of the Apple Developer Program to deploy applications
- Developers must be registered with Appcelerator to install and use Titanium Studio
- Ready? Let's learn Titanium!

Anatomy of a Titanium Project

- tiapp.xml
- app.js
- Resources folder
- manifest
- build folder

Application Metadata

- There are a few resources which are standard for each application:
- Default.png (splash screen)
- appicon.png (application icon)
- iTunesArtwork (iTunes icon, iOS only)
- Localised strings files

Getting to know the API

- Titanium is the main top level namespace
- Others include JSON and timer functions
- Titanium.App contains application information, and also the Properties object
- Titanium.Android contains Android-only functionality like intents
- Titanium.UI contains cross-platform user interface views

Getting to know the API

- There are a few API modules that you’ll find invaluable when developing using Titanium
- Log data with Titanium.API.*
- Get device metadata using Titanium.Platform.*
- Open URLs with Titanium.Platform.openURL()

Getting to know the API

- Titanium.UI.iOS contains Adview logic
- Titanium.UI.iPhone contains the UI components that are specific to iPhone / iPad
- Titanium.UI.iPad contains those specific to the iPad
- Titanium.UI.Android contains the same, but for Android

Other notable APIs

- Titanium.Geolocation
- Titanium.Database
- Titanium.Filesystem
- Titanium.Locale
- Titanium.Map
- Titanium.Media
- Titanium.XML

Views and Windows

- A UIView is a 2D region on the screen into which we can draw
- A UIWindow is UIView subclass which fills the screen. Only one window can be shown at a time
- All visual components inherit from UIView
- Views can be added to other views to build a view hierarchy

Try it out!

- Add view to a window and make it green
- Position the view so it is 10px from each side of the screen
- Add a red border to the view and give it rounded corners
- Resize the view so it is 200px by 200px and position it so it is 100px away from the left

Modular applications and components

- CommonJS is a JavaScript standard for modules
- Titanium has native support for CommonJS
- We can use it to package up our UI components into discrete elements
- We can also use it to make it easier to work with the Titanium API by creating shortcuts

Try it out!

- Implement a CommonJS module so the last view we created can be created from a function
- Create an instance of the module and use it to make a new instance of the view, and add it to the window

Displaying content

Labels

- Labels are used to display small amounts of static text
- If you’ve used labels in HTML they’re pretty much the same
- The text can be altered by changing the font, size, colour, alignment, etc.

Try it out!

- Add a label to a view and make it say ‘Hello World’
- Change the label’s font size to 20px and the colour to blue
- Make the label center aligned
- Give the label a grey background and rounded corners

TextAreas

- TextAreas are used to display text that's too big to fit into a label
- They are also similar to their HTML counterparts, but they do not resemble a form field
- They are editable by default, but this can be disabled if you just want to display text

TextAreas and Keyboards

- If a TextArea is editable, the keyboard will appear (if necessary) when the user taps the control
- The keyboard can be manipulated depending on the task the user's performing
- The Titanium.UI.KEYBOARD_* and Titanium.UI.RETURNKEY_* constants control how the keyboard looks

Try it out!

- Add a TextArea sized 200px by 300px to the view, fill it with Lorem Ipsum and make it uneditable
- Make the view editable again, and set up the keyboard so it captures telephone numbers
- Change the configuration so tapping the return key inserts a new line

TextFields

- TextFields are essentially single-line TextAreas, but they are only used to capture text
- They are useful for when the text input from the user is likely to me small in length
- TextFields share much of the same configuration as TextAreas, so they are easy to work with

Try it out!

- Create a TextField that is 10 pixels away from the left, right and top edges of the screen and is 25 pixels high
- Disable the TextField so it cannot receive user input
- Configure the TextField so it capitalises user input and has some placeholder text

ImageViews

- ImageViews are used to display images!
- The image can either be local (on the device) or remote (hosted on a web server)
- Those on iOS should use the *hires* property to display images that suit the retina display
- The *defaultImage* property can be used to display a placeholder while a remote image loads

Try it out!

- Add an ImageView to a view and use it to display a local image stored under the resources folder
- Change the ImageView so it uses the local image as a placeholder while loading a very large image from the internet
- Give the image a rounded border, 10 pixels in width

WebViews

- WebViews provide a way to display HTML content within your app
- They can work with both markup, and a URL to a remote resource
- They can also work with PDFs
- We can use them to output content that would be difficult to reproduce with individual components

Try it out!

- Use a webview to display an image, with a caption underneath it
- Try using the *url* parameter to display the Cyfle homepage

Buttons

- Buttons let the user interact with your app, and are usually used to initiate an action
- On iOS there are some system-defined button styles which you may wish to reuse: see *Titanium.UI.iPhone.SystemButtonStyle*
- Buttons can be placed into other components like *Toolbars* to group them together logically

Events

- In order to respond to a button press you will need to respond to an Event, which works the same way as it does in HTML
- Events are used in many places in Titanium, and it’s difficult to build an app that doesn’t use them
- Any Titanium object is capable of firing events, and there is also a system-wide event center

Try it out!

- Create a button which hides a label when it is clicked
- Create a toolbar containing three buttons which change the text in the label to something different when they are clicked

Organising Views

View positioning

- The basic way to position views is to use their x,y coordinates to place them absolutely on screen
- Views also feature two types of automatic positioning using the *layout* parameter:
- *Horizontal* positioning places views alongside each other
- *Vertical* positioning places them on top of each other

Try it out!

- Use absolute positioning to place some Labels and ImageViews on screen
- Use horizontal positioning to place them alongside each other
- Use vertical positioning to place them underneath each other

TabGroups

- TabGroups are interface components that allow you to have multiple Windows in your application
- Each Tab in the group is responsible for a single Window
- On iOS you are limited to 5 tabs before the platform starts organising them for you
- Android doesn't seem to have a limit!

Try it out!

- Create three windows with different background colours and use tabs to toggle between them
- If you're on iOS, try adding 6 tabs and see what happens. Then try setting the *allowsUserCustomisation* property!

Navigation

- The other way to organise many windows is to navigate between them
- On Android this is a simple as opening more windows, and using the device back button to dismiss them
- On iOS, a unique component called a NavigationGroup allows us to construct a window hierarchy
- Your apps need to allow for both!

Try it out!

- Implement Appcelerator’s recommended cross platform approach and use it to open two windows with different background colours and titles

TableViews

- TableViews are typically used to display lists of data
- They can also be utilised to arrange views, for example to present the user with a list of options
- They have been designed to handle thousands of rows of data without affecting performance

TableViewRows

- A TableViewRow object represents one row in a TableView
- Usually a row has a title, and can also be given an image and a ‘disclosure icon’
- When a TableViewRow is tapped by the user an event is fired which we can listen for, and act appropriately

Try it out!

- Create a TableView that lists the members of the Beatles
- When a row in the table is tapped, show an alert with the name of the tapped Beatle
- On iOS, change the style of the tableview to the grouped style

The Internet: The HTTP Client

- Titanium’s HTTP client lets us make HTTP requests against web resources
- We use *listeners* and *callbacks* to act when a response has been received
- The JSON object is often used to decode the data that we get back from the request

Try it out!

- Make a request to Twitter to get a list of tweets for a user http://api.twitter.com/1/statuses/user_timeline.json?screen_name=billgates
- Decode the response using the JSON object
- For each tweet in the response, add an entry to a tableview with the text from the tweet

Localisation

- Localisation allows us to translate our application into other languages
- Titanium stores translations of our strings and automatically chooses the right string for the device’s language
- Unfortunately not all languages (such as Welsh) are supported by iOS and Android, in which case we have to manage it ourselves

Try it out!

- Create an English strings file and a Spanish strings file and put some content into each
- Add a label to a view which displays some translated text using the L() function
- Try changing the device’s language to see localisation at work

MapViews

- MapViews can be used to embed maps into your application, and interact with them
- We can use Annotations to put markers on a map to identify places to the user
- Events can be used so we can respond when the user taps on the map or on an annotation
- Android devices need to obtain a key from Google before they can use mapping

Try it out!

- Add a MapView to a window and set its coordinates so it shows Cardiff
- Add an annotation to the map that indicates where the Millennium Stadium is located
- Enable user location tracking on the map

Geolocation

- Geolocation enables us to detect where the user is on Earth using the GPS unit in the phone
- Once geolocation is enabled, we can ask for location updates by attaching listeners to appropriate events
- This process can use a lot of battery power, so be careful how often you use it, and try not to leave it running unnecessarily

Try it out!

- Configure Geolocation so it updates us on the user’s location and prints it out to the console

Let’s build an app!

- We’ll bring together some of the things we’ve learned to build an application
- The app will present some data from Yahoo
- It will use many UI components, and the HTTP client, to asynchronously receive data