DIAGNOSING ERROR

A PRESENTATION BY MICHAEL LANGFORD

A BIT ABOUT ME:

» Embedded Programming Background

» iOS Developer since 2008, some Android off and on

» Independent Consultant: risk management, app development, and getting new teams going

» Founder of App Cartographer (www.AppCartographer.com), a development service that instruments apps with analytics, crash reporting, then trains people to make good decisions off those

PRINCIPLED APPROACH

» RTFM

» Fix All The Windows

» Reliable Reproduction

» Cartography First

» Use Your Eyes, not Confidence

» Split the Map

» Act Like You Have A Fast Computer

TOOLS

If you need a machine and don't buy it, then you will ultimately find that you have paid for it and don't have it.

— Henry Ford

READ THE MANUAL

» Don't guess how it works, read the doc or source code

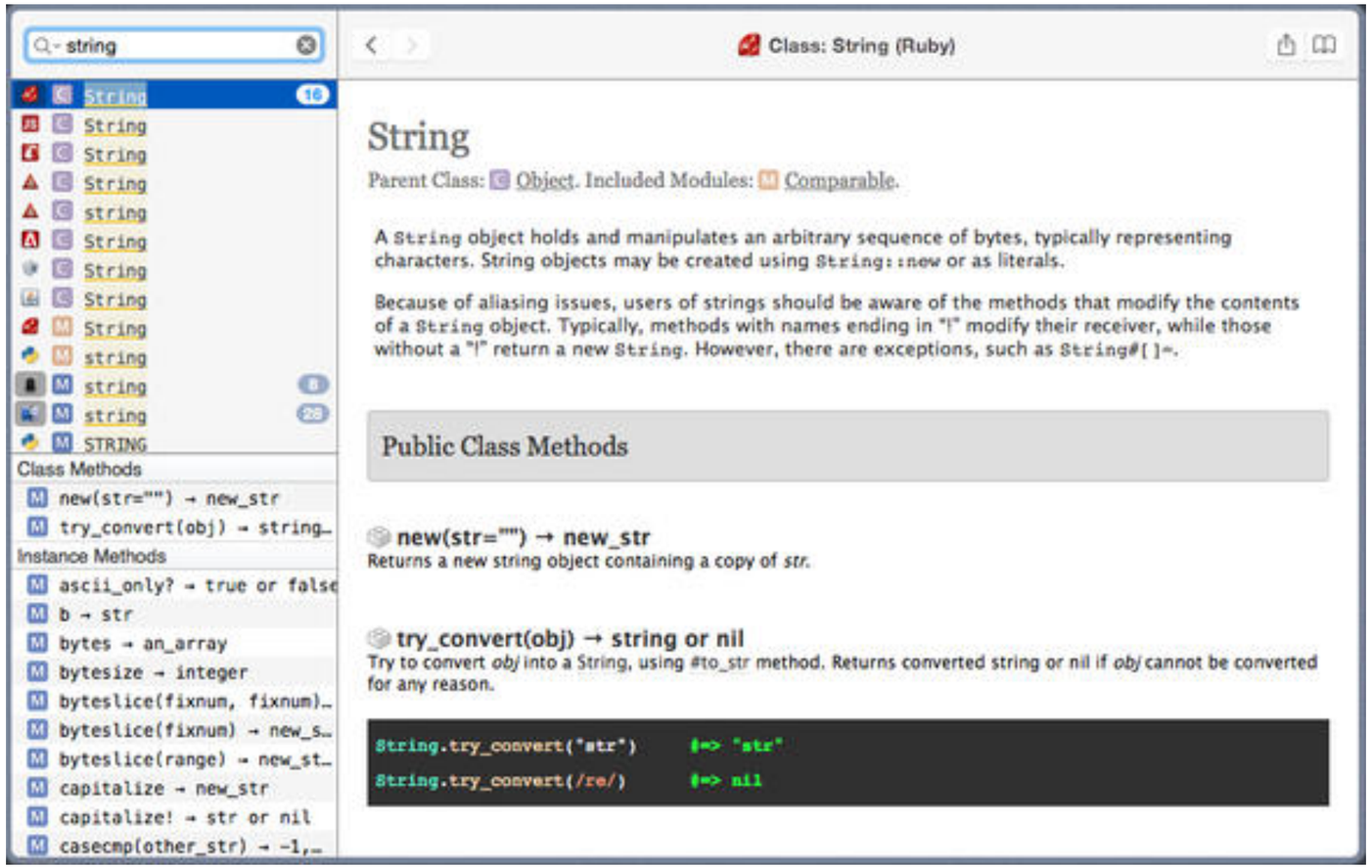
» Check the reliability of comments by checking git modification dates of doc lines vs code via git blame

» You're going to have to read some. Use your bed more, your coffee cup less

» Stack overflow is not the doc

READ THE MANUAL: TOOLS

Dash: https://kapeli.com/dash $25



FIX ALL THE WINDOWS

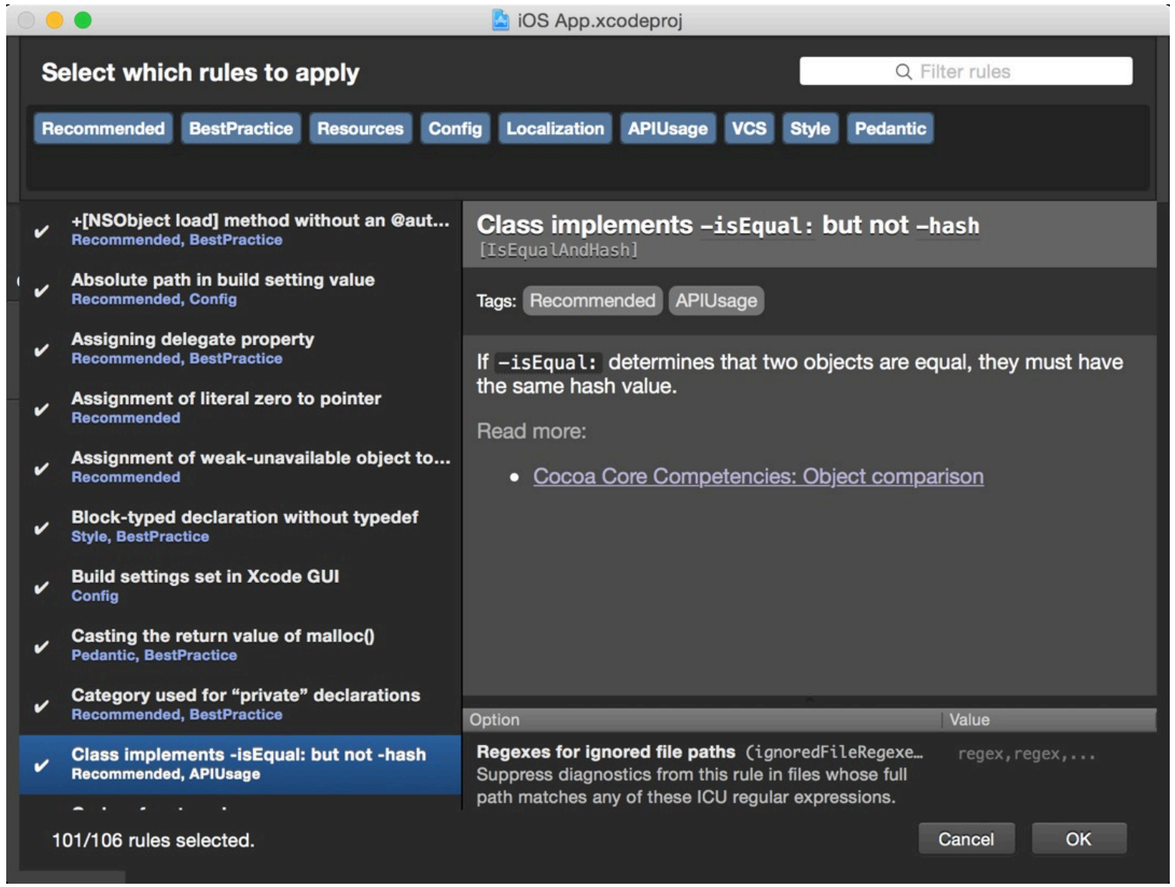
» There are a lot of automated systems that we have today that will detect minor errors

» Leaving warnings unfixed or unsuppressed stops you from seeing new bugs that are a problem. Warnings in your build cause warning fatigue.

» Fix all warnings and analyzer errors...or suppress them with comments explaining why it's not a problem

FIX ALL THE WINDOWS: FAUXPAS

`http://fauxpasapp.com $89`



RELIABLE REPRODUCTION

» Use faked input data to speed up reproduction

» Use generative/property based testing frameworks to stimulate failure

» Use chaos causing systems to stimulate failure

» Use UI "Testing" frameworks to speed up failure

» Have the ability to turn off abuse systems on the backend of the app when required

RELIABLE REPRODUCTION: TOOLS

» UI Testing: https://developer.apple.com/videos/play/wwdc2015/406/

» Generative Testing Reference: http://www.infoq.com/presentations/property-based-testing

» Generative Testing Frameworks: https://github.com/jeffh/Fox, https://github.com/typelift/SwiftCheck

CARTOGRAPHY FIRST

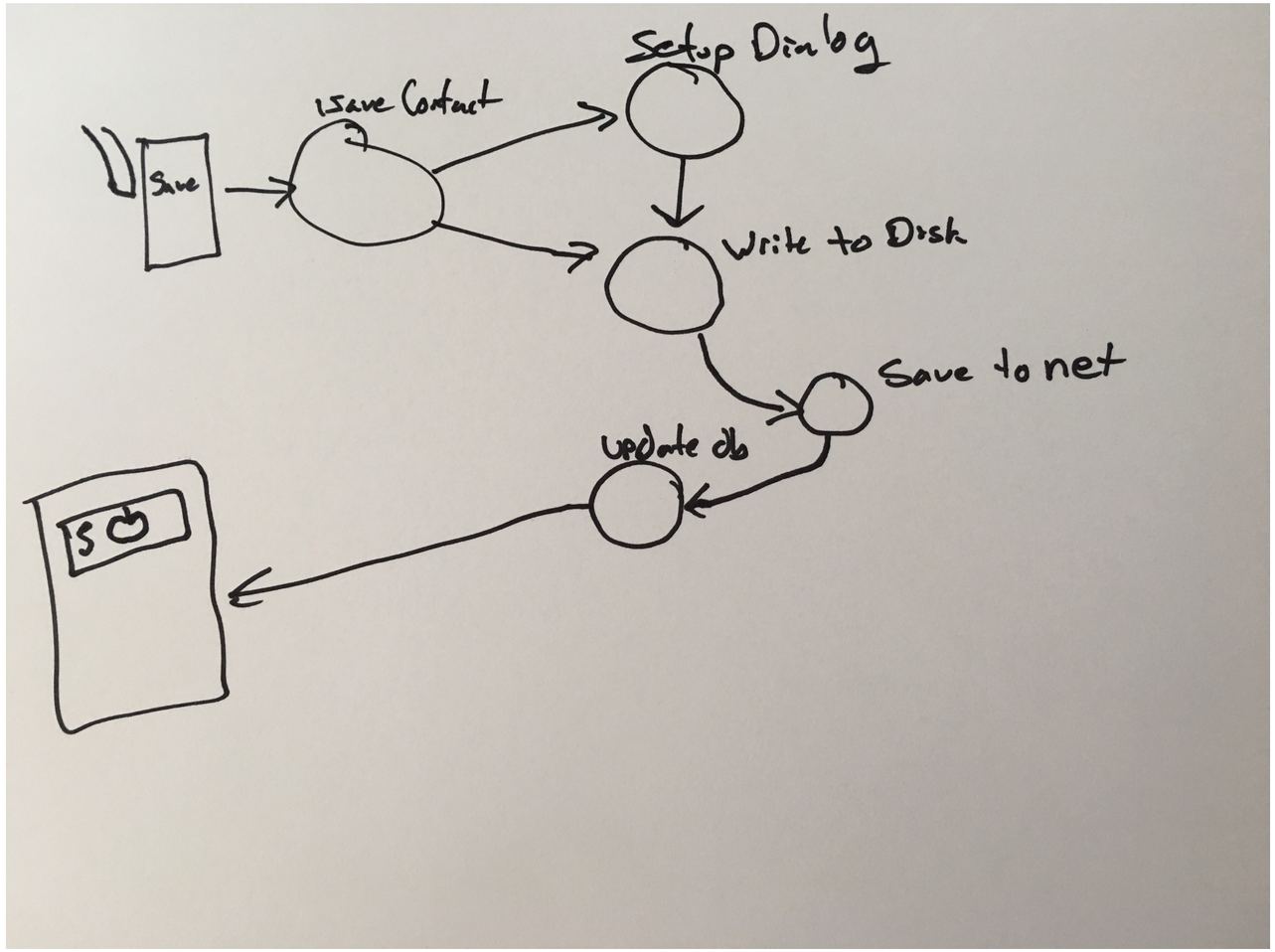
» Don't draw out the whole system, just the data/ states that are near the flow of the program that the problem is in

» Physically draw it out, don't go to the depths of documentation typically: this type of doc always goes stale and is misleading

» You may have to read more about how subsystems of iOS such as the View Controller lifecycle, release cycles, Core Data, Grand Central Dispatch.

» Remember to think of corner asynchronous cases

CARTOGRAPHY FIRST: LEVEL OF EFFORT WE'RE GOING FOR



EYES, NOT CONFIDENCE

» You must break yourself of the habit of telling yourself stories, you must determine real truth. Don't let habits of bullshitting from meetings bite you here.

» Be very careful when determining what you have verified to be true and what is a hypothesis (avoid the word "theory")

» Add assumptions to the code in the form of preconditions, asserts and guards (check these in!)

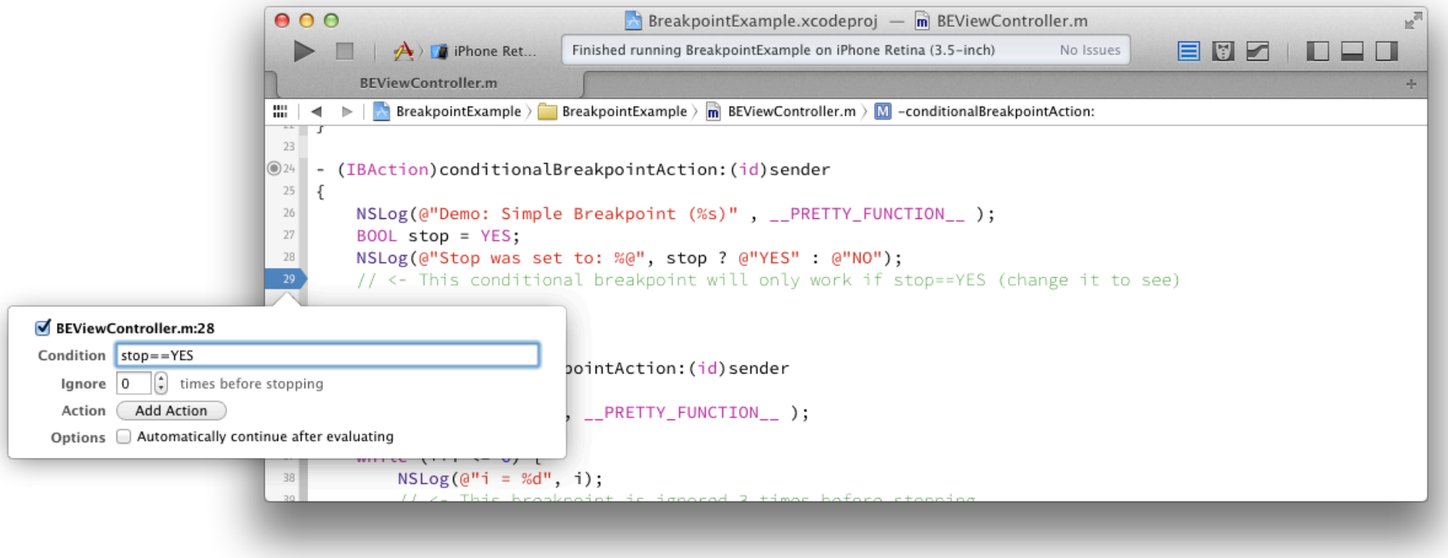
USE YOUR EYES, NOT CONFIDENCE

» Use debuggers, logging, artificially colored views, named constraints, debug modes in libraries and more

» Turn off all of the things that tell you stuff you don't need to know right now

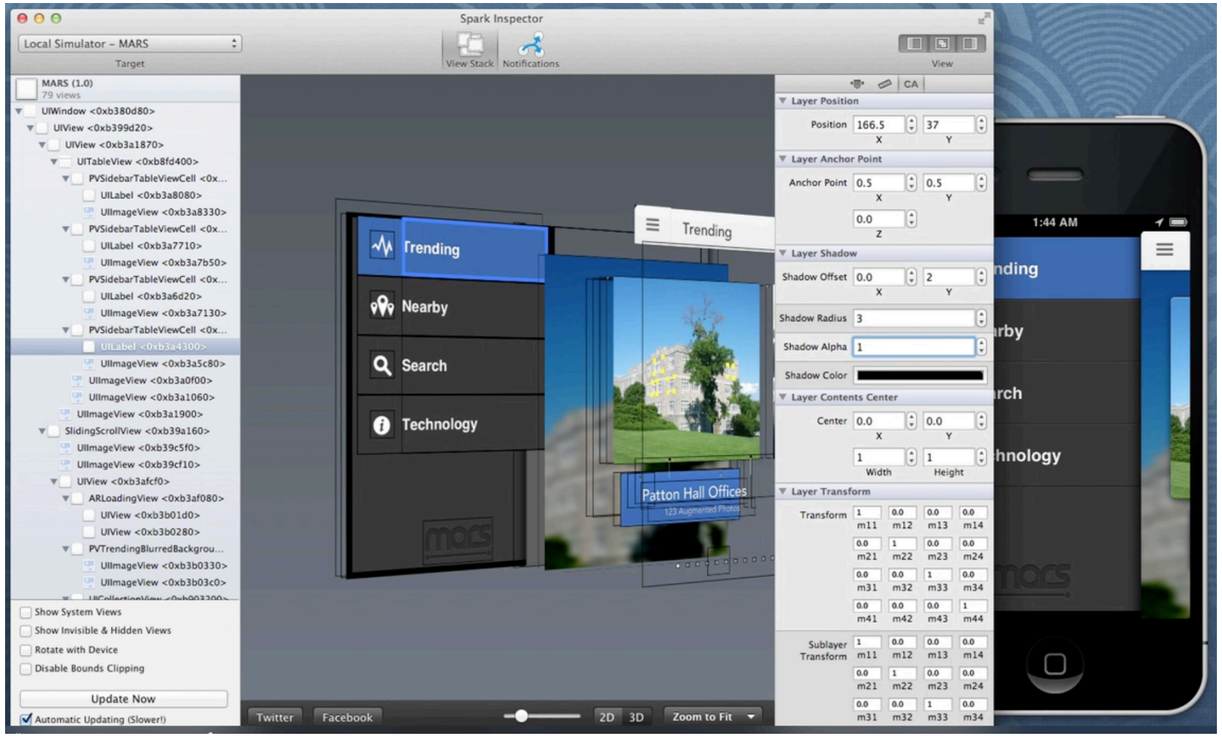
» Make instrumentation panels to expose state in the middle when simple print calls do not work

USE YOUR EYES NOT CONFIDENCE: CONDITIONAL BREAKPOINTS



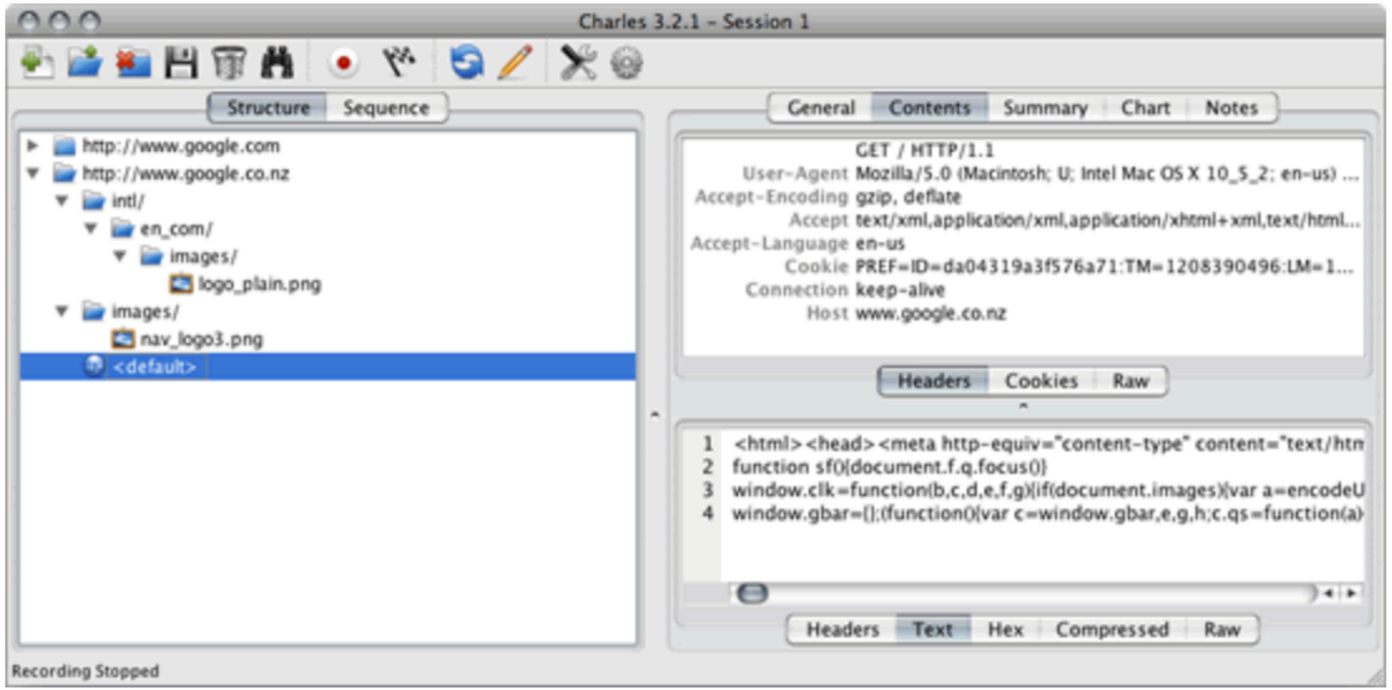
From: http://jeffreysambells.com/2014/01/14/using-breakpoints-in-xcode

USE YOUR EYES NOT CONFIDENCE: SPARK INSPECTOR



http://sparkinspector.com $50

USE YOUR EYES NOT CONFIDENCE: CHARLES PROXY



http://www.charlesproxy.com $50

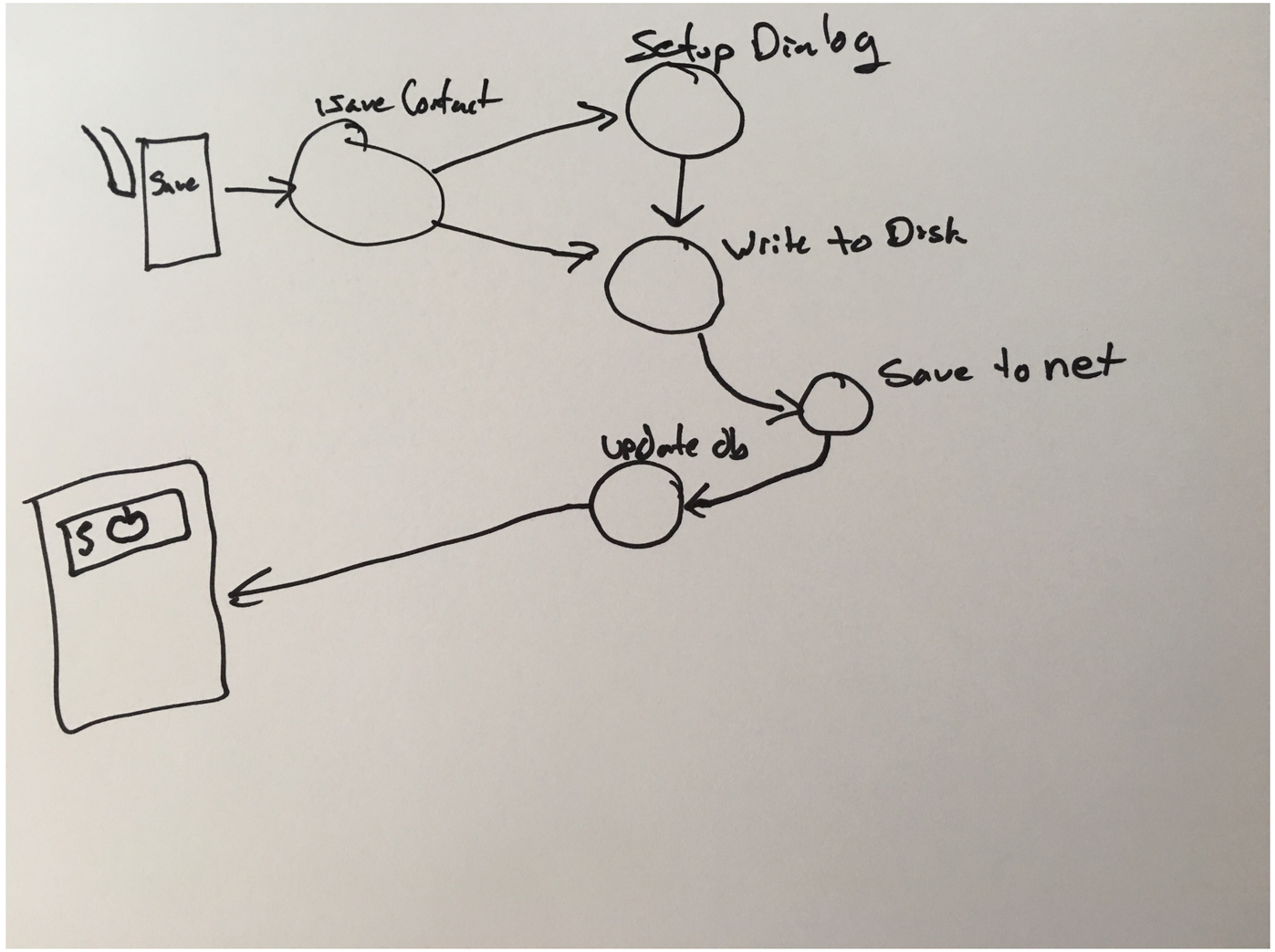
SPLIT THE MAP

» Cut the Map in Two

» Determine which of the two pieces of the map has faulty data by looking at the middle to see if the data is good there

» Repeat

SPLIT THE MAP: EXAMPLE



ACT LIKE YOU HAVE A FAST COMPUTER

» Don't change multiple things just because compiles take a long time. One change, one build.

» Force the problem to the simulator if at all possible

» Write yourself notes while you wait about your hypothesis: Makes distractions less painful

PRINCIPLED APPROACH

» RTFM

» Fix All The Windows

» Reliable Reproduction

» Cartography First

» Use Your Eyes, not Confidence

» Split the Map

» Act Like You Have A Fast Computer

CONTACT ME?

Consulting: www.RowdyLabs.com

Email: michael.langford@gmail.com

Twitter (and most iOS/Clojure slacks): @mj_langford

AppCartographer: www.appcartographer.com

THANKS

References: Debugging by Agans, Various courses at Georgia Tech, The tutelage of many great engineers I've gotten to work with and the articles referenced herein