# 345 Tables API Deep Dive

## Objective

* To create a custom table with rows containing imageViews and labels.
* To add event listeners that swap images when row children are tapped.

## Summary

In this lab assignment, you will create a custom table that doesn’t fill the entire viewport. The table will contain customized rows with background images that differ based on the row’s location within the table. Each row will contain two images and two labels. When you tap a row, an event listener will determine if either of the images was the object that received the tap. If so, that image will be swapped with an alternate graphic.

## Starting Point Code and Source Graphics

Download the starting code from {WIKI LOCATION}

## Lab Specification

Step 1: Download, extract, and then import the TableView project into Studio. Confirm that the tiapp.xml file has appropriate values then close that file. Open app.js in Studio.

Step 2: Following the comments included in the starting app.js file, add these elements to the app:

* Set the window background to images/gradientBackground.png
* Add a page heading of “Custom Table” with a dark blue, 18 px, bold font that is positioned at the top-left of the window
* Define a table that is positioned below the label and which is 90% of the width of the screen and 85% of its height. Set the table’s background color to transparent and for iOS, set the separator style to NONE.

Step 3: Write a function named makeRow() following the specifications included in the comments within the code.

* Your makeRow() function should accept an object that will be used to pass in four values: the row number, primary label text, secondary label text, and a custom “which image” indicator string. Possible values for the row number parameter will be an integer or the string ‘last’. Possible values for the myImage string property will be a, b, c, blue, and red. You’ll use this myImage property to determine which element in the row is tapped and to swap images accordingly.
* Each row should contain two images:
  + Left image: if an even-numbered row = ‘images/imageA.png’ otherwise ‘images/imageB.png’. It should have a custom property named myImage that is set equal to the myImage parameter passed to your makeRow() function at run time.
  + Right image: use ‘images/notificationBadge.png’ and set its custom myImage property equal to ‘blue’
* Each row should contain two labels:
  + Primary label: use a bold, 16 px font positioned to the right of the left image with its text set equal to the primary label parameter passed to makeRow() at run time. Be sure to set a height for the label.
  + Secondary label: use a bold, 13 px font positioned below the primary label with its text set equal to the secondary label parameter passed to makeRow() at run time. Be sure to set a height for the label.
* Each row’s background image should be set to ‘images/middleRow.png’ and the selected background image to ‘images/middleRowSelected.png’. If the row number parameter is 0, set the row’s background image to ‘images/topRow.png’ and selected background image to ‘images/topRowSelected.png’. If the row number parameter equals ‘last’ then use ‘images/bottomRow.png’ and ‘images/bottomRowSelected.png’ for the background images and use ‘images/imageC.png’ for the left image.
* The makeRow() function should return a Titanium.UI.TableViewRow object.

Step 4: Use a for-loop to create an array of 8 rows for your table, calling makeRow() in each iteration of the loop. Set the primary label to “This is row” plus the row-number indicator. Set the secondary label to “Subtitle” and the row-number indicator. Set the myImage value to either ‘a’ or ‘b’ depending on whether your loop counter is odd or even. (Hint: use the modulus operator, %, to calculate this odd/even value.)

Push one additional row into the rows array. This row should pass these values:  
 row number: ‘last’  
 primary label: ‘This is the last row’  
 secondary label: ‘The last subtitle’  
 myImage: ‘c’

Step 5: Add a click event listener to your table. Check the myImage property of the event source. Using a switch or if-else test, determine if myImage is set equal to a, b, c, blue, or red. If a, b, or c, swap the left image such that imageA becomes imageB, imageB becomes imageC, and imageC becomes imageA again. If the myImage property equals blue, the right image should change to images/notificationUnreadBadge.png otherwise it should swap back to images/notificationBadge.png.

Step 6: Save and run the project in the simulator. Confirm that the user interface matches the screenshots shown below. Confirm that your event listener functions properly:

* When the row is tapped, the background image should swap to the green “selected” version.
* If the left image is tapped, the letter should advance to the next letter, looping back to the beginning properly.
* If the right image is tapped, it should swap between the red and blue versions properly.

## Development Resources

### Development flow tip

{MOVIE LINK GOES HERE}

### Resources

* TableView API docs: <http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI.TableView-object>
* TableViewRow API docs: <http://developer.appcelerator.com/apidoc/mobile/latest/Titanium.UI.TableViewRow-object>
* Kitchen Sink code: <https://github.com/appcelerator/KitchenSink>

## Expected UI examples

 