Simplicity in Code

Christian Neukirchen
http://chneukirchen.org

Curucamp 2009
What is simplicity?

*Technical skill is mastery of complexity, while creativity is mastery of simplicity.*

—Christopher Zeeman
What is simplicity?

Avoiding complexity
e specially *needless* complexity

Finding apt solutions

Elegance

Asking yourself "why didn’t I come up with this?"
Measuring simplicity

Lines of Code (hrm…)

Time it took to develop (hrm… hrm…)

Amount of programmers
Measuring simplicity

Lines of Code (hrm…)
Time it took to develop (hrm… hrm…)
Amount of programmers
WTF/Minute
The only valid measurement of code quality: WTFs/minute

(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics
A diagram

- hacky
- messy
- trivial
- bloaty
The sweet spot

WTF/min

Lines of Code

trivial

hacky

messy

bloaty

simple
Everyone knows that debugging is twice as hard as writing a program in the first place. So if you’re as clever as you can be when you write it, how will you ever debug it?

—Brian Kernighan, The Elements of Programming Style
Achieving simplicity

Avoid magic! Magic is never simple.

Find appropriate abstractions/generalizations

Avoid over-design/over-generalization/do-everythings

Modularize. Reuse. Use simple, flexible interfaces.

Discover. Follow.
Achieving simplicity

KISS. YAGNI. DRY?

```ruby
- %w[before after].each do |block_type|
  - (instance_variable_get("#{@#{block_type}"") || []).each { |b|
  -     context.send(block_type, &b)
  -   }
- end
+ @before.each { |b| context.before(&b) }
+ @after.each { |b| context.after(&b) }
```

def get; req.get?; end
def post; req.post?; end
def put; req.put?; end
def delete; req.delete?; end
PITAs, run if you can

backward compatibility
i18n, L10n
GUIs
selfmade parsers for config files
other users
Wabi-Sabi

Wabi-Sabi is the quintessential Japanese aesthetic. It is a beauty of things imperfect, impermanent, and incomplete. It is a beauty of things modest and humble. It is a beauty of things unconventional…

Applied Wabi-Sabi

Accept edge cases

*Worse is better*

It’s not worth to be correct all the time, esp. if its easy to see *when* not.

Don’t fear rewrites, no code is permanent
Towards mastership

Know thy tools; use thy tools; do not work against the tools

Pet peeves: dæmonization, output coloring (use external tools!)

Keep things hackable; don’t fear radical solutions

Mastering these rules is hard.
Why simple code?

Faster development, faster to the finish
Focus on what matters
More efficient solutions; more security; tools that work better
More elegance, better, easier to extend
Examples

Operating systems: Multics ⇝ Unix ⇝ Plan 9

Marshalling: XML ⇝ YAML ⇝ JSON
My examples

Nukumi1 $\rightarrow$ Nukumi2 $\rightarrow$ Trivium

$\sim1000$ LoC XSLT
Ruby

$\sim1300$ LoC
Ruby

59 LoC
Ruby

a terrible overdesigned mess

a terrible mess

simple & beautiful
My examples

CGI ⇝ WSGI ⇝ Rack

RSpec ⇝ test/spec ⇝ bacon

~650 LoC  ~300 LoC
more general than test/spec
My examples

CGI ⇔ WSGI ⇔ Rack

RSpec ⇔ test/spec ⇔ bacon ⇔ Knock?

~650 LoC  ~300 LoC  ~80 LoC

more general than test/spec
Resources

Bell Labs (Research Unix, Plan 9)
D. J. Bernstein (http://cr.yp.to/)
Chuck Moore (ColorForth etc.)
http://suckless.org/
John Maeda (http://lawsofsimplicity.com/)
Math helps
The Master doesn’t try to be powerful; thus he is truly powerful. The ordinary man keeps reaching for power; thus he never has enough.

The Master does nothing, yet he leaves nothing undone. The ordinary man is always doing things, yet many more are left to be done.

—Lao Tzu, Tao Te Ching, Chapter 38
Questions?
Thank you.

Thanks to: Alexander Kellett, Linus Sellberg, Dominik Honnef, Pistos Christou, and Aria Stewart for reviewing the slides. Verbatim copying is allowed as long as this message is preserved. Duplication is encouraged.