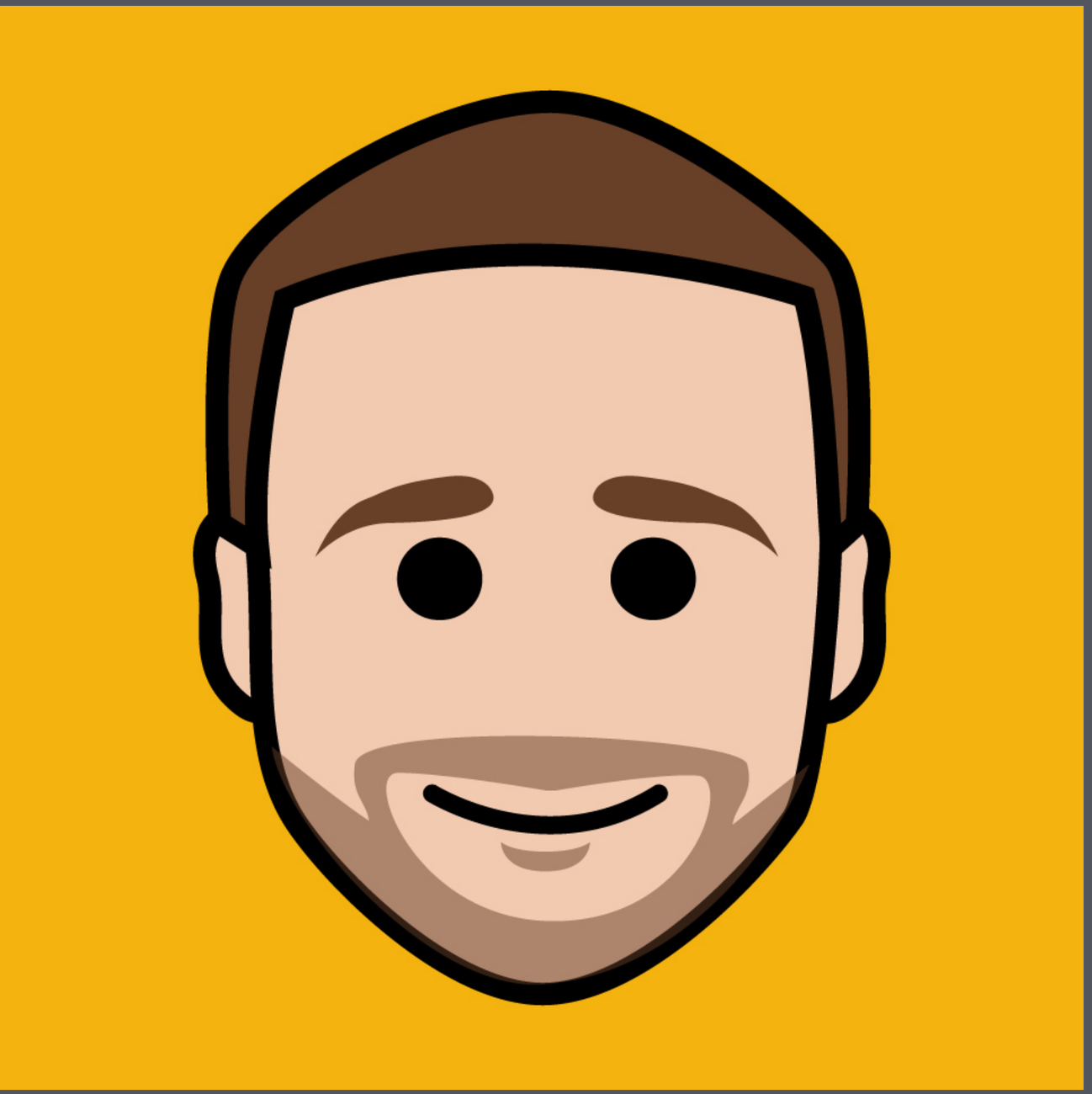
iOS 101 for Android Developers

Stephen Barnes - @smbarne

AnDevCon Boston 2015

github.com/smbarne/AndroidForiOS



Who is this guy?

- Previously Senior Mobile Dev @Raizlabs, Now Senior iOS Dev @Fitbit
- Twitter: @smbarne
- Github: github.com/smbarne
- Misc: engineeringart.io


Why?
...Come to the Dark side

- Learn something new
- Cross platform teams
- Reach more users

Overview

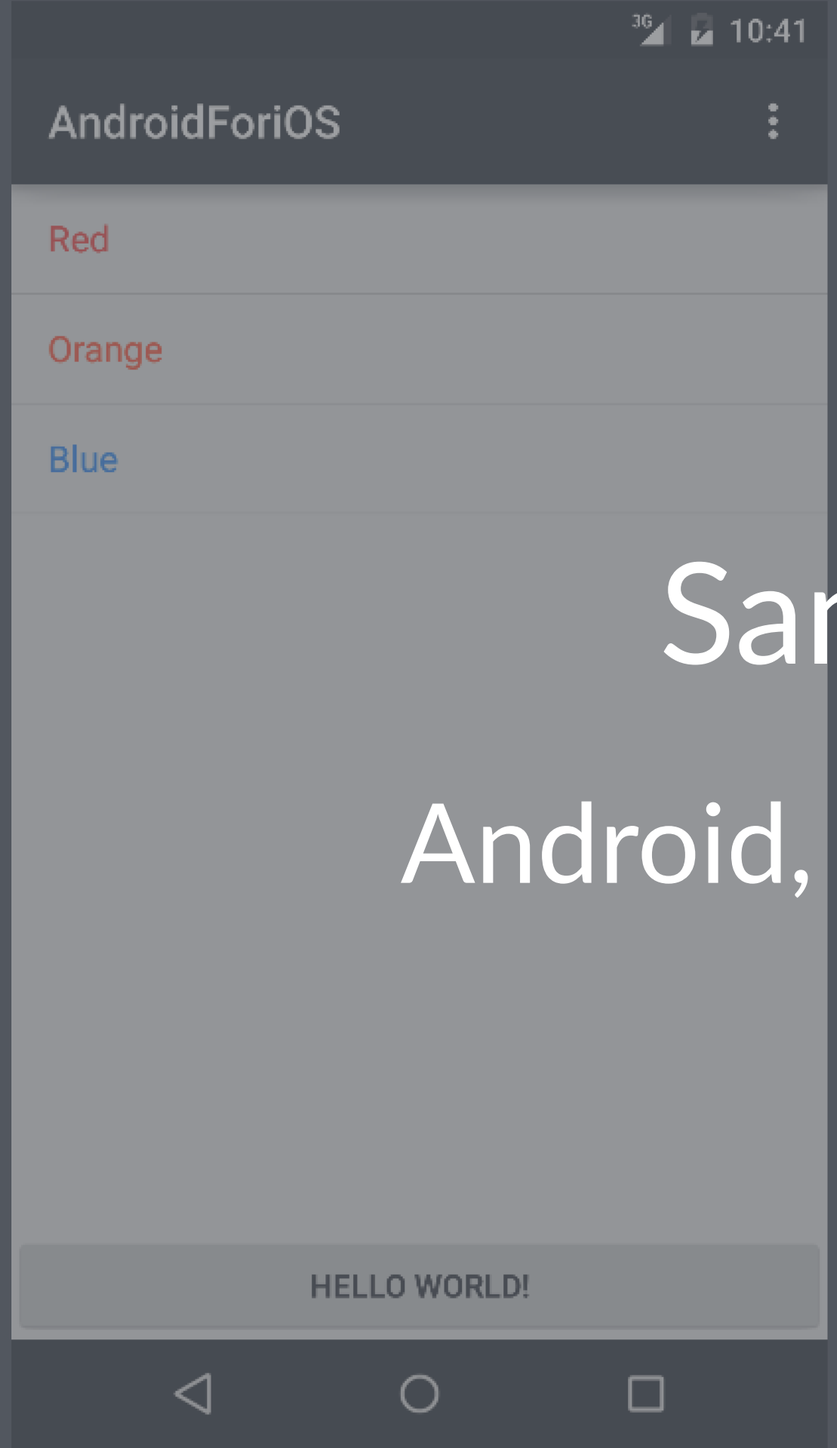
- A Word on UI Design
- Application Structure and Language
- Building Blocks: ViewControllers, TableViews, Cells, Tabbars, and more
- iOS Lifecycle
- iOS Layouts
- Data Management

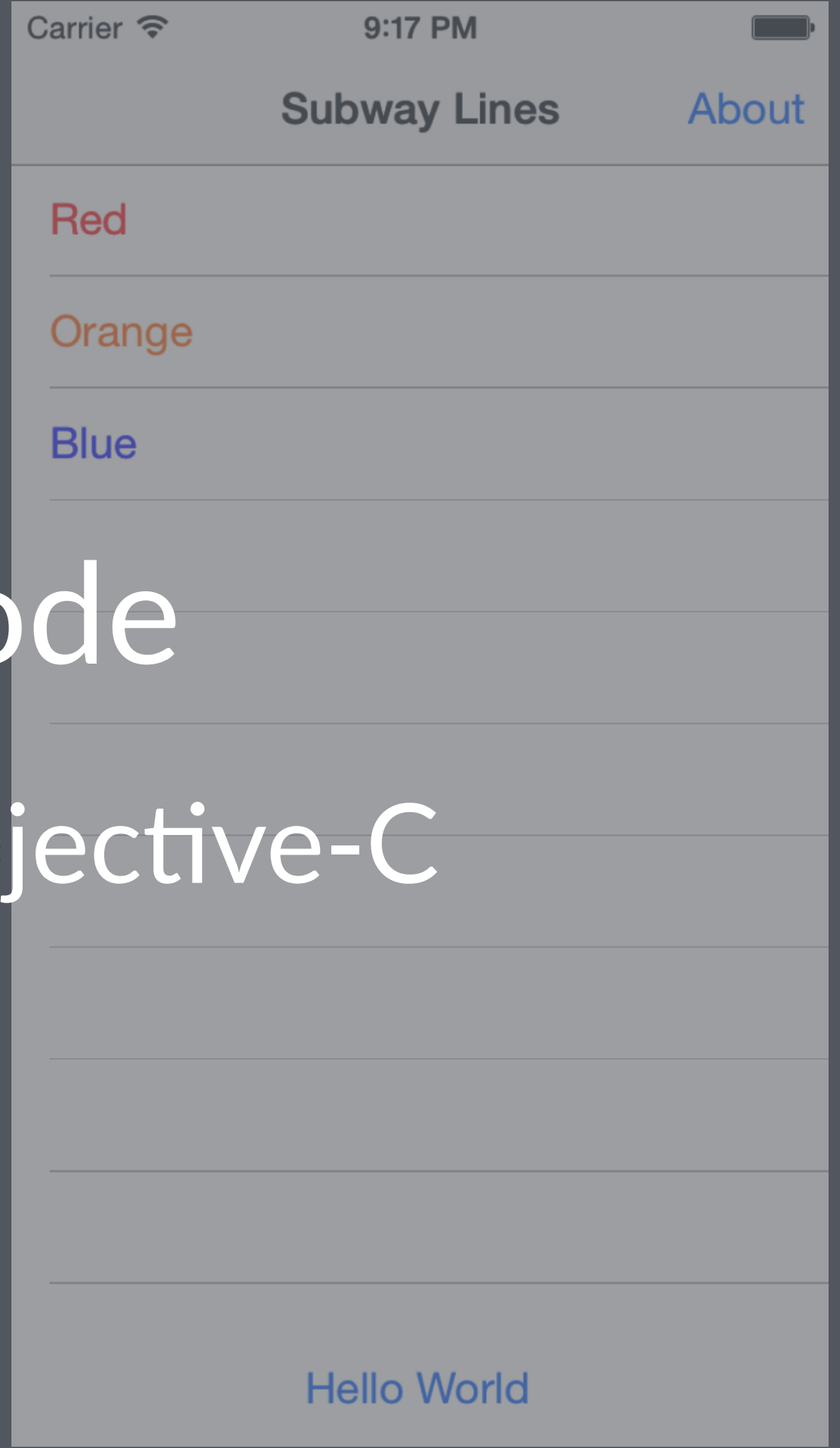
Requirments

- ~~Apple Developer Account~~
- Mac

Recommendations

Read through the Start Developing iOS Apps Today tutorial by Apple when possible.





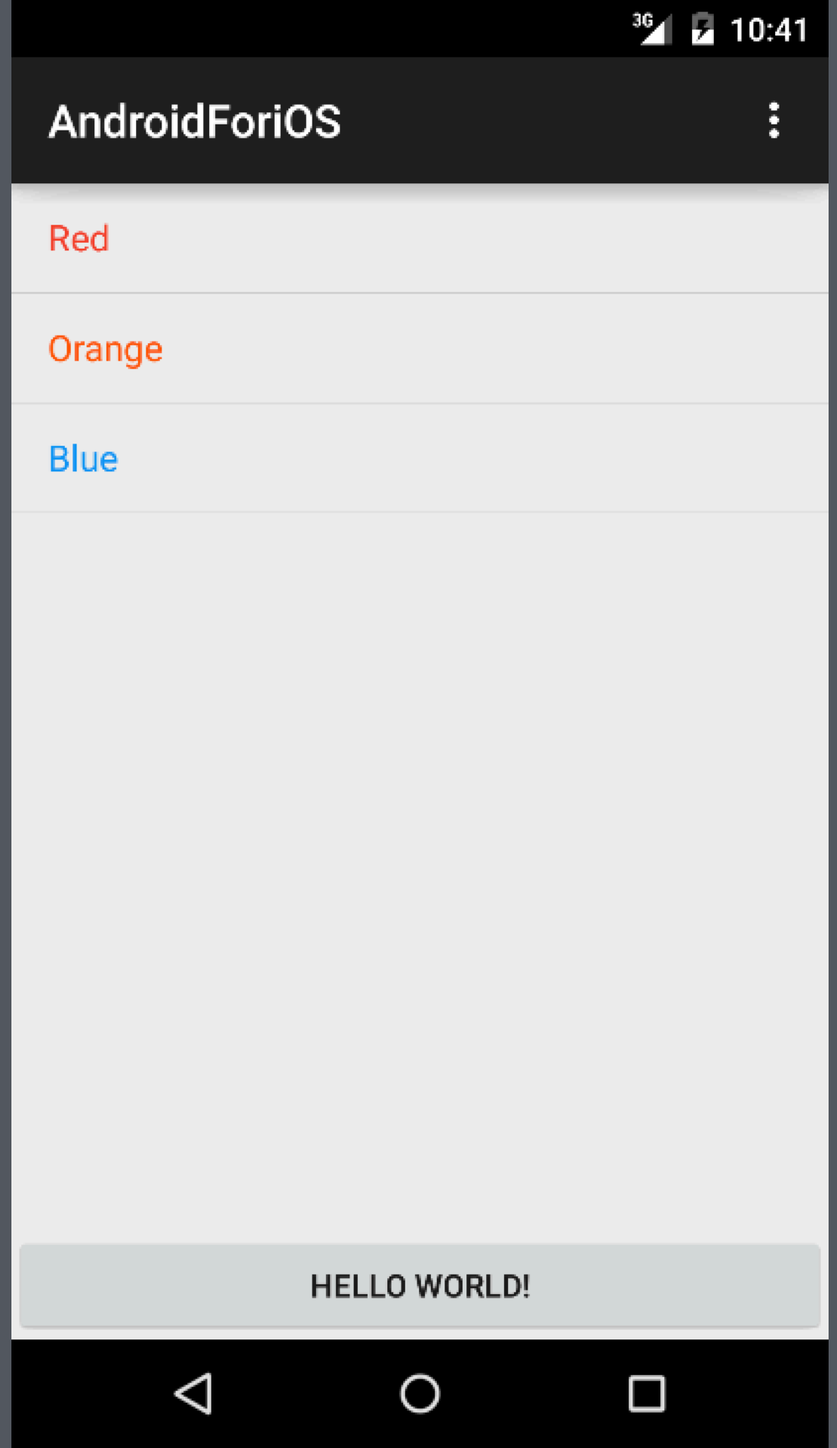
Sample Code

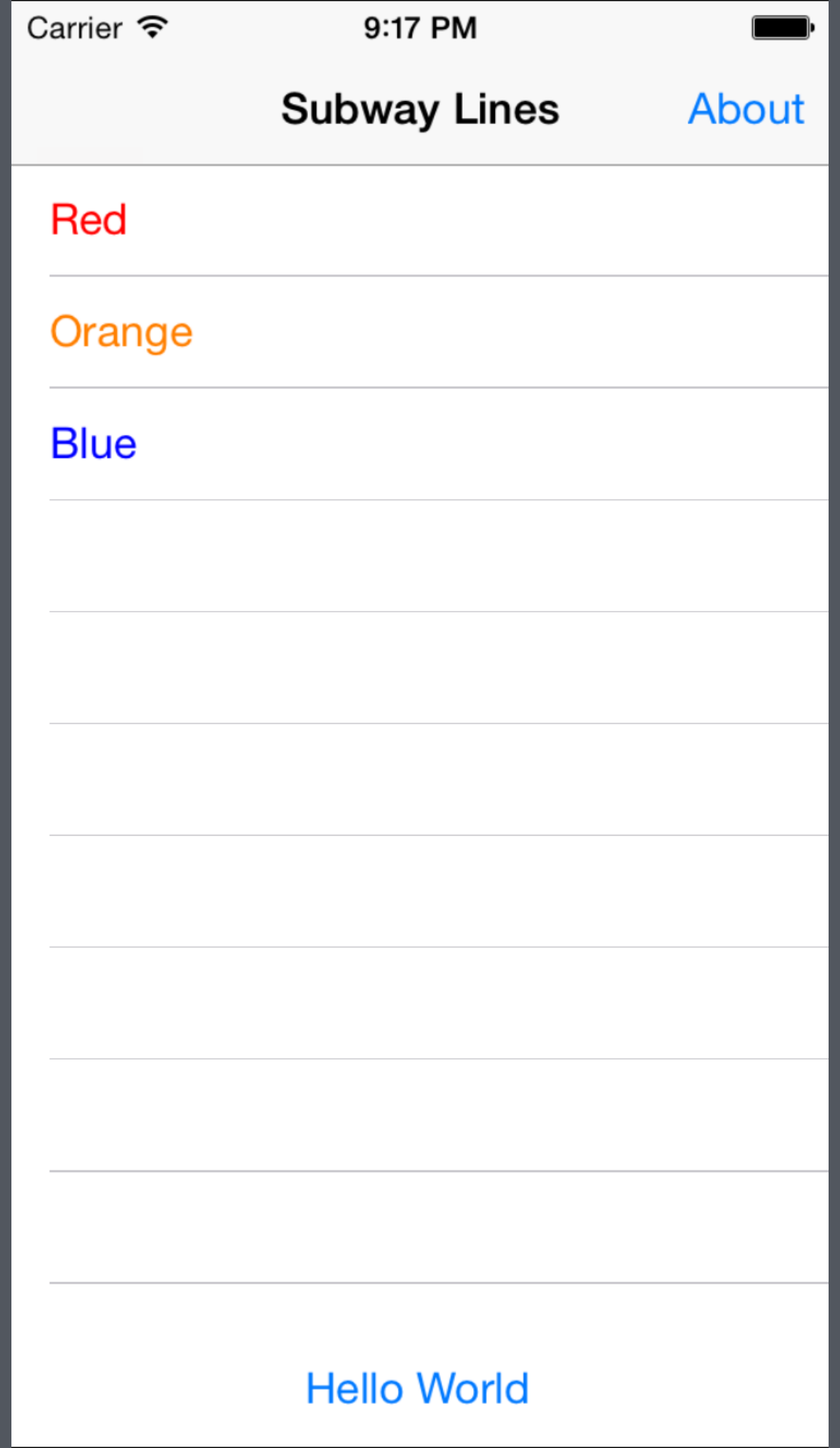
Android, Swift, Objective-C

Resources

github.com/smbarne/AndroidForiOS

A Word on UI Design


10:41
AndroidForiOS
Red
Orange
Blue
HELLO WORLD!

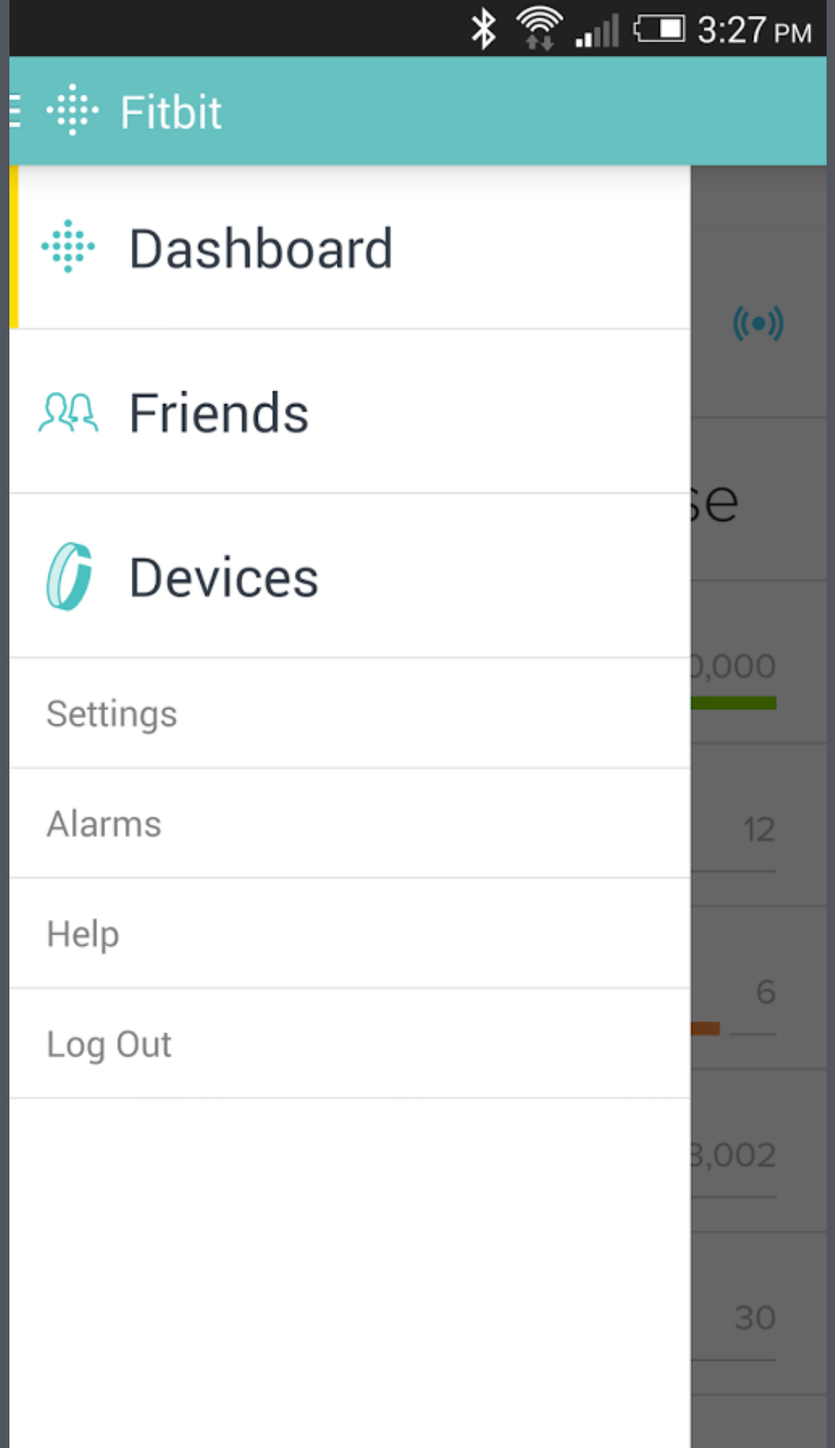

Carrier
9:17 PM
Subway Lines
About
Red
Orange
Blue
Hello World

Death to the Basement Menu (Navigation Drawer)

Long live the Tab Bar

Recent trends have advocating getting rid of the Basement Menu pattern, at least on iOS.

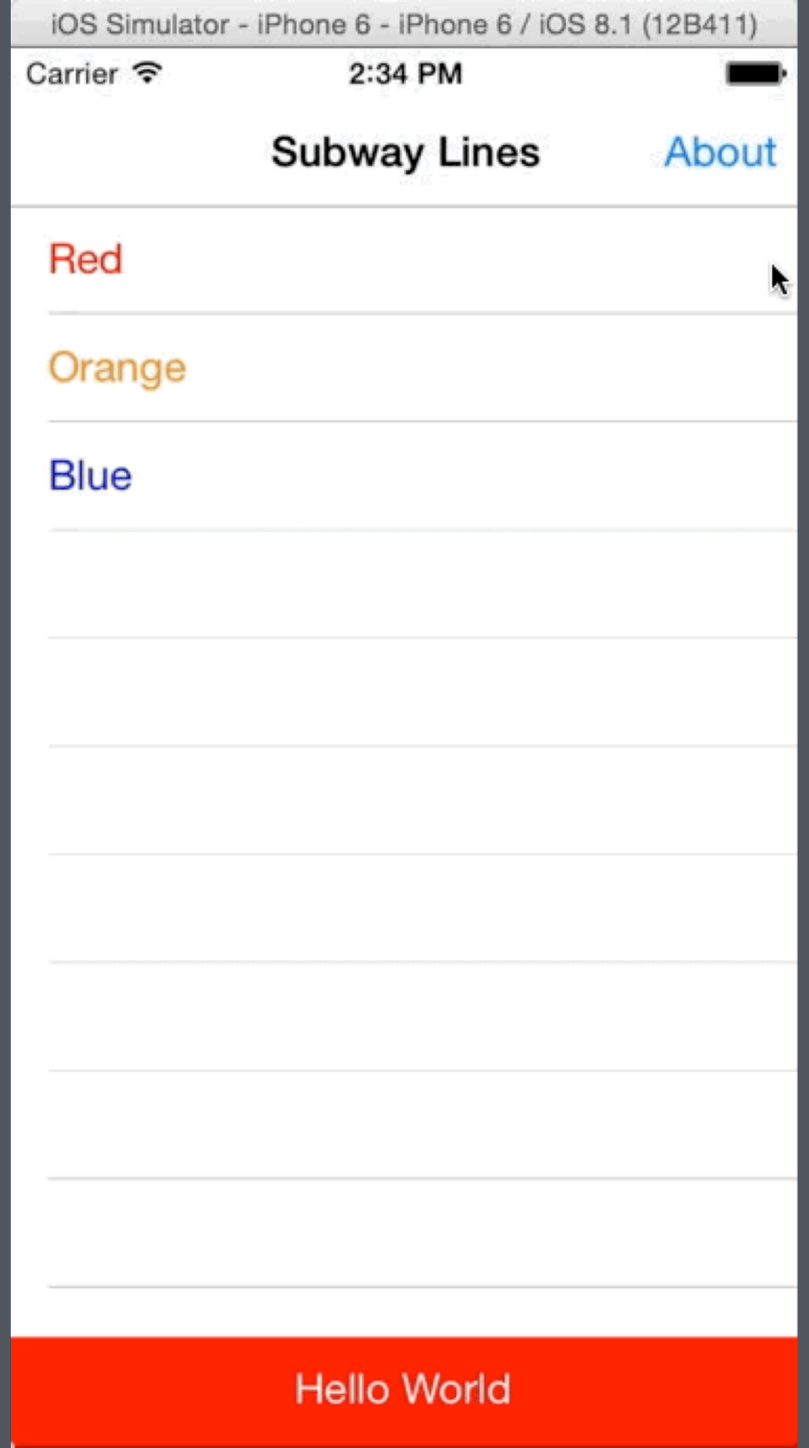

AT&T
9:50 AM
fitbit
Edit
Today
5,151 steps
2.29 miles
970 calories burned
Track exercise
150 lbs
Dashboard
Challenges
Friends
Account

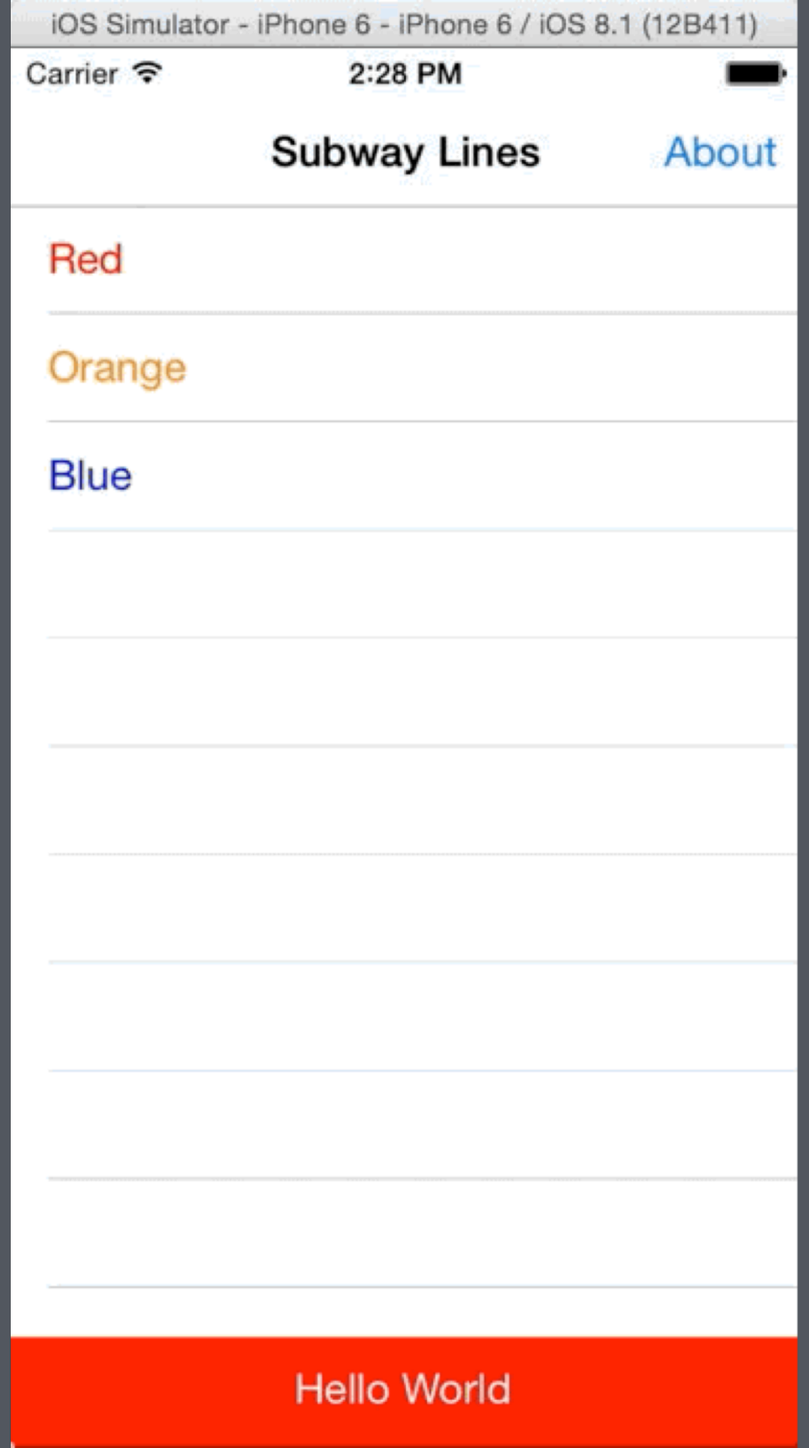

3:27 PM
Fitbit
Dashboard
Friends
Devices
Settings
Alarms
Help
Log Out

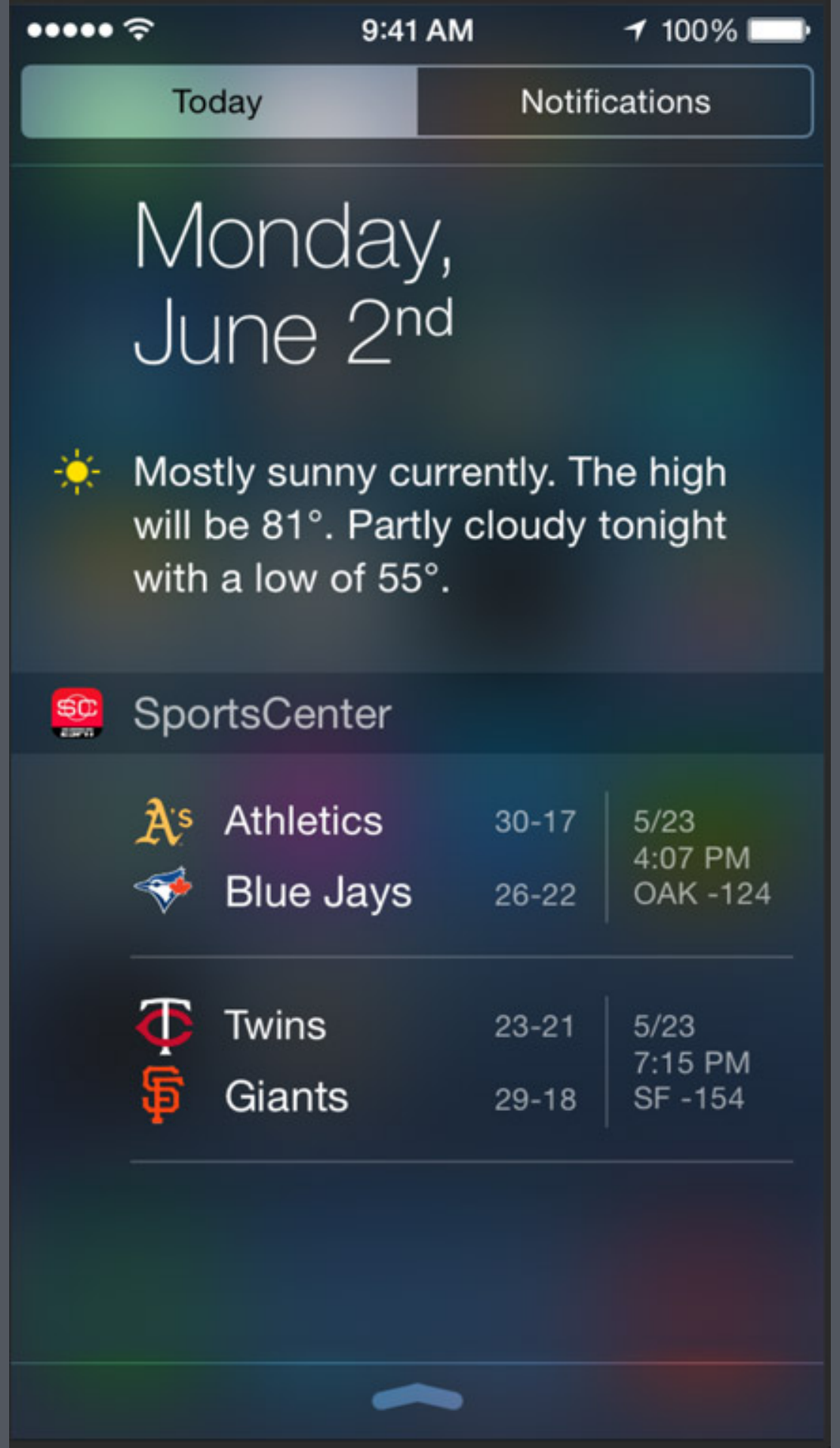
Push/Pop vs Present/Dismiss

Android has a built in Activity Stack, iOS does not

- Use a `UINavigationController` to Push and Pop `UIViewControllers`
  - Use this by default and for nested content
- Any `UIViewController` can modally Present another `UIViewController`
  - Use this for presenting modular or ediable content


iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.1 (12B411)
Carrier
2:34 PM
Subway Lines
About
Red
Orange
Blue
Hello World


iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.1 (12B411)
Carrier
2:28 PM
Subway Lines
About
Red
Orange
Blue
Hello World



Widgets

As of iOS 8

Other Extentions

- Custom Keyboards
- Photo/Media Editing
- Storage Provider
- Document Picker
- More: https://developer.apple.com/app-extensions/

Missing Patterns

- Global Back Button
  - Back is *within* app, not OS wide
- Overflow Menu
  - Try ActionSheets or Popovers instead
- Toolbar/Actionbar
  - Somewhat similar with UINavigationBar

Others

- Phone vs tablet[1]
- Rotation handling
- Aspect ratios[1]
- Overscroll

[1] iOS 8 now brings iOS to a similar architecture

Swift 2.0

For iOS 9 and Xcode 7, Apple annouced **Swift 2.0**

Swift is an innovative new programming language for Cocoa and Cocoa Touch. Writing code is interactive and fun, the syntax is concise yet expressive, and apps run lightning-fast. Swift is ready for your next iOS and OS X project — or for addition into your current app — because Swift code works side-by-side with Objective-C.

What does this mean?

- `Objective-C` and `Swift` can be used in the same project
- Lots of `Objective-C` code still in use.
- Most documentation is in Swift now.

For this talk, we're going to focus on Objective-C for practical purposes

Swift Basics

- Functional
- Optionals
- Closures
- First class values types
- Protocol Oriented

Objective-C Code Styling and Patterns

- Hello class prefixes - there is no name spacing in Objective-C.
- `Null` is `nil`, and Objective-C handles gracefully handles message sending to nil objects gracefully[2].
- `Protocols` `<->` `Interfaces` and are preferred to direct inheritance

[2] Gotchas may still include runtime crashes when sending nil into a method that cannot handle nil.

- Instance variables are prefixed with `_`, not `m`.
- `@Properties`! A handy tool for synthesizing getters and setters.
  - Setters and Getters can be customer overriden and are called event when referencing `.property`

Misc

- Provisioning
- App Store Review period
- Simulator, not emulator

Project Structure

- `Swift` uses modules
  - Note: `Objective-C` has no namespacing, so use class prefixes
- **info.plist** is required and similar to the **AndroidManifest.xml** on Android
- No **build.gradle** - All build info in ***.xcodeproj**
  - You can build workspaces of multiple projects as well

Asset Bundles

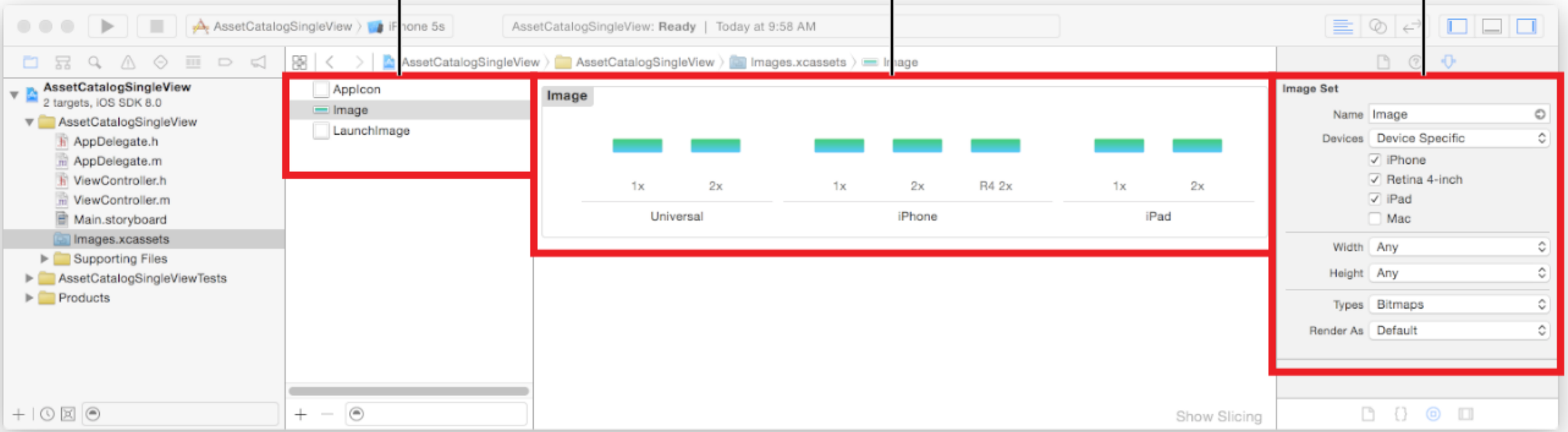
For images, Xcode provides XCAssets Bundles

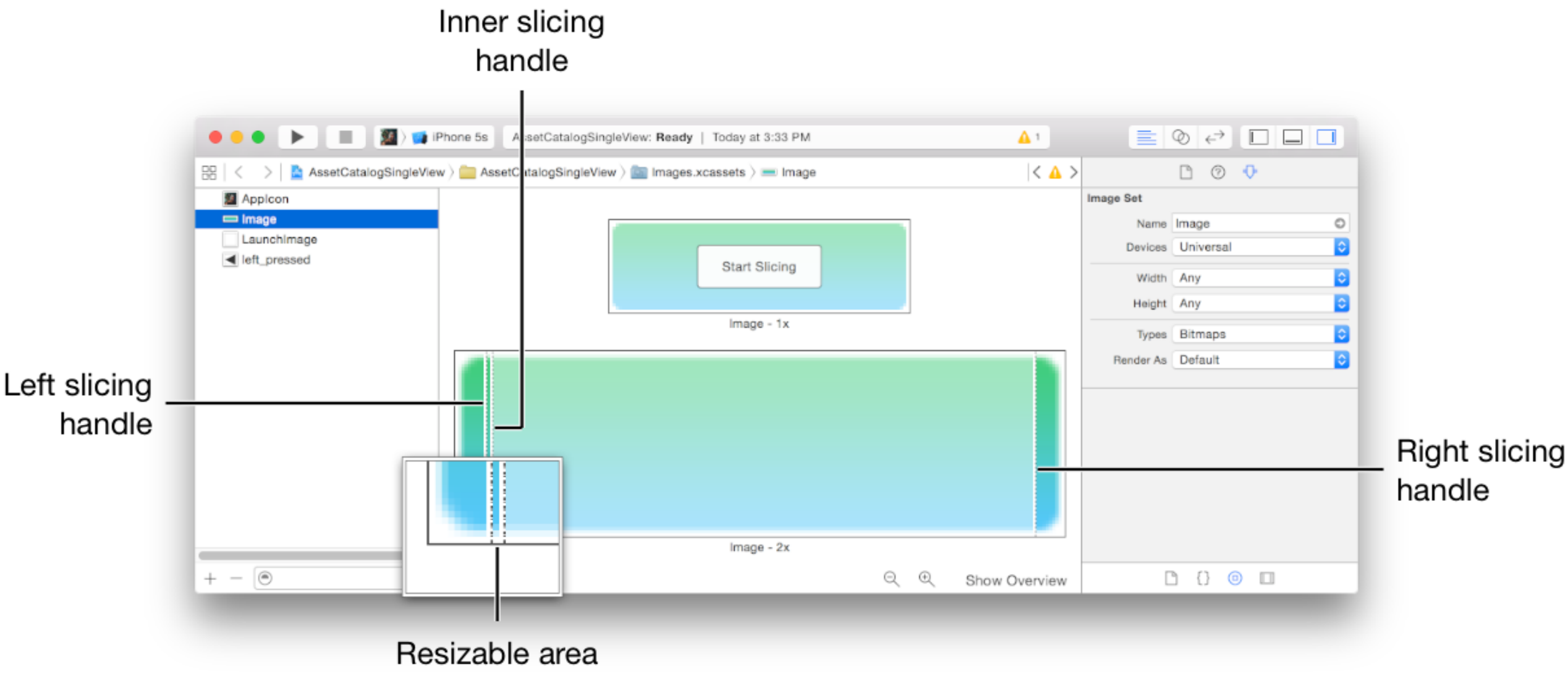
- There are **Three** buckets, instead of the many buckets for Android
  - **@1x**, **@2x**, and **@3x**
- The iPhone 6+ is the first device to have @3x and a non-1 : 1 or 2 : 1 pixel ratio

Set list

Set viewer

Set attributes inspector




Inner slicing handle
Left slicing handle
Right slicing handle
Resizable area
iPhone 5s
AssetCatalogSingleView: Ready | Today at 3:33 PM
AssetCatalogSingleView
Images.xcassets
Image
AppIcon
Image
LaunchImage
left_pressed
Start Slicing
Image - 1x
Image - 2x
Show Overview
Image Set
Name Image
Devices Universal
Width Any
Height Any
Types Bitmaps
Render As Default

Dimens and Strings

```
NSLocalizedString("This is your string value", "A descriptive comment for localization")
```

iOS uses a strings file for each language which can be generated by running `genstrings` on the project.

```
static let constantValue:Double = 1.0
```

Tools

IDEs

- Xcode
- AppCode

Distribution

- Testflight
- Cross-Platform: HockeyApp, Beta by Crashlytics, Appblade, etc

Building Blocks

Building Blocks: ViewControllers, TableViews, Cells, Tabbars, and more

UIViewControllers <-> Activities

UIViewController

The UIViewController class provides the fundamental view-management model for all iOS apps. You rarely instantiate UIViewController objects directly. Instead, you instantiate subclasses of the UIViewController class based on the specific task each subclass performs.

iOS is strongly architected around the **MVC** framework. `UIViewControllers` are the glue between model objects and the view displayed to the user.

- Easy to create large `UIViewControllers` - be careful
- Dumber views and more View Controller logic
- View Controllers do no return values. No Intents as well.

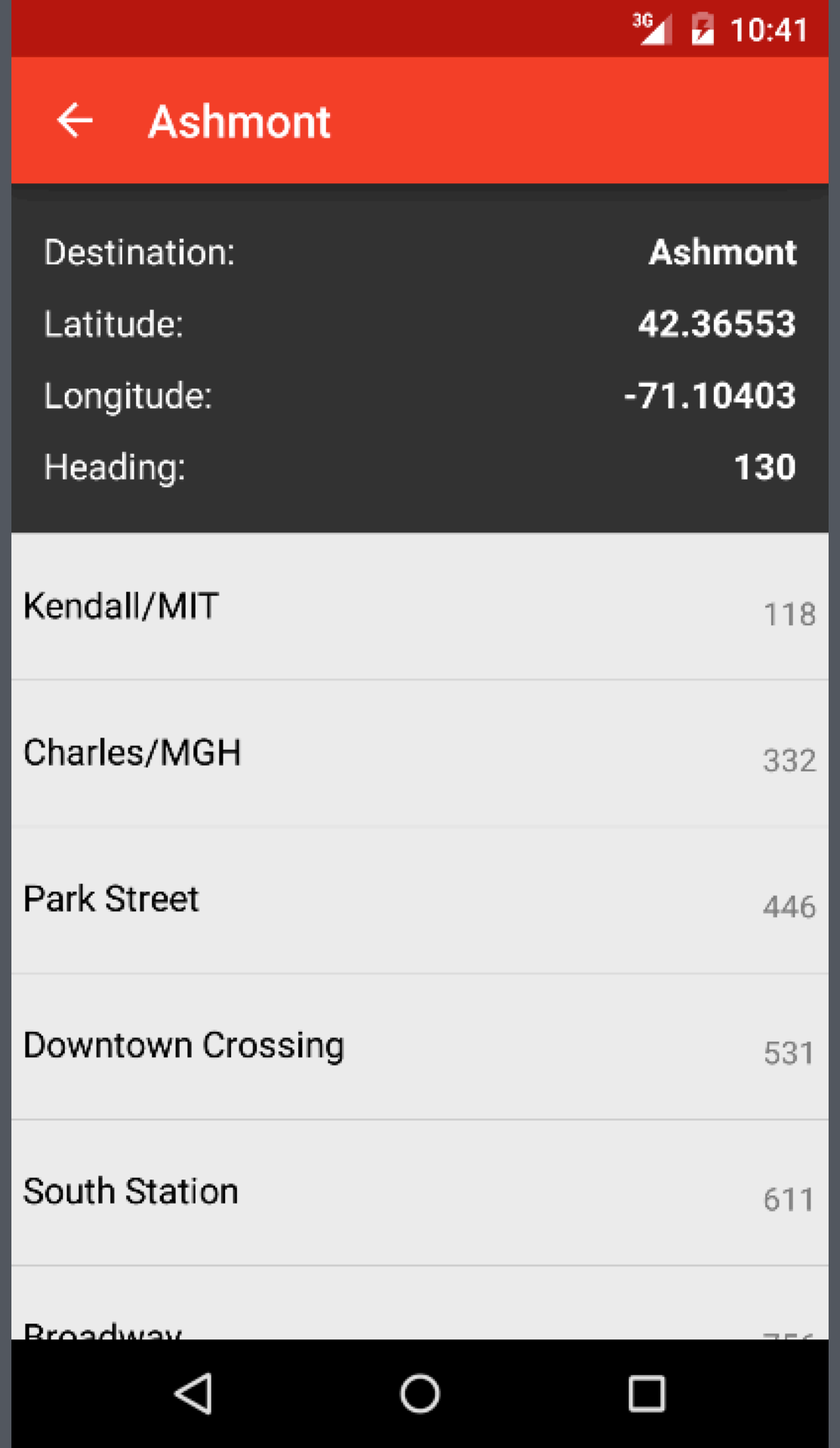
Fragments do not exist on iOS

If designing a modular controller system, use `Child View Controller Containment`.

Explain it to me with code

Let's look at a sample **UITableViewController** and a sample **ListFragment** that show a list of prediction times for a subway trip curtesy of the MBTA (Massachusetts Bay Transportation Authority).

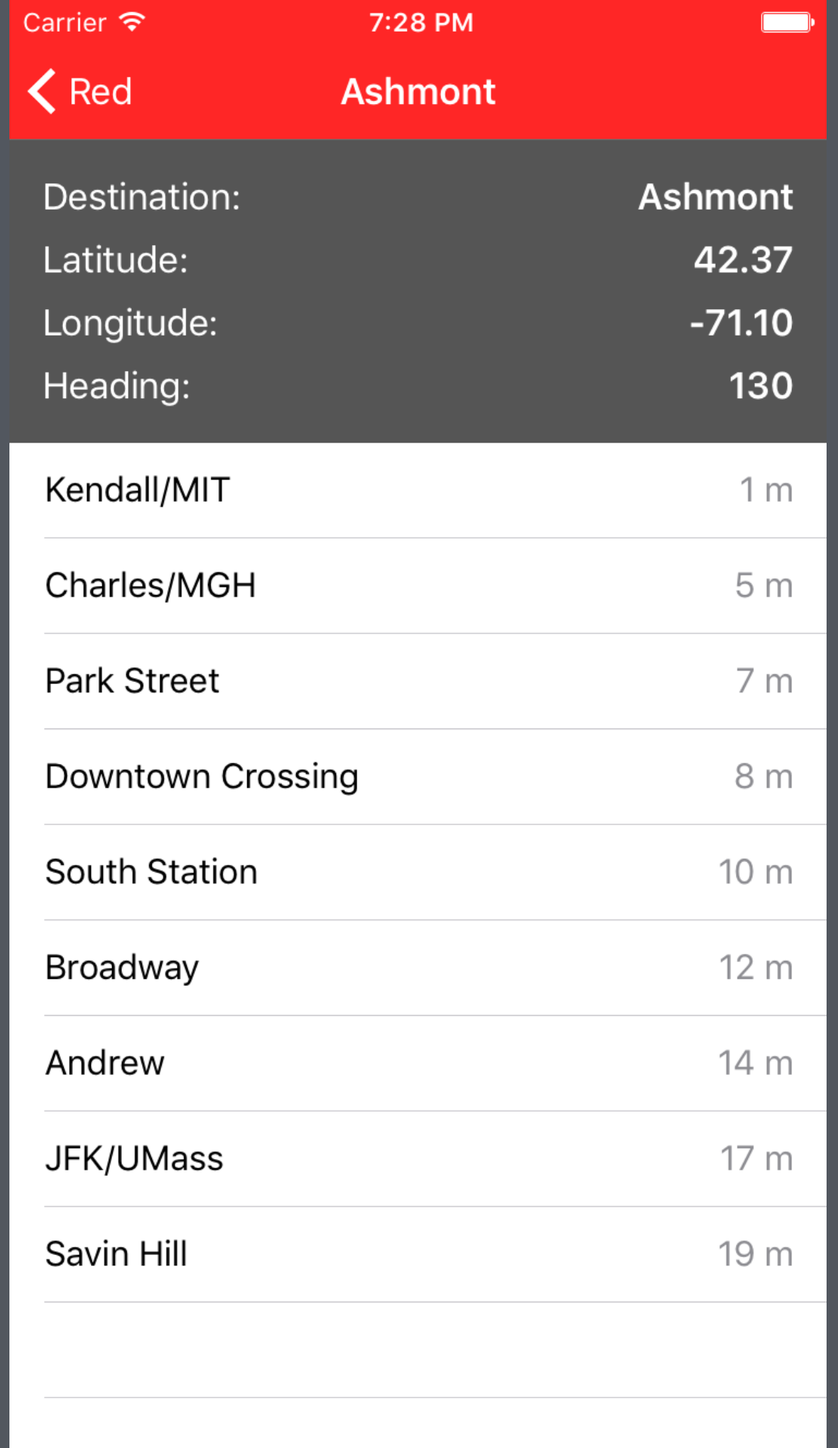
ListFragment Implementation



```java
public class TripDetailFragment extends ListFragment {
    /**
     * The configuration flags for the Trip Detail Fragment.
     */
    public static final class TripDetailFragmentState {
        public static final String KEY_FRAGMENT_TRIP_DETAIL = "KEY_FRAGMENT_TRIP_DETAIL";
    }

    protected Trip mTrip;

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param trip the trip to show details
     * @return A new instance of fragment TripDetailFragment.
     */
    public static TripDetailFragment newInstance(Trip trip) {
        TripDetailFragment fragment = new TripDetailFragment();
        Bundle args = new Bundle();
        args.putParcelable(TripDetailFragmentState.KEY_FRAGMENT_TRIP_DETAIL, trip);
        fragment.setArguments(args);
        return fragment;
    }

    public TripDetailFragment() { }
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        Prediction[] predictions =
             mTrip.predictions.toArray(new Prediction[mTrip.predictions.size()]);

        PredictionArrayAdapter predictionArrayAdapter =
             new PredictionArrayAdapter(getActivity().getApplicationContext(), predictions);

        setListAdapter(predictionArrayAdapter);
        return super.onCreateView(inflater,container, savedInstanceState);
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        TripDetailsView headerView = new TripDetailsView(getActivity());
        headerView.updateFromTripObject(mTrip);
        getListView().addHeaderView(headerView);
    }
}
```

iOS Tableview Implementation



```
public class TripDetailViewController : UITableViewController {
  @IBOutlet weak var tripDetailHeaderView: TripDetailHeaderView!
  var trip:Trip?

  public func prepareWithTrip(trip:Trip) {
      self.trip = trip
  }

  // MARK - View Lifecycle
  public override func viewDidLoad() {
    self.title = trip?.destination
    if let trip = self.trip {
      self.tripDetailHeaderView.updateHeader(trip)
    }
  }
```

```
// MARK - UITableViewDataSource
public override func tableView(tableView: UITableView,
  numberOfRowsInSection section: Int) -> Int {
  guard let dataCount = self.trip?.predictions?.count else {
      return 0
  }
  return dataCount
}

public override func tableView(tableView: UITableView,
    cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
  let cell:UITableViewCell = tableView
    .dequeueReusableCellWithIdentifier("TripPredictionTableViewCell",
      forIndexPath: indexPath)

  if let prediction = self.trip?.predictions?[indexPath.row] {
      cell.textLabel?.text = prediction.stopName
      cell.detailTextLabel?.text = String(prediction.stopSeconds / 60) + " m"
  }

  return cell;
}
```

```
// MARK - UITableViewDelegate
public override func tableView(tableView: UITableView,
    didSelectRowAtIndexPath indexPath: NSIndexPath) {

  tableView.deselectRowAtIndexPath(indexPath, animated: true)
}
```

Let's break down some of that piece by piece...

UITableViews and Datasources

ListView, meet UITableView

Both are sturctured around showing a linear list of Views smoothly

- iOS has *cells* - any content you want to show in a `UITableView` should be a subclass of `UITableViewCell`
- Reuse your views in a `UITableView`! You do this by `dequeing` a cell from the `UITableView`
- Default `UITableViewCells` are available that you can populate just like Android has default `ListView` Views

iOS Uses the Delegation Pattern Often

- `UITableViewDelegate` is a `protocol` a `UIViewController` implements to control a `UITableView`'s behavior.
  - This includes row height[3], header views, row selection actions

[3] iOS8 added fully automatic table view cell sizing

UITableViews are populated via **Datasources**

Goodbye adapters, hello datasources

Instead of having a adapters, iOS has **datasource delegation**

- Datasources provide the data and are responsible for dequeing and setting up the cells
- It is important that the number of sections and rows match any additions and removals to the `UITableView`

```
public override func viewDidLoad() {
  self.title = trip?.destination
  if let trip = self.trip {
    self.tripDetailHeaderView.updateHeader(trip)
  }
}
```

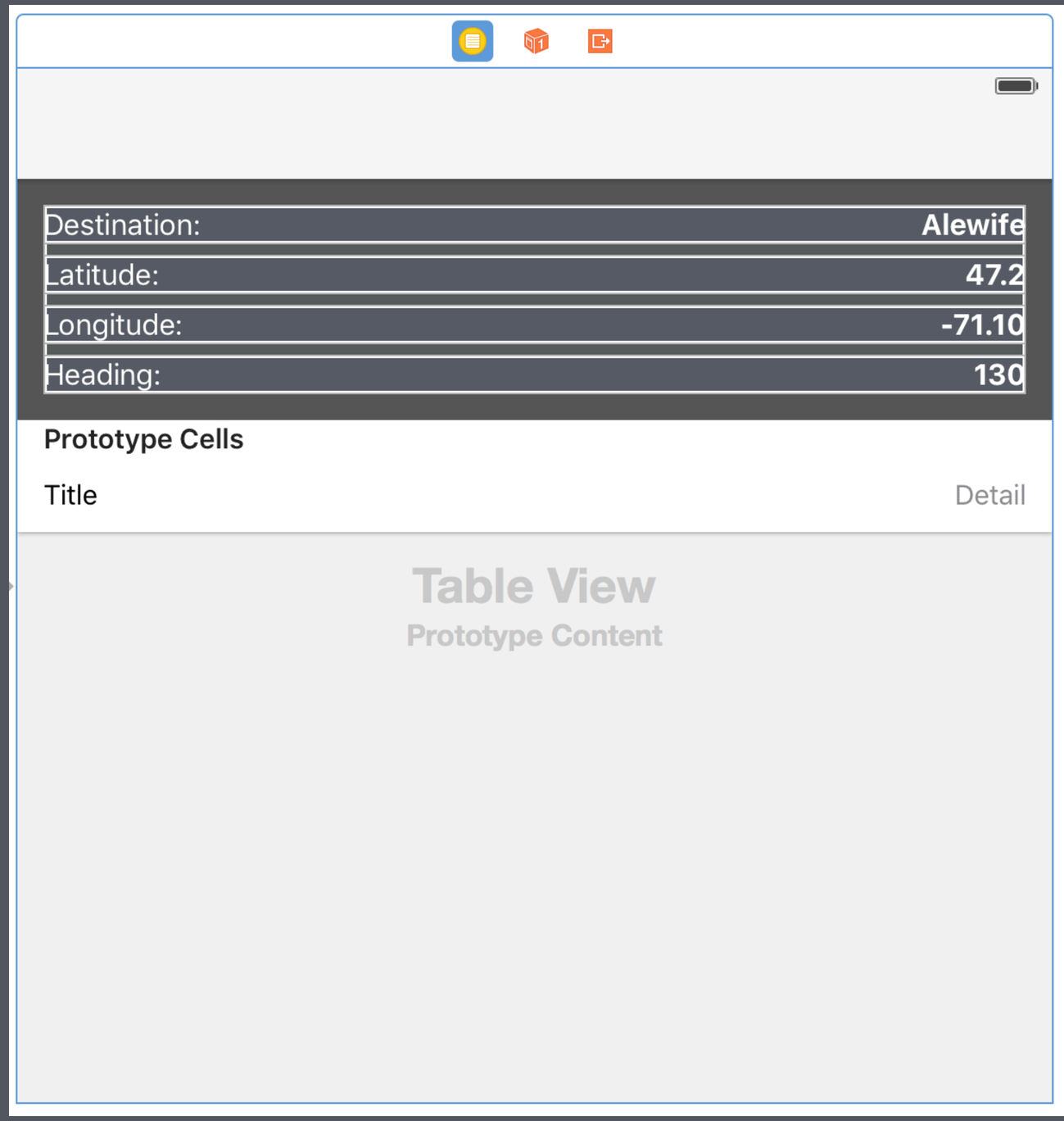
```
public override func tableView(tableView: UITableView,
      numberOfRowsInSection section: Int) -> Int {
  guard let dataCount = self.trip?.predictions?.count else {
      return 0
  }
  return dataCount
}

public override func tableView(tableView: UITableView,
    cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
  let cell:UITableViewCell = tableView.
    dequeueReusableCellWithIdentifier("TripPredictionTableViewCell",
      forIndexPath: indexPath)

  if let prediction = self.trip?.predictions?[indexPath.row] {
      cell.textLabel?.text = prediction.stopName
      cell.detailTextLabel?.text = String(prediction.stopSeconds / 60) + " m"
  }
  return cell;
}
```

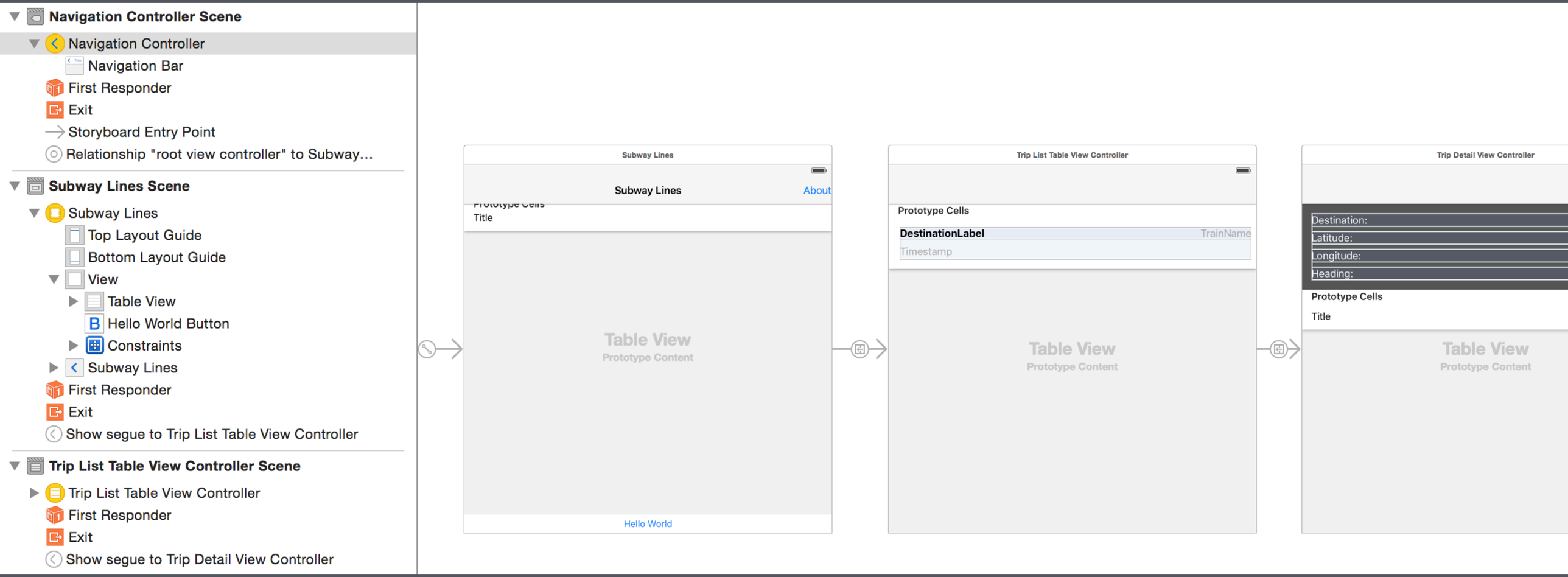
Now compare that to our Adapter

```
public class PredictionArrayAdapter extends ArrayAdapter<Prediction> {
    int LAYOUT_RESOURCE_ID = R.layout.view_three_item_list_view;

    public PredictionArrayAdapter(Context context) {
        super(context, R.layout.view_three_item_list_view);
    }

    public PredictionArrayAdapter(Context context, Prediction[] objects) {
        super(context, R.layout.view_three_item_list_view, objects);
    }
```

```
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Prediction prediction = this.getItem(position);
        View inflatedView = convertView;
        if(convertView==null)
        {
            LayoutInflater inflater = (LayoutInflater)getContext()
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflatedView = inflater.inflate(LAYOUT_RESOURCE_ID, parent, false);
        }

        TextView stopNameTextView = (TextView)inflatedView
            .findViewById(R.id.view_three_item_list_view_left_text_view);
        TextView middleTextView = (TextView)inflatedView
            .findViewById(R.id.view_three_item_list_view_middle_text_view);
        TextView stopSecondsTextView = (TextView)inflatedView
            .findViewById(R.id.view_three_item_list_view_right_text_view);

        stopNameTextView.setText(prediction.stopName);
        middleTextView.setText("");
        stopSecondsTextView.setText(prediction.stopSeconds.toString());

        return inflatedView;
    }
}
```

Storyboard


Destination:
Alewife
Latitude:
47.2
Longitude:
-71.10
Heading:
130
Prototype Cells
Title
Detail
Table View
Prototype Content


Navigation Controller Scene
Navigation Controller
Navigation Bar
First Responder
Exit
Storyboard Entry Point
Relationship "root view controller" to Subway...
Subway Lines Scene
Subway Lines
Top Layout Guide
Bottom Layout Guide
View
Table View
Hello World Button
Constraints
Subway Lines
First Responder
Exit
Show segue to Trip List Table View Controller
Trip List Table View Controller Scene
Trip List Table View Controller
First Responder
Exit
Show segue to Trip Detail View Controller
Subway Lines
Subway Lines
About
Title
Table View
Prototype Content
Hello World
Trip List Table View Controller
Prototype Cells
DestinationLabel
TrainName
Timestamp
Table View
Prototype Content
Trip Detail View Controller
Destination:
Latitude:
Longitude:
Heading:
Prototype Cells
Title
Table View
Prototype Content

Closures

Closures

- Closure are Apple's lambda addition to `Swift`
  - Or `Blocks` in Objective-C
- Largely replace the delegation pattern
- Similar use to anonymous functions and AsyncTasks

Using Closures

```
weak var weakSelf = self
DataManager.sharedInstance.importData(subwayLineType)
  { (tripList, succeeded, error) -> () in
    if (succeeded) {
        weakSelf?.data = tripList?.trips
        weakSelf?.tableView.reloadData()
    } else {
        weakSelf?.showViewController(
            UIAlertController(
                title: NSLocalizedString("Uh Oh", comment: "Unknown Error Title"),
                message: error?.localizedDescription,
                preferredStyle: UIAlertControllerStyle.Alert),
            sender: weakSelf)
    }
}
```

Grand Central Dispatch

Apple's centralized threading dispatching system for iOS and OSX. It makes threading safer and more convienent. Used largely via dispatch blocks such as:

```
dispatch_async(<#dispatch_queue_t queue#>, <#^(void)block#>)
```

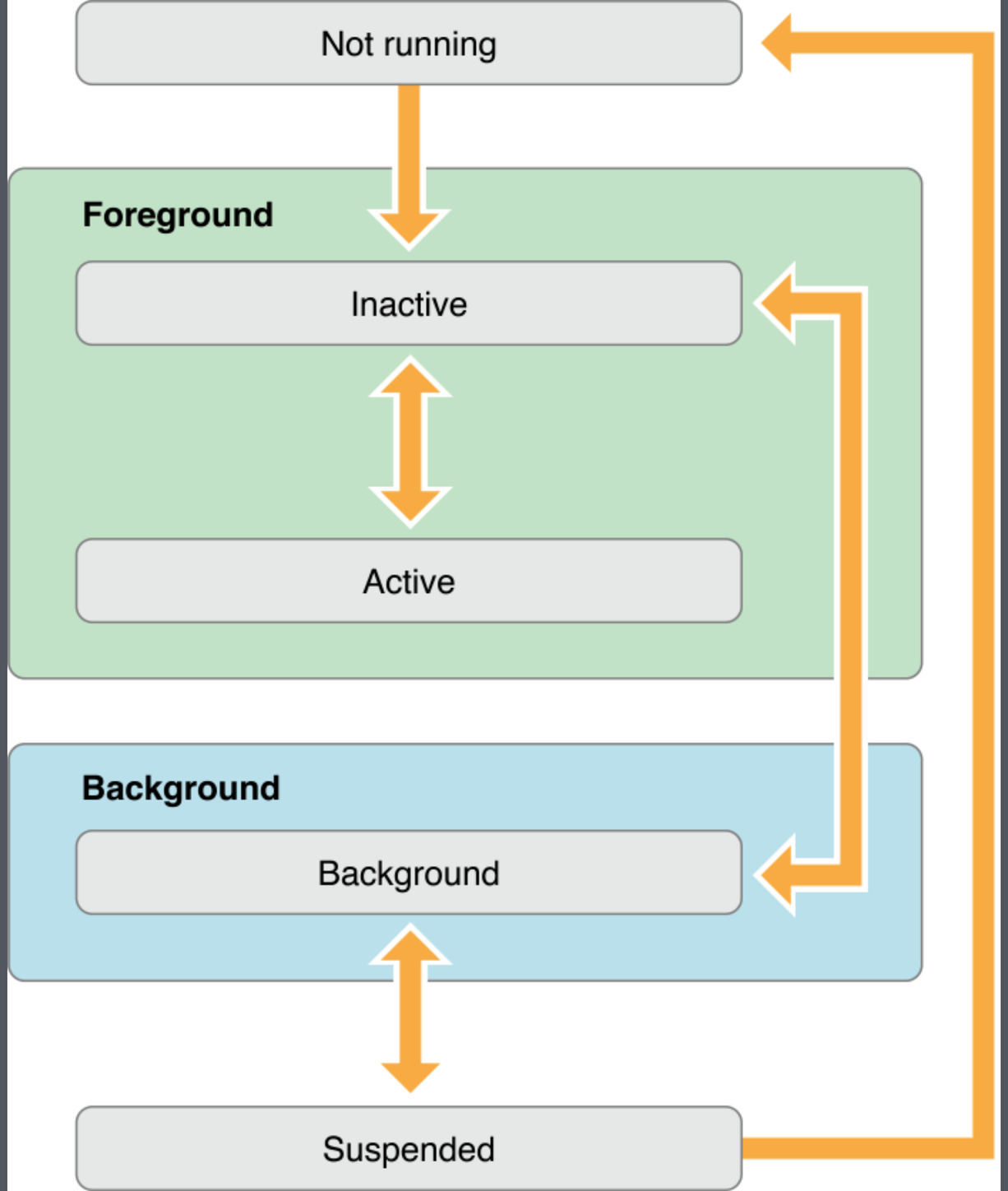
```
dispatch_sync(<#dispatch_queue_t queue#>, <#^(void)block#>)
```

```
public func importData(
   key:LineType,
   completion: ((tripList :TripList?,
                 succeeded :Bool,
                 error :NSError?) -> ())? ) {
```

```
dispatch_async(DataManager._backgroundQueue) { () -> Void in
do {
  let data = try self.parseJSONToDictionary(key.filename())
    guard let tripListData = data.objectForKey("TripList")
      as? NSDictionary else {
      throw DataManager.importError
    }
    let tripList = try TripList.parseData(tripListData)

    if let completion = completion {
      dispatch_async(dispatch_get_main_queue(), { () -> Void in
        completion(tripList: tripList, succeeded: true, error: nil)
      })
    }
  }
```

iOS Lifecycle


THAT'S $37
#TheLEGOMovie



- iOS apps my **not** run fully in the background
  - They can, however, register to be awoken at periodic intervals and through push and silent push notifications
- There is one entrance point when the app is launching

```
func application(application: UIApplication,
 didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
```

- If removed from memory, the app will be relaunched from the beginning

Benefits

- Rotation without destruction
- Properties - no need to save and restore state data
- A simpler lifecycle for `VCs` instead of the `Activity->Fragment` lifecycle

Layouts

Main View Components

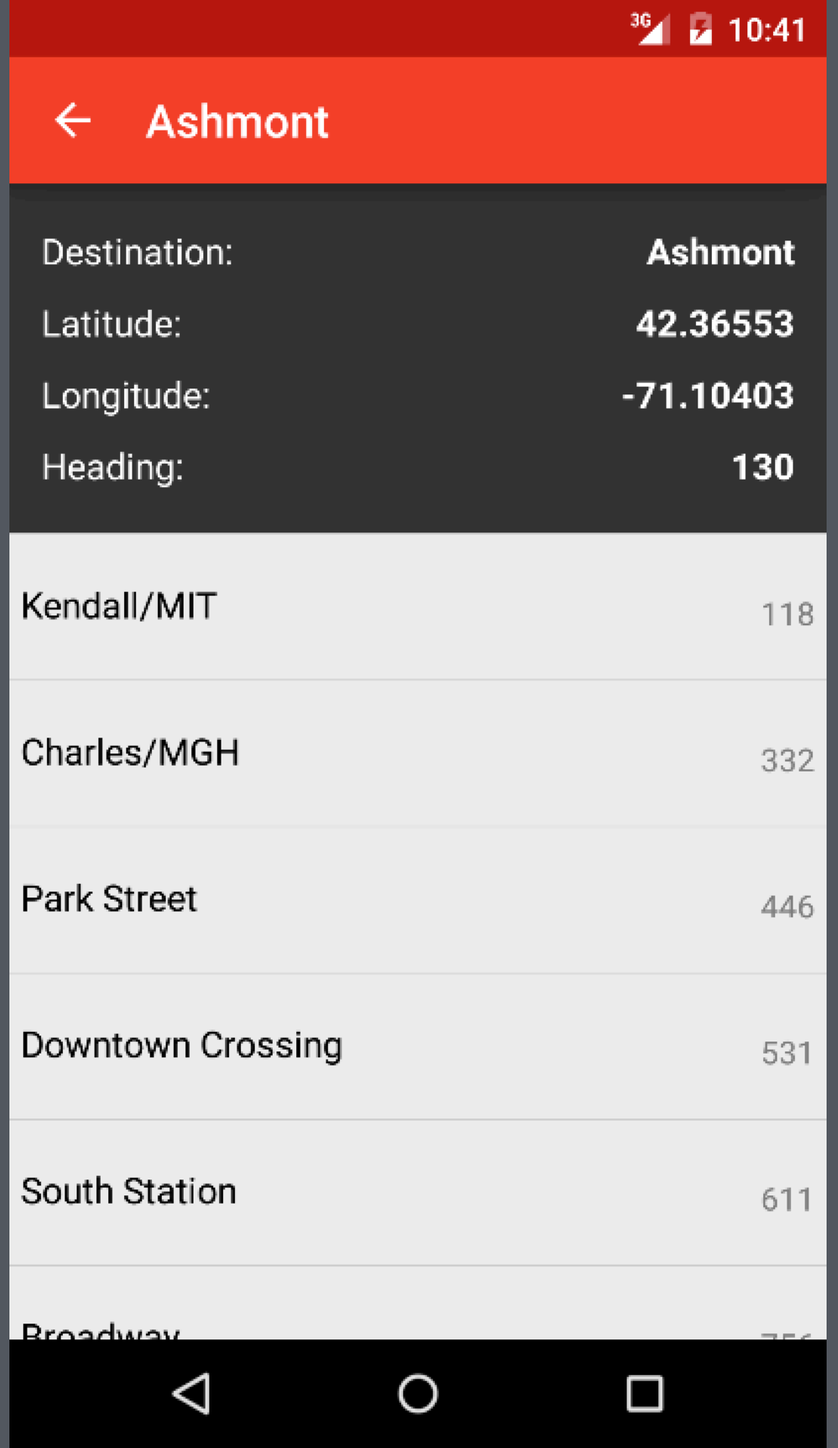
- UITabBar <-> ViewSwitcher
- UITableView <-> ListView
- UICollectionView <-> GridView

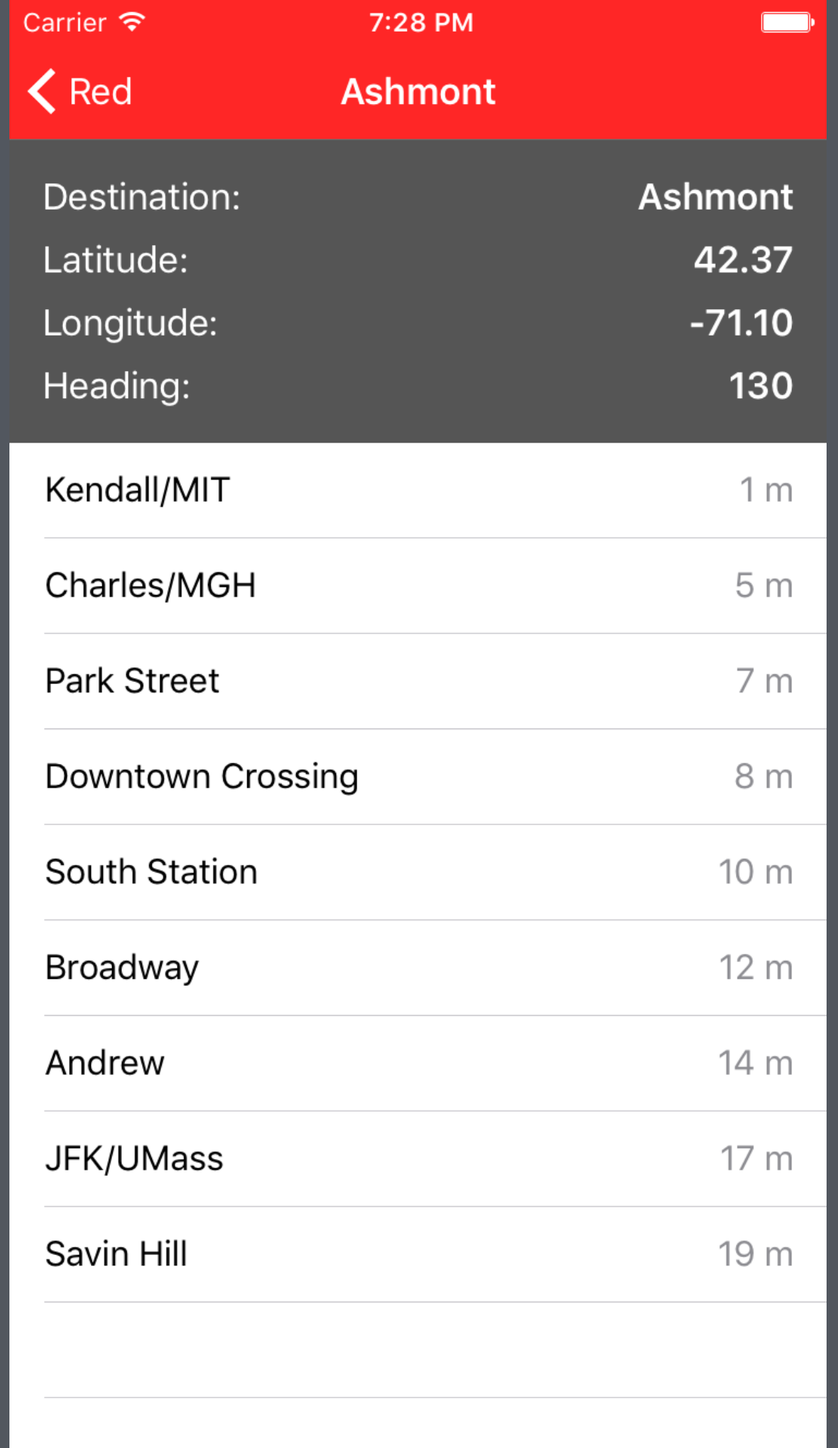
Layout Basics

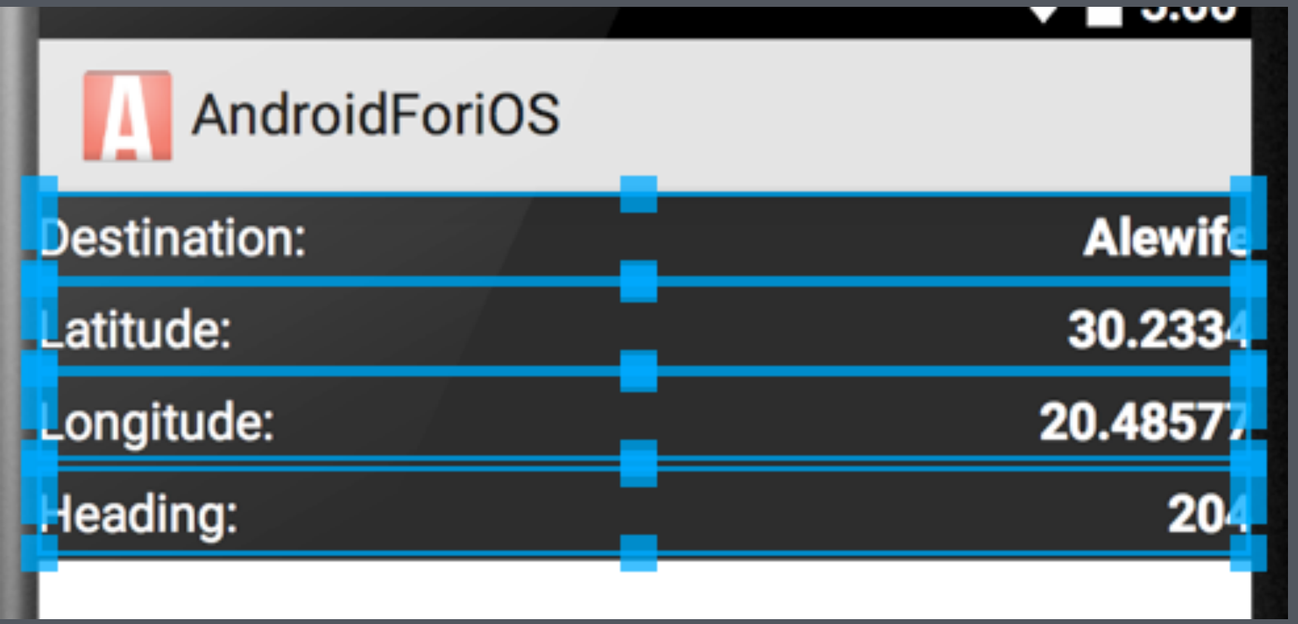
iOS is Heading More and More Towards Android's Layout Structure

Autolayout is about applying **constraints** to views that determine how they will layout.

- Interface Builder is your friend
  - Storyboards are powerful
- When building constraints, make sure you have enough to fully calculate the view
- Constraints can be modified and created in code as well


10:41
Ashmont
Destination: Ashmont
Latitude: 42.36553
Longitude: -71.10403
Heading: 130
Kendall/MIT 118
Charles/MGH 332
Park Street 446
Downtown Crossing 531
South Station 611


Carrier
7:28 PM
Red
Ashmont
Destination: Ashmont
Latitude: 42.37
Longitude: -71.10
Heading: 130
Kendall/MIT 1 m
Charles/MGH 5 m
Park Street 7 m
Downtown Crossing 8 m
South Station 10 m
Broadway 12 m
Andrew 14 m
JFK/UMass 17 m
Savin Hill 19 m

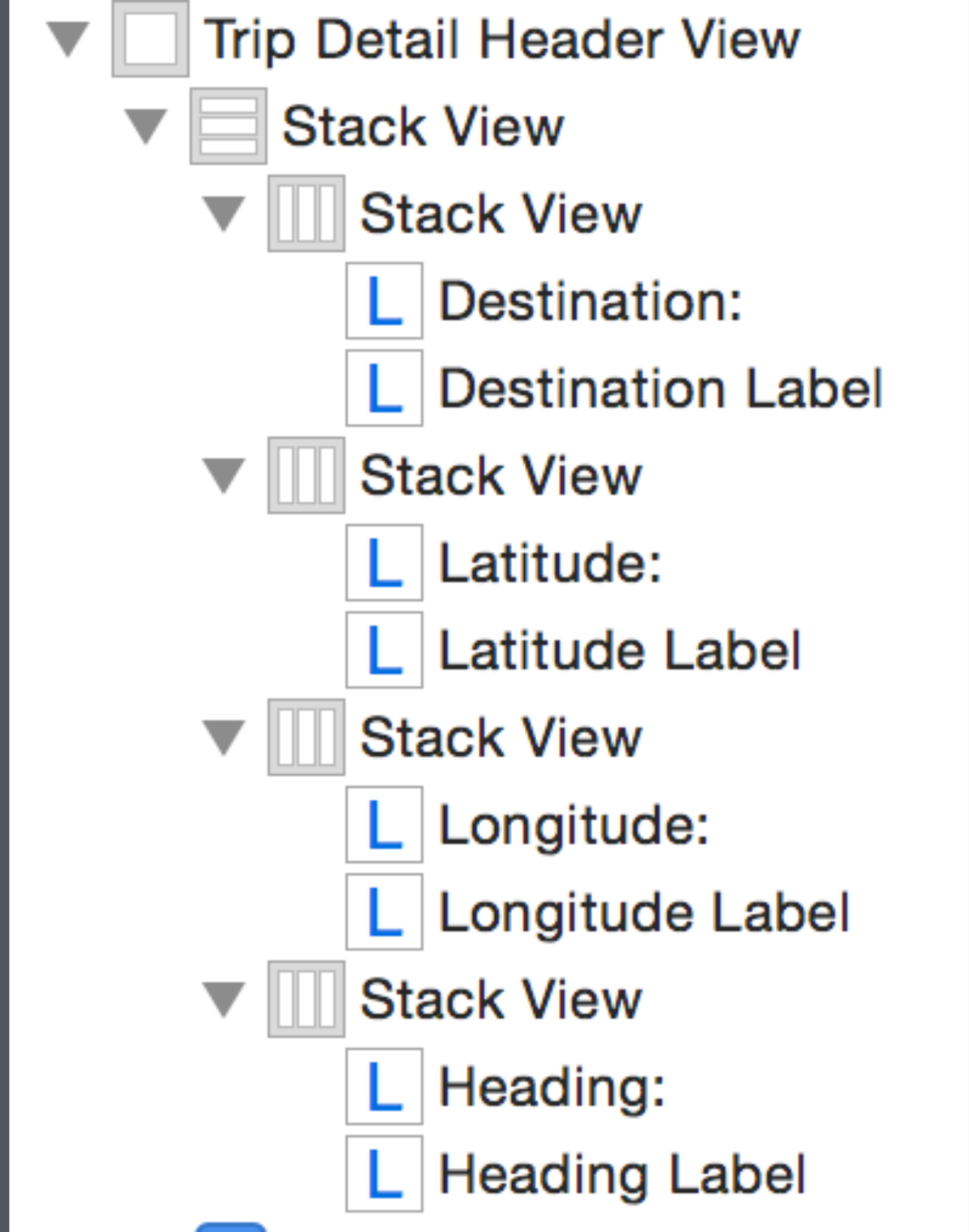


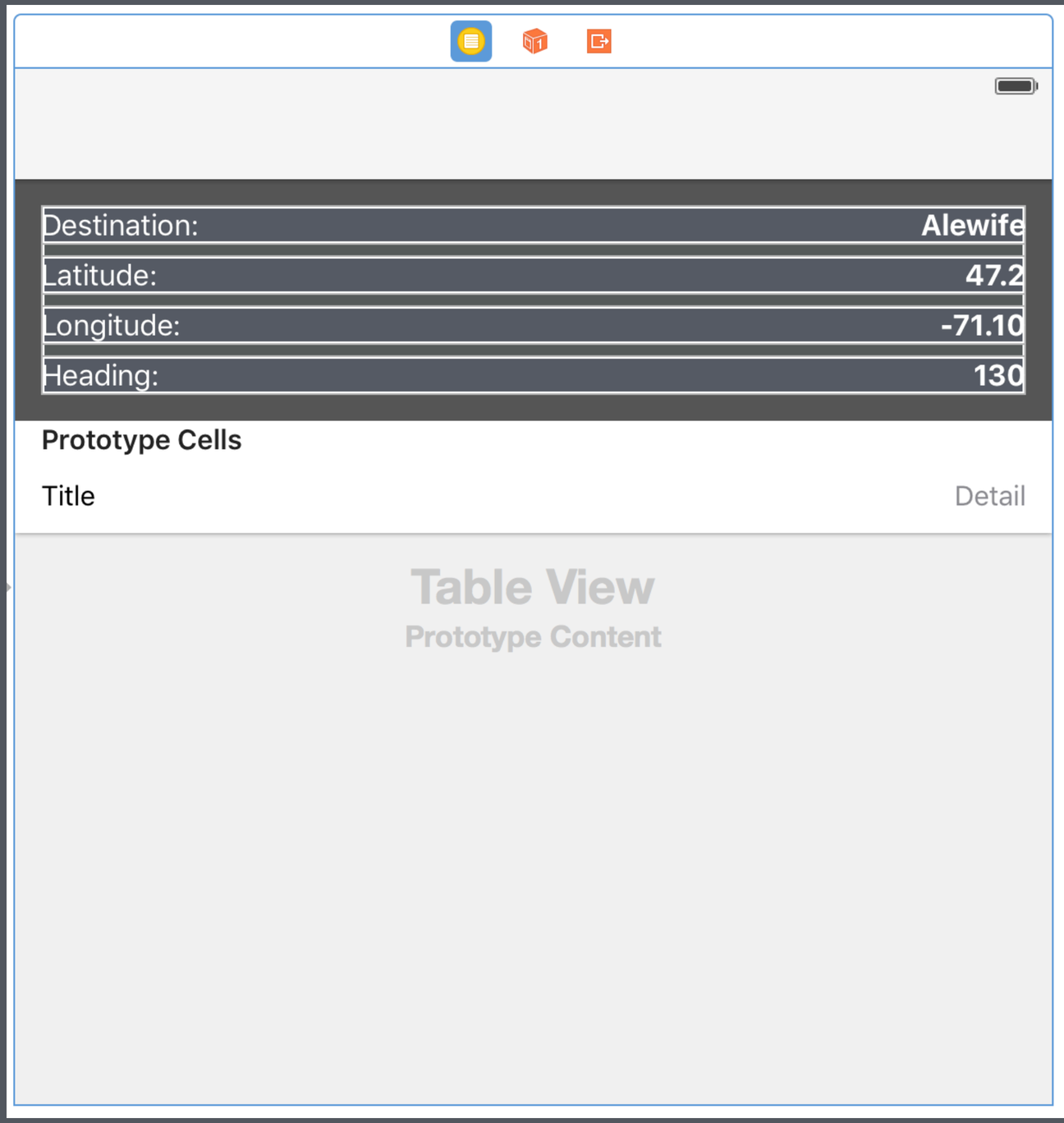

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#222222"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView />        <TextView />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <TextView />        <TextView />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView />        <TextView />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView />        <TextView />
    </LinearLayout>
</LinearLayout>
```

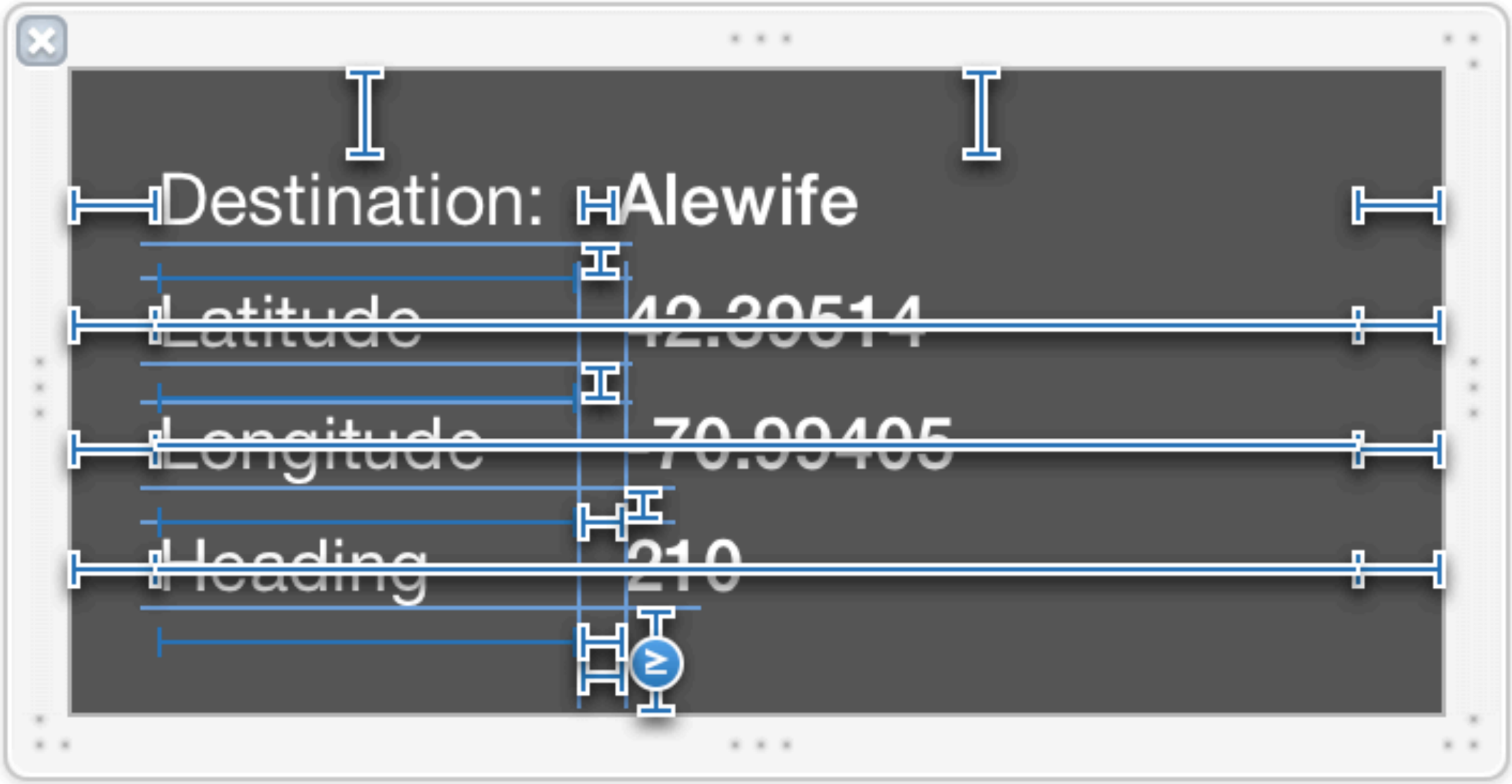
UIStackViews

iOS Linear Layouts

New in iOS 9


Trip Detail Header View
Stack View
Stack View
Destination:
Destination Label
Stack View
Latitude:
Latitude Label
Stack View
Longitude:
Longitude Label
Stack View
Heading:
Heading Label


Destination: Alewife
Latitude: 47.2
Longitude: -71.10
Heading: 130
Prototype Cells
Title
Detail
Table View
Prototype Content


Destination: Alewife
Latitude
Longitude
Heading
210

Data

Options

- NSUserDefaults <-> **Shared Preferences**
- In memory objects
- Saving to the documents directory <-> Saving to and fetching from file structure via the internal or external file storage
- **Core Data** <-> **SQLLite**

Core Data

- Model Layer Framework designed to be easy to use and reliability.
- Similar to Object-relational mapping (ORM), but it is not.
- Native solution to storing and managing model data for iOS.

Features:

- Direct ties to UI components such as `UITableView` through `NSFetchedResultsController`
- Grouping, filtering, and organizing data in memory and in the user interface.
- Relationship maintenance.
- Schema migration.
- And more...

Example: Fetching a Root User Object

```
let moc:NSManagedObjectContext = NSManagedObjectContext()
let defaults:NSUserDefaults = NSUserDefaults.standardUserDefaults()
let uri:NSURL? = defaults.URLForKey("rootItem")
let moid:NSManagedObjectID? =
  moc.persistentStoreCoordinator?.managedObjectIDForURIRepresentation(uri!)
let item:NSManagedObject = try moc.existingObjectWithID(moid!)
```

Example: Searching and Filtering Data

```
let moc:NSManagedObjectContext = NSManagedObjectContext()
let entityDescription:NSEntityDescription? = NSEntityDescription.
  entityForName("Employee", inManagedObjectContext: moc)
let request:NSFetchRequest = NSFetchRequest()
request.entity = entityDescription

// Set Search Predicate
let minimumSalary = 55
let predicate:NSPredicate = NSPredicate(format:
    "lastName LIKE[c] `Barnes` AND (salary > %@)", minimumSalary)
request.predicate = predicate
```

```swift
// Set Ordering Descriptor
let sortDescriptor:NSSortDescriptor = NSSortDescriptor(key: "firstName",
   ascending: true)
request.sortDescriptors = [sortDescriptor]

// Fetch Objects
do {
  let results:[AnyObject] = try moc.executeFetchRequest(request)
} catch {
  // Handle error
}
```

Popular Libraries

- **AFNetworking** <-> **Volley**
- **MagicalRecord** <-> **ActiveAndroid**
- **SDWebImage** <-> **Picasso**

Go Out and Build Something Cross Platform!

@smbarne - iOS 101 for Android Developers

Code: github.com/smbarne/AndroidForiOS