Memory and CPU Profiling

📊⏱️

Why should you do it

you are running blind if not

What to look for

red flags

How often

on every new of functionality/release

Some knowledge of the code base

can still do it without

Deep knowledge of the iOS SDK

ARC, GCD

How to prepare

device

Configuration

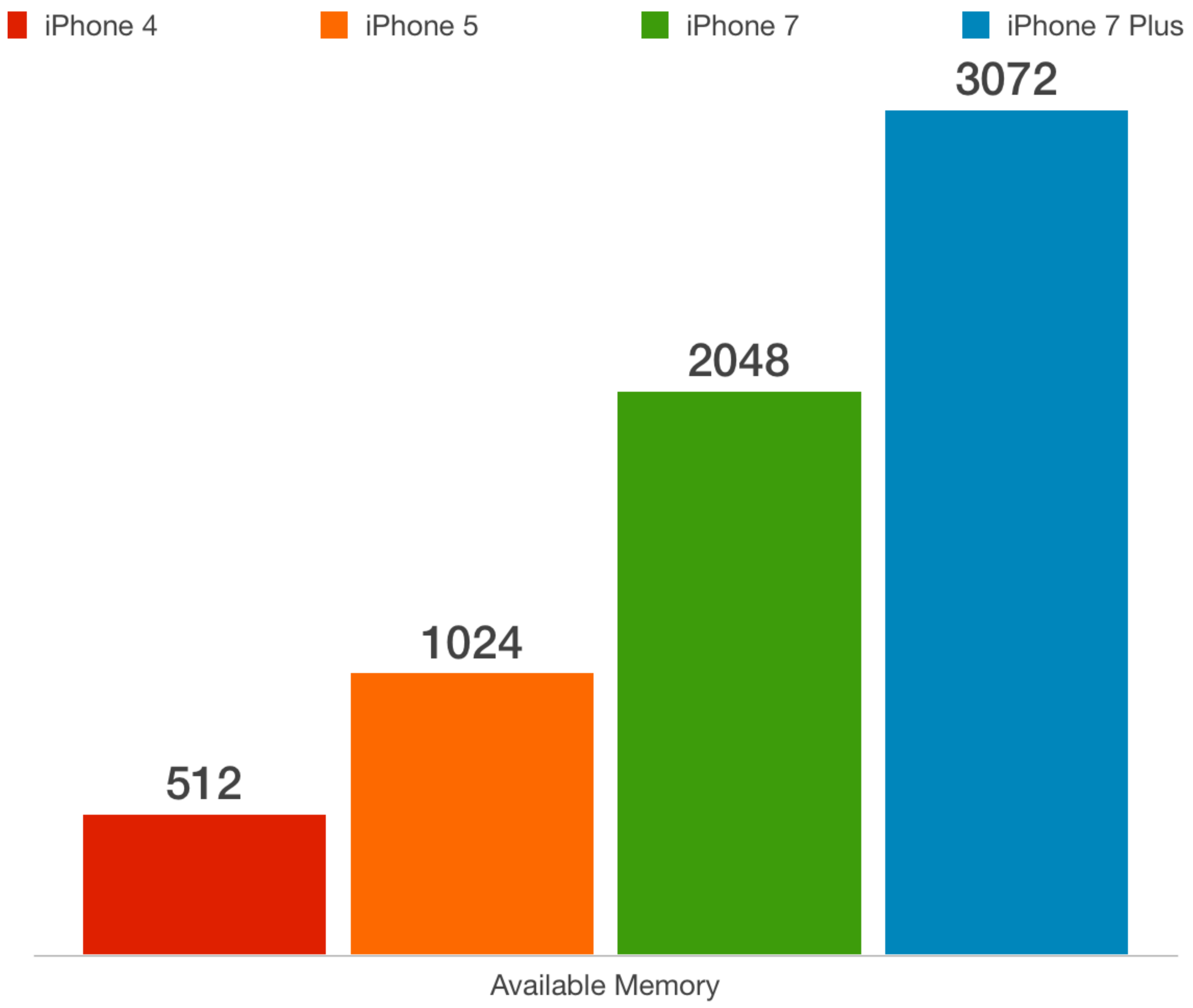
release, base model

Red flags

lead the investigation

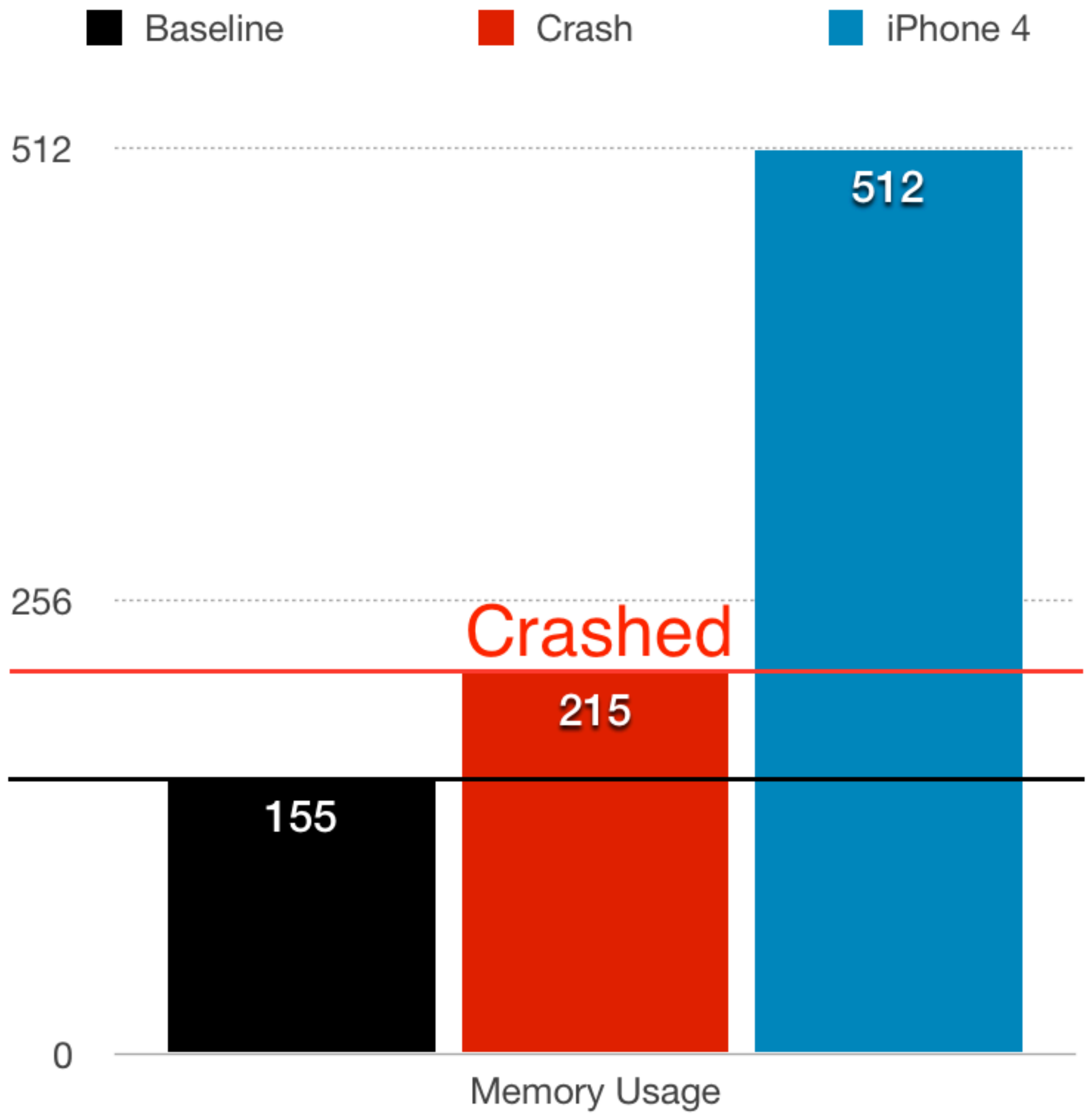
Memory

how much can you use before crashing


iPhone 4
iPhone 5
iPhone 7
iPhone 7 Plus
512
1024
2048
3072
Available Memory

Usage

not all of it is yours

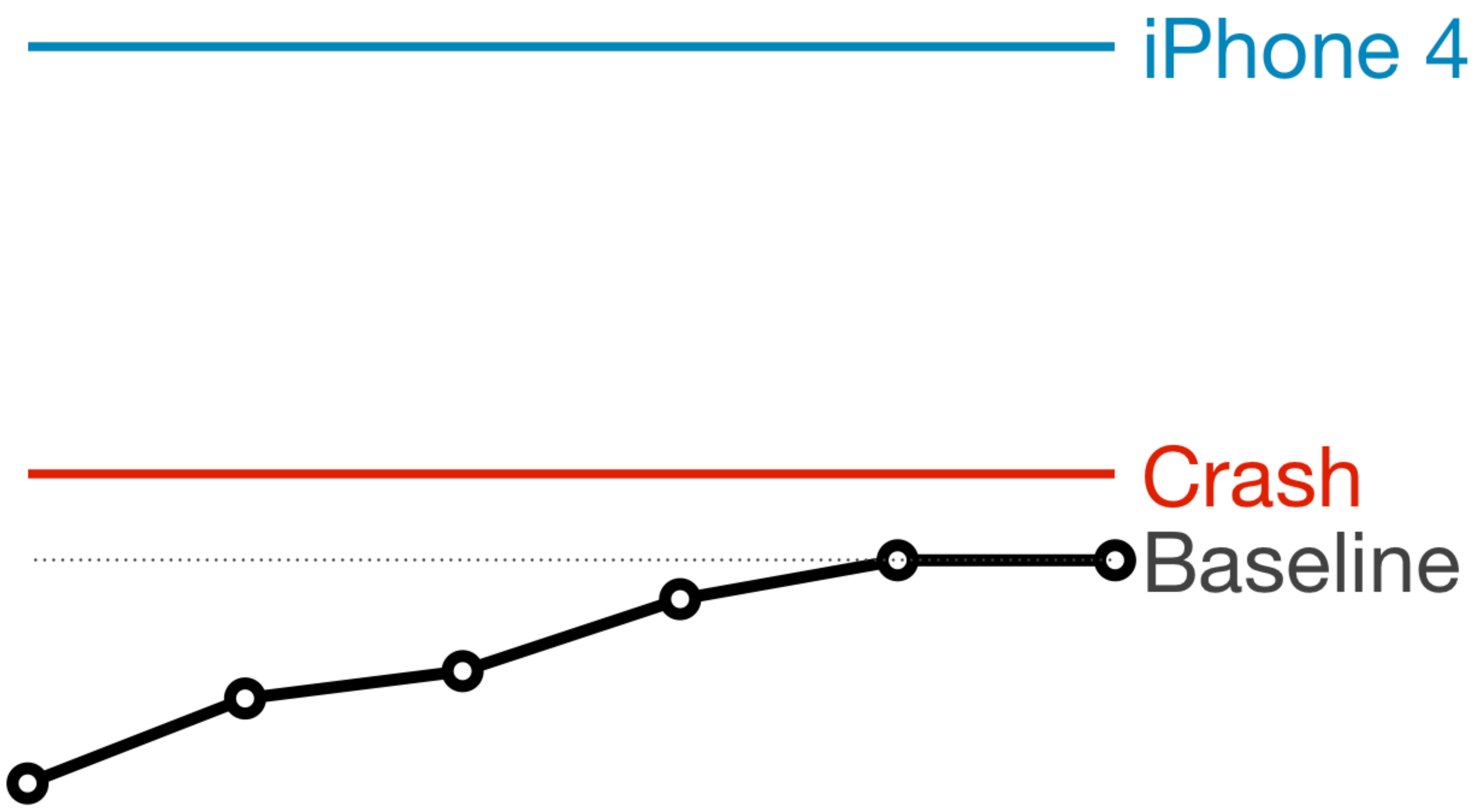

Baseline
Crash
iPhone 4
512
256
0
512
Crashed
215
155
Memory Usage

Establish a baseline

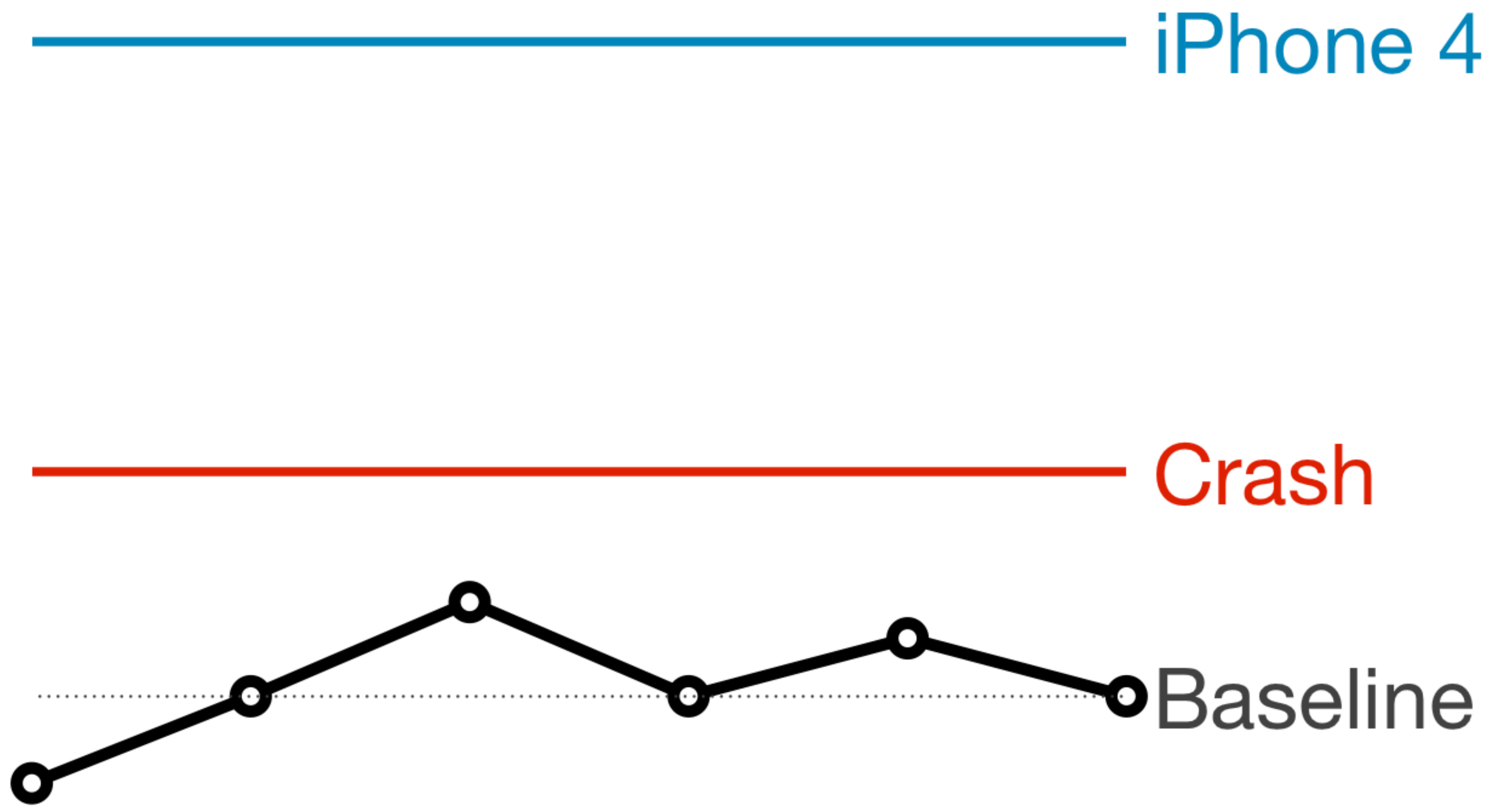
user’s “home” screen

Draw a trend

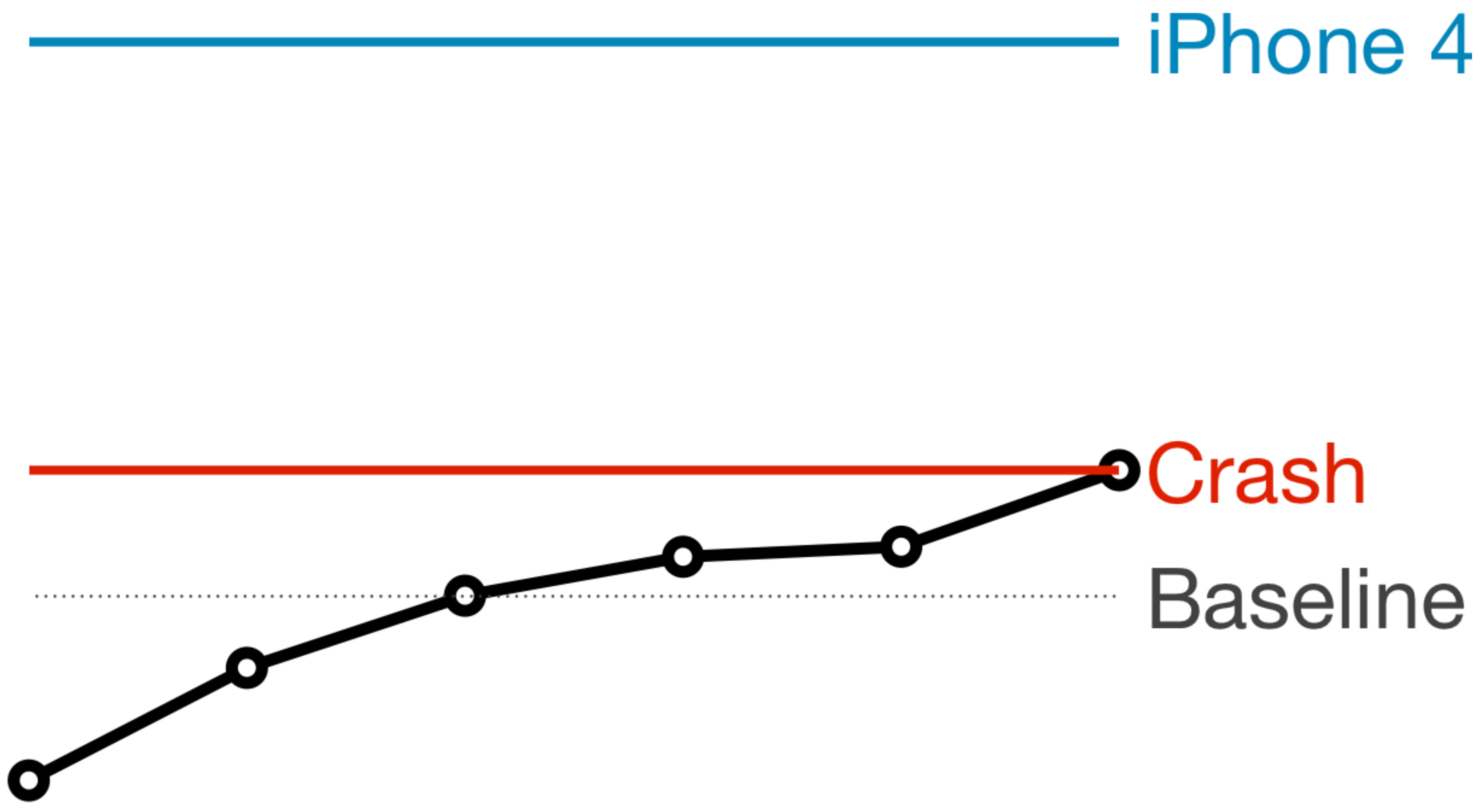
what goes up, should come down



fixed baseline



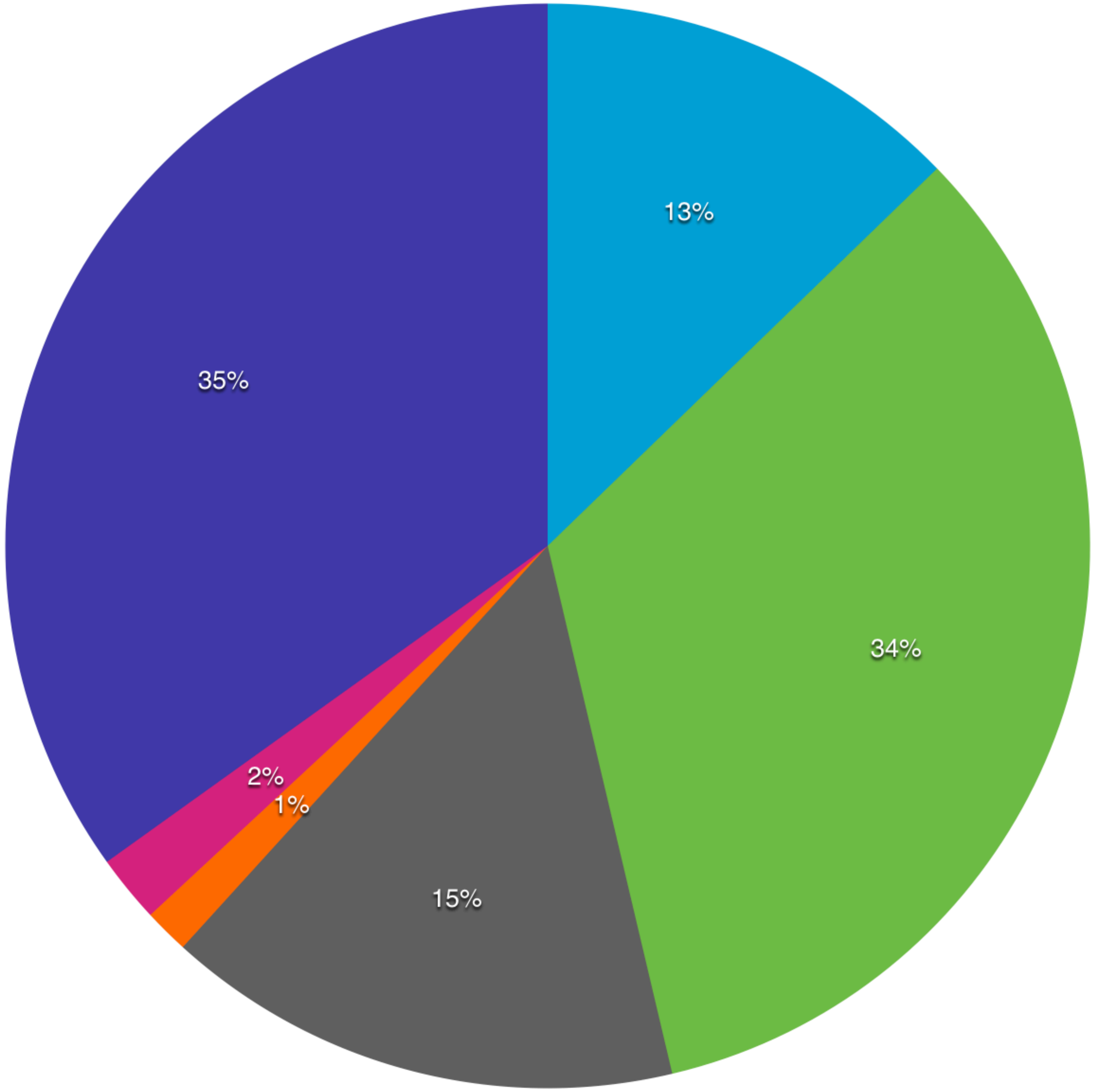
bounded on the baseline

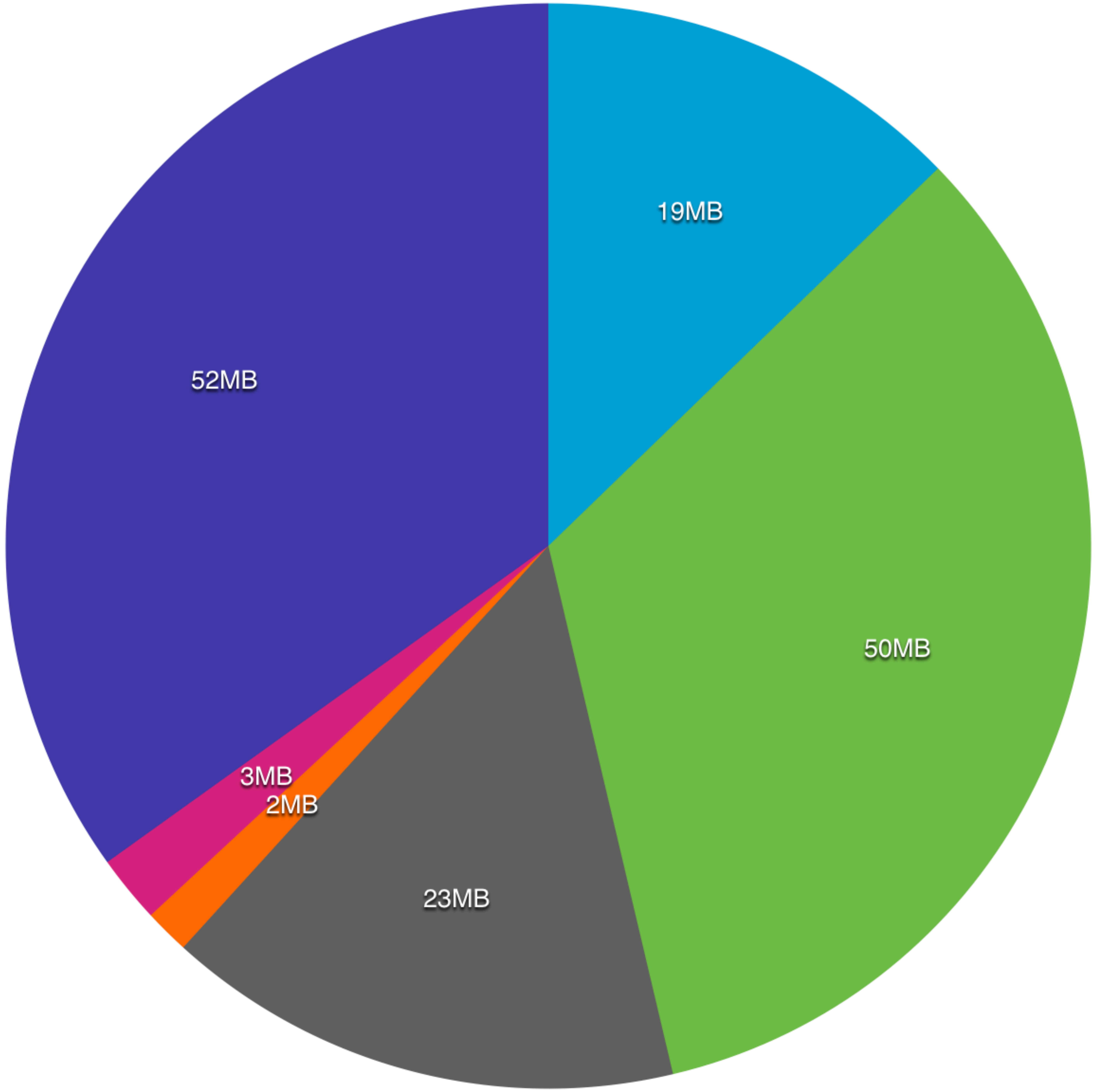


unbounded growth

Calculate usage per feature

memory hungry


13%
35%
34%
2%
1%
15%


19MB
50MB
23MB
2MB
3MB
52MB

Memory warnings

respond to them

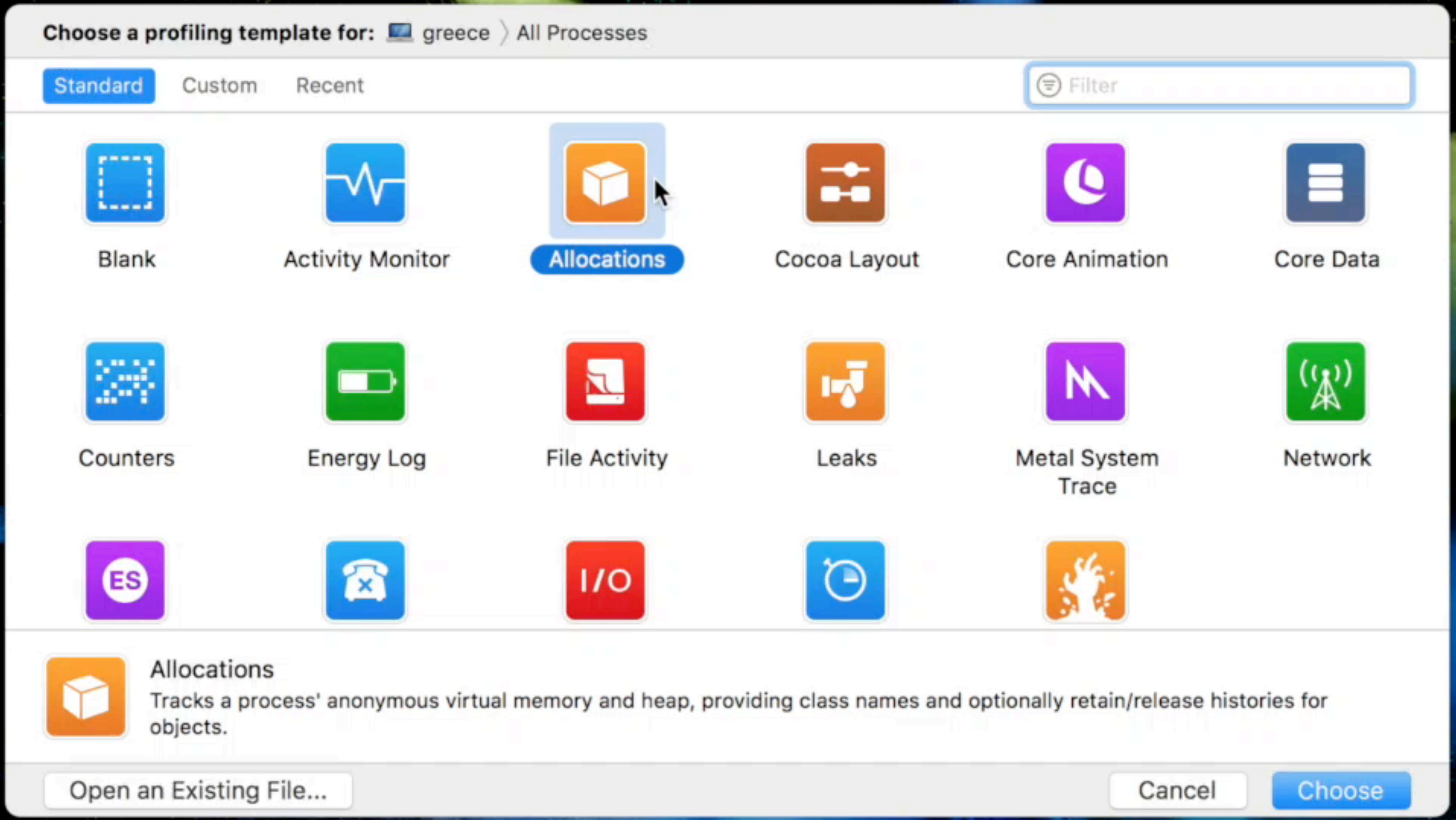
Abandoned memory

referenced

Generational Analysis

see what’s left behind*

* WWDC 2014 Session 418, Improving Your App with Instruments



Allocations

Tracks a process' anonymous virtual memory and heap, providing class names and optionally retain/release histories for objects.

Stacktrace

finding the offending source code

iPhone 7 (10.2)
iosConf (25840)
Run 1 of 1
00:04:47
00:00.000
00:40.000
01:20.000
02:00.000
02:40.000
03:20.000
04:00.000
All Heap & Anonymous VM
Dirty Size
Swapped Size
Resident Size
Details
Generations
All Generations
iosConf

Snapshot	Timestamp	Growth	# Persistent
▶ Generation A	00:26.269.683	152.75 KiB	379
▼ Generation B	00:49.804.943	188.77 MiB	1,024
▶ iosConf.Image		2.06 KiB	66
▼ Generation C	01:10.376.577	192.41 MiB	471
▶ iosConf.Image		1.81 KiB	58

Track Display
Style: Current Bytes

Generation Analysis
Mark Generation

Allocation Lifespan
- All Allocations
- [✓ Created & Persistent]
- Created & Destroyed

Allocation Type
- [✓ All Heap & Anonymous VM]
- All Heap Allocations
- All VM Regions

Call Tree
- Separate by Category
- Separate by Thread
- Invert Call Tree
- Hide System Libraries
- Flatten Recursion

Call Tree Constraints
- Count: 0 to ∞
- Bytes: -∞ to ∞

Data Mining

130
131
132
133
134
135
136
137
138
139
140
141
142
143
144

Debug the View UI Hierarchy

Peaking behind the curtain

RootViewController.swift
RootViewController.swift
Main.storyboard
UI Hierarchy
UI Hierarchy
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
UI Hierarchy
UI Hierarchy
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
RootViewController.xib
RootViewController.swift
Products
Navigation Controller
Navigation Controller
View Controller
Next
View Controller
Next
Done
Watch
Extras
iCloud Drive
iosConf
iosConf
Reference...
Safari
Messages
Screen Shot 2017-03...11.01.02

leaked memory

not referenced (reference cycle), not used

Main.storyboard
LeakedViewController.swift
ViewController.swift

Reference Cycle 〉 Reference Cycle 〉 Main.storyboard 〉 Main.st...(Base) 〉 View Controller Scene 〉 View Controller

Reference Cycle PID 27431

CPU	0%
Memory	22.4 MB
Disk	Zero KB/s
Network	Zero KB/s

Navigation Controller
Navigation Controller
Next
View as: iPhone 7 (wC hR)
82%

weak

reference

Main.storyboard
ViewController.swift
ViewController.swift
Reference Cycle
Reference Cycle
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Navigation Controller
Navigation Controller
Next
Watch
Extras
iCloud Drive
iosConf
iosConf
Reference...
Safari
Messages

bad access memory

the unowned trap

BadAccess
BadAccess
Main.storyboard
Main.storyboard (Base)
View Controller
Next
Watch
Extras
BadAccess
Busy Main
iCloud Drive
iosConf
iosConf
Reference...
Safari
Messages

risks

why high memory usage is a bad thing

App terminated while in use

Random and hard to reproduce (unknown user usage pattern)

Peaks of memory usage can drive the app over the "limit" thus terminated.

A high baseline that puts a constraint

on the features that can be added later

App terminated in the background

or why the app always launches state preservation

Other apps are getting killed

(including yours too!)

getting the blame

actions

what to do

bring the baseline down

reduce image sizes

reuse memory

release unused memory

when is no longer used

return to the baseline

reuse memory

don’t allocate

CPU

getting things done fast and responsive

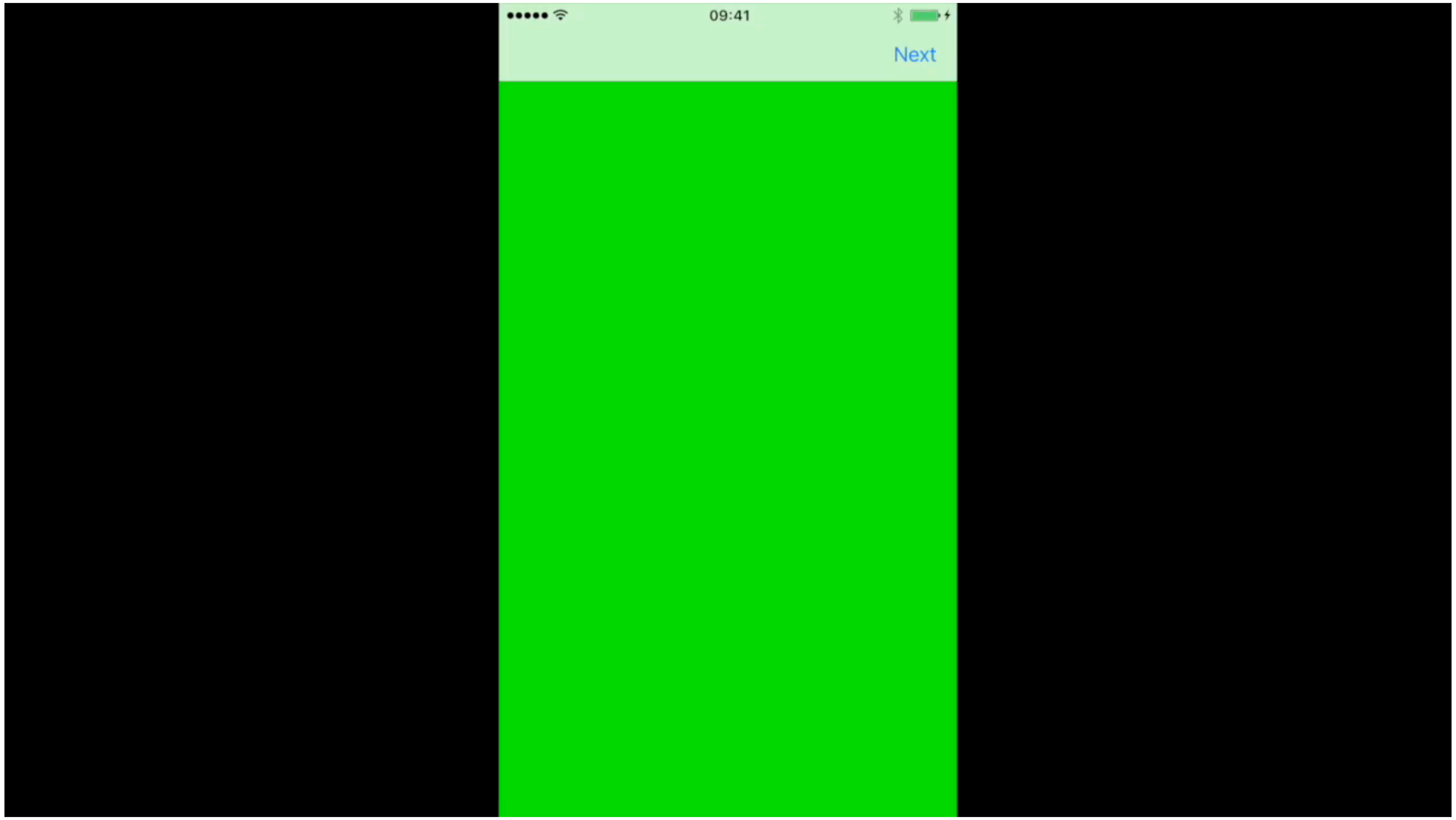
Responsiveness

What responsive UI means

responsive to touch

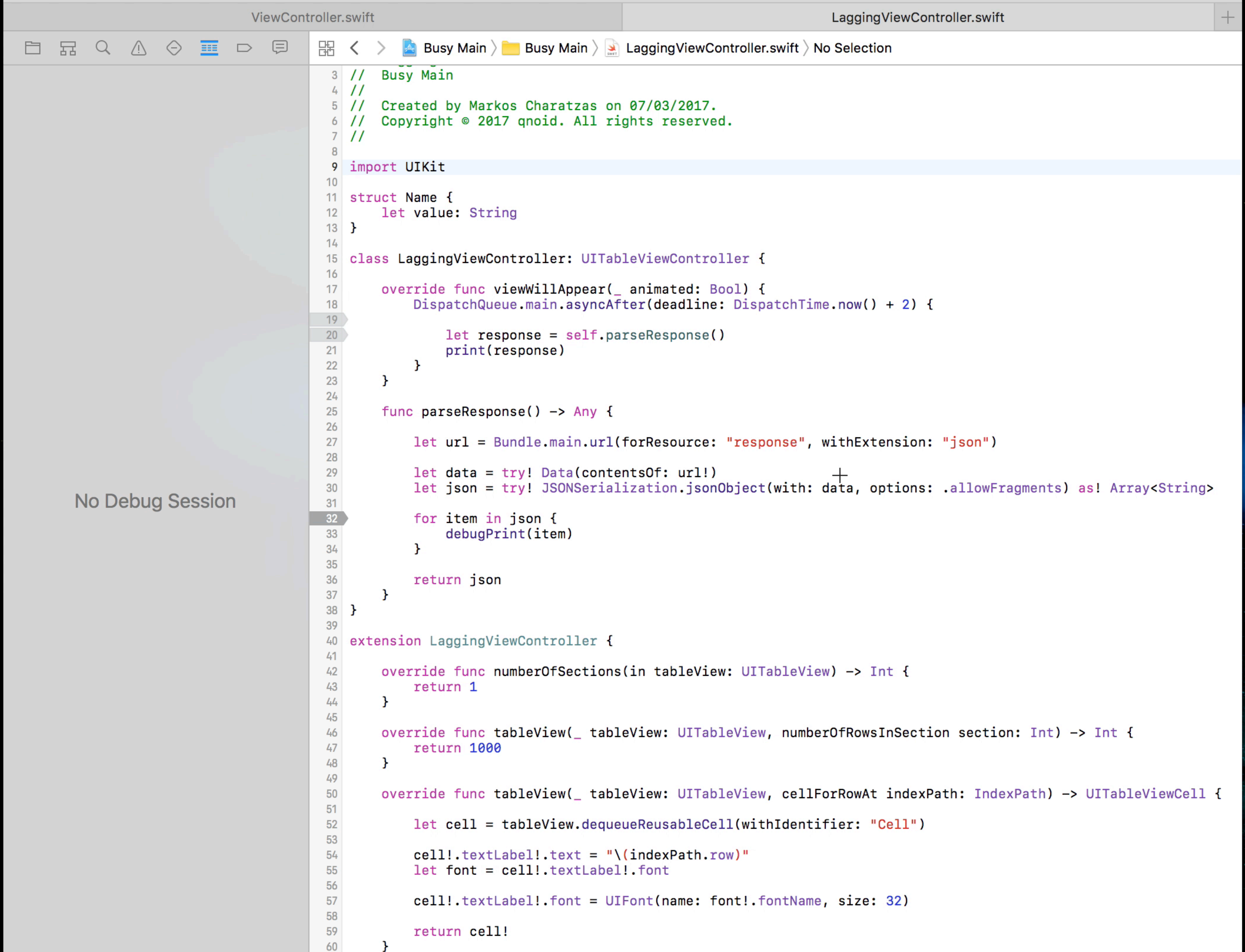
What responsive UI means

responsive to user actions


09:41
Next

UI lag

time profiler



```swift
//  Busy Main
//
//  Created by Markos Charatzas on 07/03/2017.
//  Copyright © 2017 qnoid. All rights reserved.
//

import UIKit

struct Name {
    let value: String
}

class LaggingViewController: UITableViewController {

    override func viewWillAppear(_ animated: Bool) {
        DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 2) {

            let response = self.parseResponse()
            print(response)
        }
    }

    func parseResponse() -> Any {

        let url = Bundle.main.url(forResource: "response", withExtension: "json")

        let data = try! Data(contentsOf: url!)
        let json = try! JSONSerialization.jsonObject(with: data, options: .allowFragments) as! Array<String>

        for item in json {
            debugPrint(item)
        }

        return json
    }
}

extension LaggingViewController {

    override func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 1000
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell")

        cell!.textLabel!.text = "\(indexPath.row)"
        let font = cell!.textLabel!.font

        cell!.textLabel!.font = UIFont(name: font!.fontName, size: 32)

        return cell!
    }
```

Generational Analysis.mp4

60fps

or 1 frame per ~17ms

User waiting time

i.e. at login

the 0.1s / 1.0s / 10s rule

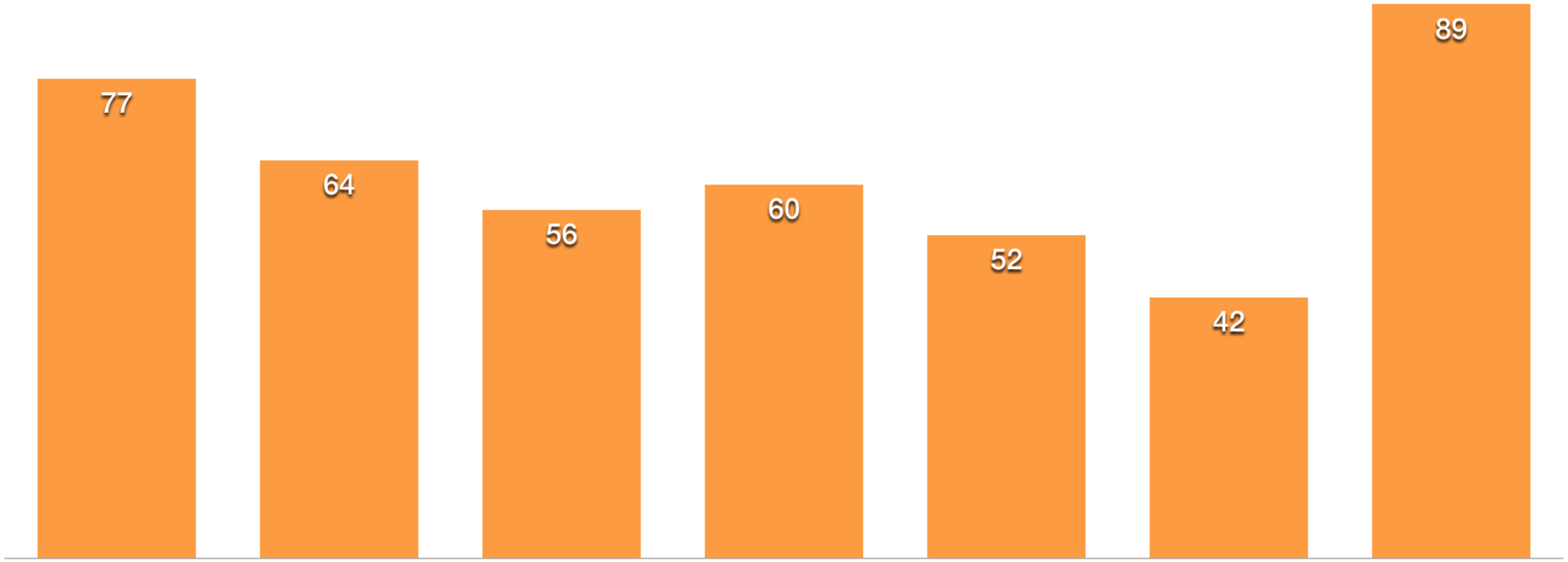
know the limits

* https://www.nngroup.com/articles/response-times-3-important-limits/

Graph

how fast can you go

% CPU per functionality



Stay idle

do you let the CPU rest

All Cores
All Processes / Threads
00:00.000
00:10.000
00:20.000
00:30.000
00:40.000
00:50.000
01:00.000
01:10.000
Time Profiler
Details
Profile
Root
Process
Weight
Self Weight
Symbol Name
Calendar
Photos
Maps
Wallet
Reminders
News
Health
Settings
Safari
Messages

Options

- High Frequency
- Record Kernel Callstacks
- Record Waiting Threads

risks

why high CPU usage is a bad thing

Main thread is busy

unresponsive to touch

miss opportunity to respond to memory warnings

Degrades user experience

app does not "feel" smooth

time wait processing

Unresponsive due to high cpu usage

whether on the main thread or not

battery drain

device runs hot

actions

what to do

offload processing from main thread

on background operations

Postpone processing for when required

being lazy

Partition the work

2 cores = 200% CPU

Find what’s killing your CPU

keeping you busy

In general

tips and tricks

don’t tie user interface to processing

i.e. network

didFinishLaunchingWithOptions

400ms is a good target

(no more than 20 seconds)*

* WWDC 2016 Session 406, Optimizing App Startup Time

applicationDidEnterBackground

release resources

persist state

stay low

IBAction

low hanging fruit

touch handling

low hanging fruit

completion blocks

low hanging fruit

Put it on schedule

on every release

Keep a record of your measurements

to compare and contrast

Go forth and release

confident / knowing the risks

www qnoid.com

One more thing…

allwallpapersfree.blogspot.com