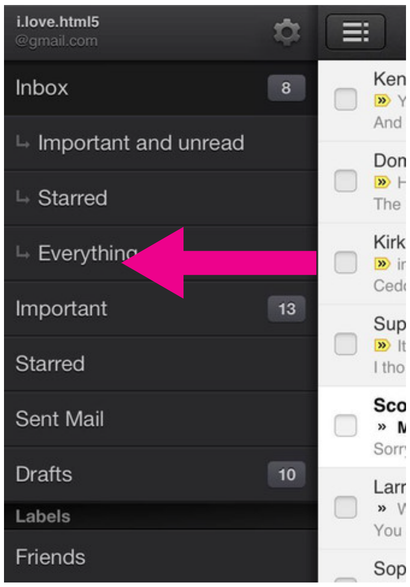
iOS Custom CrossCut Menu in Titanium Appcelerator

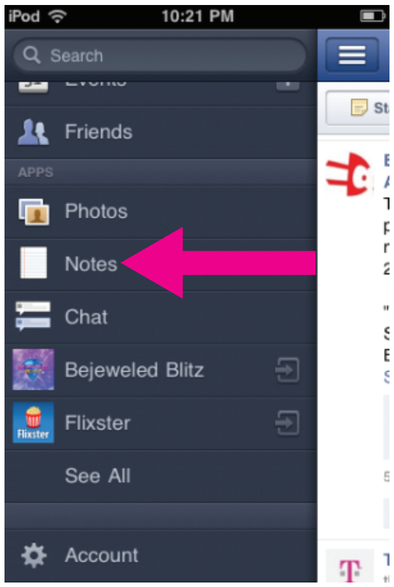
Step by Step: Tutorials in understanding customizing Crosscut Menu

Humber College - MultiMedia Design and Development
AVIS 322: Multimedia Production 2, Autumn Term 2012
Instructor: George Paravantes MFA IxD
Date:Nov 21, 2012

Crosscut Menu Gmail app

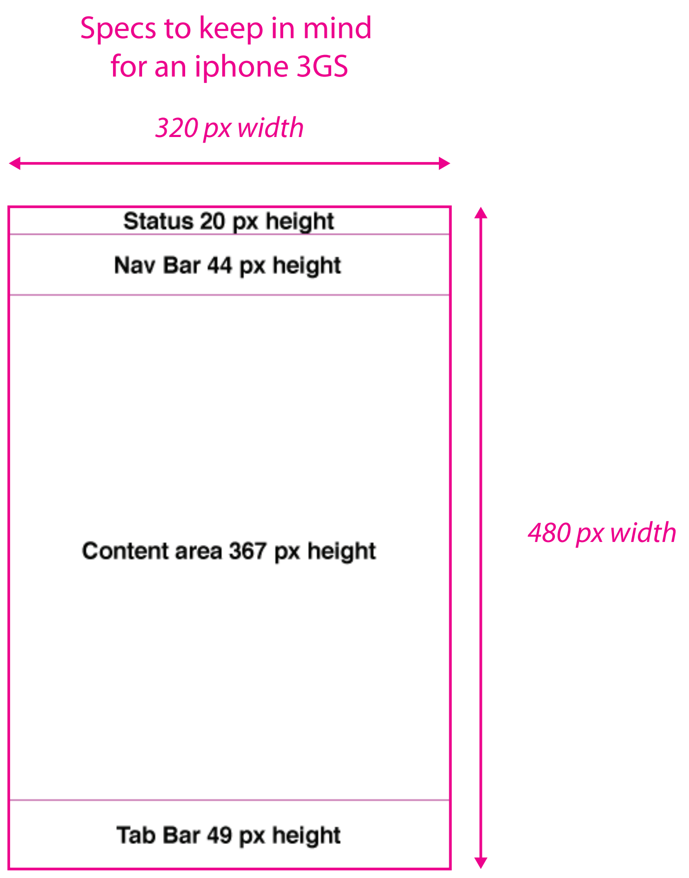


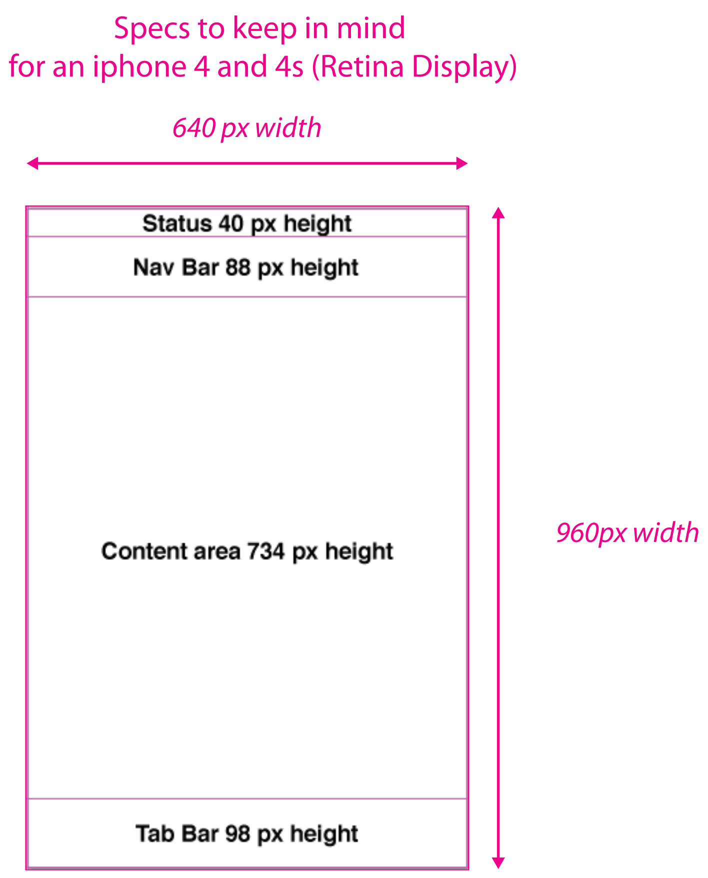
Crosscut Menu Facebook app



Intro to crosscut menu customization:
This week we will look at creating cross cut menus that has become increasingly more popular with mobile application. The facebook app and the gmail app are good examples of this cross cut menu. They are also know as top-level navigation systems.

Step 00: This illustration will help us keep in mind the default size Titanium Studio starts an iOS project is 480x320 px @ 72 dpi



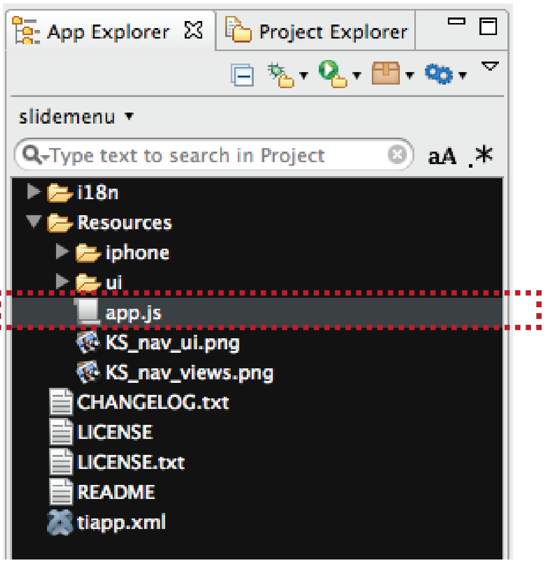


iOS Custom CrossCut Menu in Titanium Appcelerator

Step by Step: Tutorials in understanding customizing Crosscut Menu

Step 01: We will start with app.js file and start scratch.

Start with a blank app.js file



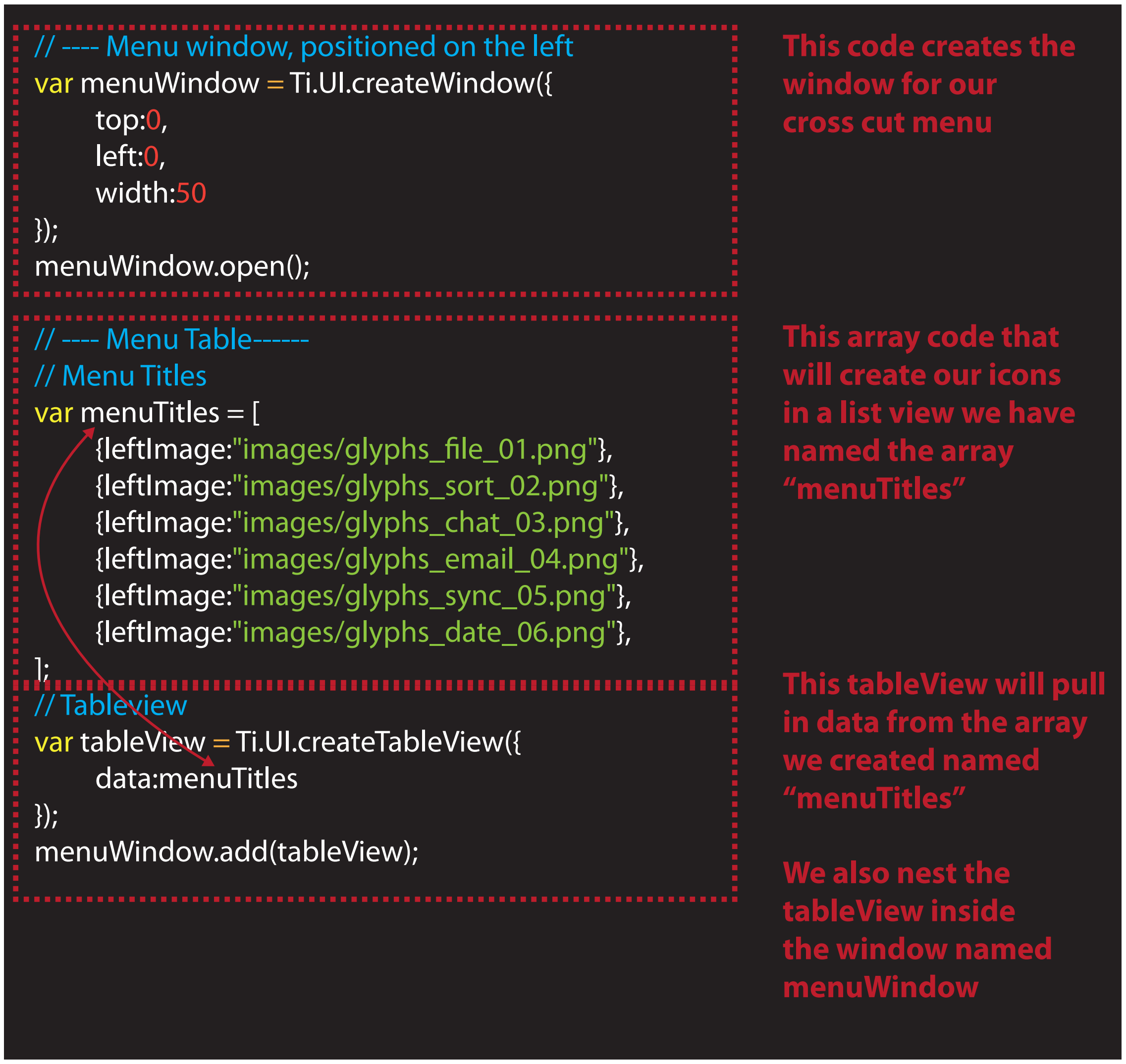
Step 02: We will start with app.js file and start scratch.

Here we add a window called menuWindow.

We create and array to declare out icon set in out menu

We also create a tableView to place our array data inside this view.

Finally we nest the tableView inside the window menuWindow.



```
// ---- Menu window, positioned on the left
var menuWindow = Ti.UI.createWindow({
        top:0,
        left:0,
        width:50
});
menuWindow.open();
```

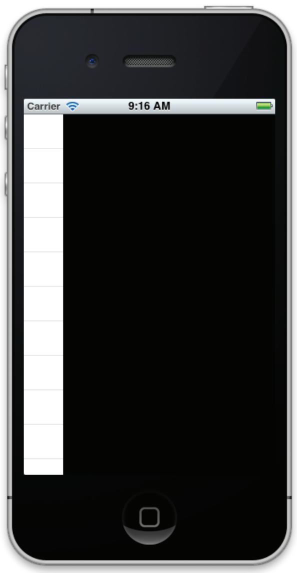
This code creates the window for our cross cut menu

```
// ---- Menu Table------
// Menu Titles
var menuTitles = [
        {leftImage:"images/glyphs_file_01.png"},
        {leftImage:"images/glyphs_sort_02.png"},
        {leftImage:"images/glyphs_chat_03.png"},
        {leftImage:"images/glyphs_email_04.png"},
        {leftImage:"images/glyphs_sync_05.png"},
        {leftImage:"images/glyphs_date_06.png"},
];
// Tableview
var tableView = Ti.UI.createTableView({
        data:menuTitles
});
menuWindow.add(tableView);
```

This array code that will create our icons in a list view we have named the array "menuTitles"

This tableView will pull in data from the array we created named "menuTitles"

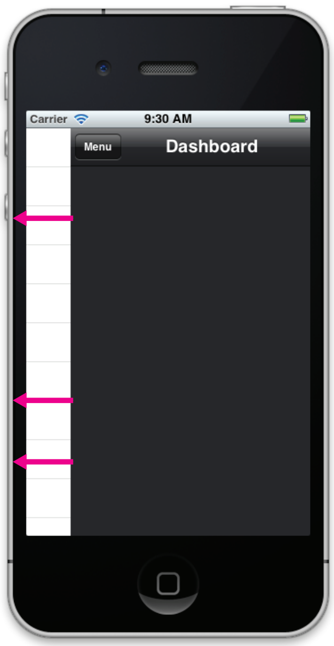
We also nest the tableView inside the window named menuWindow



iOS Custom CrossCut Menu in Titanium Appcelerator

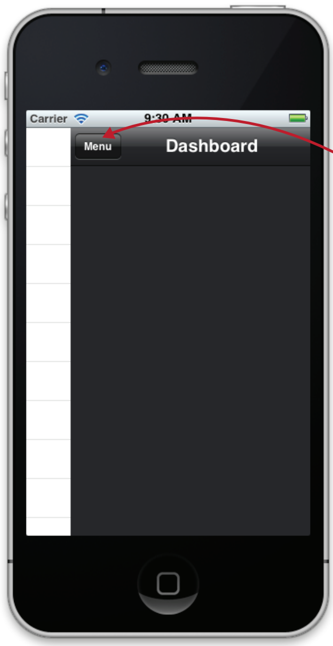
Step by Step: Tutorials in understanding customizing Crosscut Menu

Step 03:
Lets create more windows for our Navigation bar and our main window Dashboard.



Now we need a window with a navigation bar and a button in it that will allow us to open the menu with an animation.

So, in order to achieve this, we actually need two windows: one containing a navigationGroup (indispensable in order to have a navigation bar) and another one that is in the navigationGroup:



```
// ---- Window with navigationGroup
// Set the width of the sliding window to avoid cut out from animation
var navWindow = Ti.UI.createWindow({
        width:320
});
navWindow.open();
```

Here we create the code to declare a window for our navigation bar

```
// Main window
var win = Ti.UI.createWindow({
        title:'Dashboard',
        backgroundColor:'#28292c',
        backgroundImage:'/images/background.png',
        barColor:'#28292c',
        moving:false, // Custom property for movement
        axis:0 // Custom property for X axis
});
```

Here we create the code to declare our main window that is titled Dashboard.

```
// NavigationGroup
var navGroup = Ti.UI.iPhone.createNavigationGroup({
        window:win
});
navWindow.add(navGroup);
```

Next I call on the API createNavigationGroup and place it inside navWindow

```
// Top left button
var menuButton = Ti.UI.createButton({
        title:'Menu',
        toggle:false // Custom property for menu toggle
});
```

Lets create and place our menu button in our nav bar

```
win.setLeftNavButton(menuButton);
menuButton.addEventListener('click', function(e){
        // If the menu is opened
        if(e.source.toggle == true){
                navWindow.animate({
                        left:0,
                        duration:300,
                        curve:Ti.UI.ANIMATION_CURVE_EASE_IN_OUT
                });
                e.source.toggle = false;
        }
        // If the menu isn't opened
        else{
                navWindow.animate({
                        left:50,
                        duration:300,
                        curve:Ti.UI.ANIMATION_CURVE_EASE_IN_OUT
                });
                e.source.toggle  = true;
        }
});
```

Now we need to add an EventListener when button is pressed.

We have a if statement that says if the menu is open and the use clicks the menu button then ease out position to 0 over 300milliseconds.

But if the menu is closed and user presses the menu button then the window will slide over 50pixels over duration of 300 milliseconds.

The line of code that declares toggle:false property in our button, right? It doesn't really exist; it's a custom property that we created.

We can name name it however we want or even create a variable for it (like var toggle = true;) if it makes more sense. However, its suggested we use this ittle workaround because it is really useful when you have a lot of custom properties in your app.

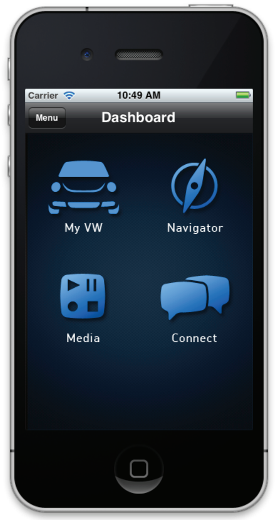
We will notice that when we click the button, we call function(e), where e is our object (the button). By calling e.source.toggle, we are checking the custom "toggle" property discussed to the left (you can also use menuButton.toggle, it's the same thing).

If it is false, we're moving our window to the right and switching the property to true. So, of course, if it's true, the window goes back to normal and our property is then set to false again.

iOS Custom CrossCut Menu in Titanium Appcelerator

Step by Step: Tutorials in understanding customizing Crosscut Menu

Step 04:
Next we add the imageViews to create our dashboard buttons



```
// A locally stored image in the folder named images
var imageView01 = Titanium.UI.createImageView({
        image:"images/dsh_icon_01.png",
        //notice that we created an 'images' folder
        height:145, //Use the dimensions of the image
        width:145,
        top:10, // push off 10px from the nav bar
        Left:10 //push off 10px from the left side
        });
        win.add(imageView01);
```

```
// A locally stored image in the folder named images
var imageView03 = Titanium.UI.createImageView({
        image:"images/dsh_icon_03.png",
        //notice that we created an 'images' folder
        height:145, //Use the dimensions of the image
        width:145,
        top:165, // push off 165px from the nav bar
        Left:10 //push off 10px from the left side
        });
        win.add(imageView03);
```

```
// A locally stored image in the folder named images
var imageView02 = Titanium.UI.createImageView({
        image:"images/dsh_icon_02.png",
        //notice that we created an 'images' folder
        height:145, //Use the dimensions of the image
        width:145,
        top:10, // push off 10px from the nav bar
        right:10 //push off 10px from the right side
        });
        win.add(imageView02);
```

```
// A locally stored image in the folder named images
var imageView04 = Titanium.UI.createImageView({
        image:"images/dsh_icon_04.png",
        //notice that we created an 'images' folder
        height:145, //Use the dimensions of the image
        width:145,
        top:165, // push off 165px from the nav bar
        right:10 //push off 10px from the right side
        });
        win.add(imageView04);
```

We have created four imageViews to place in the main dashboard window to plave our four icons in a 2x3 matrix.

iOS Custom CrossCut Menu in Titanium Appcelerator

Step by Step: Tutorials in understanding customizing Crosscut Menu

Step 05: Final step lets add touch slide properties to our main dashboard window

1

```
win.addEventListener('touchstart', function(e){
        // Get starting horizontal position
        e.source.axis = parseInt(e.x);
});
```

2

```
win.addEventListener('touchmove', function(e){
        // Subtracting current position to starting horizontal position
        var coordinates = parseInt(e.globalPoint.x) - e.source.axis;
        // Detecting movement after a 20px shift
        if(coordinates > 20 || coordinates < -20){
                e.source.moving = true;
        }
        // Locks the window so it doesn't move further than allowed
        if(e.source.moving == true && coordinates <= 50 && coordinates >= 0){
                // This will smooth the animation and make it less jumpy
                navWindow.animate({
                        left:coordinates,
                        duration:20
                });
                // Defining coordinates as the final left position
                navWindow.left = coordinates;
        }
});
```

3

```
win.addEventListener('touchend', function(e){
        // No longer moving the window
        e.source.moving = false;
        if(navWindow.left >= 25 && navWindow.left < 50){
        // Repositioning the window to the right
                navWindow.animate({
                        left:50,
                        duration:300
                });
                menuButton.toggle = true;
        }else{
                // Repositioning the window to the left
                navWindow.animate({
                        left:0,
                        duration:300
                });
                menuButton.toggle = false;
        }
});
```

1. Let's use the touchstart event to get the position of our finger when it touches the screen:
Here we take the horizontal coordinate (e.x) of our event, parse it as an integer, and then save it in our custom property axis.

2. Next we are going to animate the window depending on the position of our finger:

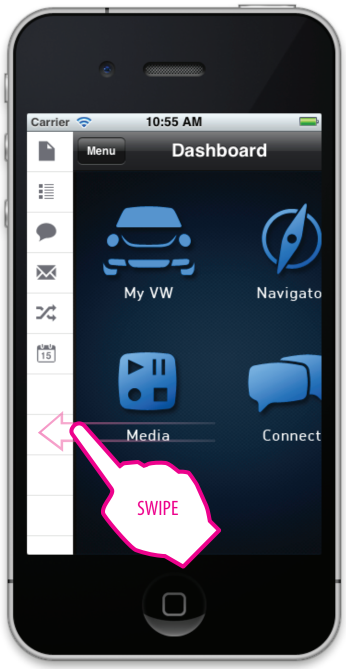
To prevent the window from moving everytime we touch it, we are waiting for a movement over 20 pixels before animating. We track our touch coordinate with e.globalPoint.x and subtract it to our starting point (axis) so we can move the window. Also, it can not slide beyond the menu width (50 pixels) or beyond the left side of the screen.

3.Now if we run the app, you'll see that your window will stay exactly where you leave it.

That's not what we want. We need to reposition it when the touch event is over, so it will open/close itself depending on where it is:

When our finger no longer touches the screen, the touchend event is fired, so we can adjust the position of our window.

Step 06: Now let run the final project and test out the swiping behaviour.


Carrier
10:55 AM
Menu
Dashboard
My VW
Navigato
Media
Connect
SWIPE