Mobile Development

The Good, the Bad, and the Ugly

DEVSTAFF
MEET UP

The hottest word of the last x > 5 years

MOBILE

(and the apps that make it great)

Always remember

- CPU is bad
- RAM is low
- Network is slow
- Battery is running low
- Users are anxious

Actually they are not that bad, but you need to be aware of limitations and follow some useful patterns for making complex stuff

1. CPUs are mainly ARM based designed for power efficiency
2. RAM varies from 512MB-3+GB, but OS doesn't allow mem allocation like in desktop
    - Android apps run in a VM (16 - 512MB, depends on screen size amongst others)
    - iOS (~80% of total mem per app)
3. Network speeds and latency vary a lot
    - Home network (@Greece: avg 8.93Mbps)
    - Public network (too many users / bad network devices)
    - Mobile Networks
        i. 2G (100-400Kbps, 300-1000ms Latency)
        ii. 3G (0.5 - 5Mbps, 100-500ms Latency)
        iii. 4G (1-50Mbps, <100ms Latency)
4. Most phones last 1-2 days with moderate usage (don't make it last less) Display - Radio - CPU all use a lot of power(!)
5. If nothing happens then it is dead (== don't block the main thread) - strive for 60fps and feedback to the user (Android ANR 5-10 secs)

Common parts*

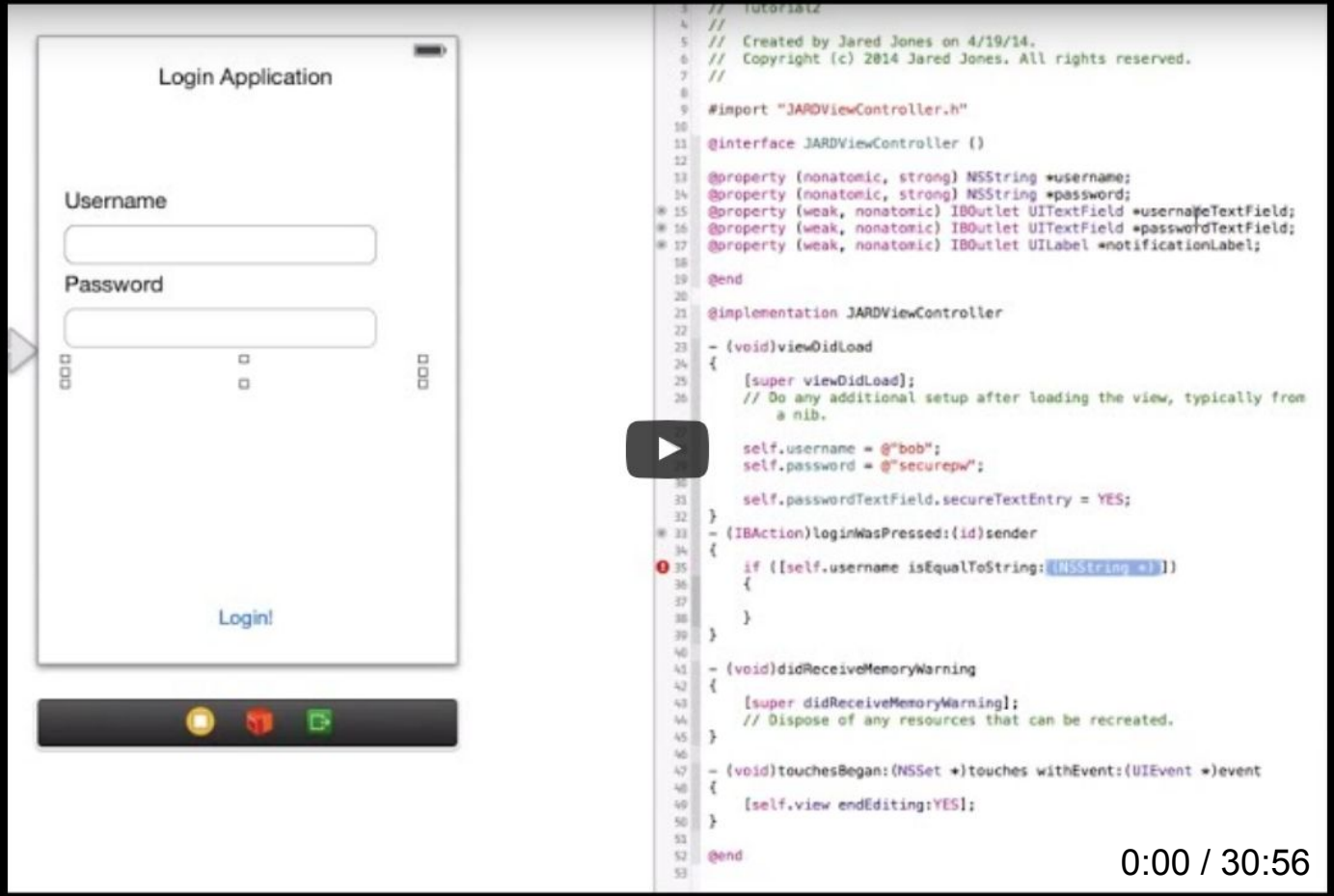
- Radios/Antennas
  - Wifi
  - Bluetooth
  - GPRS
  - GPS
- I/O
  - Touch screen (Small in size - Big in resolution)
  - Speaker(s) - 3.5mm headphones jack
  - Microphone(s)
  - Camera(s)
- Sensors
  - Accelerometer
  - Gyroscope
  - Proximity sensor
  - Ambient light sensor

* Not all parts available for all devices.
Also some devices might have extra parts.

All these sensors/radios/i/o devices packed in a single device, which users almost always carry around, can provide a lot of opportunities

- Location awareness (GPS, GPRS, Wifi, Bluetooth)
- Proximity awareness (Bluetooth, Microphone, Wifi, Bluetooth, Proximity Sensor, NFC)
- Ambient awareness (Microphone, Ambient light sensor, Bluetooth, *Wifi*)
- Activity awareness (Accelerometer, Gyroscope, Location awareness)

Objective-C Tutorial: Creating an iOS App for Absolute Beginners (Tut: 1)



0:00 / 30:56

Intro

- Xcode (Mac OS X only)
- Objective C || Swift
- Advanced Tools
- Quality Control

Languages

Objective-C

- Initially Used
- Extends C (=awesome)
- Object Oriented
- Most Frameworks, ...

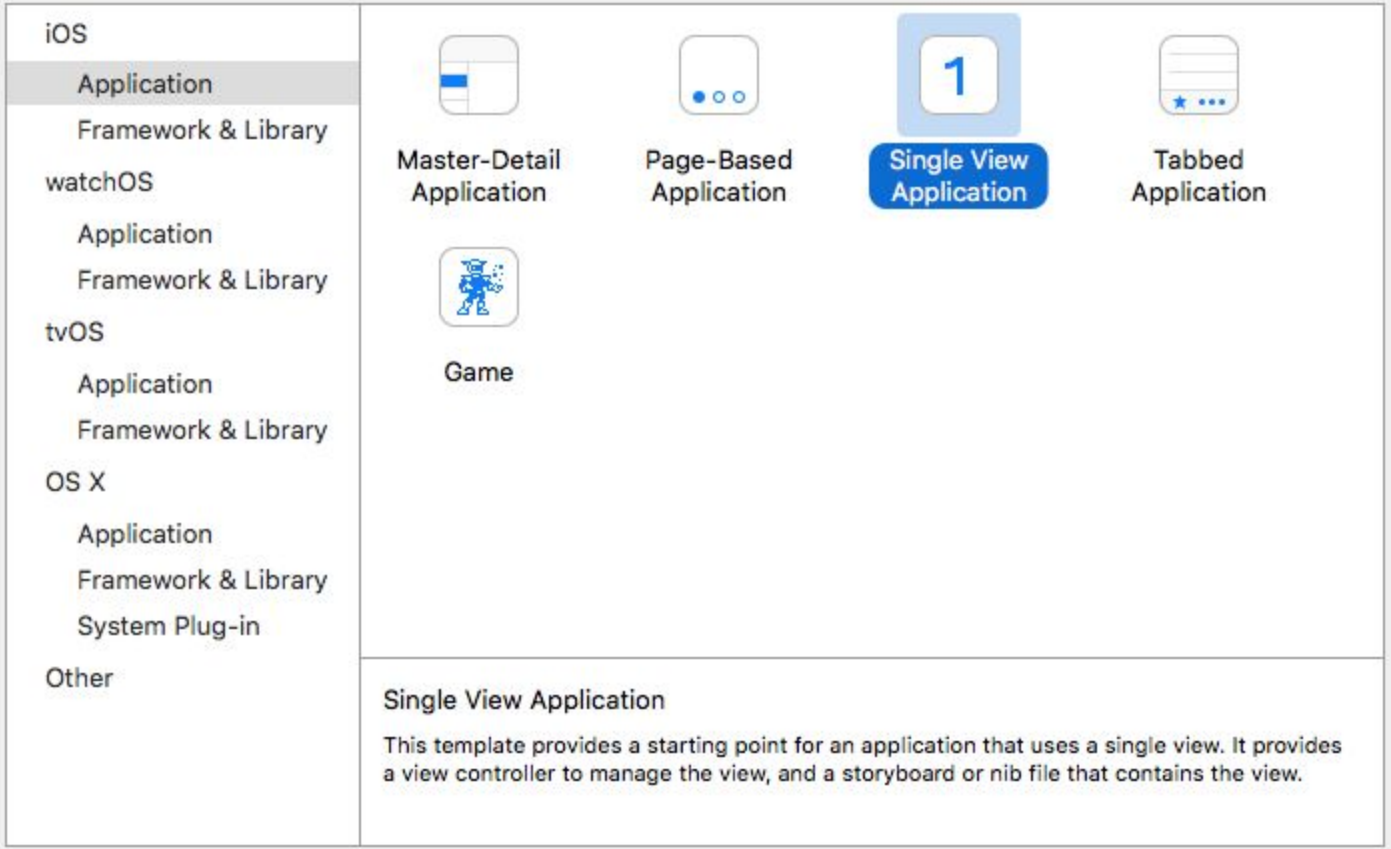
Swift

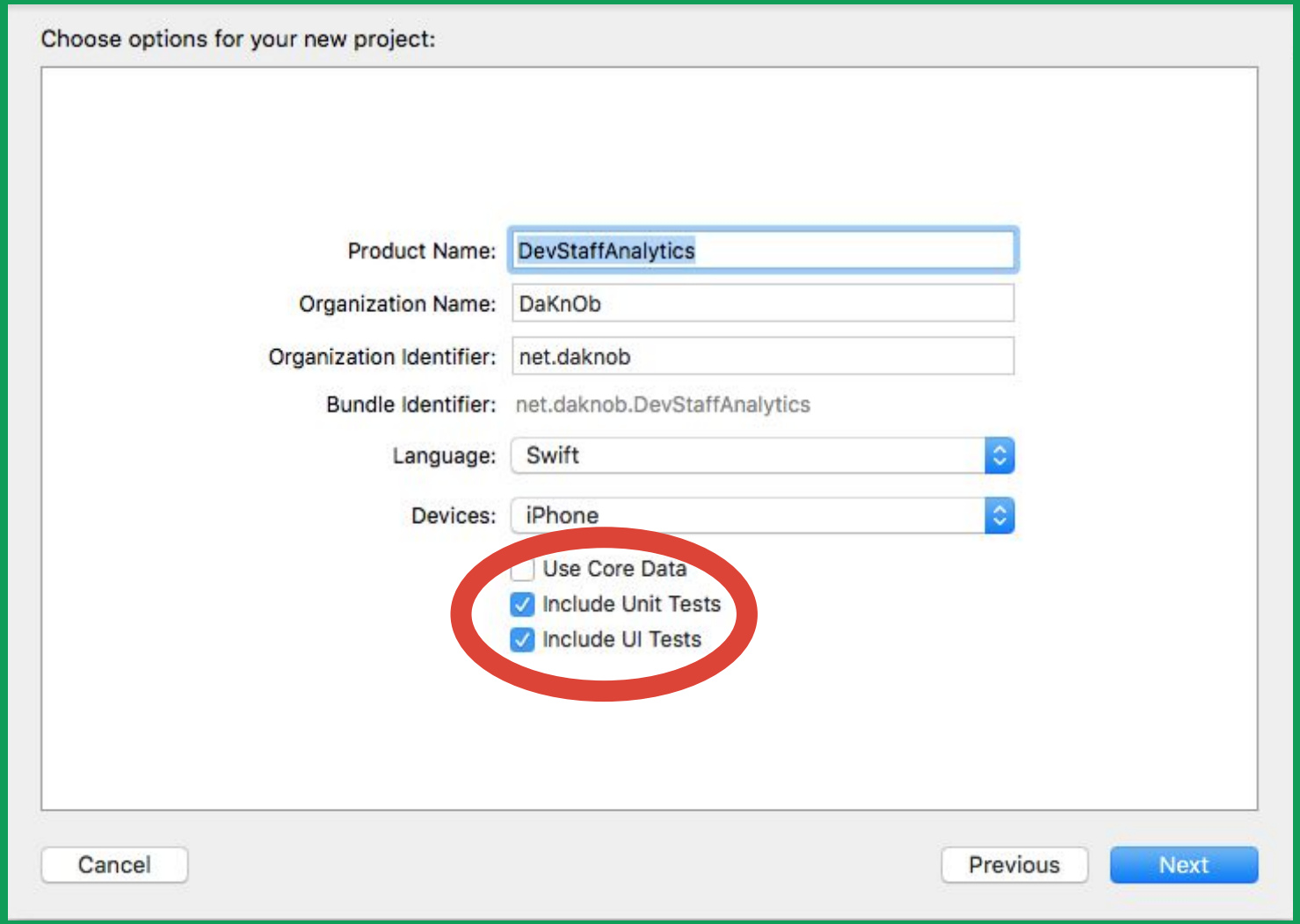
- Newly added
- Open Source on GitHub
- Object Oriented
- Interfaces perfectly w/ Obj C
- Named after Taylor Swift (?)

Tools

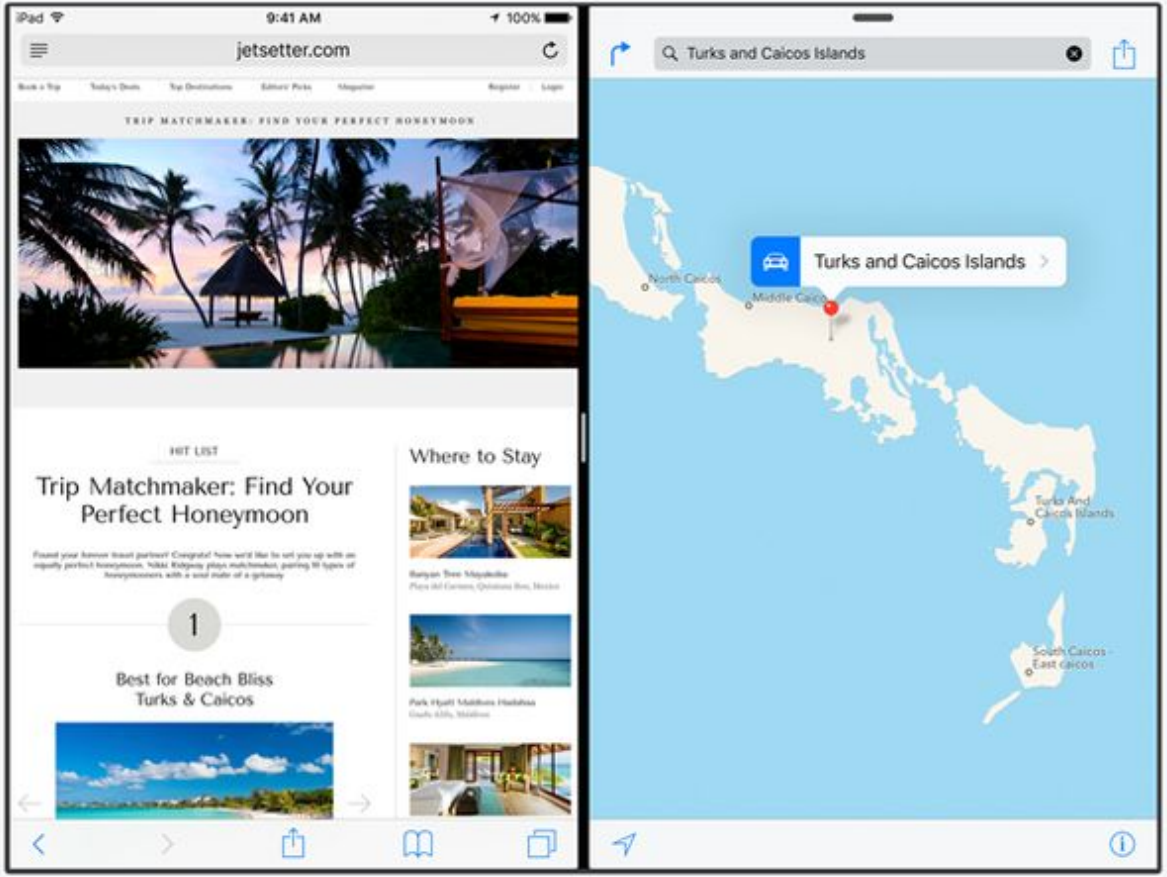

XCODE

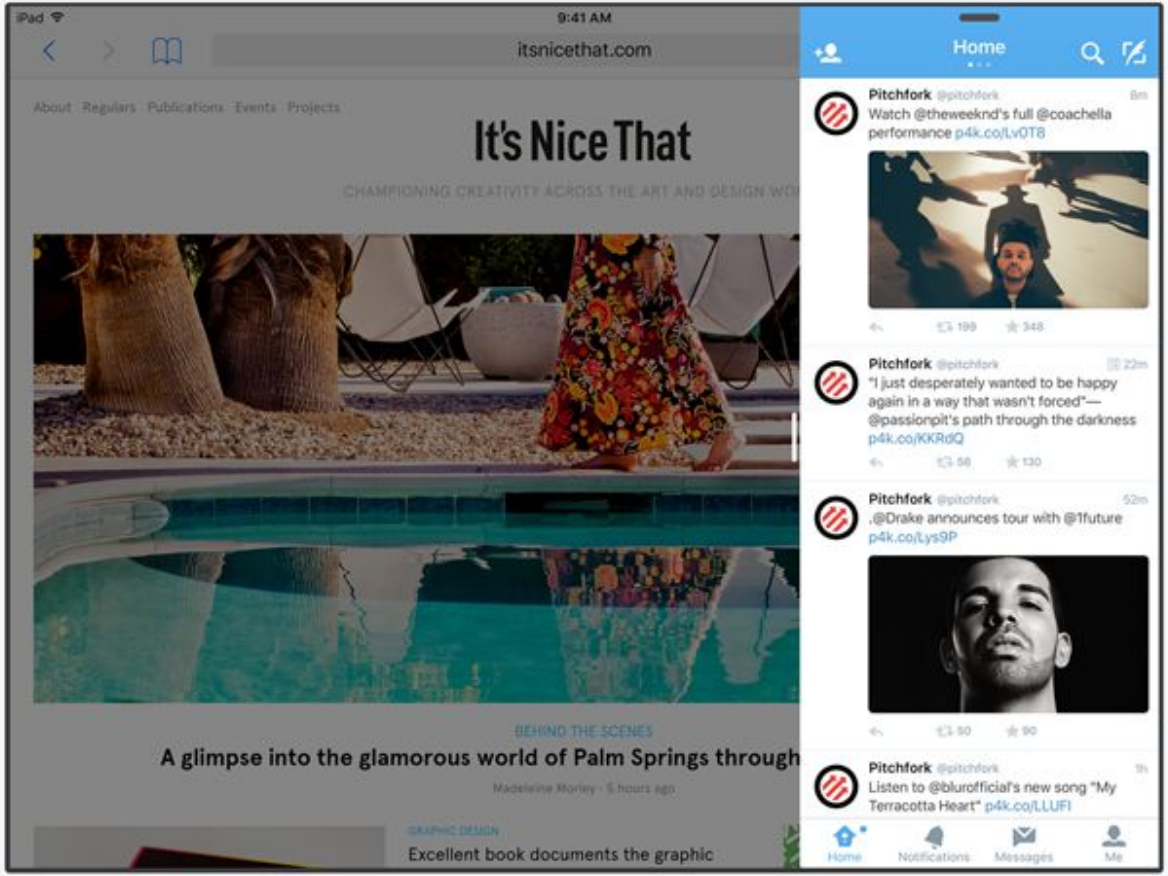
Choose a template for your new project:

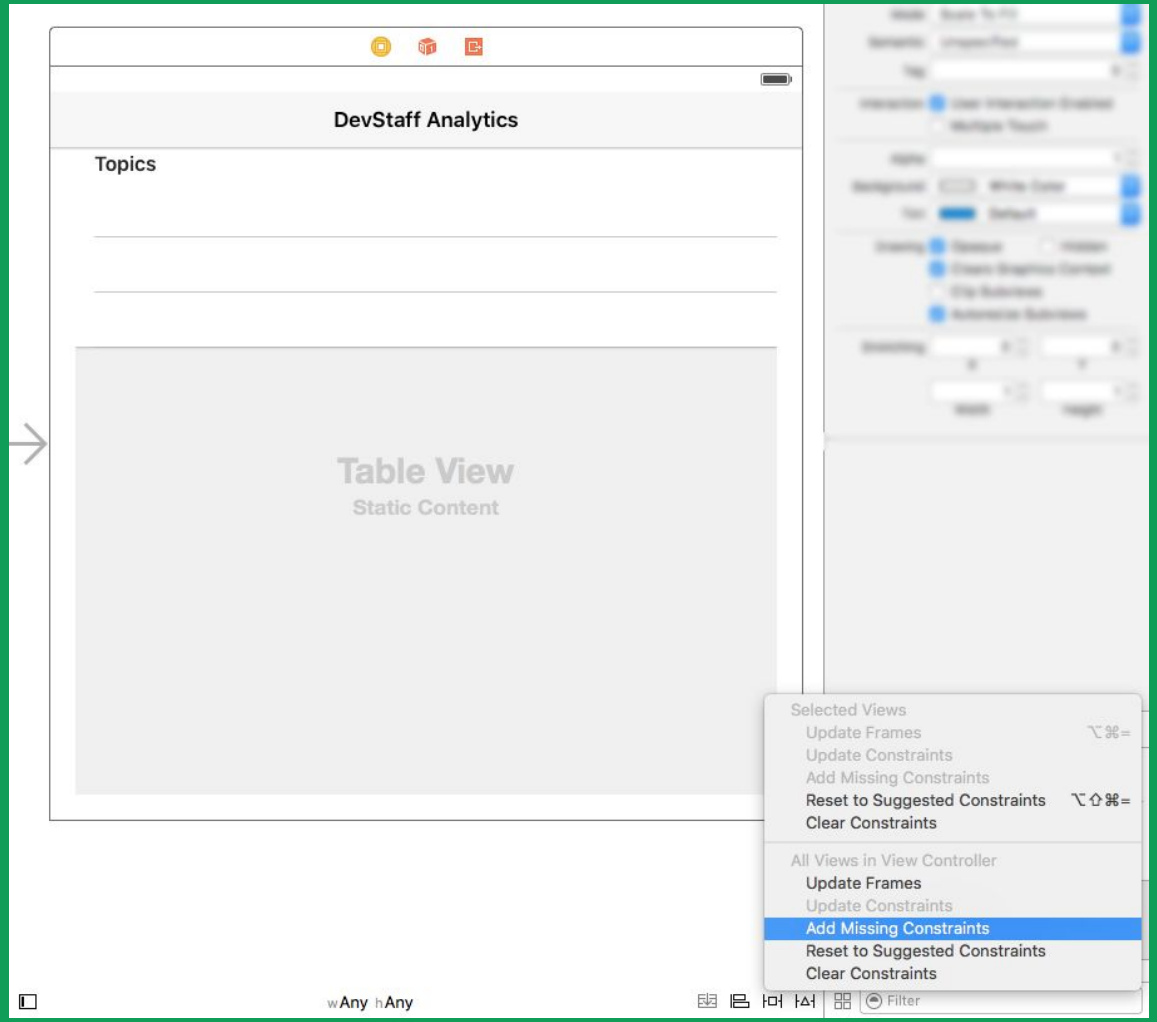


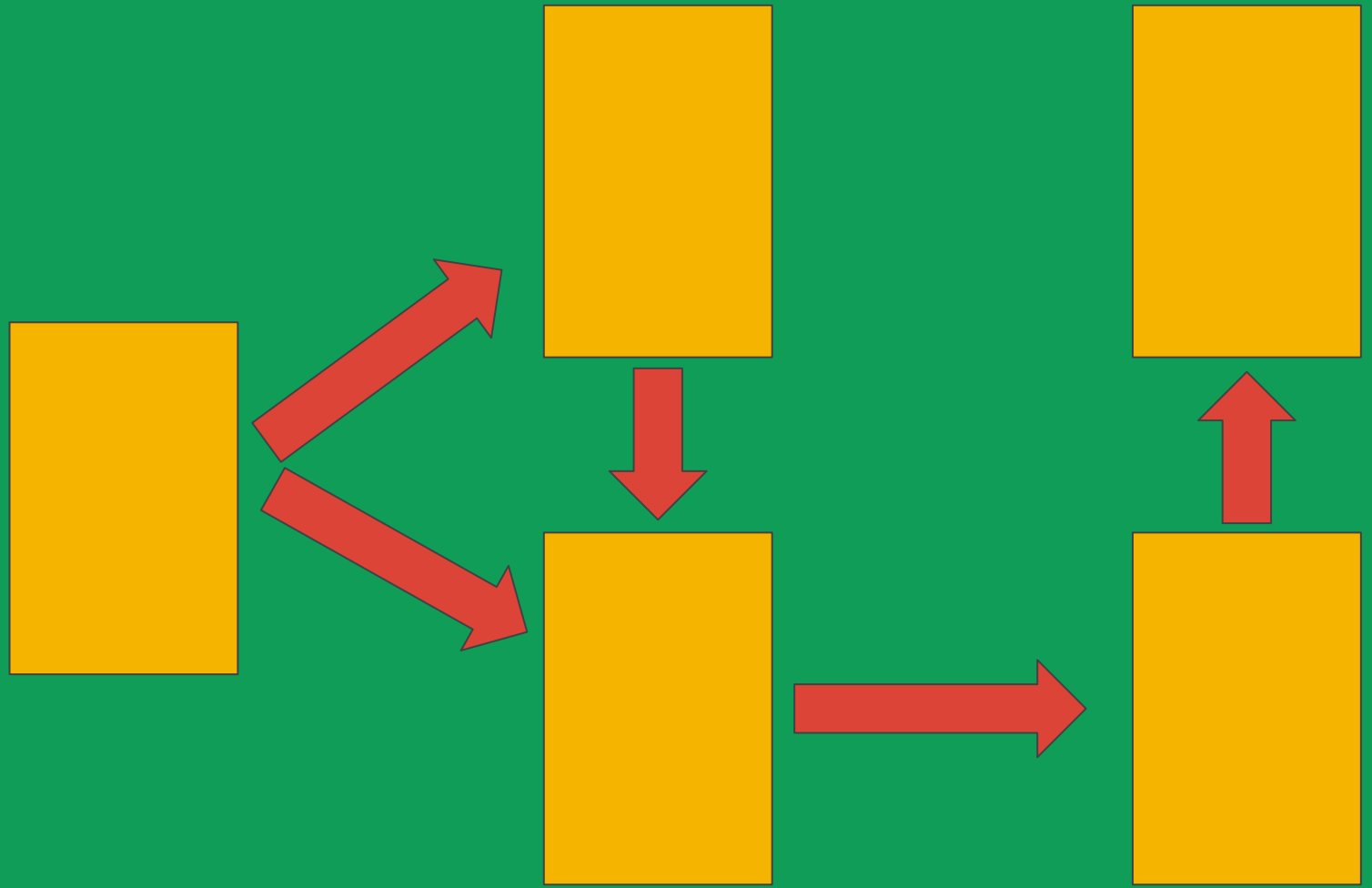

Choose options for your new project:
Product Name: DevStaffAnalytics
Organization Name: DaKnOb
Organization Identifier: net.daknob
Bundle Identifier: net.daknob.DevStaffAnalytics
Language: Swift
Devices: iPhone
Use Core Data
Include Unit Tests
Include UI Tests
Cancel
Previous
Next

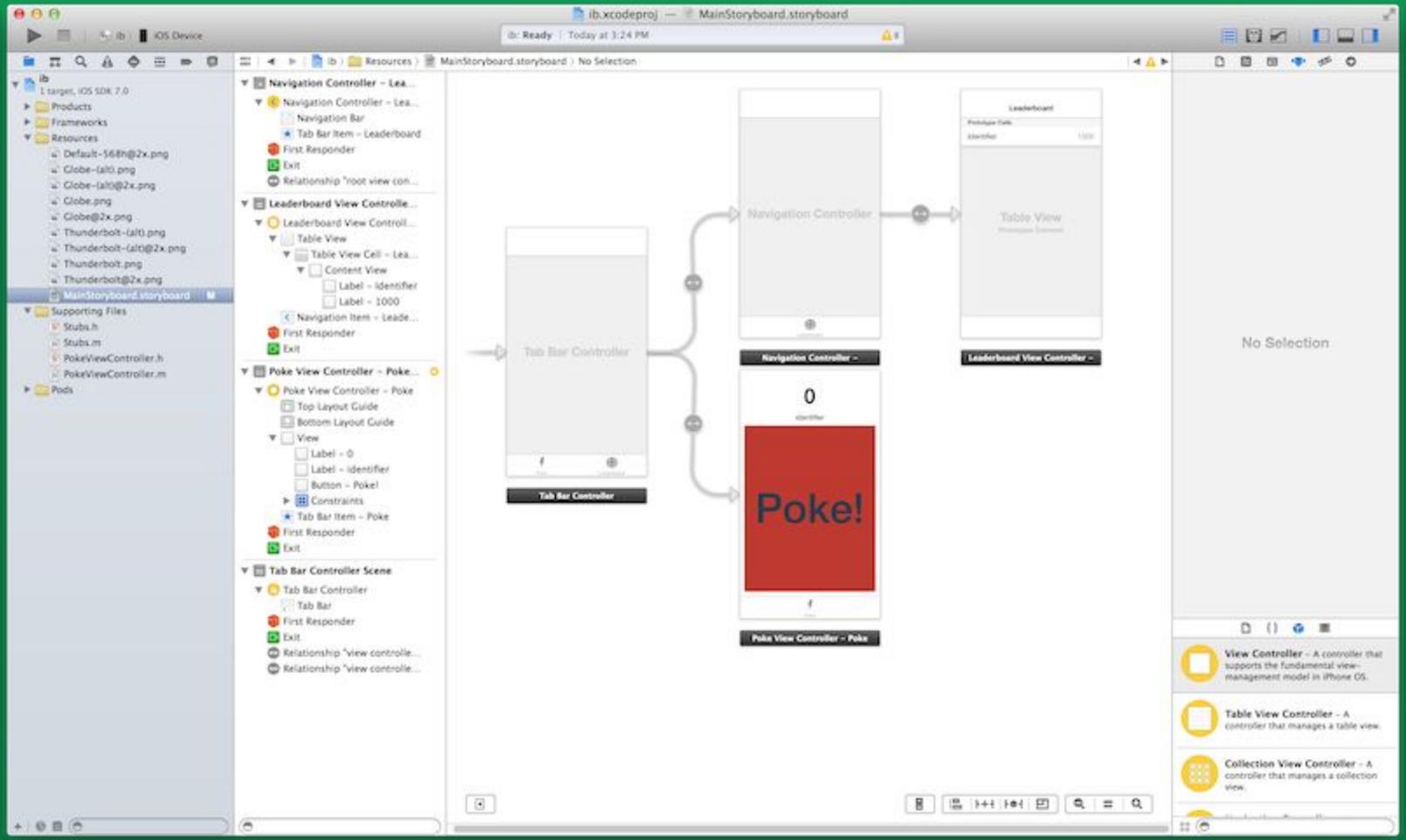

FaceTime
Calendario
Fotos
Cámara
Contactos
Reloj
Mapas
Videos
Notas
Game Center
Configuración
Música
Mensajes
Mail
Recordatorios
Photo Booth
Revistas
iTunes Store
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Notes
Reminders
Stocks
Game Center
Newsstand
iTunes Store
App Store
iBooks
Health
Passbook
Settings
Phone
Mail
Safari
Music
Wallet
News

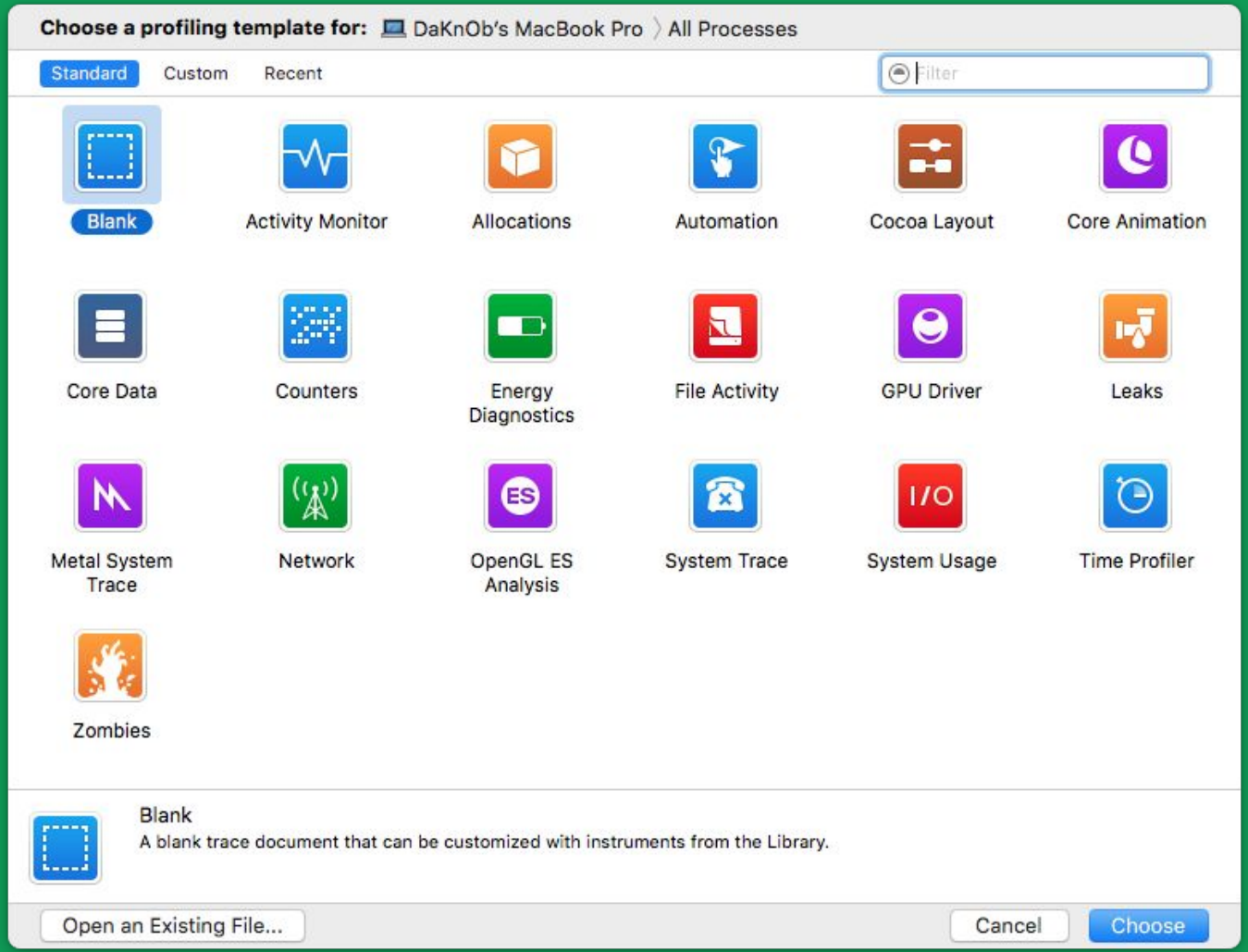

iPad
9:41 AM
100%
jetsetter.com
TRIP MATCHMAKER: FIND YOUR PERFECT HONEYMOON
HIT LIST
Trip Matchmaker: Find Your Perfect Honeymoon
1
Best for Beach Bliss
Turks & Caicos
Where to Stay
Turks and Caicos Islands
Turks and Caicos Islands
North Caicos
Middle Caicos
Turks And Caicos Islands
South Caicos
East Caicos


iPad
9:41 AM
itsnicethat.com
About Regulars Publications Events Projects
It's Nice That
BEHIND THE SCENES
A glimpse into the glamorous world of Palm Springs through
Excellent book documents the graphic
Home
Pitchfork @pitchfork 8m
Watch @theweeknd's full @coachella performance p4k.co/Lv0T8
199 348
Pitchfork @pitchfork 22m
"I just desperately wanted to be happy again in a way that wasn't forced"— @passionpit's path through the darkness p4k.co/KKRdQ
56 130
Pitchfork @pitchfork 52m
.@Drake announces tour with @1future p4k.co/Lys9P
50 90
Pitchfork @pitchfork 1h
Listen to @blurofficial's new song "My Terracotta Heart" p4k.co/LLUFI
Home
Notifications
Messages
Me


DevStaff Analytics
Topics
Table View
Static Content
Selected Views
Update Frames
Update Constraints
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints
All Views in View Controller
Update Frames
Update Constraints
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints
wAny hAny
Filter




ib.xcodeproj — MainStoryboard.storyboard
Navigation Controller – Lea...
Navigation Controller – Lea...
Navigation Bar
Tab Bar Item – Leaderboard
First Responder
Exit
Relationship "root view con...
Leaderboard View Controlle...
Leaderboard View Controll...
Table View
Table View Cell – Lea...
Content View
Label – Identifier
Label – 1000
Navigation Item – Leade...
First Responder
Exit
Poke View Controller – Poke...
Poke View Controller – Poke
Top Layout Guide
Bottom Layout Guide
View
Label – 0
Label – Identifier
Button – Poke!
Constraints
Tab Bar Item – Poke
First Responder
Exit
Tab Bar Controller Scene
Tab Bar Controller
Tab Bar
First Responder
Exit
Relationship "view controlle...
Relationship "view controlle...
Tab Bar Controller
Navigation Controller
Table View
Navigation Controller –
Leaderboard View Controller –
0
Poke!
Poke View Controller – Poke
No Selection
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.


Choose a profiling template for: DaKnOb's MacBook Pro 〉All Processes
Standard
Custom
Recent
Filter
Blank
Activity Monitor
Allocations
Automation
Cocoa Layout
Core Animation
Core Data
Counters
Energy Diagnostics
File Activity
GPU Driver
Leaks
Metal System Trace
Network
OpenGL ES Analysis
System Trace
System Usage
Time Profiler
Zombies
Blank
A blank trace document that can be customized with instruments from the Library.
Open an Existing File...
Cancel
Choose

Frameworks

iOS Frameworks

- Over 100 System Frameworks
- Do (almost) anything allowed
- Well Documented w/ Examples
- Many Available Public Methods

Some Examples

Accelerometer & Gyroscope
Social Accounts (Twitter, fb)
iAd (Drag & Drop Ads)
Access Images / Videos
Play & Transform Audio
Calendar Events
Camera Access
iCloud and iCloud Drive
View and Modify Contacts
Access User Location
Interface w/ Siri and Search

External Accessories
Games (incl. Multiplayer)
Open{A,C,G}L
Health Information
Smart Home Integration
JavaScript (for Browsers)
Maps
Data Protection & Authentication
Metal
Music
(Much more)

Distribution

Distribute iOS Apps

- App Store
- App Store
- Enterprise Deployment
- App Store

App Store

- Apple Developer Program ($99/yr)
- Review by humans (1-20 days)
- 30% of App Price to Apple
- Free Apps bare no cost at all

How to send apps to App Store

- Complicated PKI Stuff
- Upload Signed Binary
- Wait for Tests / Review by Apple
- Publish App

Control over App Store Apps

- Select Only Specific Countries
- Change Price
- Offer Discount Bundles
- Delete App (Red Switch)

Why native?

- ➔ Performance
- ➔ Native look and feel
- ➔ Better support for interaction with peripherals/sensors, OS capabilities
- ➔ Solid languages (Java / C -better performance/best for games-)
- ➔ Excellent tools
- ➔ Excellent SDK and 3rd party libraries
- ➔ Excellent compatibility libraries (don’t mind about fragmentation)

Tools

- Android Studio (Open source version of IntelliJ with extras)
  - Full featured IDE that can do most of what is required by your project, including integration with tools listed below
  - Code templates to help you build common app features
  - Rich layout editor with support for drag and drop theme editing (XML editing is also wonderful)
  - Build in support for Google Cloud Platform
- Gradle
  - Build + Dependency management system
    - Build variants and multiple apk files generation ROCKS
- Lint
  - Static code/files/structure analysis
- Proguard (Optional)
  - Obfuscation and minification of code
- Android Emulator
  - Run your app in every device (* check out v2.0)

Basic Concepts (1/2)

- Activities

An Activity is an application component that provides a screen with which users can interact in order to do something. Each activity is given a window in which to draw its user interface. The window typically fills the screen, *but may be smaller than the screen and float on top of other windows.*

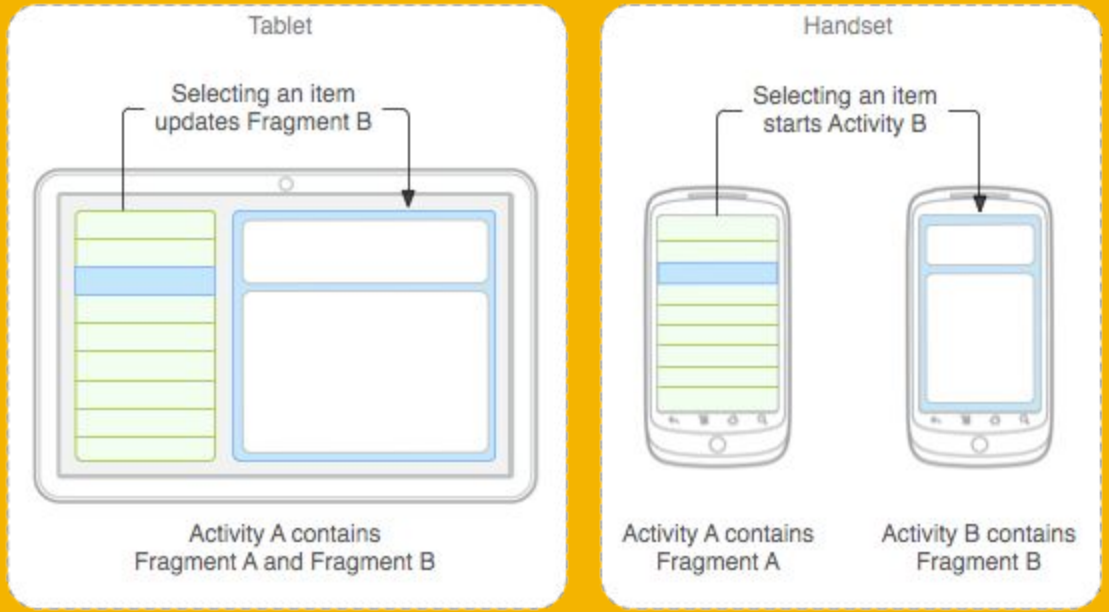
- Fragments

A Fragment represents a behavior or a portion of user interface in an Activity.
A fragment must always be embedded in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.
When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy and the fragment defines its own view layout.
However, a fragment is not required to be a part of the activity layout; you may also use a fragment without its own UI as an invisible worker for the activity.

- Services

A Service is an application component that can perform long-running operations in the background. It continues to run even if user switches application, possible even when on low memory. Can be used for Interprocess Communication (IPC)

Activities and fragments



Basic Concepts (2/2)

- Broadcast receivers

A Broadcast receiver is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Can also be generated from the system. No UI but can display notifications or start an activity or a service.

- Intents

An Intent is a messaging object you can use to request an action from another app component
Used for: Starting an Activity, Service, or sending a Broadcast

- Content providers

Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the standard interface that connects data in one process with code running in another process. e.g. Calendar Provider, Contacts Provider

- Views & ViewGroups

View obviously contains buttons, textviews, edittexts etc. ViewGroup (e.g. ListView, ScrollView, LinearLayout etc) contain Views and Fragments and other ViewGroups

App + lifecycle

AndroidManifest.xml

Appears exactly once at the root of the project. Contains information used by the OS, like: app's package name (unique per app), permissions, activities, broadcast receivers, content providers etc.

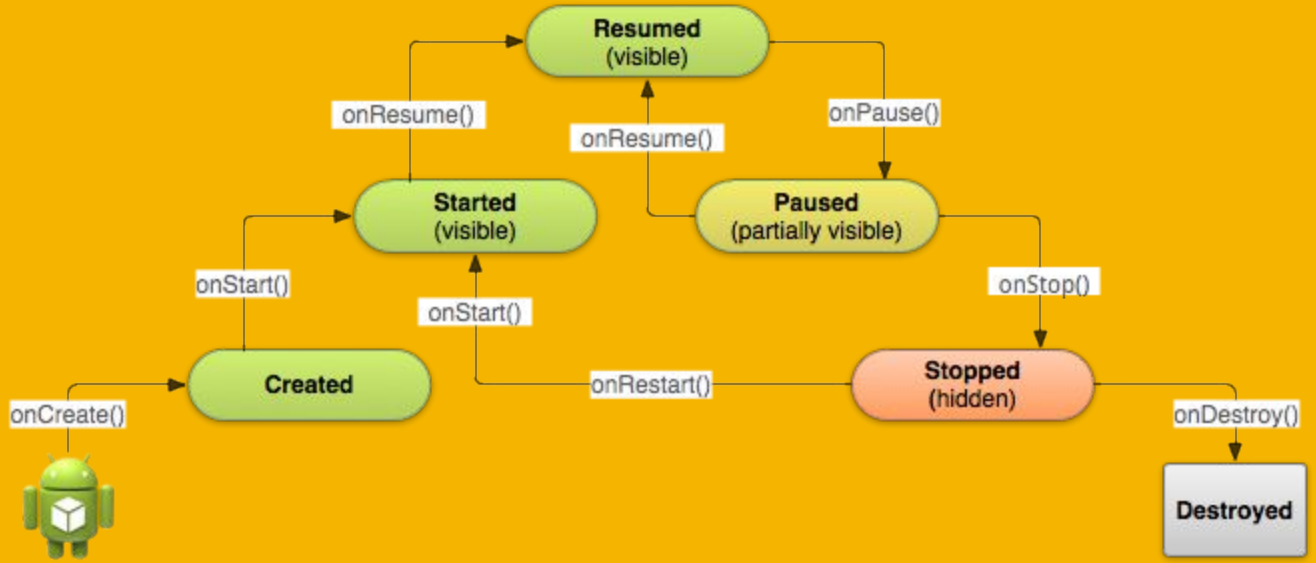
Multiple entry points, Activities, Services and Broadcast Receivers can give ways to access your application.

This is optional, if you have an activity and want to be visible to the launcher(s)

```
<activity android:name=".MainActivity" android:label="@string/app_name">
     <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
</activity>
```

Lifecycle

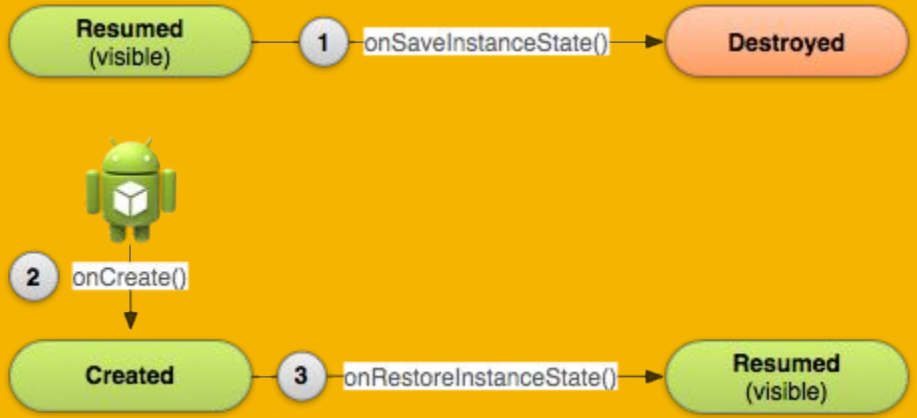
Activity Lifecycle



Lifecycle and state

Activity can be destroyed and re-created in many ways, e.g device rotation, some other configuration change, low memory conditions (when not visible)
Activities within the same app can be destroyed while another activity is visible!

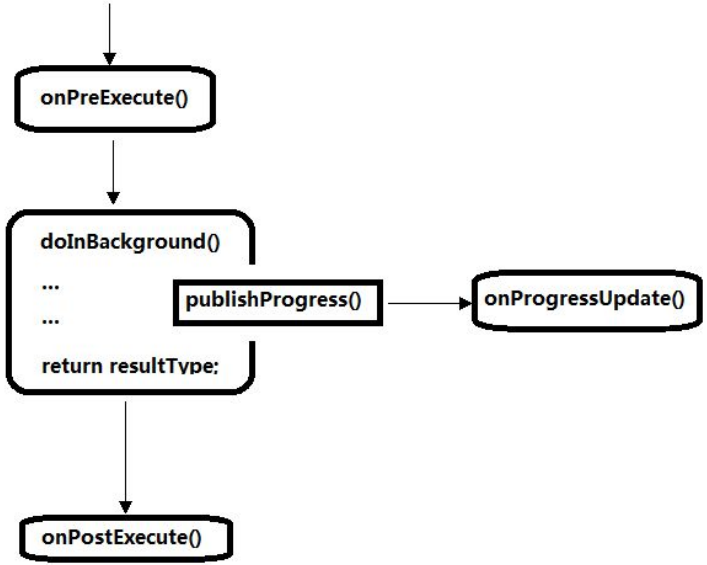
Bundle to the rescue, passed in onCreate, onSaveInstanceState and onRestoreInstanceState.
Built-in Views and well built custom views can do things automatically if an id is assigned to them.



Bundle is a key-value<String, Parcelable> store where you can store primitives, CharSequencies, Parcelables and Serializables

Threads

Application runs on a *single process* and most of the time one thread is used, so everything that does I/O or a heavy job needs to be done in a separate thread to avoid issues with ANR and UI rendering.



Loaders and AsyncTasks must be used in such cases. They facilitate into making heavy tasks and I/O into a separate thread and that callbacks are made in the main thread.

For longer running tasks a Service that then starts a new thread is preferable, or a remote Service.

A whole lot of different devices

Due to the large variety of screen sizes and densities px, in, mm are of no good use.
What to use: dp - Density-Independent Pixels and sp - Scale-Independent Pixels (for font sizes)

1dp == 1px on a 160 dpi (Dots per inch) Screen - mdpi

ldpi - 0.75x
mdpi - 1.0x
hdpi - 1.5x
xhdpi - 2.0x
xxhdpi - 3.0x
xxxhdpi - 4.0x

Android allows setting resource (images, strings, numbers, colors, etc) based on qualifiers with a best match approach if nothing fits perfectly

- MCC & MNC
- Language and region
- Layout direction
- Smallest width
- Available width
- Available height
- Screen size
- Aspect ratio
- Round screen
- Screen orientation
- UI mode
- Night mode
- Screen pixel density (dpi)
- Touchscreen type
- Keyboard availability
- Primary text input method
- Navigation key availability
- Platform Version (API level)
- Primary non-touch navigation method

Reading material

http://developer.android.com/guide/index.html

Cross Platform / Hybrid Dev

Thanks!

Contact us:

Antonios Chariton
<daknob@daknob.net>

George Peponakis
<greenman18523@gmail.com>
<peponakis@cytech.gr>

Manolis Kritsotakis
<kritsot@gmail.com>