Everything I know about software, in 90 minutes

Jon Skeet

40% discount:
ctwsdd18
at manning.com

What this talk is not

- A C# lesson
- A set of rules to follow without thinking
- An attempt to belittle those who disagree with approaches to testing etc
- Definitive

What this talk is

- Reflections on my personal experience
- More about people than technology
- The start of a conversation

Look how far we've come!

Time: 4

Getting started

- Learn almost any language in a browser
- For free
- With lots of people willing to help you, also for free

Open *

- Open source, open APIs, even open hardware
- Big-player acceptance of benefits
- Broad theme of collaboration and sharing
- Not problem-free, of course

Hard-to-believe power

- Cloud
- ML
- Phones!
- PCs

Look what a mess we've made!

Time: 10

Old protocols

- SMTP is here forever. Oh.
- HTTP2 is gradually happening.
- IPv6 is very gradually happening

Insecure by design

- We’re coming to security too late, along with a bunch of other things
- Privacy

I know, I'll build a DI framework…

- Or a new Javascript framework
- Or a new web paradigm
- Or a new methodology
- Or a new logging framework
- Sometimes, more choice is less

CSS

- Maybe it's just me
- Web in general – although I note Javascript is improving, and I know I'm not a web developer.
- But if we started from scratch, knowing what we know, how different would it be?
- And how much of a mess would it be in 10 or 20 years time when requirements change again?
- Hindsight is wonderful.

Demographics

- This won't be the last time this comes up.
- But yes, we've made a mess of the industry itself.
- Hat tip to Emily Chang for Brotopia

Traits of a good developer

Time: 20

Confidence

- Every day, software changes the world.
- Confidence can be ludicrous: anything amazing was probably deemed impossible by someone once
- Confidence let's you start, even for simple things
- Confidence that it's okay to try
- Confidence that it's okay to fail
- Confidence correlates with privilege (assertion)

Humility

- When your code fails, it's probably your fault.
- Compilers, runtimes and libraries have bugs – but your code probably has more
- Having a bug in your code doesn't make you a bad person
- Having a bug in your code and assuming the problem is somewhere else makes you an unproductive developer

Curiosity

- This is more debatable
- Some developers can keep working in the same tech stack forever without learning anything new, and be productive
- Most developers will need to be adaptable
- Better to explore something new and interesting, then make that your job, than be told what to do next

Nevertheless, she persisted

- There are plenty of times when software is hard in a drudgery way.
- Persisting when nothing makes sense is difficult
- Persisting can give you one tiny victory (against a bug) at a time
- Two-edged sword: persistence can be a problem
- Persisting in upholding your beliefs can win out in the long term – or just make you difficult to work with
- Persistence with flexibility can be powerful: “I will do my best to meet your needs without compromising my long-term goal”

Laziness (of a kind)

- The right kind of lazy is hard to define
- Automation and timing
- Knowing when to make a decision, when to invest
- Automation has a high up-front cost, but a great payoff.
    - Do it when you know the cost won't be repeated!

Don’t be a jerk

- This will be a recurring theme.
- I would like to be remembered as a good person who could code reasonably than a “rock star” programmer who was a jerk
- There are examples of people succeeding in both software and business despite being jerks. Don’t emulate that.

“Soft” skills are hard

Time: 35

People matter

- Team mates
- Other colleagues
- Potential colleagues
- Open source collaborators
- Developers who work on open source projects you use
- Users
- The people whose questions you answer
- The people who answer your questions
- You

Communication matters

- Everything we do in software is communication
- Writing code is only part of it, but that's still comms-focused
- Think about how much of the day is about communicating ideas and responses
- Clarity and empathy are not the same thing

DI matters

- Diversity and inclusion. Not dependency injection. That matters too.
- Software is changing the world. Whose problems do we want to solve?
- Good for business too.

Be a non-jerk

- Go out of your way to avoid being a jerk
- That involves learning what a jerk looks like
- Can be challenging (check your privilege)
- Reflective nature mirrors reflection on coding success / failure / new perspectives
- Allies, not knights

What doesn't need to be hard

Time: 50

Diagnostics

- Still haven't worked out why this isn't as common as it should be

Learning

- Software community can be awful – or incredible
- We have people skills to learn, for empathizing with those who don't have our knowledge
- We have people skills to learn in empathizing with those who have more knowledge
- We have tech skills to learn to make best-in-field ideas for discovery more common
    - Jupyter notebooks
    - Interactive documentation
    - API docs in familiar formats

Date/time stuff

- For the most part, anyway
- Don’t be scared
- Use the right tools

What's still techy-hard?

Time: 65

Versioning

- Predicting the future is hard
- Thinking in time is hard

Inclusive UIs

Internationalization
Accessibility
Inclusive language and documentation

Privacy and security

- Security in the face of CPU bugs and design flaws
- Privacy in many axes
- Verifiability

Integration

- Increasingly large proportion of software effort, it feels

Testing

- Time-consuming to write
- Time-consuming to run
- Error-prone
- Brittle
- Incomplete

Don’t be a jerk

- Final reminder

Coding tips

Know what your data means

Use the right representation

- Parse strings early
- Format late

Know what you know

- If you're using a language/platform all the time, it's worth learning it thoroughly
- Otherwise, keep it simple; don't be too *ambitious*

Learn systematically

Performance *may* matter

- Be deliberate about trade-offs between performance and simplicity
- Have concrete goals where possible
- Test!

Testing

- Mocks, fakes etc
- Pragmatism over dogmatism

Questions

Time: 85