VICTOR AUGUSTEO

SOFTWARE CRAFTSMANSHIP AND YOU

hi everyone, welcome to the talk you've been waiting for! please put your hands up if you consider yourself software developer. how about software engineer? why do you consider yourself that? how about software craftsman?

now my purpose is to hopefully convert more of you into craftsmen way of thinking.

for the non-devs, this talk will still beneficial for you as you can be design craftsmen or hardware craftsmen.

WHO'S THIS GUY?

- Victor Augusteo
- Senior iOS Dev @ Tabcorp
- Mostly Apple and web apps
- augusteo@gmail.com
- linkedin.com/in/victoraugusteo

two parts talk:

- what is software craftsmanship and how does it help you

- my goal is by understanding this may we produce better software with fewer bugs and better customer experience

second part is

- how sharing and teaching will help you learn and grow
- my goal is you may learn more efficiently and share more of what you learned.

they are very closely related to each other and the concepts can be applied to both techies and non-techs alike.

Why not engineering?

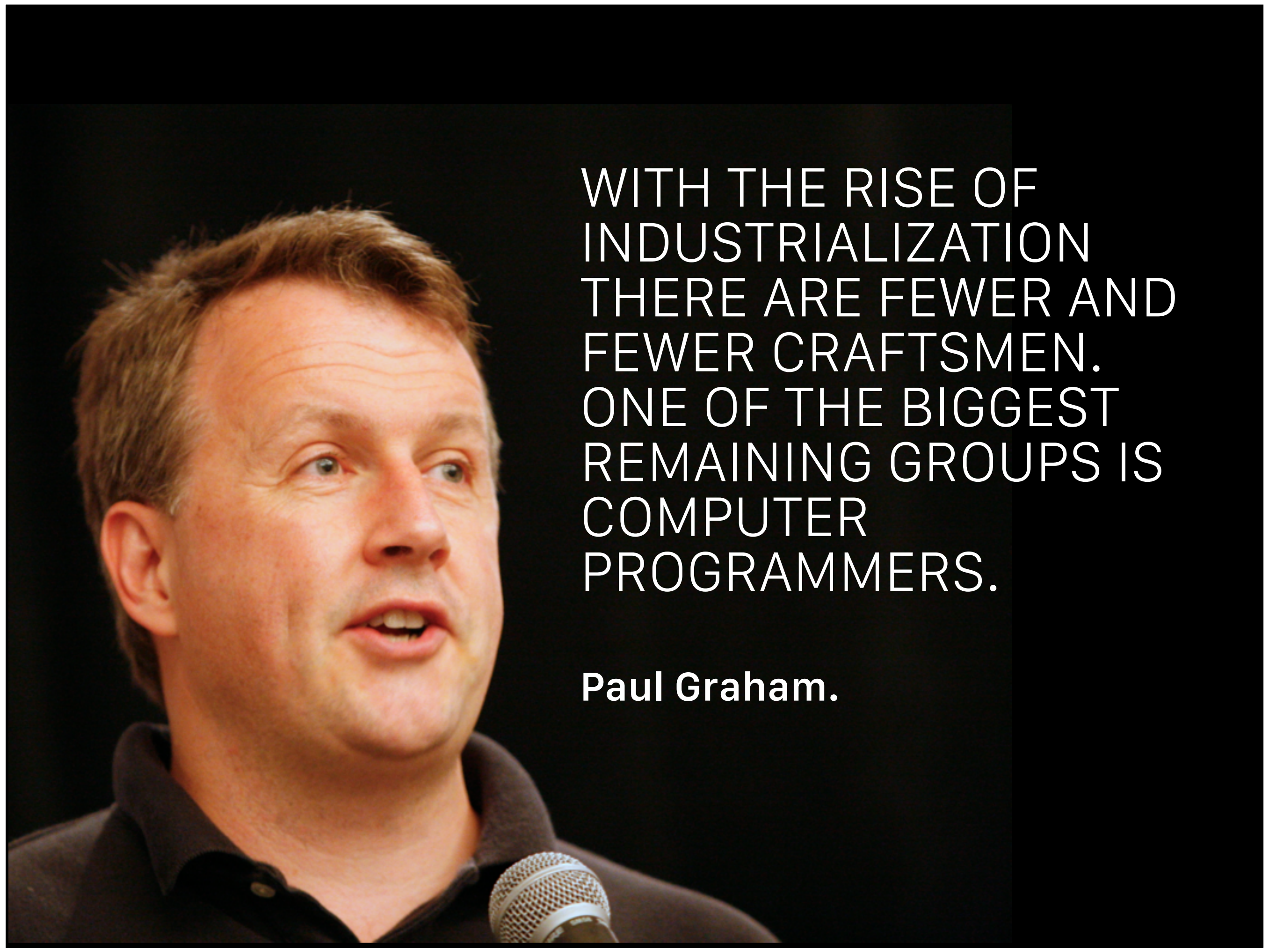
first of all, why not software engineering?

engineering:
-treats devs equally as automatons/resources, but as we know, even with equal year of experiences, devs are vasly different
-assumes highly degree of control and predictability, but software is often unpredictable.
-approved/set body of knowledge, exams and certification before we can call ourself engineer, we don’t really have that in software
-generally end-goal is set in advance, like a bridge from A-B or nuclear power plant. software doesn’t have that luxury and generally changes requirements daily

craftsmanship places the craftsman at the centre of the process

Craftsmen

– Dedicates their working life to getting better at the craft
– Learns their skills through apprenticeship to a master craftsman
– Aims for mastery of the craft
– Teaches others through apprenticeship
– Builds a reputation based on what they deliver • Not what exams they have passed
- passion for your craft is optional. it helps, but definitely you can be successful craftsman without passion.



WITH THE RISE OF INDUSTRIALIZATION THERE ARE FEWER AND FEWER CRAFTSMEN. ONE OF THE BIGGEST REMAINING GROUPS IS COMPUTER PROGRAMMERS.

Paul Graham.

so who is this guy and why should we listen to him?

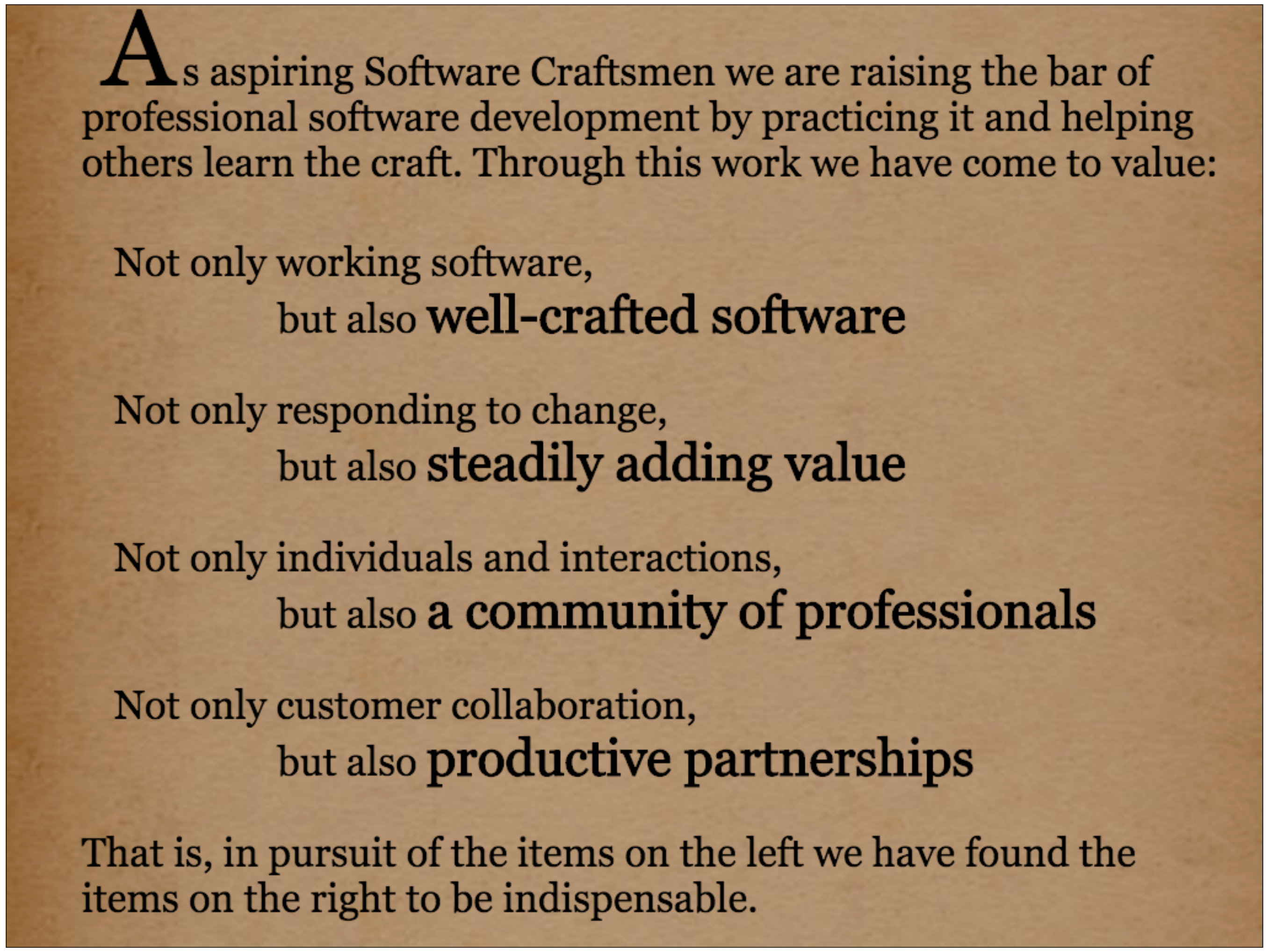
he is cofounder of YCombinator, the famed startup incubator that launches thousands of profitable startups. he writes many esteemed essays. this quote is from one titled ‘How to make wealth’

concept isn't new. the classics embrace it.

Pragmatic programmer's subtitle is 'from journeyman to master' and it is filled with extremely useful tips and tricks to help you master the craft.

Software craftsmanship new imperative talks about the situation that we are facing now. As the demand for software has exploded, the software engineering establishment has attempted to adapt to the changing times with short training programs that teach the syntax of coding languages. But writing code is no longer the hard part of development; the hard part is figuring out what to write. This kind of know-how demands a skilled craftsman, not someone who knows only how to pass a certification course. still true 15 years after the book is published ya.

and the apprenticeship patterns writes itself similar to the design patterns and refactoring books. it goes through many patterns that we can follow on the path to mastery.


As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

Not only working software,
but also **well-crafted software**

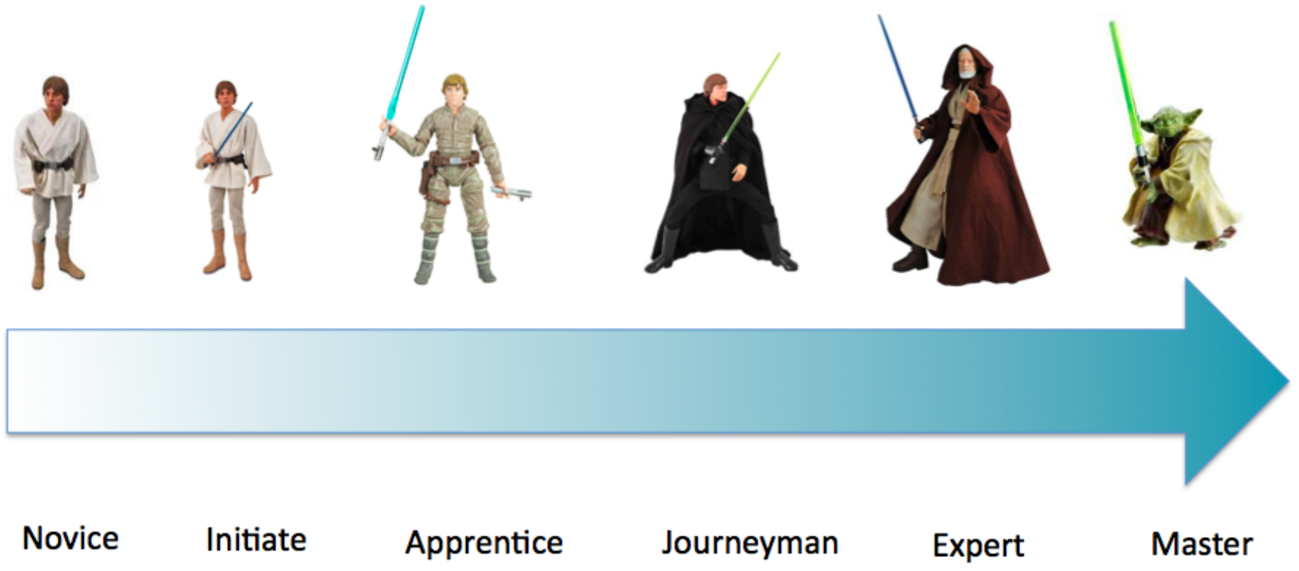
Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

Not only customer collaboration,
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

copy pasted from the ‘software craftsmanship manifesto’ website. just google it.
a followup of the agile manifesto.



Hoffman, 1998

i started as novice back in high school around 15 years ago. developing simple html-js webpages for my MMO guild.

My proper apprenticeship started when I enrolled into uni degree because that's where i started to learn from masters.

so what's apprenticeship anyway?

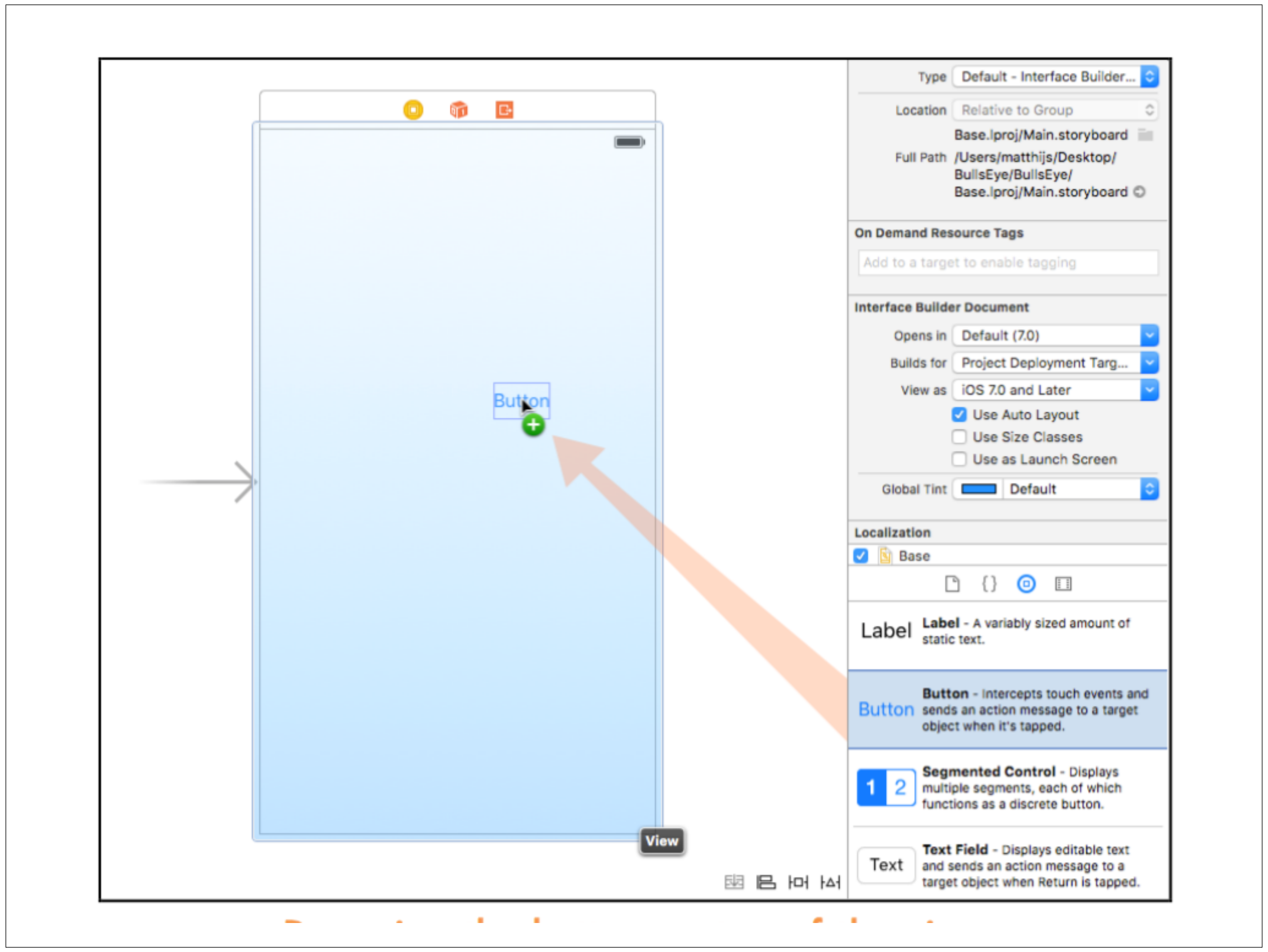
apprentice is the beginner. the student. the squire.

the most important thing for an apprentice to do is learn from the right people.

in game of thrones or medieval era, an apprentice who wants to learn new craft will have to look for a master to study from. just like podrick learns better from brienne to be a knight, and not from tyrion. Tyrion is a master in his own right, but definitely not master of swordfighting.

thankfully for us, with the advent of the internet and printing press, it's easy to learn from the masters of the craft through books, videos and courses.

to recap, Apprenticeship is the state/process of evolving and looking for better ways and finding people, companies and situations that force you to learn those better/ smarter/faster ways.


Type Default - Interface Builder...

Location Relative to Group

Base.lproj/Main.storyboard

Full Path /Users/matthijs/Desktop/BullsEye/BullsEye/Base.lproj/Main.storyboard

On Demand Resource Tags

Add to a target to enable tagging

Interface Builder Document

Opens in Default (7.0)

Builds for Project Deployment Targ...

View as iOS 7.0 and Later

- [x] Use Auto Layout
- [ ] Use Size Classes
- [ ] Use as Launch Screen

Global Tint Default

Localization

- [x] Base

Label — **Label** - A variably sized amount of static text.

Button — **Button** - Intercepts touch events and sends an action message to a target object when it's tapped.

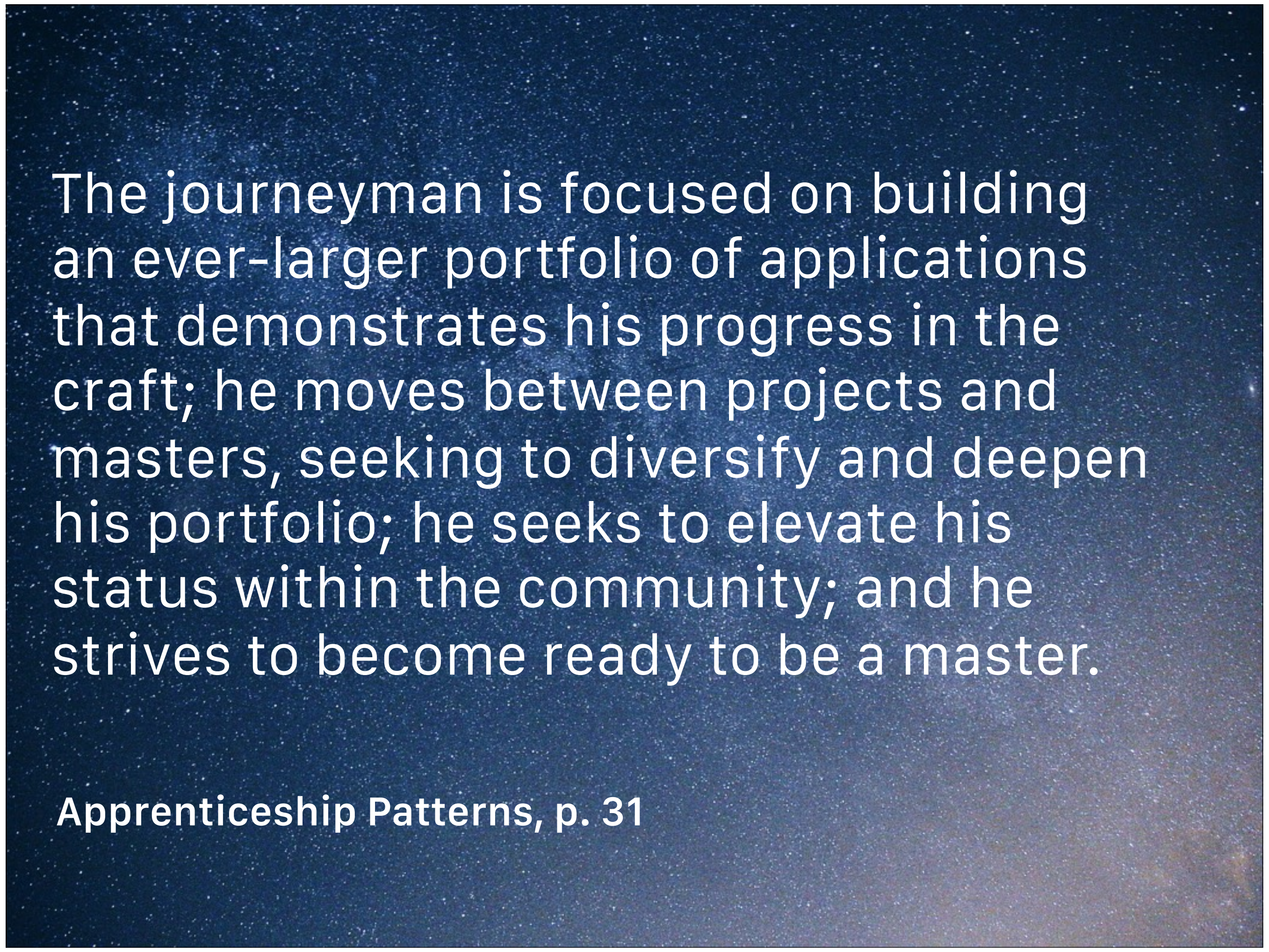
1 2 — **Segmented Control** - Displays multiple segments, each of which functions as a discrete button.

Text — **Text Field** - Displays editable text and sends an action message to a target object when Return is tapped.

in iOS, apprentices would learn about

1. interface builders and iOS controls
2. swift basics, OOP, MVC
3. CocoaTouch framework

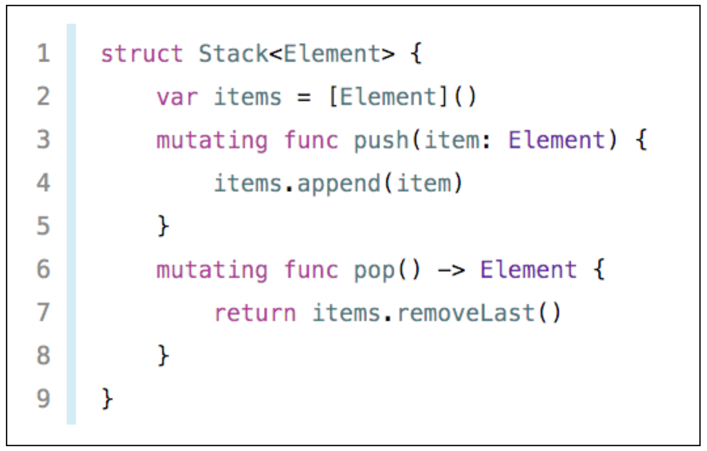
so what's the next step after apprenticeship?



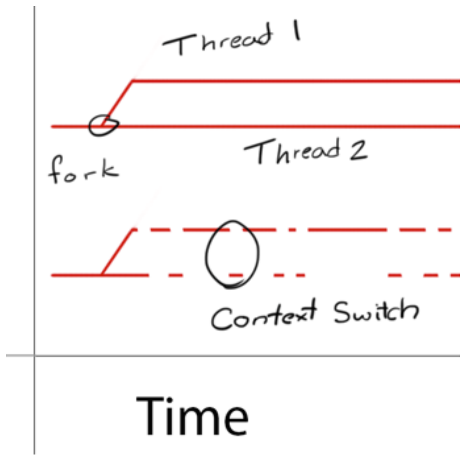
The journeyman is focused on building an ever-larger portfolio of applications that demonstrates his progress in the craft; he moves between projects and masters, seeking to diversify and deepen his portfolio; he seeks to elevate his status within the community; and he strives to become ready to be a master.

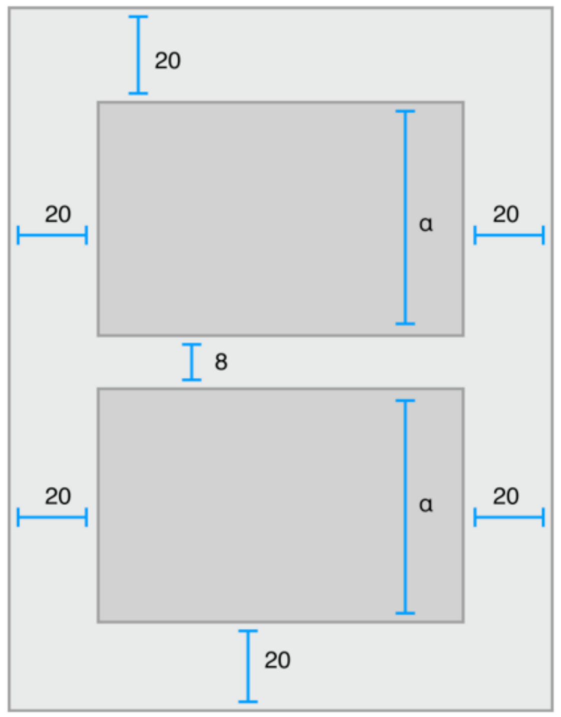
Apprenticeship Patterns, p. 31

i consider myself a journeyman. good enough to be paid to craft software, but not good enough to be a master yet.



```swift
struct Stack<Element> {
    var items = [Element]()
    mutating func push(item: Element) {
        items.append(item)
    }
    mutating func pop() -> Element {
        return items.removeLast()
    }
}
```





journeyman iOS devs would be expected to master

1. swift generics and protocol oriented programming
2. autolayout and size classes in code and storyboards
3. networking - parsing/sending Json
4. GCD and threading
5. the list grows everyday.

lastly we have the masters.
in the old days, a expert journeyman has to produce a ‘Masterpiece’ that has to be accepted by other masters before allowed to join the guild.

in software craftsmanship, this would be the people who has mastered their craft, produced masterpieces and shares their superior skills to advance their professions and the wider world. there is no certificate for it. we can’t just copy the masterpiece either, because you have to create your own.

on scale of 1-10, on a good day, you might rate yourself as 8s or 9s. but once you see the works of the masters, you realise the scales goes up to 100s or 1000s.

1. linus torvalds is one of those people. his masterpiece is of course linux and git which changed the whole computing landscape forever
2. chris lattner for creating LLVM, CLang, and Swift. the iOS would be a very different world without him
3. richard stallman for creating the GNU projects which is the base for Linux kernel; and the Free Software Foundation that started the open-source movement
4. many others!

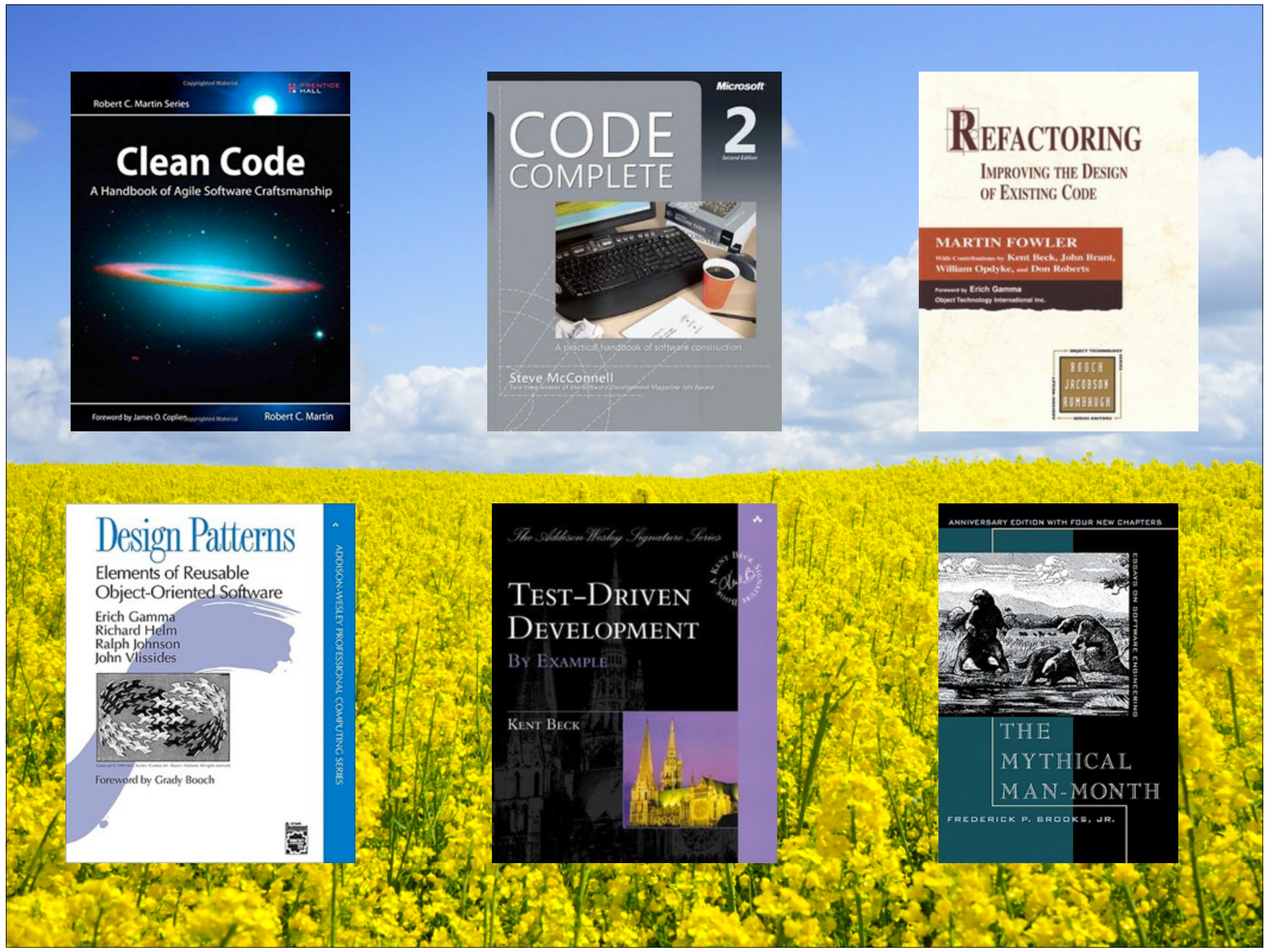
part 1 summary:
we want to grow from apprentice to master. maybe to the point where thousands of people enjoys and benefit from our work.

software development is more craft than engineering
as craftsman we strive to better ourself through path of mastery

any questions for this before we move on to the second part?

so now we know that to progress from journeyman to master, we need to grow our skill. did you know that sharing those skills will actually help you grow as well as help others grow?

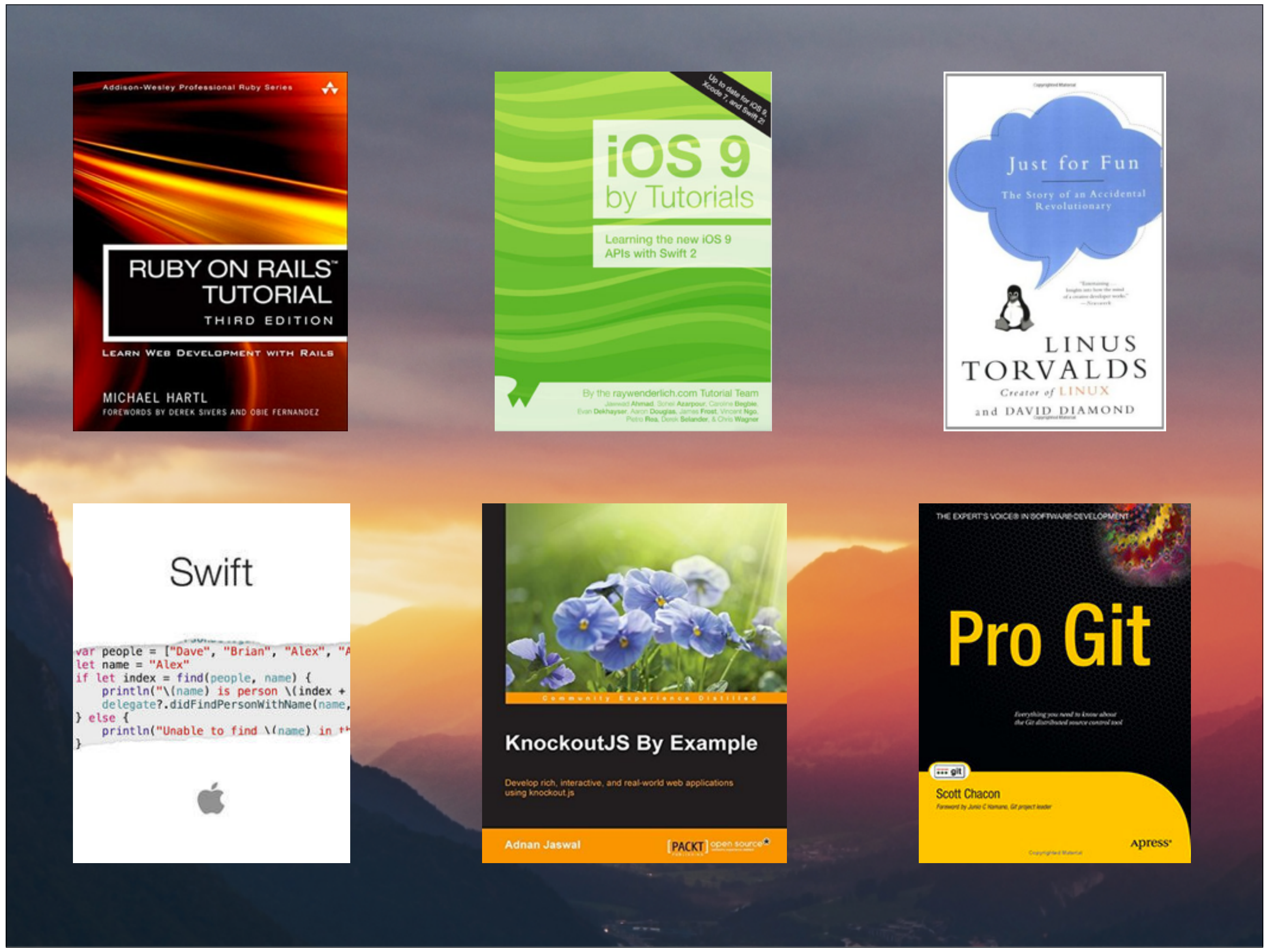
before you share, you need to learn first. the simplest way to learn is by reading books of course. Personally I learn much of my dev skills from books. Do not read those books like a novel; from start to finish. try to understand parts of it and try them out. and comes back to it later on.



here are some books that i recommend for learning more about software development, in addition to the 3 that we saw about software craftsmanship.

how many of you has read more than 3 books? 2? 1?

i've read one of those and currently reading 'refactoring'. sometimes it's hard to find application in our daily life.


Addison-Wesley Professional Ruby Series
RUBY ON RAILS™ TUTORIAL
THIRD EDITION
LEARN WEB DEVELOPMENT WITH RAILS
MICHAEL HARTL
FOREWORDS BY DEREK SIVERS AND OBIE FERNANDEZ
iOS 9 by Tutorials
Learning the new iOS 9 APIs with Swift 2
By the raywenderlich.com Tutorial Team
Just for Fun
The Story of an Accidental Revolutionary
LINUS TORVALDS
Creator of LINUX
and DAVID DIAMOND
Swift
KnockoutJS By Example
Develop rich, interactive, and real-world web applications using knockout.js
Adnan Jaswal
PACKT open source
THE EXPERT'S VOICE® IN SOFTWARE DEVELOPMENT
Pro Git
Scott Chacon
Apress®

those are the books that i have read and can personally recommend.

so what do we do beside reading books? practice of course!

the old adage says ‘Practice makes perfect’. but not all kinds of practice works the same way. what we want to do is deliberate practice. it has to be:
improving specific skill, repeated with continuous feedback and it must be challenging/hard.
there is no point repeating something that you already mastered over and over again.

one great way to practice is by doing Code Kata. Kata is a pattern in karate that can be practiced alone over and over again.
one website with kata template is codekata.com
each kata will be a short exercise 30-60minutes long and can be solved in multiple ways. just remember to get feedback/review to improve as doing it in isolation can develop bad habits.

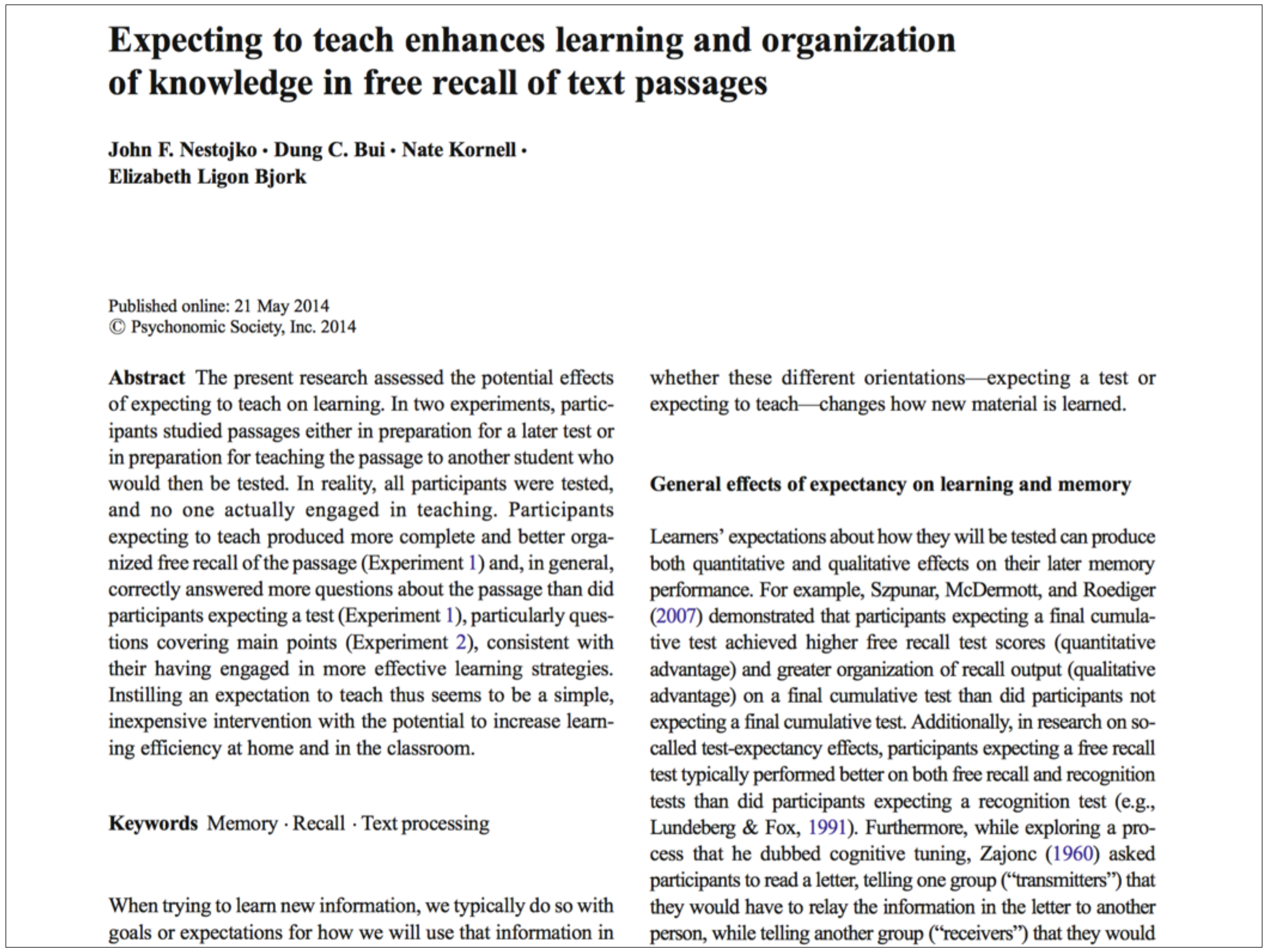
you can also create your own katas and share it with us!

now beside growing your skill through study and practice, you must also share you learnings with others.
why is it so important?

imagine that you have two groups of people. on the left you have team mango. on the right team avocado.
they are about to attend a lecture.
team mango was told that they will have to teach the material to others.
team avocado was told that they will be tested on this.

which team do you think scores better on a test?
hands up if you think team mango wins.
hands up if you think team avocado wins.

team mango performs significantly better in the test.



Expecting to teach enhances learning and organization of knowledge in free recall of text passages

John F. Nestojko · Dung C. Bui · Nate Kornell · Elizabeth Ligon Bjork

Published online: 21 May 2014

Abstract The present research assessed the potential effects of expecting to teach on learning. In two experiments, participants studied passages either in preparation for a later test or in preparation for teaching the passage to another student who would then be tested. In reality, all participants were tested, and no one actually engaged in teaching. Participants expecting to teach produced more complete and better organized free recall of the passage (Experiment 1) and, in general, correctly answered more questions about the passage than did participants expecting a test (Experiment 1), particularly questions covering main points (Experiment 2), consistent with their having engaged in more effective learning strategies. Instilling an expectation to teach thus seems to be a simple, inexpensive intervention with the potential to increase learning efficiency at home and in the classroom.

Keywords Memory · Recall · Text processing

When trying to learn new information, we typically do so with goals or expectations for how we will use that information in whether these different orientations—expecting a test or expecting to teach—changes how new material is learned.

General effects of expectancy on learning and memory

Learners' expectations about how they will be tested can produce both quantitative and qualitative effects on their later memory performance. For example, Szpunar, McDermott, and Roediger (2007) demonstrated that participants expecting a final cumulative test achieved higher free recall test scores (quantitative advantage) and greater organization of recall output (qualitative advantage) on a final cumulative test than did participants not expecting a final cumulative test. Additionally, in research on so-called test-expectancy effects, participants expecting a free recall test typically performed better on both free recall and recognition tests than did participants expecting a recognition test (e.g., Lundeberg & Fox, 1991). Furthermore, while exploring a process that he dubbed cognitive tuning, Zajonc (1960) asked participants to read a letter, telling one group ("transmitters") that they would have to relay the information in the letter to another person, while telling another group ("receivers") that they would

"When teachers prepare to teach, they tend to seek out key points and organize information into a coherent structure," the author said said. "Our results suggest that students also turn to these types of effective learning strategies when they expect to teach."

this is by a study from St Louis Washington University.

even the intention to teach will already improve your retention compared to just listens or read. so I want you to approach your learning from now on with different perspective.

i know most of us codes everyday and can’t really be bothered to write after we get home. but blogging is one of the most cost effective way to share knowledge. it benefit you too as you need to deconstruct what you have learned and structure it.

another good way to share is to do brown bag sessions in the office. or you can come to meetups like this one and give a short talk! come talk to us and we will help you with your talk.

- email me augusteo@gmail.com
- connect on linkedin.com/in/victoraugusteo
- http://psych.wustl.edu/memory/nestojko/NestojkoBuiKornellBjork%282014%29.pdf
- http://www.paulgraham.com/wealth.html
- http://manifesto.softwarecraftsmanship.org/
- images from: pexels.com & unsplash.com & raywenderlich.com
- book covers from goodreads.com

augusteo@gmail.com