

# iPhone and Android apps how to create native apps with Appcelerator Titanium

Part 1 of a series



Jonathan Carter co-founded Glimworm IT BV in 2001 with partners Paul Manwaring, Marten Hoekstra and Colin Williams. At 15 his professional career began as a game programmer. After more than 10 years of innovative experience Jonathan was asked to join CMG's prestigious Advanced Technology Department consulting for blue chip companies. Eventually, he shifted his attention to new media and created Glimworm, bringing his talents to the world of web development. With nearly 30 years in the industry his list of accomplishments is huge including everything from project management to robot building and he continues to keep on top of new developments by constantly evaluating emerging technologies.

By Jonathan Carter co-founder and Chief Technical Officer of Glimworm IT BV in Amsterdam. In this white paper Jonathan explains how he created his first native app with Titanium, a cross platform development tool made by Appcelerator.

## INTRODUCTION

I am an experienced developer but for the last 6 months I have started using a MAC with XCODE to make some basic apps and, quite frankly, I find it pretty hard work. Not to say that programming should not be difficult, but making apps is rapidly becoming more important for my company so I would like to be able to get a bigger 'bang for my buck' (buck in this case being hours).



I have built a sort of wrapper for quick and easy deployment of HTML5 based apps built with sencha touch and jqtouch but they are not quite as good as native apps. It is for this native building that I want to maximize my efforts and, therefore, the idea of a cross browser platform is appealing so long as it really does produce native code. In the case of Titanium, I saw that after making my program I was able to export it as an XCODE project and then compile it, therefore, I do see Titanium as providing this 'real native' solution.

## CREATING THE APPLICATION

Firstly, I see that Titanium makes a native folder structure on your filesystem and from then on you are on your own. It creates a few files for you but you have to find an editor yourself and work from examples and the API Docs. From here I start with an app called test003m.

In the folder there is :

- tiapp.xml, which is apparently a static configuration file
- build(folder), which is where the output will go
- Resources (folder), where "resources" will go, I am not quite sure what "resources" are at the moment but I will soon find out.
- i18n (folder), where I can put language specific stuff, apparently

## APP.JS

In the resources folder is app.js which I suspect is quite important. it contains the following code to start with:

```
var tabGroup = Titanium.UI.createTabGroup();

var win1 = Titanium.UI.createWindow({
  title:'Tab 1',
  backgroundColor:'#fff'
});

var tab1 = Titanium.UI.createTab({
  icon:'KS_nav_views.png',
  title:'Tab 1', window:win1
});
```

Now this is very reminiscent of the structures you use in EXT.JS in order to set up the GUI. It is all JSON based syntax, so far it looks like it could be quite easy to get the hang of.

The following are the major design components in Titanium:

- Windows – windows host one or most Views
- Views – views draw content on the screen
- Widgets – widgets are special types of views that perform specific actions like buttons.

Here is an example from the Appcellerator website of code to make a view in a window

```
var win = Ti.UI.createWindow();
var view = Ti.UI.createView({backgroundColor:"red"});
win.add(view);
win.open();
```

We could restructure the above code to use a URL-based design. First, in your app.js, you would add:

```
var win = Ti.UI.createWindow({url:"view.js"});
win.open();
```

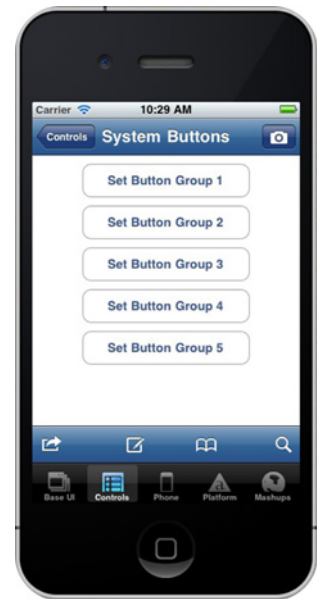
Then, create a file named view.js and add the following code:

```
var win = Ti.UI.currentWindow;
var view = Ti.UI.createView({backgroundColor:"red"});
win.add(view);
```

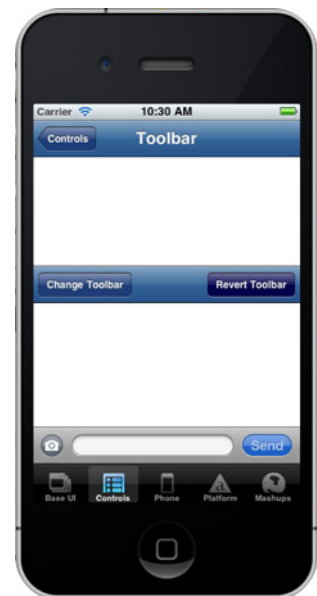
Notice that the win variable points to Ti.UI.currentWindow. Titanium defines a set of special variables in your JavaScript context automatically for you which allow you. Ti.UI.currentWindow defines the Window reference that owns (opened) the current Window so you can still reference it.

To add interactivity this should be the code according the the tutorial I am reading:

```
view.addEventListener('click',function() {
  view.animate({width:96,height:96,duration:1000});
});
```



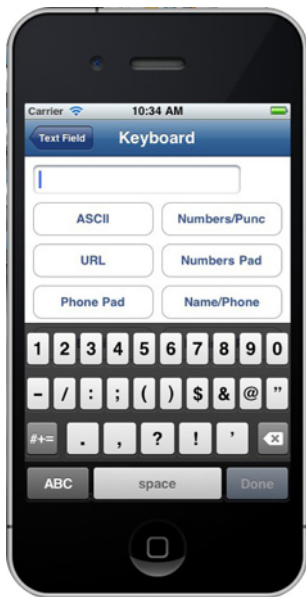
SYSTEM BUTTONS



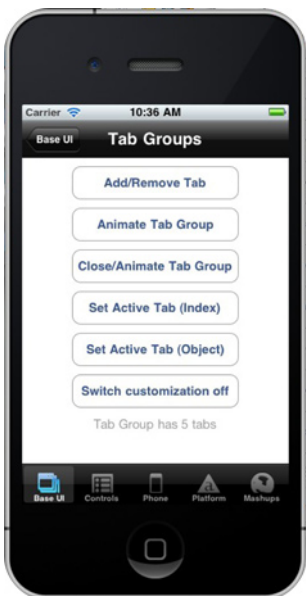
TOOLBAR

Here is a more complicated example:

```
view.addEventListener('click',function() {
    var t = Ti.UI.create2DMatrix();
    t = t.rotate(-90).scale(4);
    view.animate({transform:t,duration:1000});
});
```



KEYBOARD



TAB GROUPS

## WIDGETS

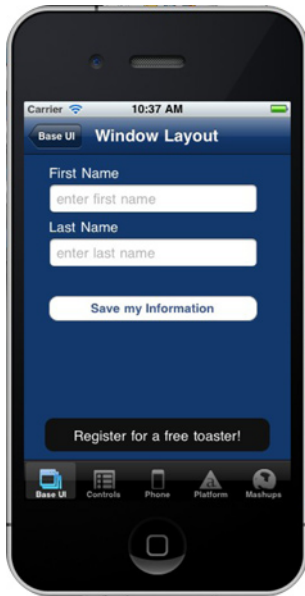
It seems to me that widgets are basically forms, i.e. views with controls on them which in turn cannot contain views. Here is another example from the website

```
var win = Ti.UI.createWindow();
var view = Ti.UI.createImageView({
    image:"myimage.png",
    width:24,
    height:24
});
var button = Ti.UI.createButton({
    title:"Animate",
    width:80,
    height:40,
    bottom:10
});
button.addEventListener("click",function(){
    view.animate({top:0,duration:500});
});
win.add(view);
win.add(button);
win.open();
```

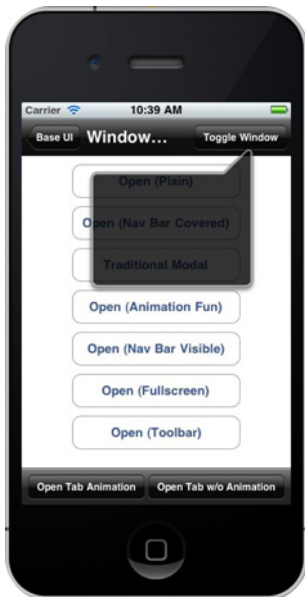
## MODAL WINDOWS

You can create modal windows at any time without adding them to another component. Here is an example of opening a modal window which is attached to a button by the use of an onclick handler

```
var b3 = Ti.UI.createButton({
    title : 'b3'
});
b3.addEventListener('click',function(e) {
    var mod1 = Titanium.UI.createWindow({
        backgroundColor:'red'
    });
    var mod1b3 = Ti.UI.createButton({
        title : 'b3'
    });
    mod1b3.addEventListener('click',function(e) {
        mod1.close();
    });
    mod1.add(mod1b3);
    mod1.open({
        modal:true,
        modalTransitionStyle: Ti.UI.iPhone.MODAL_
        TRANSITION_STYLE_COVER_VERTICAL,
        modalStyle: Ti.UI.iPhone.MODAL_PRESENTATION_
        FORMSHEET
    });
});
```



WINDOW LAYOUT



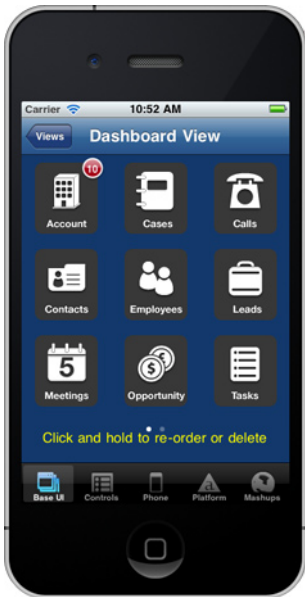
WINDOW BUTTON

## THE REFERENCE GUIDE

There is a comprehensive guide to the API but I found it difficult to determine which component was which by looking at the descriptions. I made myself a set of notes which I now share with you:

- Tab Group - this is the classic navigation with tabs below the display
- Table View - this is any sort of tabular data but NOT a navigation
- WebView - this is any sort of HTML in a window
- Options Dialog - these are the options which shoot up from the bottom of the screen
- EmailDialog - this is for sending emails
- MapView - the classic map view
- Coverflow View - this is the flipping inames
- Dashboard view - this is like the appie front screen with a series of icons
- Tab Badge - is the red number on the tabs
- WindowNavBar - this is at the top of the window and has the back buttons
- Window Toolbar - this is at the bottom of the view and I think is seldom used
- NavGroup - this is the left sliding list of items with a detail view or a 2nd level of menu to the right
- Controls
  - Slider
  - Switch (on off)
  - Activity indicator
  - Progress bar
  - Button
  - Label (any text)
  - Search Bar - this bar at the top of a view
  - Text field, text area
  - Button Bar - a group of buttons
  - System Buttons (as from the system)
  - Picker - this is a select box
- Photo Gallery - access to photos
- Orientation - access to orientation
- Contacts - access to address book
- Camera - camers
- Save to Gallery
- Shake
- Status Bar - the whole bar at the top of the page - can be hidden.
- App Badge - the red number in the app

To help with remembering which components and views to use I have made a sort of cheat sheet with screenshots from the Appcelerator Titanium kitchen sink application:



DASHBOARD VIEW

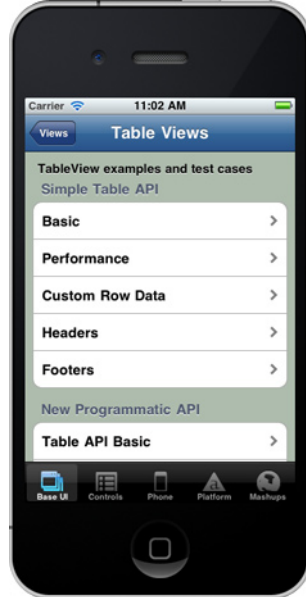
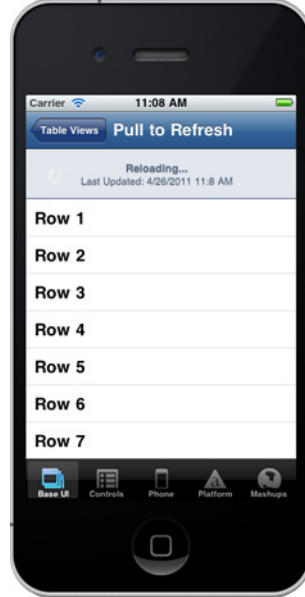


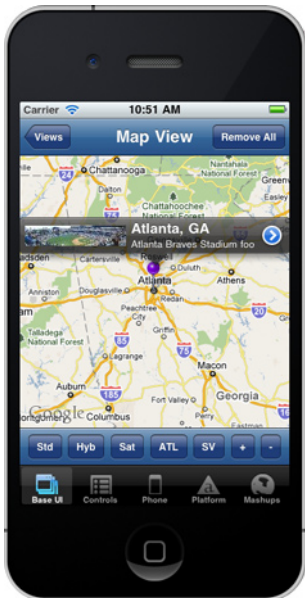
TABLE VIEWS



PULL TO REFRESH



TABLE SECTION HEADER



MAP VIEW

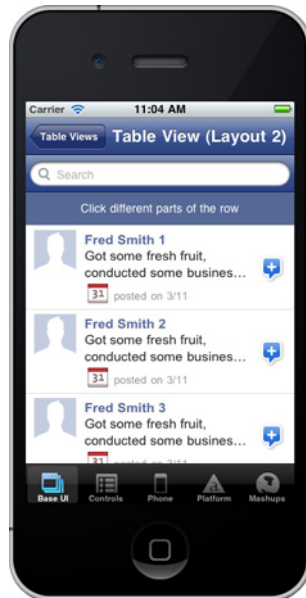
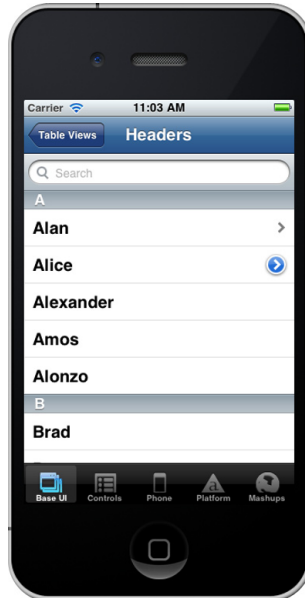
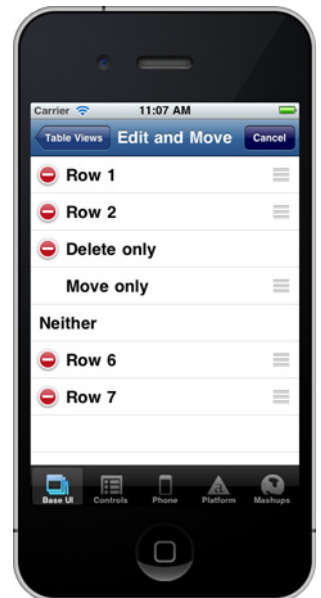


TABLE VIEW (LAYOUT 2)



HEADERS



EDIT AND MOVE

In part 2 I will be exploring the practicalities of implementing a real app with web communication.



Glimworm IT BV  
 Eerste Weteringplantsoen 8  
 1017 SK, Amsterdam  
 Nederland

Tel: +31 (0)20 - 616 56 40  
 Fax: +31 (0)20 - 240 13 57  
 E-mail: [jc@glimworm.com](mailto:jc@glimworm.com)

