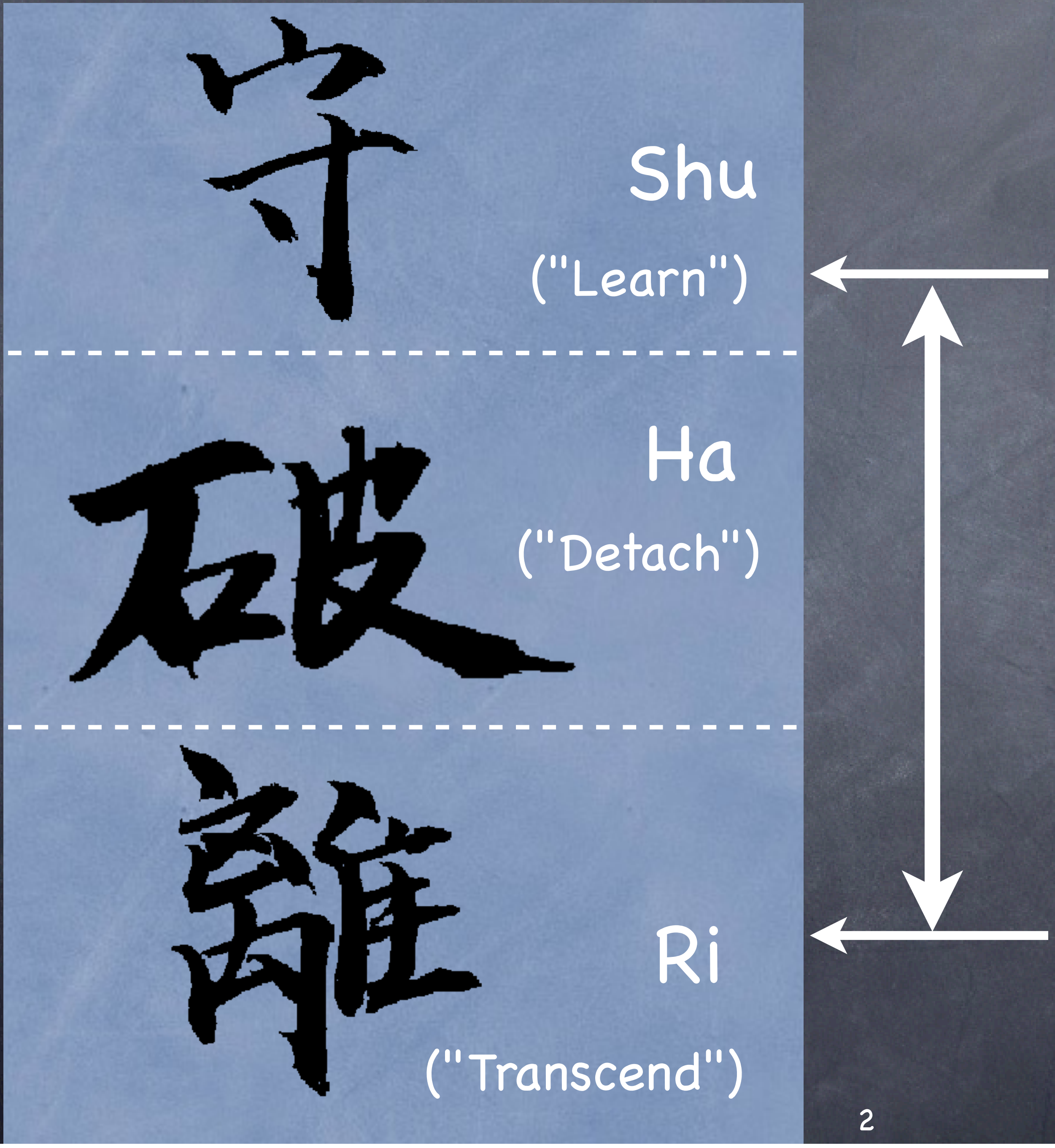
"Good Enough" IS Good Enough!

http://www.aleax.it/pybay16_geige.pdf

Google™

This talk's "level"



Some Cultural Assumptions...:

- everybody should always be striving for perfection at all times!
  - settling for a software release that's anywhere below "perfect!" is thus a most regrettable compromise.
- do you mostly agree with these...? OR...:
- keep-it-simple, just-good-enough
  - launch early, launch often!
  - get plenty feedback & LISTEN to it
  - iteratively improve, enhance, refactor...

"Worse is Better"

- Richard Gabriel, 1989, a Lisp conference
  - "New Jersey" approach, AKA "WiB"
    - vs
  - "MIT/Stanford" approach, AKA "The Right Thing"
- years of debate afterwards (plenty of it by RG, sometimes as "Nickieben Bourbaki")...
  - on BOTH sides of the issue!-)

All agree on what's good...:

- simplicity
- correctness
- consistency
- completeness

...but there are important differences in...:

- - exact definitions and nuances
- - priorities

Worse-is-better (e.g: Unix)

- simplicity
  - implementation (esp!) AND interface
  - most important consideration in design
- correctness
  - (slightly) better be simple than correct
- consistency
  - "not overly inconsistent"
- completeness
  - can be sacrificed to any of the top 3
  - MUST be, if simplicity's threatened

"The Right Thing" ("MIT")

- simplicity
  - esp. interface
- correctness
  - absolute-must, top priority
- consistency
  - just as important as correctness
- completeness
  - roughly as important as simplicity

Quoting RG himself...:

- The right-thing philosophy is based on letting the experts do their expert thing all the way to the end before users get their hands on it.
- Worse-is-better takes advantage of the natural advantages of incremental development. Incremental improvement satisfies some human needs...

G.K. Chesterton

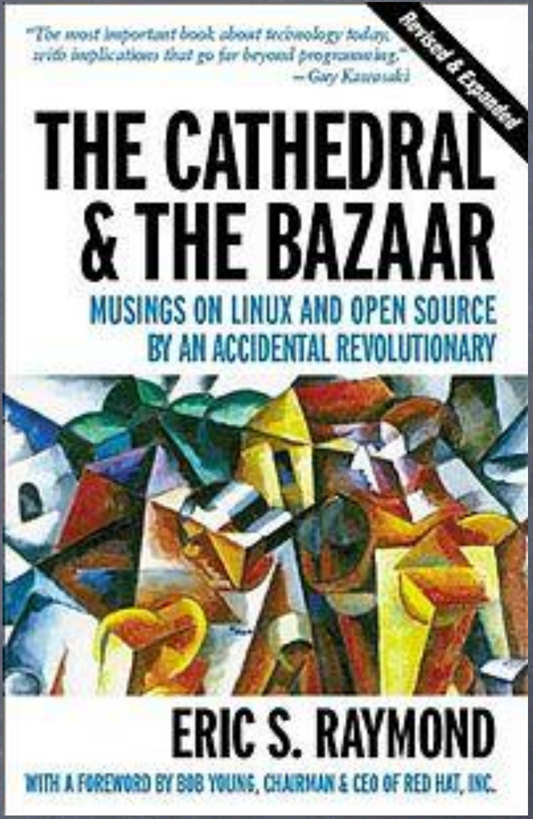
- Anything worth doing...
  - ...is worth doing badly!

Note to fontly critics

The proportional font I use is Apple Chalkboard, which is NOT MS Comic Sans :-)

Cathedral, Bazaar...?

- Eric Raymond, 1997
- focus: two diverging models of software development
  - Cathedral: close to RG's "right-thing" MIT/Stanford
    - experts in charge
  - Bazaar: chaotic, launch-and-iterate NJ-like models -- crowd in charge
- The core Bazaar idea: "given enough eyeballs, all bugs are shallow"


"The most important book about technology today, with implications that go far beyond programming."
—Gary Kawasaki

Revised & Expanded

THE CATHEDRAL & THE BAZAAR

MUSINGS ON LINUX AND OPEN SOURCE BY AN ACCIDENTAL REVOLUTIONARY

ERIC S. RAYMOND

WITH A FOREWORD BY BOB YOUNG, CHAIRMAN & CEO OF RED HAT, INC.

BUGS?! I don't DO bugs!

- my very first program ever WAS bug-free
  - 1974: 3 freshmen HW design majors and a Fortran program to compute conditional probabilities of suit-division in bridge
  - we had to punch it into punched cards
  - we got one-&-only-one chance to run it...!

- it ran perfectly that first-and-only-time...!
- ...never ever happened again in my life.
  - ...don't count on it, buddy...!-)

"Perfection" -> BDUF

- If you want to only release "Perfection",
  - you clearly need "Big Design Up Front"
- everything must proceed top-down,
  - perfect identification of requirements,
  - begets perfect architecture,
  - begets perfect design,
  - begets perfect implementations,
- **(it takes...)** forever and ever, A-MEN!
- alas! real life doesn't tend to co-operate...
  - stakeholders resent the "forever" part!-)

BDUF vs the real world

- requirements change all the time
  - you ain't ever gonna nail them perfectly!
- architecture varies with design choices
- design varies with implementation techs
- implementation _always_ has some bugs
  - only discovered in real-world deployment

-->

- ITERATIVE development's the only way to go!
  - deploy SOMEthing, fix bugs, improve, ...
  - solve SOME user problems, win mindshare

Backwards Incompatibility

- ...is your FRIEND!-)
  - if you're constrained to remain backwards compatible forever,
  - early-stage design errors drag you down
- "good enough" IS good enough, IFF...
  - ...you can make it better later!
- e.g: `raise 'some string'` now raises `TypeError` instead (since Python 2.6)

"Perfect": verb, ¬adjective!

- **perfecting** your work is great
  - keep doing it -- based on real data!
- **perfection** is a process, NOT a state
  - you never "reach" it
  - goalposts keep shifting
  - no laurels to rest on!

What **not** to skimp on

- light-weight, agile **process** and its steps
  - revision control, code reviews, testing...
  - proper release-engineering practices
- code style, clarity, elegance
- documentation

no cowboy coding!

Must be in from the start

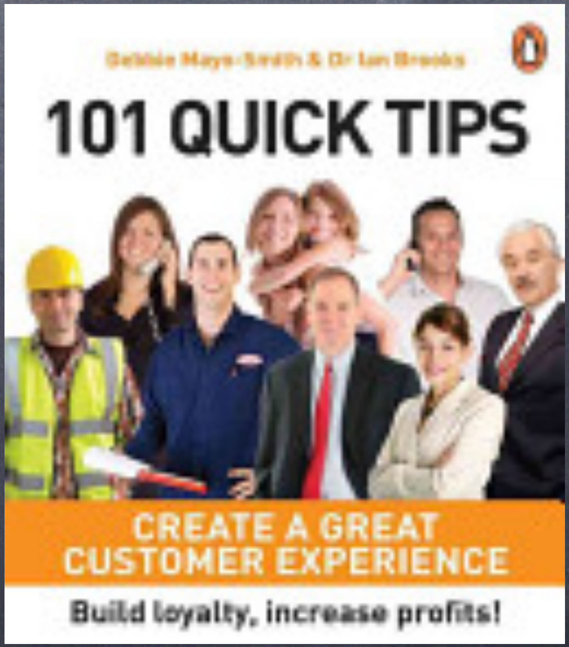
- security, in the most general sense, incl.:
  - privacy
  - auditability
- many other things would be `best` to have at the start, BUT you CAN refactor later...:
  - modularity, `plug-ins`
  - an API
  - scalability
- you CAN incur technical debt, _with care_
  - but, DO plan "repayment" as you go!

Recoverable or not?

- focus on avoiding potential errors that could cause <u>irrecoverable</u> losses
  - as long as one can/does recover, it's OK...
  - ...in a `beta`, at least!-)
- is the <u>reputational damage</u> to yourself recoverable...?
  - it depends! but, usually, YES
    - esp. w courteous, speedy response to issues that get reported (=="service")
  - "get it right the 2nd time" is usually OK

Customer service secret

- "Customers with the highest levels of satisfaction tend to be those who have had a problem resolved" -- even better than those who never had any problem at all!
- it's the "Service Recovery Paradox"
  - http://jsr.sagepub.com/content/10/1/60.abstract


Debbie Mayo-Smith & Dr Ian Brooks
101 QUICK TIPS
CREATE A GREAT CUSTOMER EXPERIENCE
Build loyalty, increase profits!

General vs ad-hoc solution

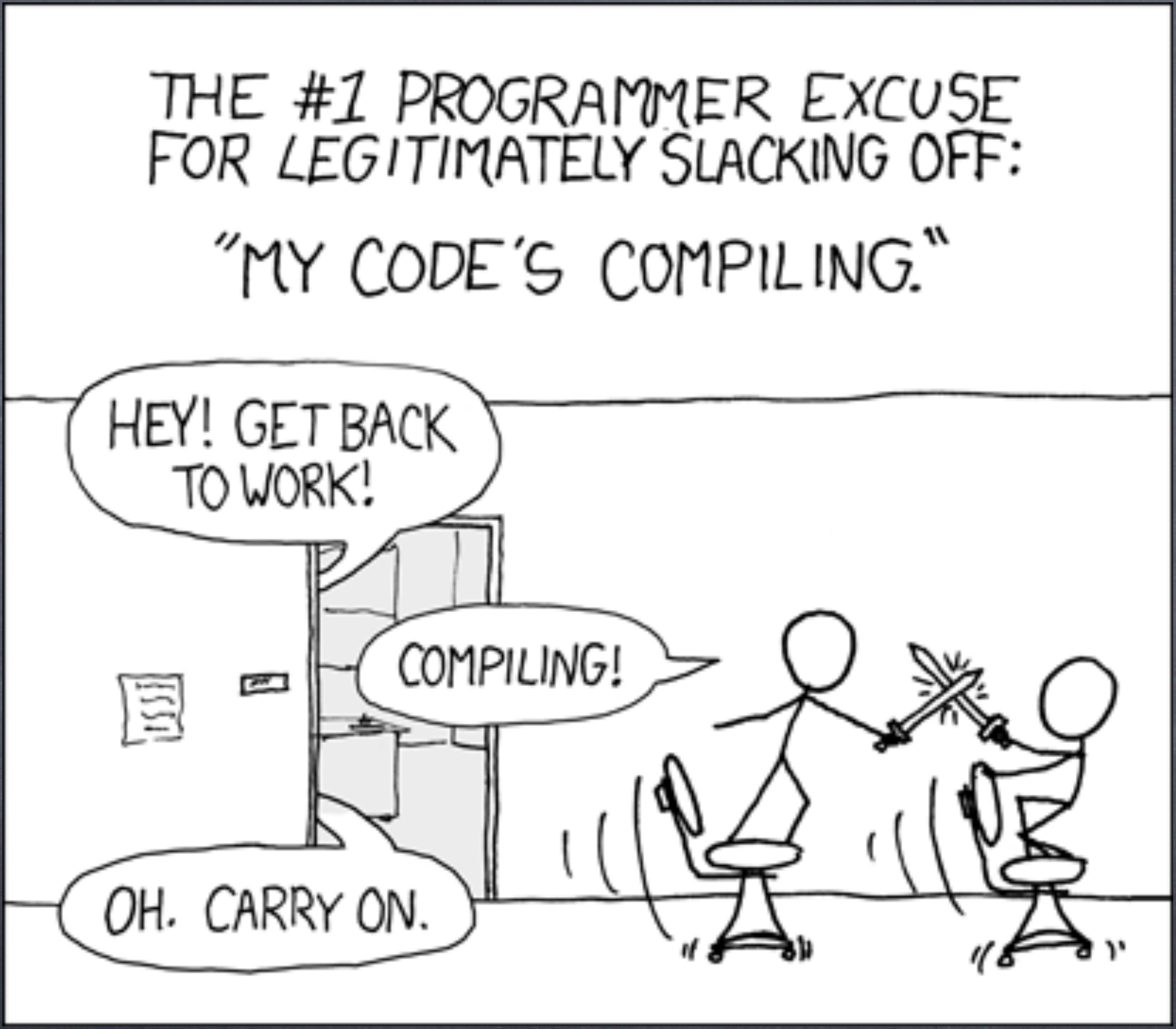
- intuition may tell us ad-hoc easier, faster
- reality: sometimes, but NOT always (DRY!)

```
def find_by_col(root, color):
  if root.color == color: yield root
  yield from (find_by_col(c, color) for c in root.cs)
def find_by_sha(root, shape):
  if root.shape == shape: yield root
  yield from (find_by_sha(c, shape) for c in root.cs)
```

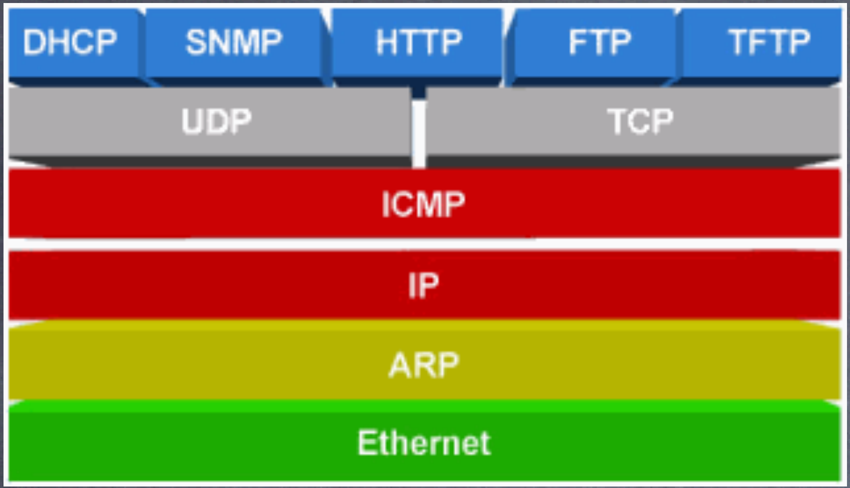
vs

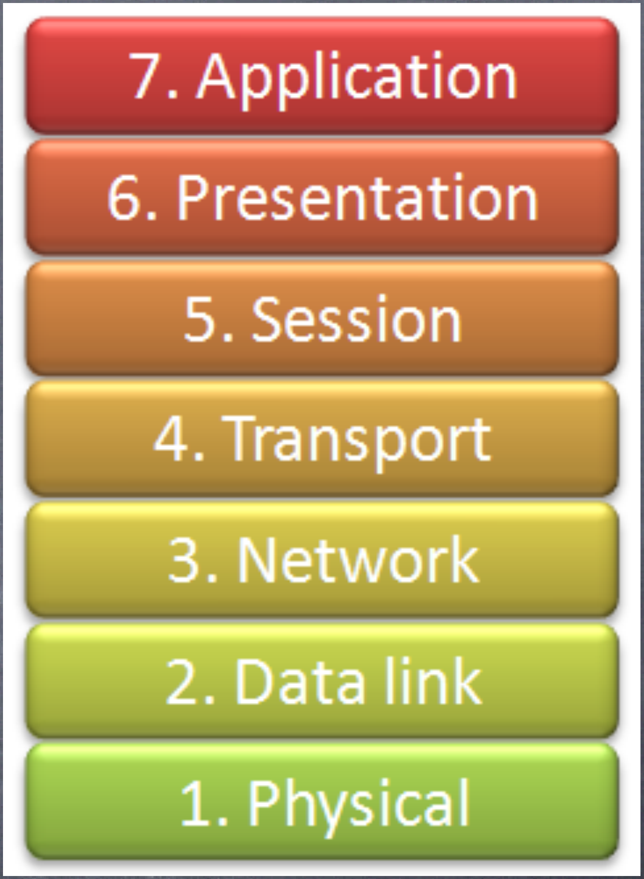
```
def find(root, n, v):
  if getattr(root, n) == v: yield root
  yield from (find(c, n, v) for c in root.cs)
```

WIB vs TRT: programming



TCP/IP vs ISO/OSI

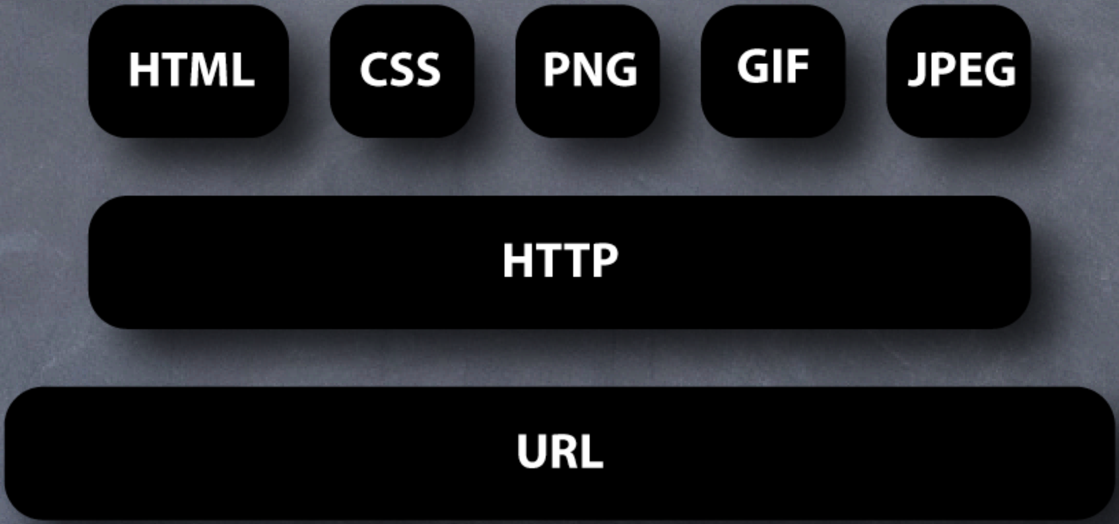




- rough consensus...
  - ...and RUNNING CODE (David Clark: MIT, but... IETF front and center!)

Xanadu vs the WWW

Perfect, ideal hypertext



Hackish, incrementally improved hypertext

Guess which one conquered the world...?-)

Intr syscall: ITS vs Unix

- MIT AI Lab's ITS:
  - every long-running syscall needs to be quasi-atomic AND interruptible...
    - so: every syscall must be able to...:
      - unwind state changes at ANY point
      - resume user-mode for intr. service
      - restart kernel-mode syscall code again
- early Unix:
  - errno←EINTR, return -1 -- that's it!-)

Metaclass vs Decorator

```
class Meta(type):
  def __new__(m, n, b, d):
    cls = type.__new__(m, n, b, d)
    cls.foo = 'bar'
    return cls
class X:
  __metaclass__ = Meta
```

...vs...

```
def Deco(cls):
    cls.foo = 'bar'
    return cls
@Deco
class Y(object): pass
```

Python incrementals

- sorting
  - once: alist.sort(cmp=...)
    - always in-place; slow; a bit cumbersome
  - then: DSU
    - x=[(k(a),a) for a in alist]
    - x.sort(); alist[:] = x
  - now: alist.sort(key=k); sorted(alist, key=k)
- generators: once yield-only, now w/`send`
- finalization: once try-finally, now `with`

WIB vs TRT: other fields

Good enough never is (or is it?)

- Eric Ries, http://www.linkedin.com/today/post/article/20121008194203-2157554-good-enough-never-is-or-is-it
- "Lean Startups" use the "middle way" to...:
- **minimum viable product**: that version of a new product which allows a team to collect the maximum amount of validated learning with the least effort
- 37signals' Hansson disagrees: "just build something awesome and ship it";-)

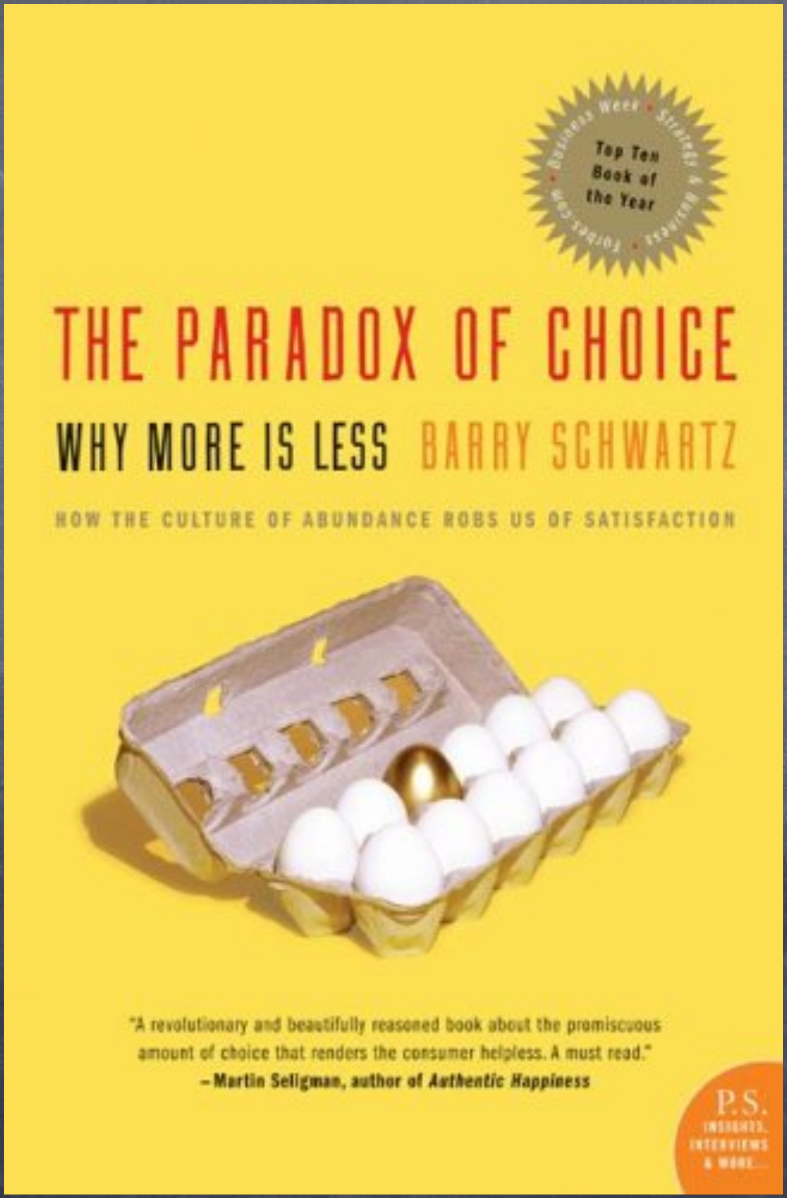
Pick a Perfect Employee...?

- http://theundercoverrecruiter.com/find-perfect-employee/ : DON'T!
  - you'll delay by months, miss opportunities
  - he/she might not be out there looking!
  - you'd likely be over budget
- rather:
  - pick a GOOD (not PERFECT!) fit
  - focus on personality & culture match
  - provide TRAINING on missing skills

Satisficer vs Maximizer

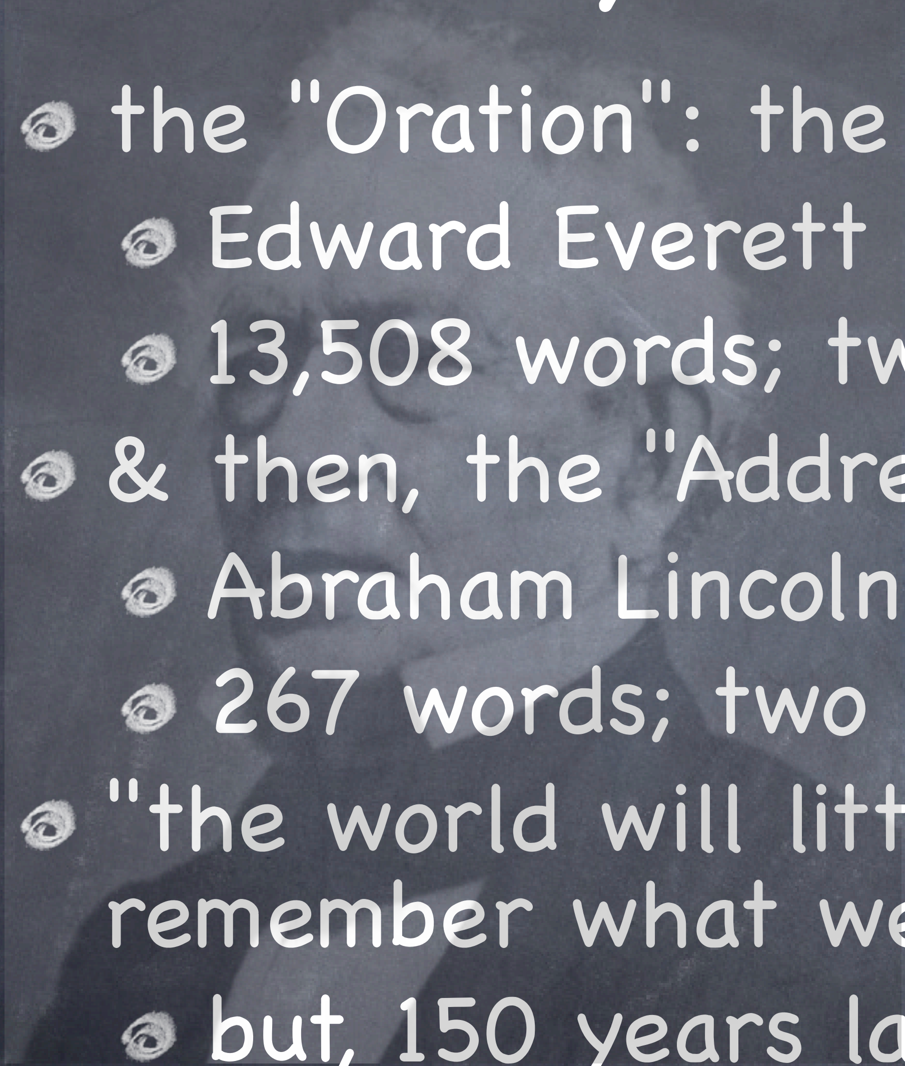
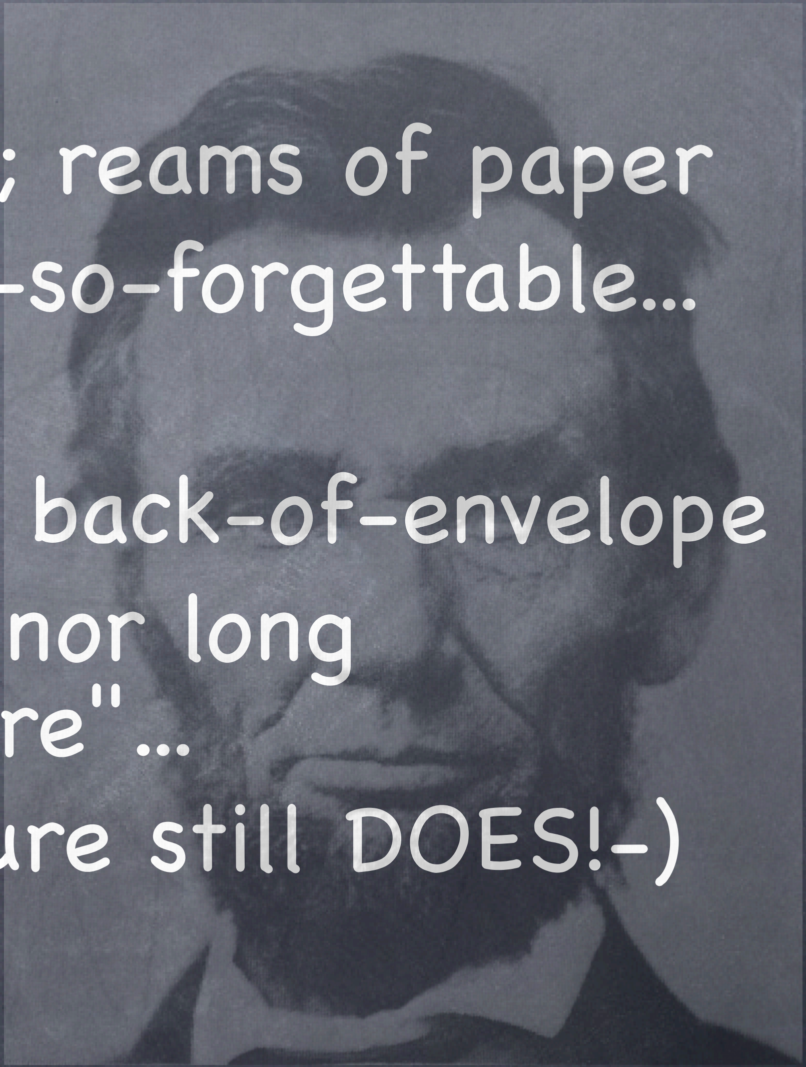
Satisficer:
90% is just fine,
take it, move on!

80% may be OK
(20% of effort:
Pareto's Law)


Business Week • Strategy & Business • Forbes.com
Top Ten Book of the Year
THE PARADOX OF CHOICE
WHY MORE IS LESS
BARRY SCHWARTZ
HOW THE CULTURE OF ABUNDANCE ROBS US OF SATISFACTION
"A revolutionary and beautifully reasoned book about the promiscuous amount of choice that renders the consumer helpless. A must read."
—Martin Seligman, author of Authentic Happiness
P.S. INSIGHTS, INTERVIEWS & MORE...

Maximizer:
99.99% is NOT
100%,
so it's A FAIL!

Gettysburg Dedication

- the "Oration": the soon-forgotten one...
  - Edward Everett
  - 13,508 words; two hours; reams of paper
- & then, the "Address": not-so-forgettable...
  - Abraham Lincoln
  - 267 words; two minutes; back-of-envelope
- "the world will little note, nor long remember what we say here"...
  - but, 150 years later, it sure still DOES!-)

"Lowering expectations"?

- NO! our dreams **must** stay big! **BHAG!**
    - Rightly traced and well ordered: what of that? // Speak as they please, what does the mountain care?
- **however**: the best way **TO** those dreams remains "release early, release often"
    - **learn** from real users' interactions
- Ah, but a man's reach should exceed his grasp // Or what's a heaven for?
- Browning's Andrea del Sarto: **less is more**!

Q & A

www.aleax.it/pybay16_geige.pdf

http://shop.oreilly.com/product/0636920012610.do

ebooks, all formats, DRM-free; at checkout: AUTHD