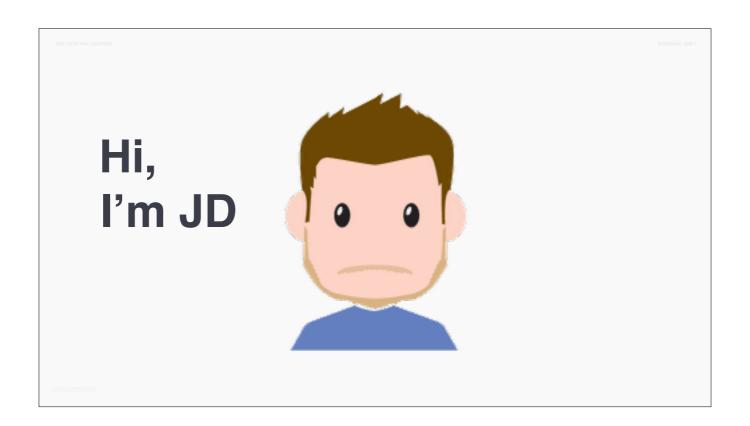
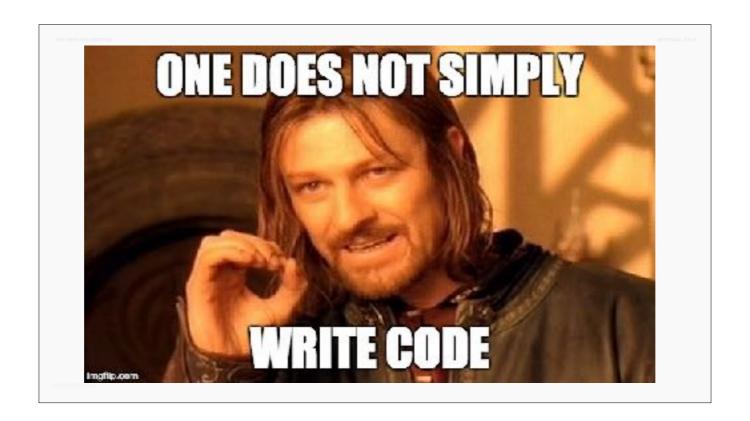
THE NEW PHILOSOPHIE INTERNAL ONL

Professional Software Development

PHILOSOPHIE



- \bullet JD Wolk, professional software developer for ~7 years, ~4.5 years as a consultant at Philosophie
- Here to give a whirlwind tour of professional software development
- Completely subjective and basically arbitrary. Based on my own experience in the field
- Combination of things you'll encounter in day-to-day software development, lessons you'll probably learn, topics you can research, and what I think might be helpful for starting out in the field
- Mostly the stuff besides the nuts-and-bolts of code, though we will talk a bit about it.



- If you only take one thing away from this, let it be that professional software development is about more than just writing code
- This probably isn't a surprise to anyone, but you probably have a lot of "unknown unknowns" if you've only ever worked on hobby software projects before
- To be successful as a professional software developer you need to develop both hard skills and soft skills so you can work effectively on a team

Hard Skills Tools Practices Principles Soft Skills Team Clients Self

This is a high-level outline of this talk

Hard Skills

■ Tools

Practices

Principles

Soft Skills

Team

Clients

Self

- Tools are one of the most valuable things to learn as a software developer
- * The sooner you refine your abilities with your tools, the sooner you can stop letting incidental things get between your thoughts and working code.
- * A side benefit is that as you gain more experience with these things you can start to pick them up faster (generally).
- * They start out as a hurdle, but eventually you take them for granted as they start to blend into your workflow, making it quicker and giving you more options



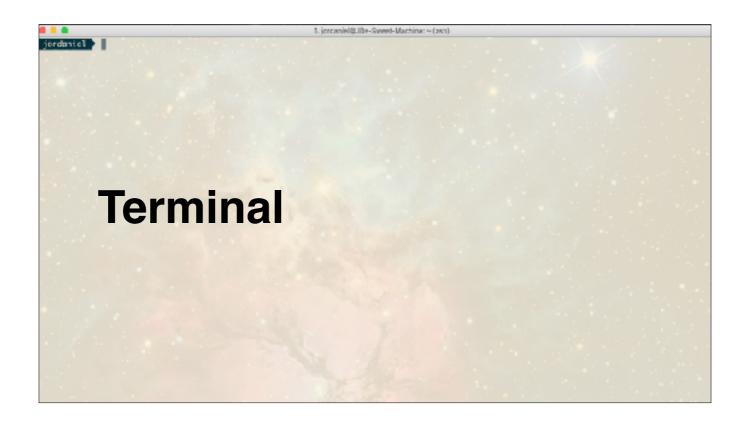
- * (Text Editor)
- * When coding, I spend about 75% of my time looking at my text editor
- * You want to have the ability to not think about what you're doing, like touch typing
- * Spend as much time as you need thinking through a problem, then execute very swiftly
- * There are 2 types of text editors

```
19 class Monber - ActiveSecord: Sese
                         20 of Extensions
21 include SertableConcern-
> 🛅 benner
> 🔳 collection
                         23 - F Attributes-
26 - enum status: { inactive: 0, ective: 1 }
                         /b betcre_save :set_activates_at, it: -> ( status_cranges/ )-
27 - after_create :generate_activation_code-
                         20 # Welifotions
30 **welldetes :first_name, presence: true
                         31 validates :lest_name, presence: true
                              - validates istatus, presence: true-
   GUI-based.
                                  return all unless q-present?
                                 where('first_name ilike :same or last_name ilike :name', name: "#(g)%")
                          41 - scope contocomplete, -s(q, limit: nil) (-
                          12 searth_same(q)
                                .limit((limit || Settings.autocomplete_limit).to_i)
.orderlids :dose)
                         45 - 461 actave-(status)
                         th - self.status = (status&.to_bool ? :active : :iractive)
                          51 private
```

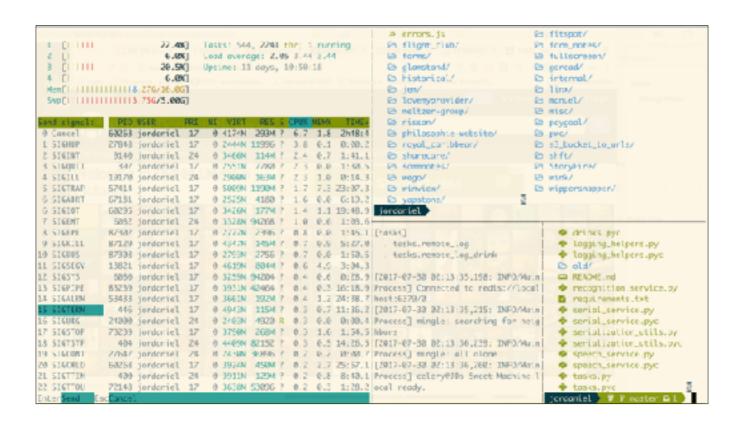
- (GUI-based editors)
- * Atom seems to be a pretty popular choice for more "modern" editors. Others in this space are sublime text & brackets

```
38 # Success / Failure handling
                                                                                      import wave
                              33 444444
 (up a dir)
                                                                                     import es.path
/drink_mixer_backup/
                                                                                     4 import paho.mgtt.client as mgtt
gudto helpers/
                              41 class Success():
                                                                                     from pygane import mixer
                              42 das __imit__(self, volue):
such_helpers/
                                                                                     E from connor settings import MOT HOST, MOT MANES
                              45 | self.value = value
                                                                                      ACE, AUDIO_CLIPS_DIR
51.07
clips/
                                                                                     from logging_helpans import log_for
old/
                              45 def wcs_successful(self):
                                                                                     log = leg_for('SPIECH')
__imit__.py
                                   return True
__main__.py
assistant_helpers.py
                              48 class follow():
                                                                                   11 class SpeechMessages():
                              49 de' _imit_(self, error):
                                                                                    12 MAKING DRINK = 'MAKING DRINK
button service.py
collection_helpers.py
                                  : self.value = error
                                                                                       NOT_UNDERSTOOD = 'NOT_UNDERSTOOD'
collection_helpers.gyc
                                                                                       GIVING_UF = "GIVING_UP"
                                                                                   IS WHAT_WOULD_YDU_LIKE - WHAT_WOULD_YDU_LIKE
drinks.py
                                                                                     MCSSAGCS - SpeechMessages()
                                                                                    8 SPEECH_TOPIC = MOTT_MANESPACE + "/speech"
drinks.pyc
togging_helpers.py
                                                                                      SPEECH_SAY_TOPIC = SPEECH_TOPIC + "/SUV"
Logging helpers.pyc
                              57 # Audio Recognition Contexts
                              SS SUPPLEE
README. Inc.
                                                                                    21 class SpeechService():
racognition_sarvice.py
                                                                                   22 def __init__(self):
requirements.txt
                              60 class GoogleContext():
                                                                                        | self.iritiplize_nottO
serial_service.py
                              61 def recognize(self, recognizer, audio, **Iwargs)
serial_service.pyc
                                                                                       def initialize_mqtt(self):
serialization_utils.py
                             return recognizerurecognize_google(ducio)
                                                                                        | self.client = mgtt.Client()
serialization utils.pyc
                                                                                         ; self.client.or_correct = self.or_correct
speech_service.py
                              66 class foodContext():
                                                                                    78 | Self-client.message_collback_ocd(SPEEOLSAN_)
speech_service.pyc
                             65 der recogniza(self, recognizer, audio, **!wargs)
                                                                                     PIC, self.message_callback)
                                                                                        | Balf.client.connect(MQTT_HOST)
basks.py
backs.pyc
                                   return recognizer.recognize_sphinx(
```

- (Terminal-based editors)
- * The two heavyweight contenders in the terminal-based editor space are vim and emacs
- * Regardless of whether you prefer a GUI-based editor or whether your preferred language requires an IDE, you should still at least know the basics of working w/ terminal-based text editors like vim or emacs so you can work when ssh'd into remote servers that don't have a GUI



- * (Terminal)
- * You'll also hear words like: command line, command prompt, shell, console...they're all _basically_ referring to the same thing
- * Skill with the terminal is pre-requisite for many tasks
- * Mastery allows you to incorporate powerful tools into your workflow
- * Ex: the command line is about more than file manipulation and navigation...



(Explain processes in terminal)



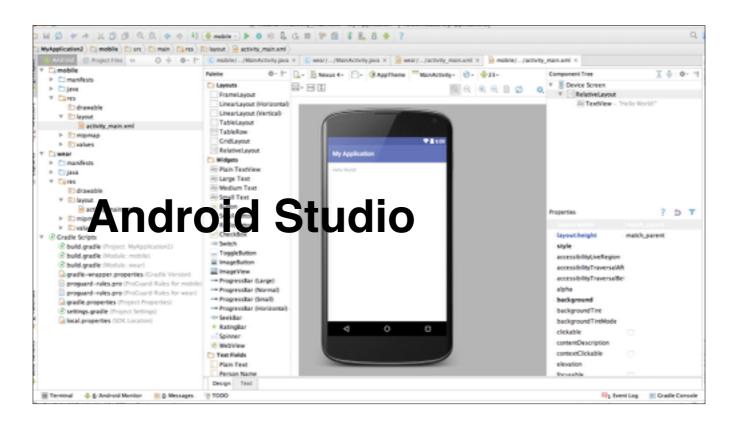
- * (IDEs)
- * Sometimes languages require more overhead as part of their workflow. Text editors are great for languages where this overhead is minimal like JavaScript, Ruby, and Python.
- * By contrast, IDEs tend to be used more often for developing in languages that DO have this overhead (you hear terms like static, strongly typed and compiled) (i.e. Java, C++, Objective C, etc)
- * Also in cases like mobile development where you need sophisticated GUI tools as part of the development workflow
- Lots of functionality besides just basic text editing: they compile your code, check for errors, run tests, have built in debugging tools, diagram editing, etc, etc
- Don't _need_ to know an IDE unless you're in a language that generally uses one

```
PlichPerfect
                                                                                                                                                                                                                                                stantity and Type
                                                            : // AppDelegate.swift
1 // PitchPerfect
  PlathParfect
                                                                                                                                                                                                                                                          Name AppDelegate.cult
                                                          . //
5 // Created by Jordaniel Wolk on 7/28/17.
5 // Copyright + 2917 Jordaniel Wolk. All rights reserved.
                                                                                                                                                                                                                                                           Type Default - Swift Spurce

    HewContributavit

   Main.story/ocers
                                                                                                                                                                                                                                                                  ADDICE/BORNESHIT
   Apopta.maspela
                                                                                                                                                                                                                                                       Full Path / Masta/lorderfel/code/fca/
      Laure/Gonen.storyles
                                                                                                                                                                                                                                                                  Prioriverted/informerrect.
Stubblartart/
AppBologate.swift
                                                           * import Ulkit
    e Info plist
                                                          10 GUTACCICATIONNEIN
10 class AppCologates WITHCopendor, UTApplicationDelagate (
 Products
                                                                                                                                                                                                                                                De Demand Resource Tags
                                                                        var windows UENIndow?
                          Farget Membership
                                                                                                                                                                                                                                              (2) A PROPRETECT
                                                                         fund applicationwillResignActive( application: UEApplication) {
    // Sent when the application is about to nowe from active to inactive
    state. This can occur for certain takes of temporary interputions
    (such as an incoming phone call or SMS message) or when the user quits
    the application and it begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate
    graphics rendering callbacks. Sames should use this method to pause
    the game.
                                                                                                                                                                                                                                                 Yest Erroring Default - Unicode (UEY-8)
                                                                                                                                                                                                                                                 Line Endings Default - mac 60 / Unix (SF
                                                                                                                                                                                                                                                  Intentibles Spaces
                                                                                                                                                                                                                                                        Militina d T Indiana
                                                                                                                                                                                                                                                               0 0 0
                                                                                                                                                                                                                                               Betton - Intercepts touch events an
areas areastine message to a target
altjest when it's tapped.
                                                                        func applicatiosDidEnterOsckground[_application: VIApplication) {
    // Use this method to release shared resources, saws user data, invalidate
    timers, and store enough application state information to restore your
                                                                                                                                                                                                                                                            Segmented Control - Displans
multiple segments, each of which
functions as a displant button.
                                                        [2]
                                                                                                                                                   button pushed!
Button pushed!
Button pushed!
Button pushed!
Bussec from debugger: Terminated due to sional 15
                                                                                                                                                                                                                                                             Fest Reid - Linguist solube set
                                                                                                                                                                                                                                                 Text and sends an action message to a 
target object when ferum is appe
```

(XCode)



(Android Studio)



- * (Source Control)
- * Also called version control
- * Source control is partly about keeping code safe, partly about change management
- * Anyone ever had a work laptop stolen? Call me careless, but I've had 2

*

```
4562f26 Only allow active coupons to be applied
* 139Fda1 More verbase staging logs
* 538c65b Pass in custom value_in_cents

    3:23344 Send Payment#value_in_cents to controller for processing

* 3u4ed6e Added Payment#value_im_cents
agef794 User can enter a coupon
* e3u45ec Added active_model_serializer gem
* a3d73c6 Busic JS to handle coupon submission started
4aa0938 Coupon controller, etc tests
* 568:321 Tests for Coupon model
* 25dela5 Admin can toggle coupons as active or inactive
* 9835100 Setter Coupon callbacks, validations

    af743th Added rilify_blanks gen, nuke empty string values in Coupon

* 9981c9e Refactored coupon Endex/creation workflow
* 58-917e Basic coupon views and controller action for create
* 30fa57b Botter note text in tables per CCS feedback

    450eedd replaced test rollback data with production.

 Zu0c8c7 Merge pull request #113 from philosophie/rollbock-script
* f/32781 final touches on rollbock script
1 * ue21589 tidied up rollbuck script.
* a338237 wrote event transition and script to move whap doc cases from awaiting_clean_up to really_opened
9chbds small bugfix, include HistoryFresenter in PostFeminderClecnibQueueFresenter
4865e6a notes count only displays number in tables
599898f gross fix to remove factor from notes page
```

- * (git)
- * Most people here have probably been exposed to git
- * Git was a leader in what's called "distributed" version control
- * Make sure you're committing often in small chunks
 - * Makes it less likely you'll lose changes + easier to follow along
- * Learning more than the basics of git allows you to maintain a clean history of what happened in the codebase. This is documentation for you and other developers
- * Also have nifty tools like git-bisect which lets you step through a series of commits to find which one introduced an error

Hard Skills

Tools

→ Practices

Principles

Soft Skills

Team

Clients

Self

(Moving onto practices)



- * (Testing)
- * Broadly speaking, testing is ensuring your code works as expected by running it
- * Manual testing is running by hand. Starts out manageable, but the more functionality you have to check the less manageable it becomes
- * When possible, you want to write automated tests basically additional test code that you run to do the checking for you
- * Running frequently while you develop increases confidence that you're not breaking things
- (Talk about tradeoffs of too little vs too many tests)
- * (Talk about tests that are too-specific)
- * Downside: One famous computer scientist, Edsgar Dijkstra, pointed out that "testing can be used very effectively to show the presence of bugs but never to show their absence"

Types of Automated Tests

Unit

Integration

- * Acceptance
- * End-to-end

BONUS: Property-based

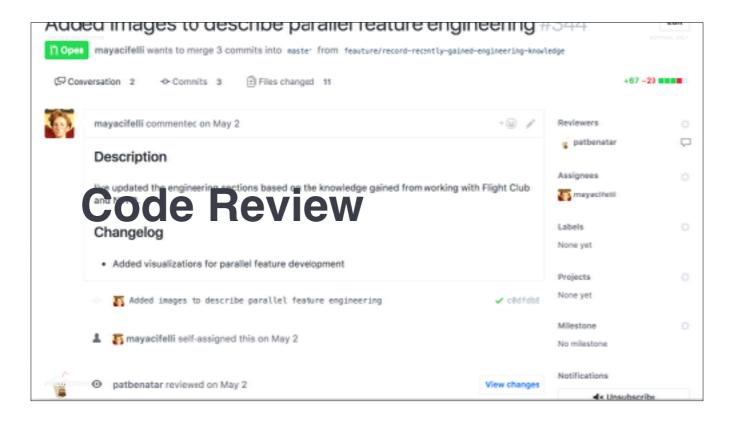
- * (Types of tests)
- $^{\star}\,$ End-to-end tests are basically a kind of acceptance test
- * Other types too: smoke tests, load tests

TDD =
Test Driven Development

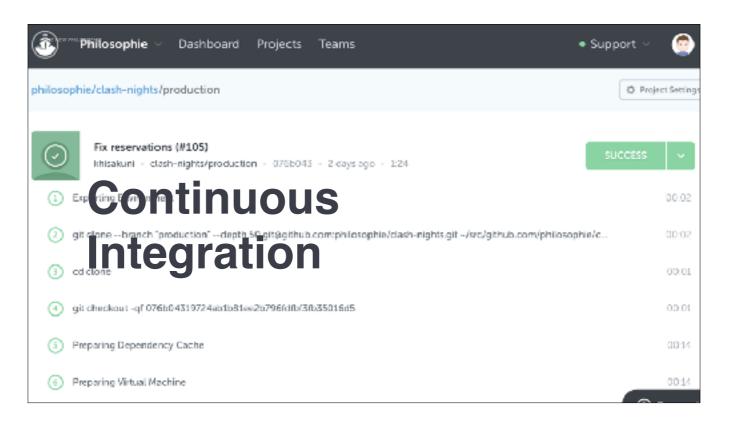
(TDD)



- * (Refactoring)
- * Improving the design of existing code (mainly so it's easier to keep working with it as things inevitably change)
- * You need to have a support structure in place, whether it's tests, types, or both, to refactor effectively
- * As much as possible, try to refactor as you go so you don't have a ton of technical debt



- * (Code Review)
- * Provided by a service like github, gitlab, bitbucket, etc
- *, i.e. "CR", also called "PR" (pull requests or peer review) or "MR" (merge requests)
 - * Increases code quality
 - * Catches errors
 - * Introduces you to new concepts if the reviewer has more experience in the tech you're writing in
- * Make sure you're providing objective, constructive criticism and make sure to point out good work!



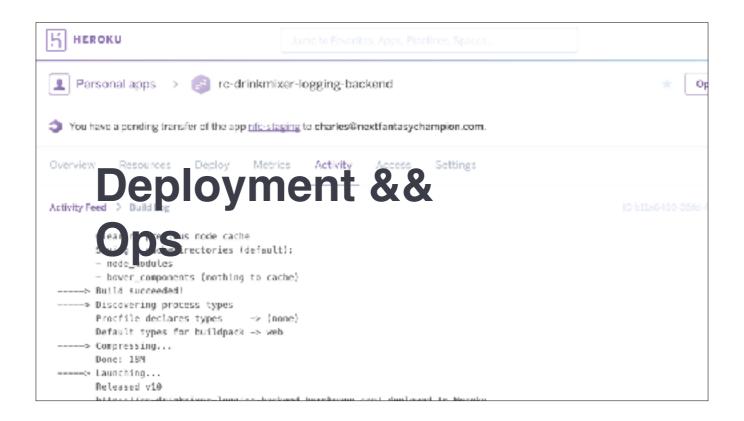
* (CI)

* You aren't the only one working in the codebase anymore

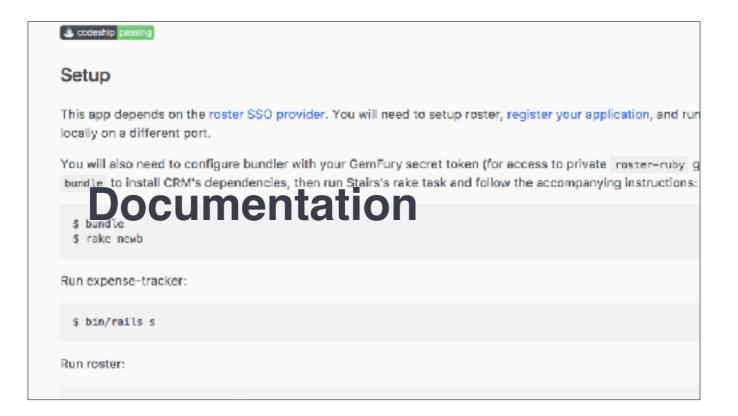
* (What this entails)

* Services: Codeship, CircleCI, Travis CI

* Tools: Jenkins



- (Deployment and Ops)
- * "Ops" == operations. Basically means system provisioning (i.e. setup) and administration
- * Tools like Heroku let you not worry as much about provisioning and managing servers
- * Tools like docker are shifting the paradigm
- * Just know that there's a lot of knowledge in ops; you can go down the rabbit hole as deep as you'd like, but you don't necessarily HAVE to these days (as long as your client can afford managed solutions)
 - * Time / Value tradeoff



- (Documentation)
- * Be wary of comments in code; try to make your code itself self-documenting if possible with:
 - * Good naming
 - * Tests
 - * Types (if appropriate)
- * Good technical documentation gives context. The lower-level the thing your explaining, the less likely you need to be explaining how it's doing what it's doing
- * Presentationally, markdown is a good tool to have in your tool belt for writing documentation
 - * As a bonus, it's a great markup format for taking notes too!

Hard Skills Tools

Practices

" Principles

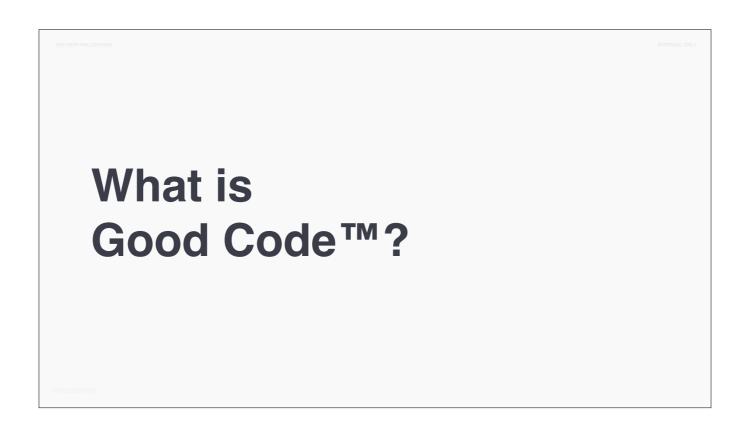
Soft Skills

Team

Clients

Self

- (Principles)
- * I tried to come up with principles that aren't specific to one language or framework
- * Guiding principles. Some of them may contradict; it's only through experience that they'll make sense. Treat them as guidelines, not hard rules



* What is Good Code?

Good Code™: Readable

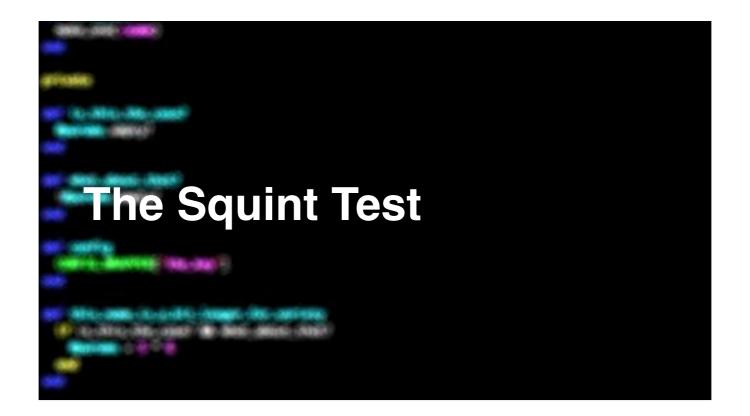
- * (Readable)
- * Code is read much more often than it is written
- * Like other forms of communication, you want to consider your audience. Don't apply some fancy technique or use some arcane knowledge if it's not common knowledge among the people who will be maintaining the code
- * There's definitely something to be said for educating people though what was once arcane knowledge can become common knowledge

Good Code™: Functional

- * (Functional)
- * Not in the functional/OO sense (though FP can help)
- * Functional in the sense that it does what it's supposed to do / serves the appropriate function

Good Code™: Consistent

- (Consistent)
- * Style
 - * Conventions
 - * Variable naming
 - * Indentation
 - * Enforce it, where possible (i.e. linting)
- * Keep functions/methods/classes short
- * Group related code
- * SQUINT TEST: http://robertheaton.com/2014/06/20/code-review-without-your-eyes/



SQUINT TEST: http://robertheaton.com/2014/06/20/code-review-without-your-eyes/

Good Code™: Simple

- * (Simple)
- * YAGNI -- you ain't gonna need it
- KISS -- keep it simple, stupid.
- $^{\star}\,$ Simple code does exactly what it needs to do no more and no less
- * You usually have to start with complicated code and pare it down to simple code via refactoring

Good Code™: Maintainable

- (Maintainable)
- * Overall, good code is code that you can maintain over time.
- * It's not overly complicated
 - * "Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."
 - * Newcomers to the codebase won't hate you
 - * You won't hate you when you have to revisit the codebase in a year and you've forgotten everything about it
- * It's flexible enough to handle changes
 - * i.e. don't make concrete decisions about things until you have to
 - * make your code easy to remove as things change

Don't Repeat Yourself (DRY)

- * (DRY)
- * Rule of 3

Don't Repeat Yourself (DRY)

Deal With The Pain Until You Can't

- (Deal with the pain)
- * Think VERY hard about the abstractions you're using. Try to derive them from actual code + the domain

Interfaces > Implementations

- * (Interfaces > Implementation)
- * Where possible, code to an interface for what something should look like rather than the concrete something
 - * Then you can easily drop in the thing or later swap it out with something else
- * Having experience with a variety of common data-level abstractions makes this easier / less likely to have you coding to the wrong abstraction. Just programming more makes this easier, but it's kind of an experience thing

Single Responsibility Principle

- (Single Responsibility Principle)
- * ,i.e. make sure a unit of code (i.e. class, module, function, whatever) has a single, well-defined purpose
- * refactor in this direction -- when you see things getting bloated w/ multiple responsibilities, it's time to break them out.



- * (Tradeoffs)
- * There are ALWAYS tradeoffs
- * From a technical perspective there are tradeoffs like time vs memory consumption
- * From a broader perspective, there is the triple constraint of Time, Cost, and Quality you can pick any two together but the third will suffer
- * Seen from an implementor's POV this usually translates to just Quality vs Speed. You have to be mindful of what mode you're operating in and how to operate in that mode to deliver as effectively as possible.

Hard Skills

Tools

Practices

Principles

Soft Skills

ਜ਼ Team

Clients

Self

(Team)



- * (Agile Process)
- * Skills re: agile process are all about learning how to move with your team in small, focused intervals toward mini-goals so you can get tighter feedback cycles from your users and/or the client
- * Try to keep meetings minimal (i.e. morning standup) + checkpoint

Team Communication

- (Team Communication)
- * Trello is also a form of living team communication if used effectively
- * Other tools: slack, google docs, sheets, drive

Hard Skills

Tools

Practices

Principles

Soft Skills

Team

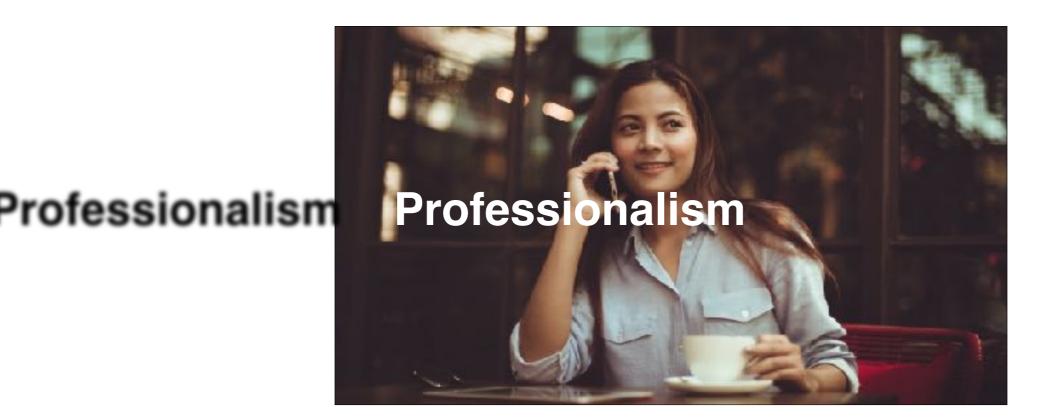
7 Clients

Self

- * (Clients)
- * "Client" doesn't have to be an external client (i.e. internal software built for other groups in the organization)
- $^{\star}\,$ Client may or may not be the end user of the software
- * Either way you should be regularly communicating with whomever your client is



- * (Team <-> Client)
- * At Philosophie one of our company values is "same team", i.e. the client is on our team and we're on theirs
- * Constantly be trying to empathize with their needs and wishes
- * The more you can be "in the trenches" with the client, the more empathy you will have for them this is a lesson I learned over time
- * And the more time the client spends with you, the more they'll understand all that goes into making a software product.
- * Tools: slack, trello, google docs, sheets, drive
- * Talked about documentation before but that was more technical documentation. For client facing documentation, think about your audience
- * I've never once regretted putting in more time in client documentation always get feedback that they appreciate / understand in more detail



- * (Professionalism)
- * Show up to meetings on time or ahead of time
- * Try to maintain a good rapport
- * Delivery early, deliver often, get lots of feedback
- * Always look for ways to delight
- * Make your expertise known don't flaunt it, but make people aware that you know what you're talking about

Hard Skills

Tools

Practices

Principles

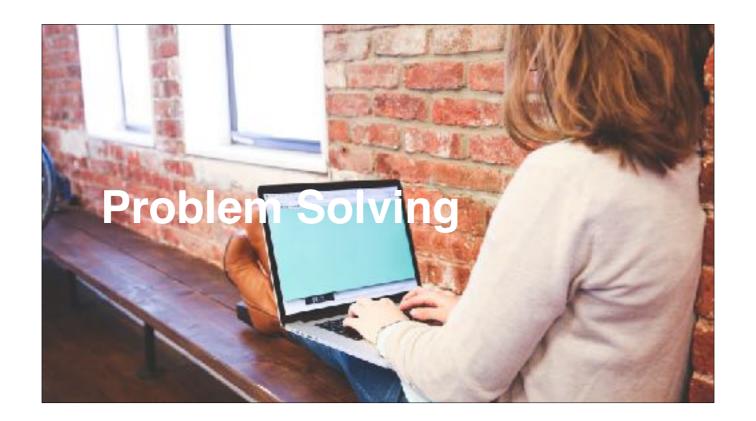
Soft Skills

Team

Clients

∏ Self

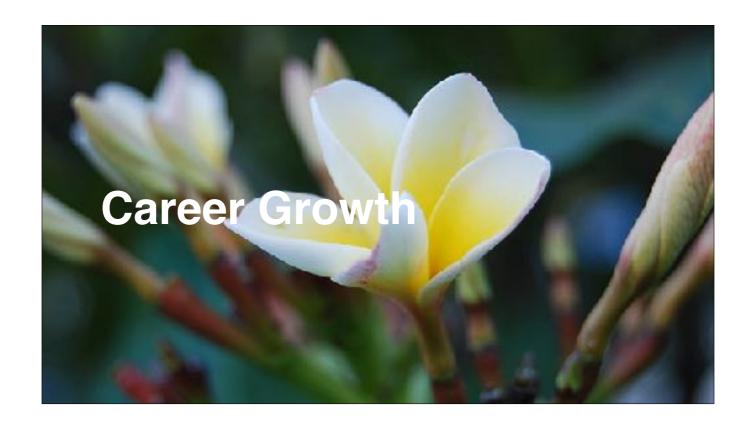
(Self)



- (Problem Solving)
- * Definitely takes tenacity
- * Have to be able to approach things from a LOT of different angles
- * Usually it helps to have a scientific mindset / approach, especially with, i.e., debugging
- * How can I subdivide the problem? What tests can I run to get me closer to a solution?
- * Sometimes you have to flip the problem on its head to think of it differently
- * Sometimes you have to draw parallels to processes or systems completely outside of the domain your working in. This requires experience
- * Sometimes you just need to step away from the problem for a bit and come back later
- * Don't be afraid to ask for help from people who are more familiar with the problem domain (or even people who aren't!)



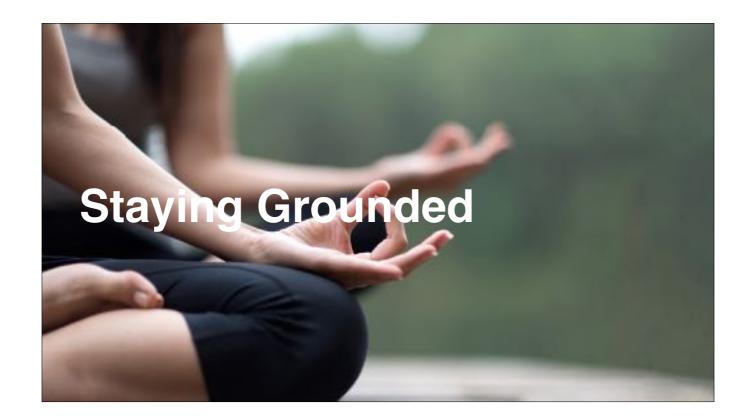
(Constant Learning)



- * (Career Growth)
- * Some good advice I've heard: when thinking about what you want in a job, think about what job you'll be applying to AFTER the current job. That will tell you what skills you want to be picking up in the current job you're applying to

Work/Life Balance

- * (Work / Life Balance)
- * Burnout is a real thing and tends to affect people in software development more frequently than in a lot of other professions
- * You have to actively combat it by regularly coping with stress and occasionally recharging for longer periods



- (Staying Grounded)
- * I personally recommend meditation and mindfulness practice
- * Need to be able to start your day with a clear head
- * Be as effective as you can during working hours so that you can leave with a clear mind
- * As much as possible, try to let work be work and home be home



- * (Hobbies)
- * Hobbies also help I'm personally a big fan of board games
- $^{\star}\,$ BONUS: increase sense of community in the office. We play lunch games practically every day



- * (Family and Friends)
- * Having a solid support network of family and/or friends also helps a lot
- * Remember that you're an actual person, not a machine. You've got to come up for air
- * Plan for time off because the pace of work will probably not slow down

NEW PHILOSOPHIE INTERNAL ONL

Thank You!

PHILOSOPHI