

CS2049

- Make sure to have:
 - Xcode 7.2 on your Mac
 - iOS 9 and a device with you
 - USB cable to connect to device
 - AppleID setup so that you can run code on device

Lecture 1

CS2049: Intermediate iPhone Development

Instructor: Daniel Hauagge

Instructor

- Daniel Hauagge (daniel.hauagge@cornell.edu)
- Runway PostDoc@CornellTech
- Founder of PeachyLabs: building systems that recognize food in images
- Background: CS PhD@Cornell
- Research Area: Computer Vision

Staff

- Guandao Yang (gy46)
- Zheng Fu (James) (zf38)
- Office hours/Lab sessions
 - Every other Saturday, same place and time as lecture
 - *Except* 1st lab session!

Course Description

- Format: every class build an App, with the instructor, from scratch
- Focus is on rapid prototyping, not fundamentals
 - Learn on the fly

Course Description

- 1 credit, pass/fail
- 3 hour lectures every other week
- lab session every other week
 - Short lecture on topics related dev tools
- Not mandatory

Course Description

www.cs.cornell.edu/courses/cs2049/2016sp/

- Announcements & Schedule: webpage
- Questions? Piazza
- Handing in HW and Grades: CMS

Course Description

- Homework after each class
 - Extend app built in class
 - 2 weeks to finish, grade is pass/fail
- Final project
 - Your choice (with some requirements)
 - Optional: present to the rest of the class at the end of the semester

Course Description

- Tools: Swift, Xcode 7.2, iOS 9
 - Students should have access to a Mac and an iOS device at class and for homework
- Requirements:
 - CS2048
 - OR
 - Basic understanding of Xcode + ObjC or Swift

Swift

Why Swift?

- It's the future
- Cleaner than ObjC
 - Goodbye @ and [] madness
 - No more header files
 - Proper namespaces (DCHMyClass -> DCH.MyClass)
- Many of the high level concepts from ObjC map nicely to Swift: MVC, delegates, extensions, protocols, etc.
- Fast
- Type safe

Why Swift?

- Avoids common errors in ObjC
 - Stricter about pointers
- Open source
- Plays well with ObjC
 - Call ObjC from Swift
 - Call Swift from ObjC

Why Swift?

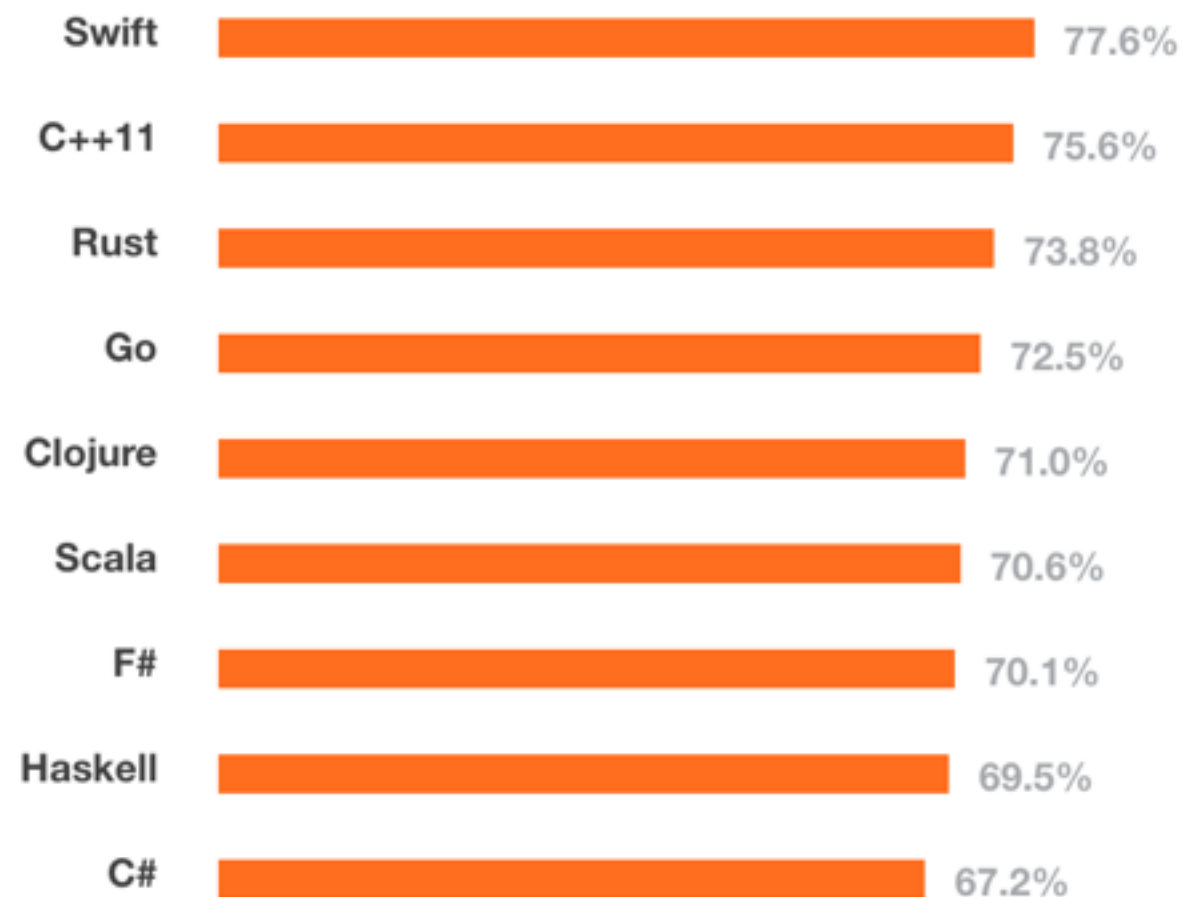
- Playgrounds
- Generics

2015 Developer Survey

Most Loved

Most Dreaded


Most Wanted

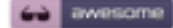


github.com/matteocrippa/awesome-swift


README.md

Awesome Swift



 awesome

A curated list of awesome Swift frameworks, libraries, and software for iOS / OSX / tvOS / watchOS and Linux.

let  = Linux.Ready

Contributing

Please take a quick look at the [contribution guidelines](#) first. If you see a package or project here that is no longer maintained or is not a good fit, please submit a pull request to improve this file. Thank you to all [contributors](#); you rock!

Contents

- [Demo Apps](#)
 - [iOS](#)
 - [Apple Watch](#)
 - [OS X](#)
- [Dependency Managers](#)
- [Guides](#)
- [Patterns](#)
- [Editor Support](#)
 - [Emacs](#)
 - [Vim](#)
- [Libs](#)
 - [Animation](#)
 - [App Store](#)
 - [Audio](#)
 - [API](#)
 - [Bluetooth](#)
 - [Chat](#)
 - [Colors](#)
 - [Command Line](#)
 - [Concurrency](#)
 - [Data Management](#)
 - [Core Data](#)

Topics

	Lecture	Lab
1	CoreMotion, AutoLayout, Segue, StackViews	Swift and Playgrounds
2	AVFoundation	TBD
3	Persistence with Realm, CocoaPods	TBD
4	SpriteKit	Demos of Final Projects

Today's Class



Drawing with the Accelerometer

CoreMotion

AutoLayout

StackViews (maybe)

Core Motion

- Gives you access to device sensors:
 - Accelerometer
 - Gyroscope
 - Magnetometer
 - Altimeter (pressure, relative altitude)
 - ~~GPS~~ → CoreLocation

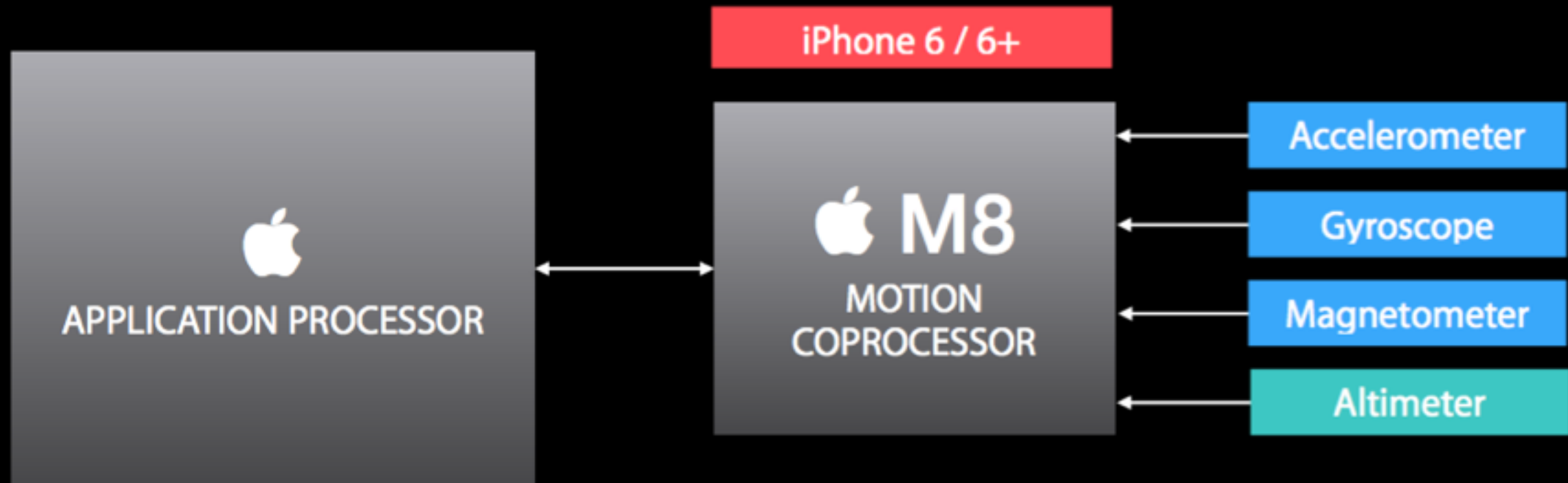
Core Motion

- Pre-processed data:
 - Acceleration - gravity
- Virtual instruments:
 - Pedometer (# of steps, distance, floors ascended and descended, pace, cadence): Uses a combination of accelerometer and GPS data

CoreMotion

- Allows for live updates
- Or queries to past data

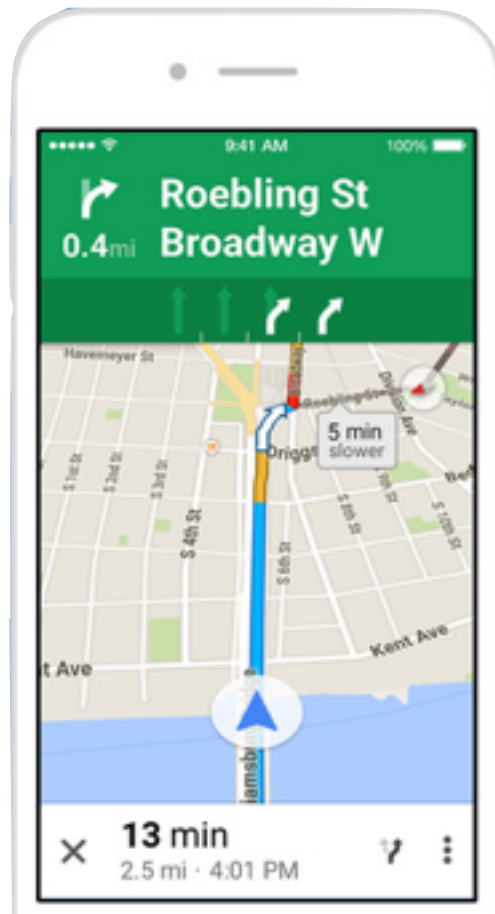
CoreMotion



Applications



Passive Activity Trackers



Maps app

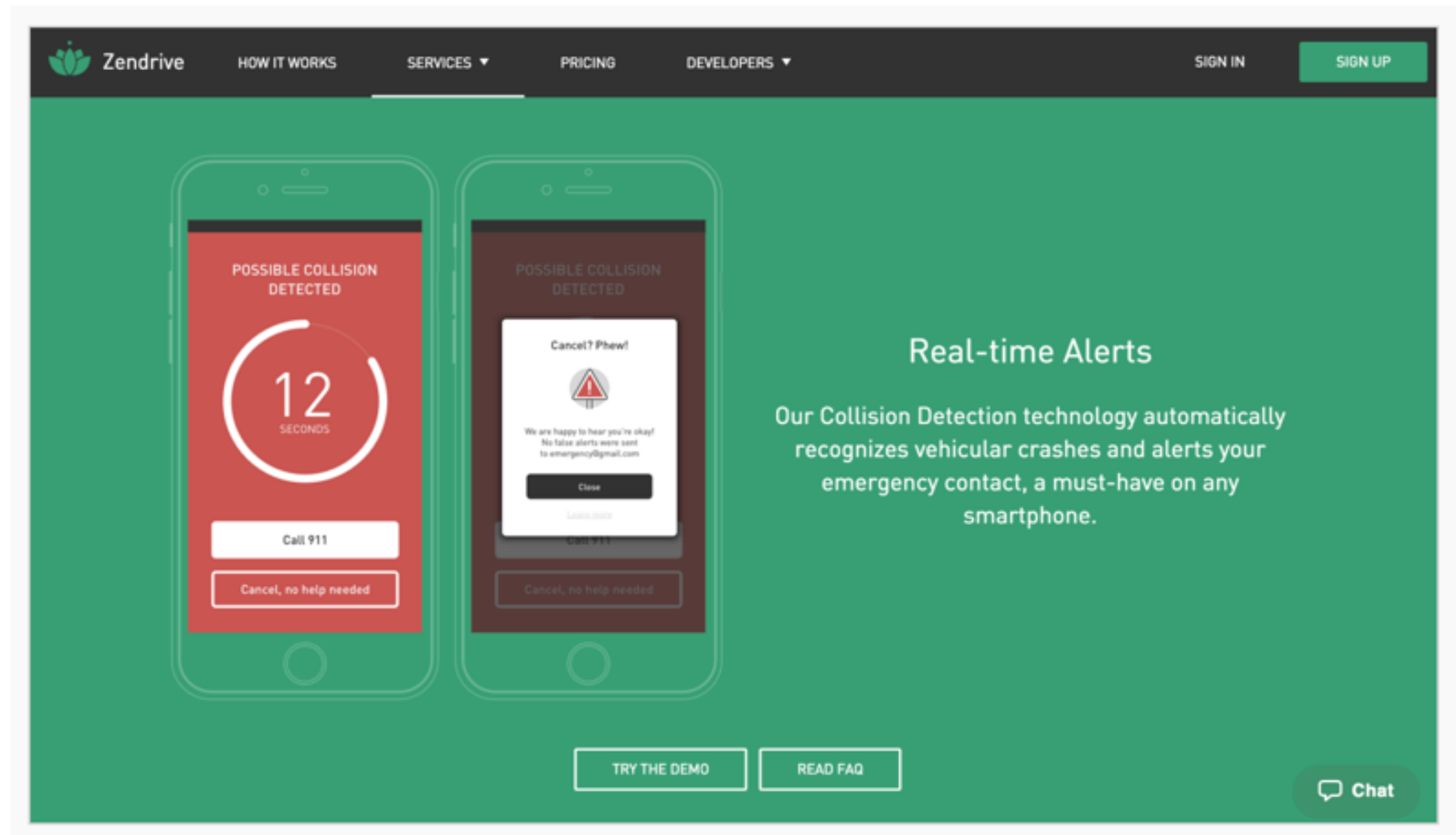


Tools



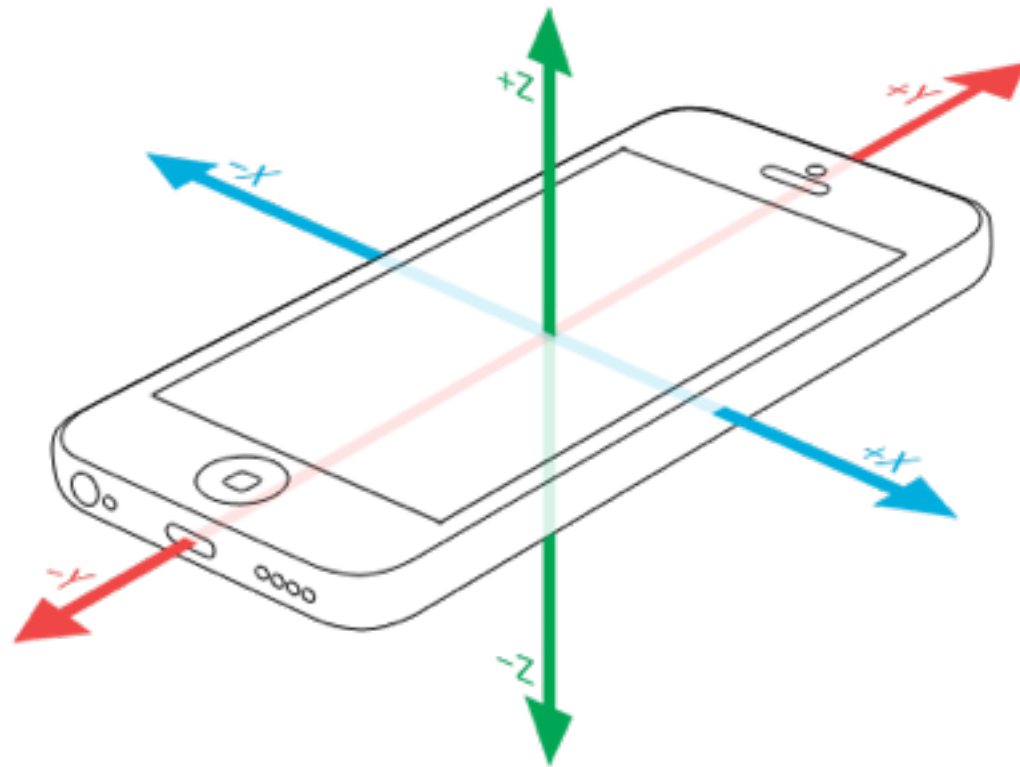
VR & Games

Applications



Coordinate Systems

Accelerometer



Drawing

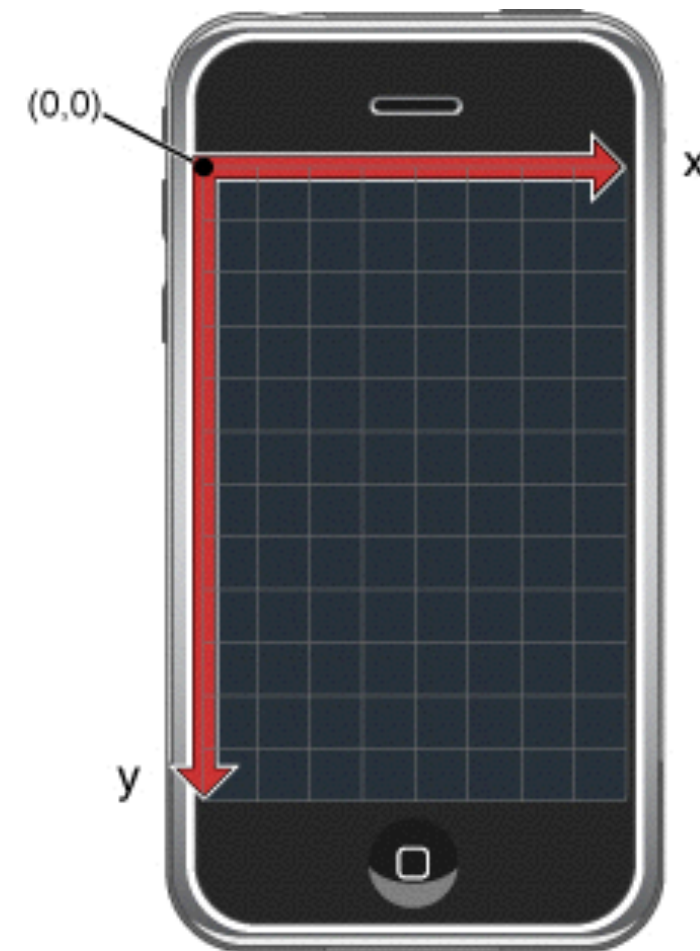
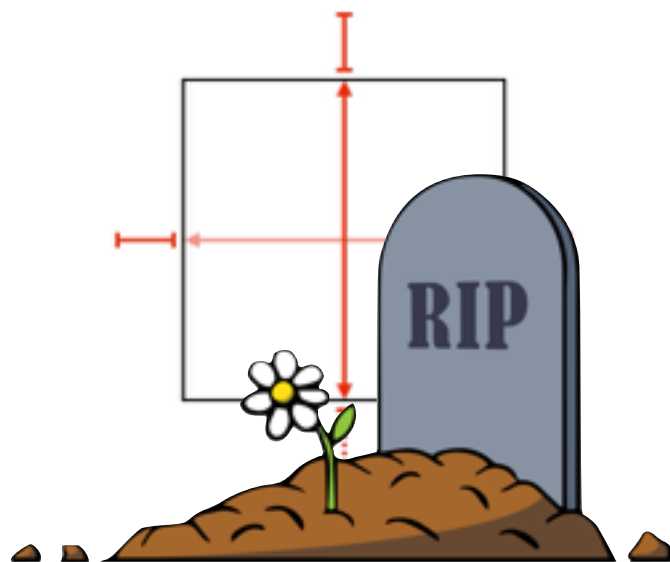


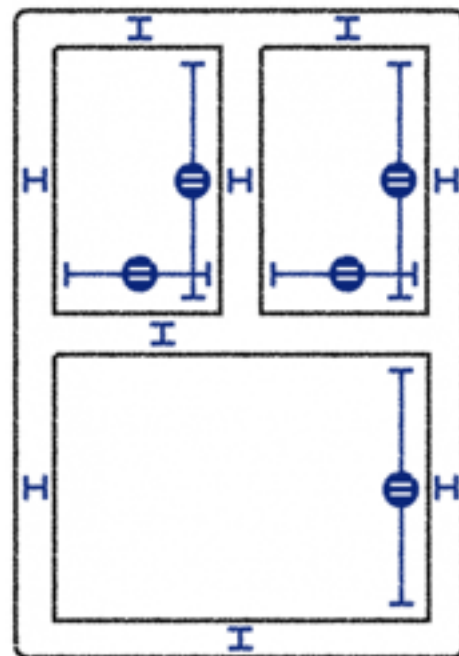
image credit: <http://nshipster.com/cmdevicemotion/>

Layout in iOS

Auto Resizing Masks



Auto layout (iOS6, 2012)



Describe relationship between objects using constraints.

Visual format language.
Describe your layouts in ASCII.

Auto resized labels for different languages.

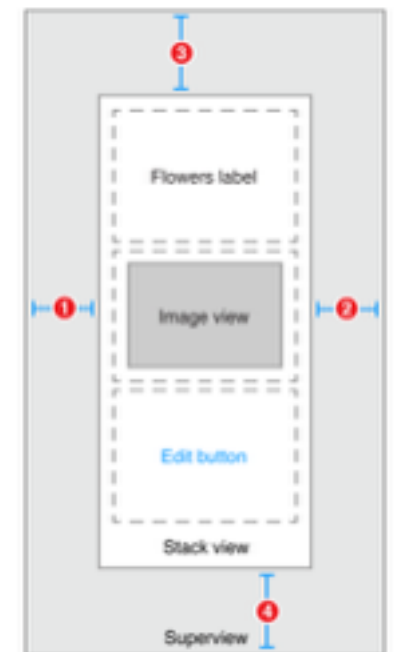
Size Classes (iOS8, 2014)



Clusters screen sizes into size classes.

Allows for tweaks that are specific to each size class.

Stack Views (iOS9, 2015)



Easier way to group widgets into vertical and horizontal bundles.

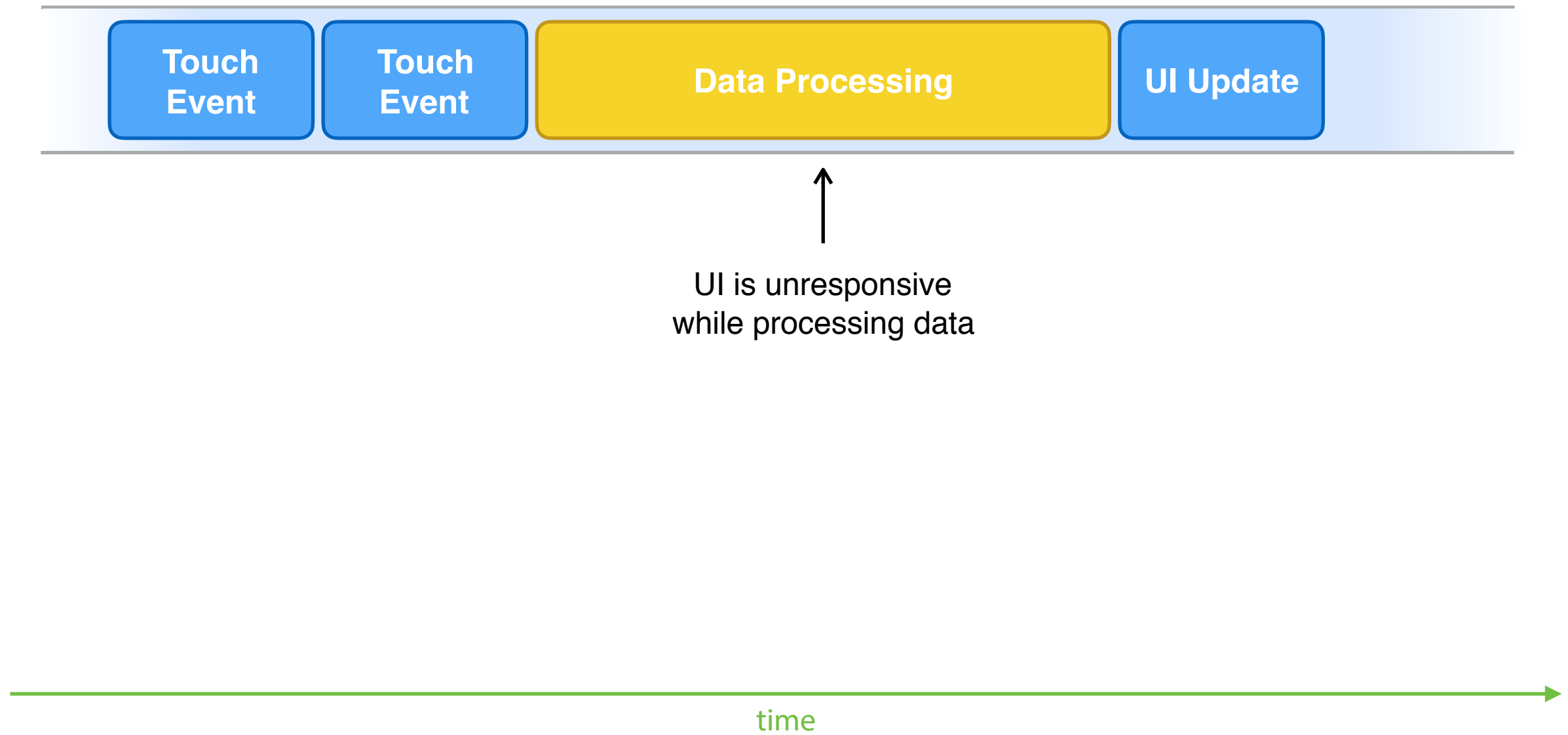
Works together with Auto-Layout.

GCD: Grand Central Dispatch

- Lightweight multi-threading lib
- Organizes concurrency into
 - queues (~thread)
 - blocks (code that should execute on a thread)

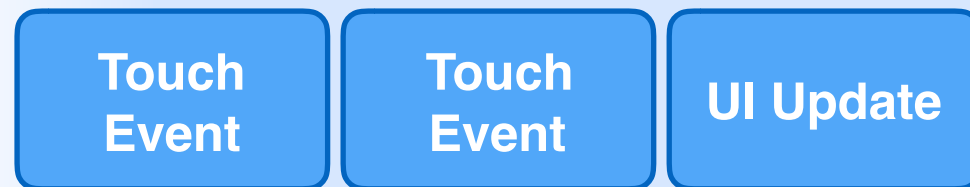
GCD

Main Queue



GCD

Main Queue



Data Processing Queue



time

Blocks

Main Queue



print(...)

```
dispatch_async(dispatch_get_main_queue(), {  
    print("Running on main thread :)")  
})
```

Blocks

Main Queue

print(...)

Data Processing Queue

processData() print(...)

```
dispatch_async(dispatch_queue_create("data-processing", DISPATCH_QUEUE_SERIAL), {  
    processData()  
    dispatch_async(dispatch_get_main_queue(), {  
        print("Back to main queue")  
    })  
})
```