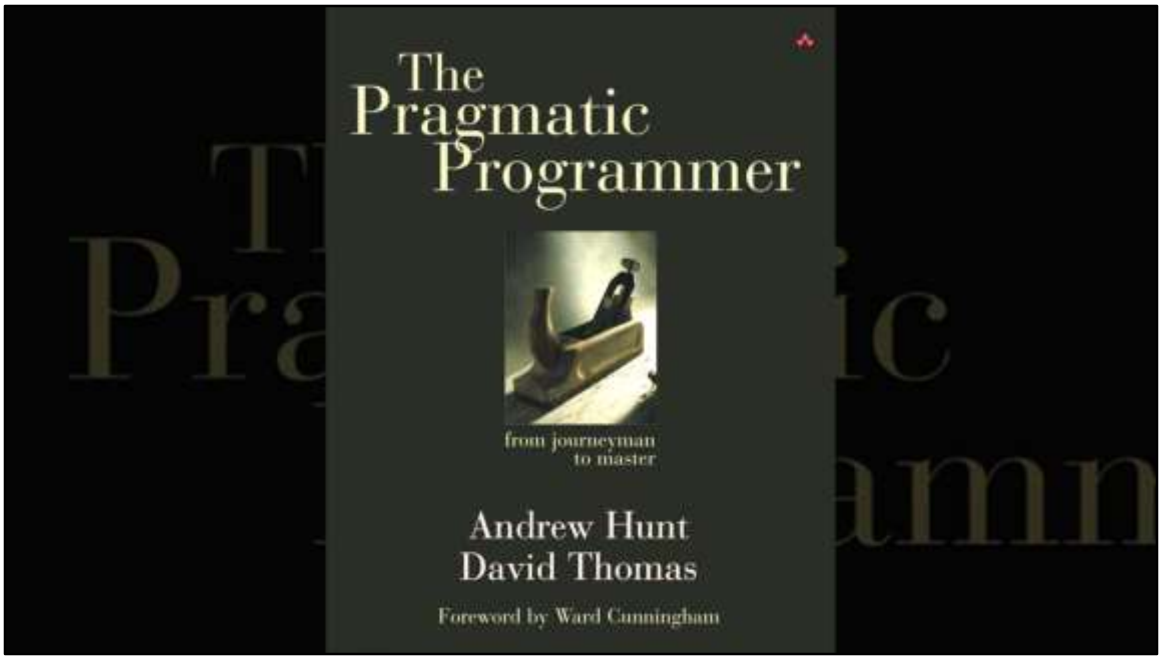
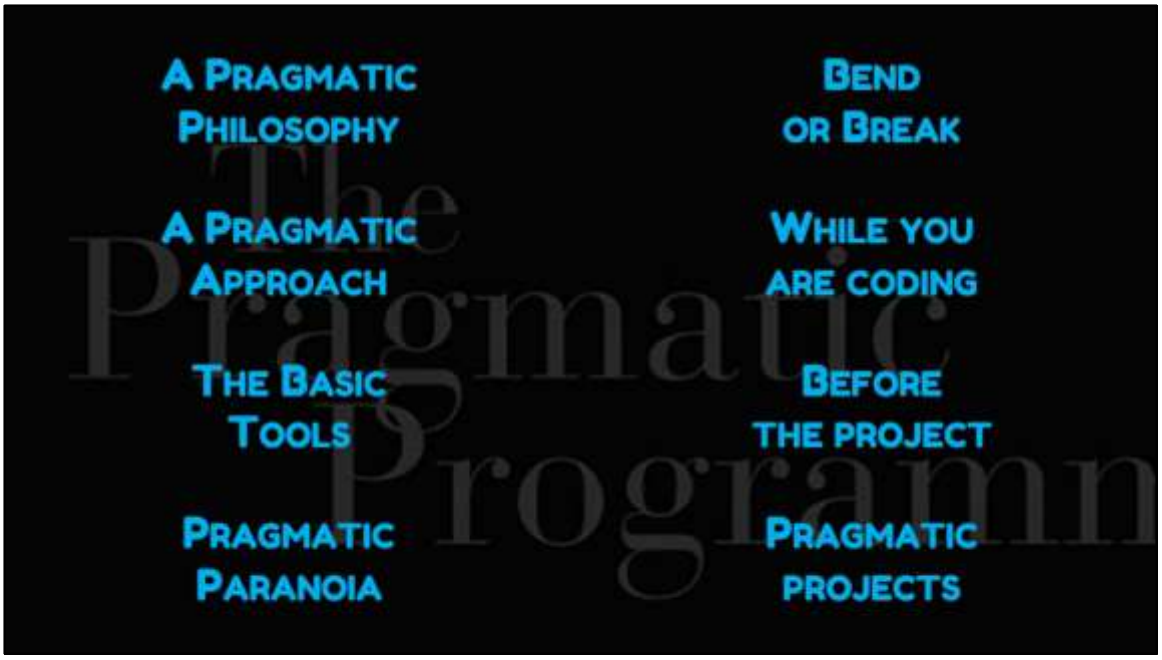
THE PRAGMATIC PROGRAMMER

FROM APPRENTICE TO JOURNEYMAN

Image credit: dumbyellowdog @ flickr


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham



A Pragmatic Philosophy

A Pragmatic Approach

The Basic Tools

Pragmatic Paranoia

Bend or Break

While you are coding

Before the project

Pragmatic projects

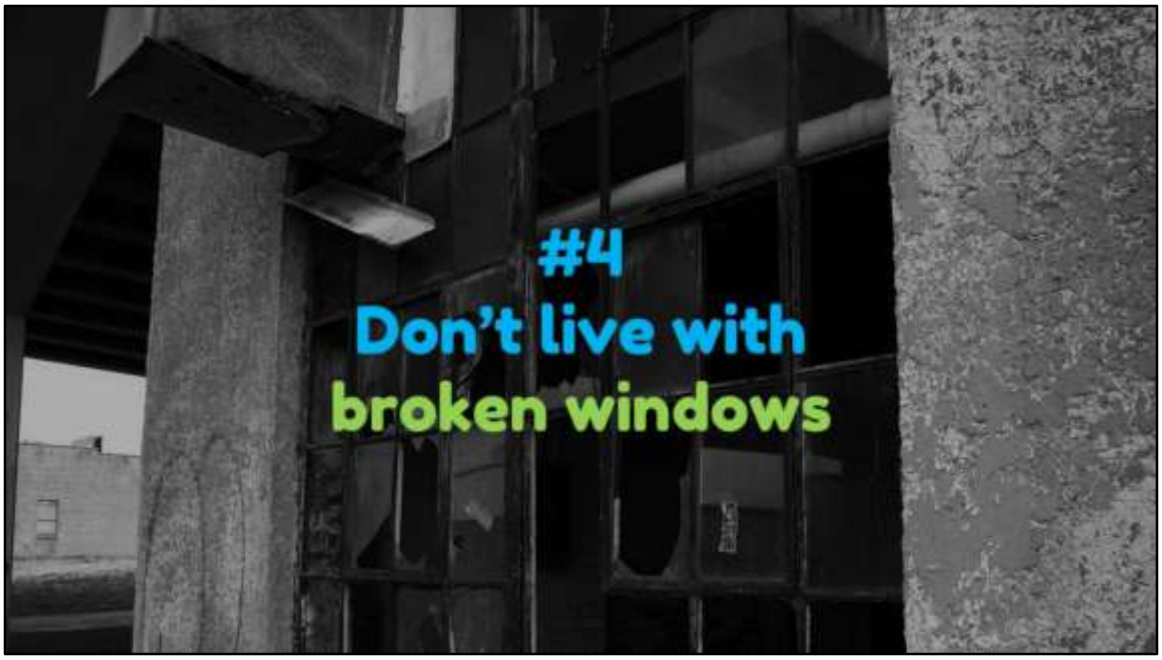
The book is divided into 8 chapters

And has it's own summary in the form of 72 tips
But don't panic … I'm not covering all of them!
(You'll have to buy the book for that.)

When an apprentice has learned all he can from his master – typically after a 7 year apprenticeship - he would undertake a journey – for 3 or so years – to find other masters and learn from them.

A PRAGMATIC APPROACH

Image Credit: mrbenn03 @ Flickr


#4
Don’t live with
broken windows

Researchers (crime and urban decay): a broken window, left unrepaired, quickly escalates into large scale damage.

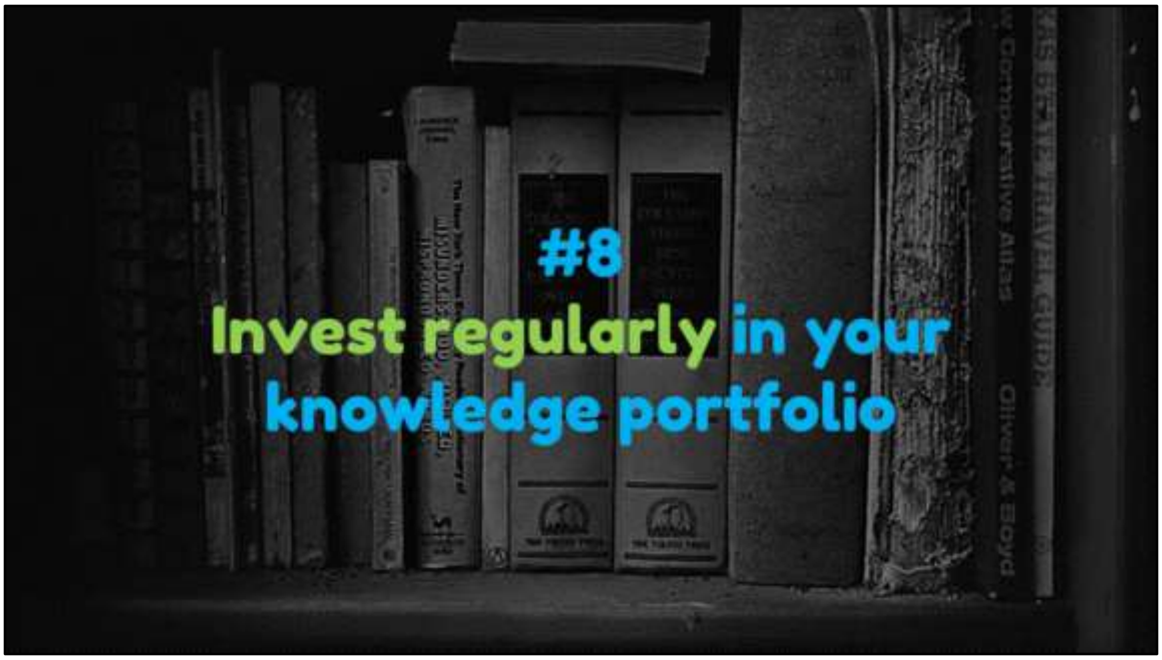
Image credit: leeadlaf @ Flickr

New York police established a special unit: encourage landlords => minor repairs quickly, => measurable drop in crime rates.

Apply the Boy Scout rule – leave the campsite better than when you arrived.

Same with our code – fix the small things and keep things tidy, so that the next developer does the same.
Keep fixes for the big things in your backlog.

Image credit: leeadlaf @ Flickr



Change is about the one true constant in our industry.

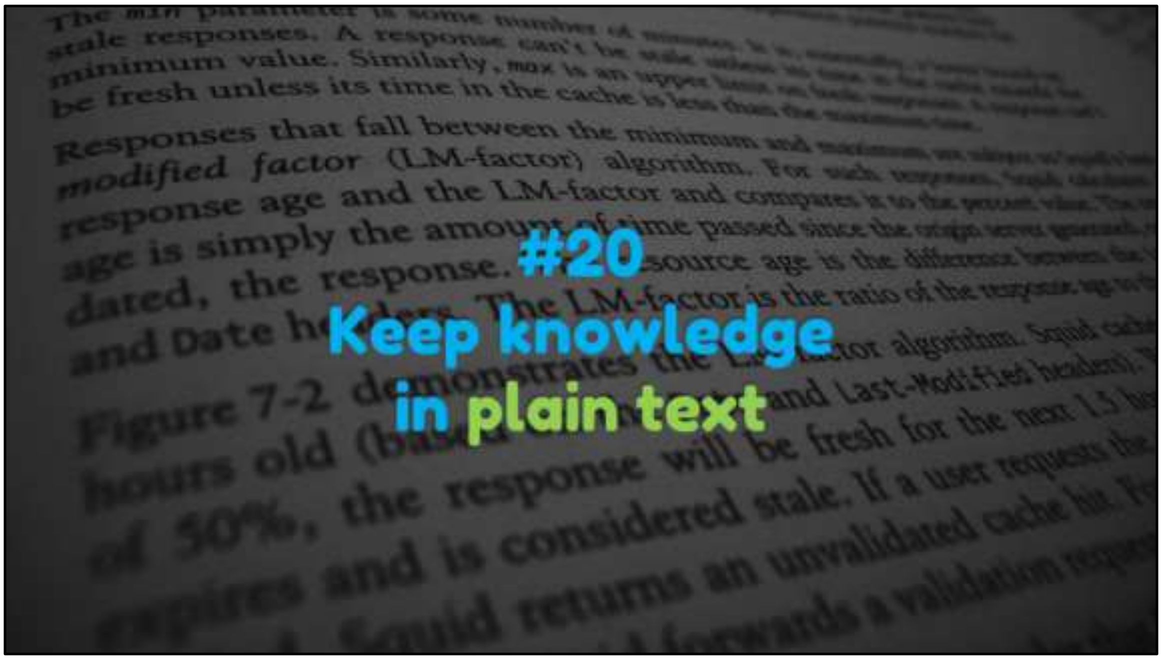
If you don't invest in yourself, who will?

Read a technical book every quarter. Start with the Pragmatic Programmer. Venture outside your comfort zone. Get your work to buy books – cheap training.

Go to your local user group. Oh, yeah, Hi.

~~Try out~~ Play with a new technology in your own time.

Image credit: quinnanya @ Flickr

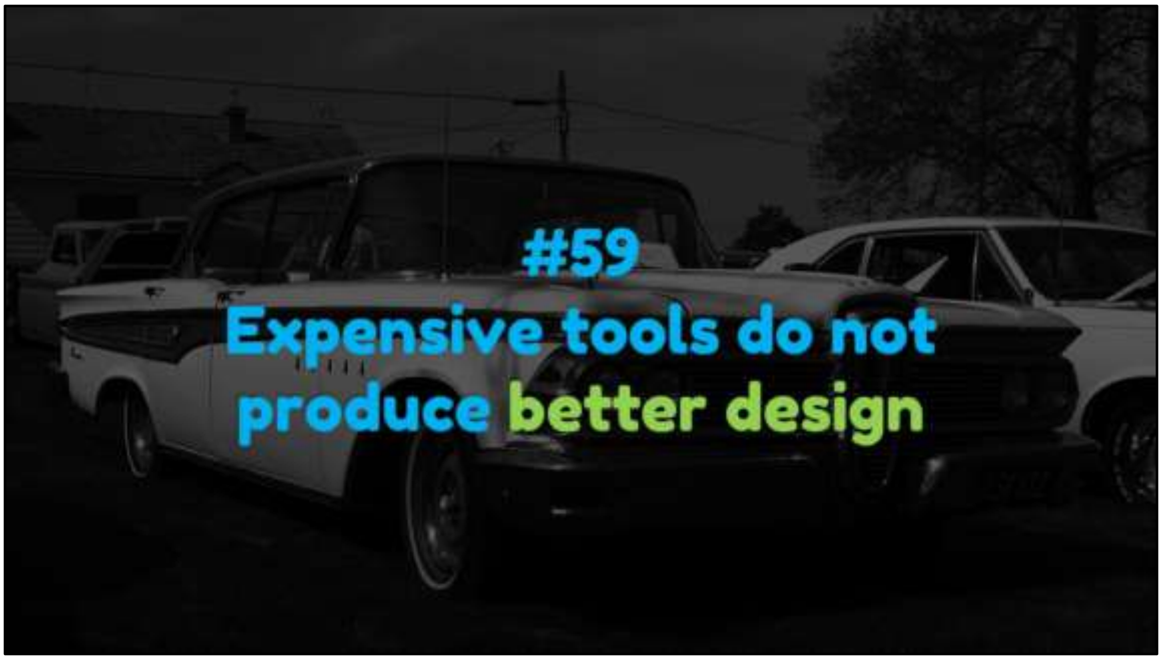


There’s a reason that most programming stacks use text files for source code.
Easily manipulated, compared, versioned and processed.

Binary data isn’t secure, just obscure.
Text data can outlive the original system.
Text includes text based formats like xml, json and yaml.

Human understandable, not just human readable. Self documenting.

Image credit: allspaw @ Flickr



#59 Expensive tools do not produce better design

Having an expensive tool doesn’t make you a better designer – at best it gives you a shinier result.

Collaborative design around a whiteboard can – and usually will – outperform a solo architect with an expensive tool.

Image credit: roadsidepictures @ Flickr

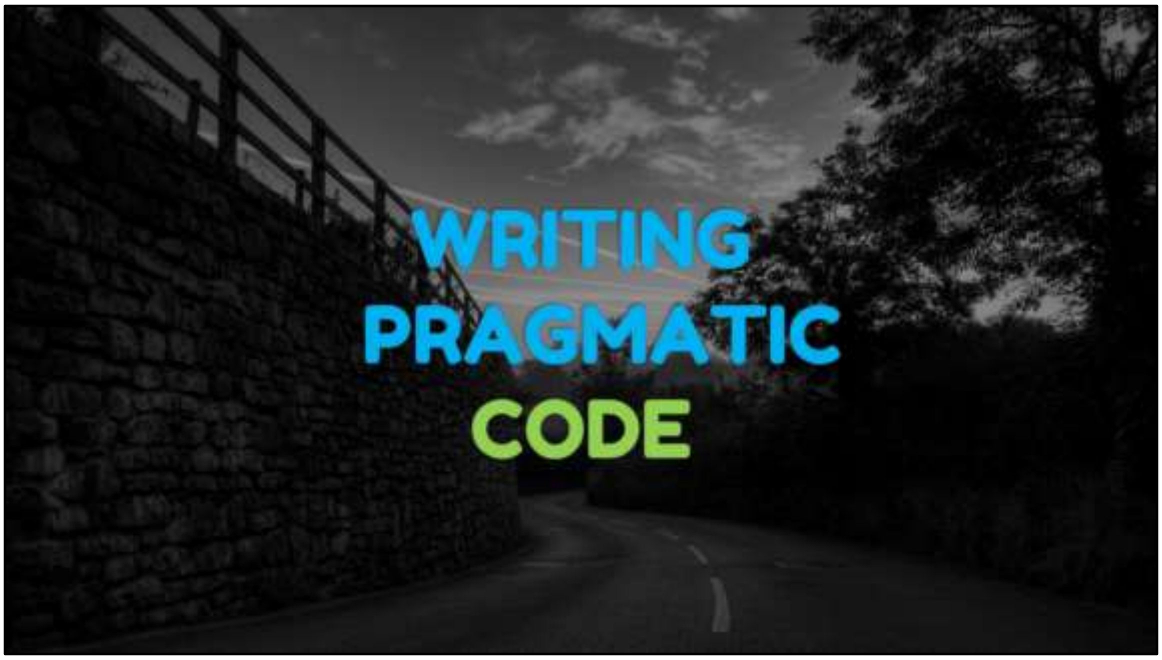
Computers will follow instructions and do the same thing every time.
People won't – even if they try.

If it can be automated, it should be automated.
Why waste valuable human time on something the computer can do?

Automate the builds.
Automate the testing of those builds.
Automate deployment of those tested builds.
Automate the creation of release notes.

It's a step by step process.
Don't try to automate everything in one go.
Make one improvement every time.

Image credit: vintage19_something @ Flickr



WRITING PRAGMATIC CODE

The DRY principle is key – in many ways, it underlies all of the SOLID principles.

It's not that a piece of knowledge can only be written down once.
But that one place should be the "source of truth"

Generate other representations where you can
Test for consistency where you can't

Image credit: inggmartinez @ Flickr

Instead of writing the same code over and over, write one piece of code that can be reused.

Passive code generation – fire once, manually updated afterwards. Convenient, limited benefits. Needn't be complex.

Active code generation – used repeatedly, though the life of the project. More work to build, vastly increased benefits.
Leverage technology stack – eg. **partial** keyword.

Image Credit : _flood_ @ Flickr

#30 You can't write perfect software

It's sad, but true: every piece of software has bugs.

Write defensively. Check assumptions. Validate inputs.
Fail fast.

Image credit: julishannon @ Flickr

Log normal and abnormal activity.
You might not get a chance to log an error on the way out – logs of normal stuff may be the only clue.
Leave clues behind when your process dies.

Image credit: julishannon @ Flickr

We’re moving into an era of massively parallel computing – even on the client.

The C library routine strtok uses background storage, so it can’t be used twice at the same time, even from different threads.
You can do the same in C# with static members.

Don’t do it.

Make everything thread aware.

Validation Framework: static storage, but thread aware to keep uses separate.

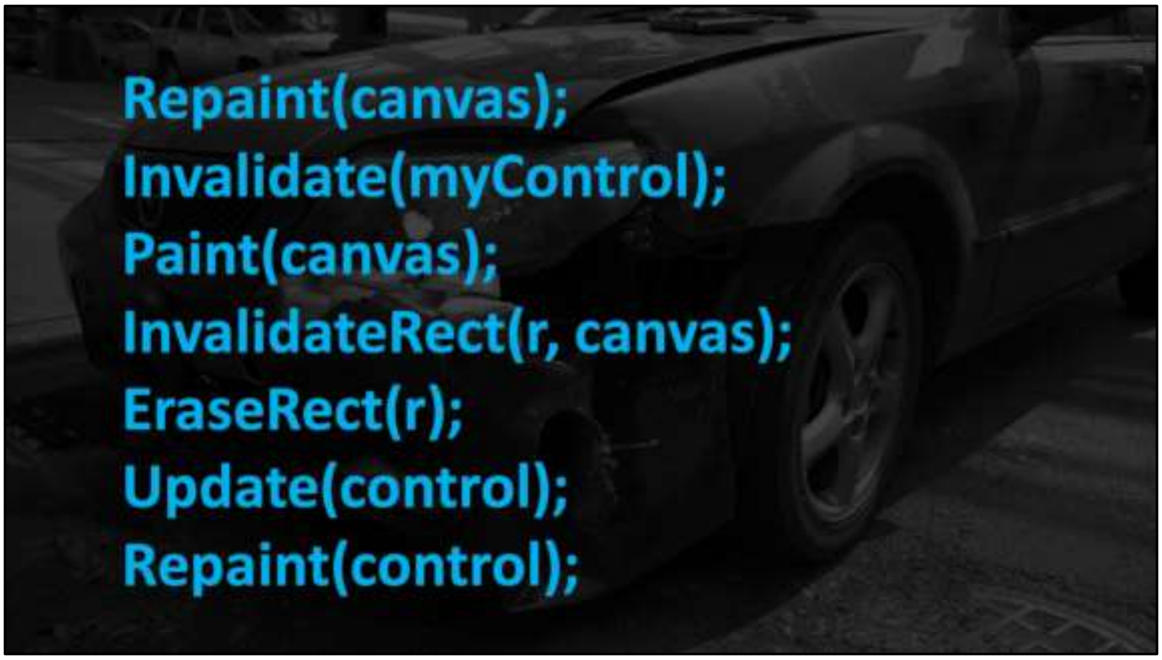
Image credit: (Mick Baker)rooster @ Flickr

#44
Don't program by coincidence

Make sure you know why code works.

Rely on what's documented, not the way it seems to work.
Truck taillight design based on undocumented characteristics of the chip. Chip revision broke the design.

Implementations change.



```
Repaint(canvas);
Invalidate(myControl);
Paint(canvas);
InvalidateRect(r, canvas);
EraseRect(r);
Update(control);
Repaint(control);
```

Imagine you came across code like that shown.

Clearly the developer was trying to achive something – and either got there and gave up, or failed.

It’s easy to write code in a way that can’t be easily tested.

E.gs.
Code that consumes data from an external feed – that can only be tested by hooking it up to that feed.
Hard coded singletons – C# static classes – make it difficult to test.

Automated testing is your friend.
Good architectures include consideration for testing – e.g. Ports and Adapters

PRAGMATIC COMMUNICATION

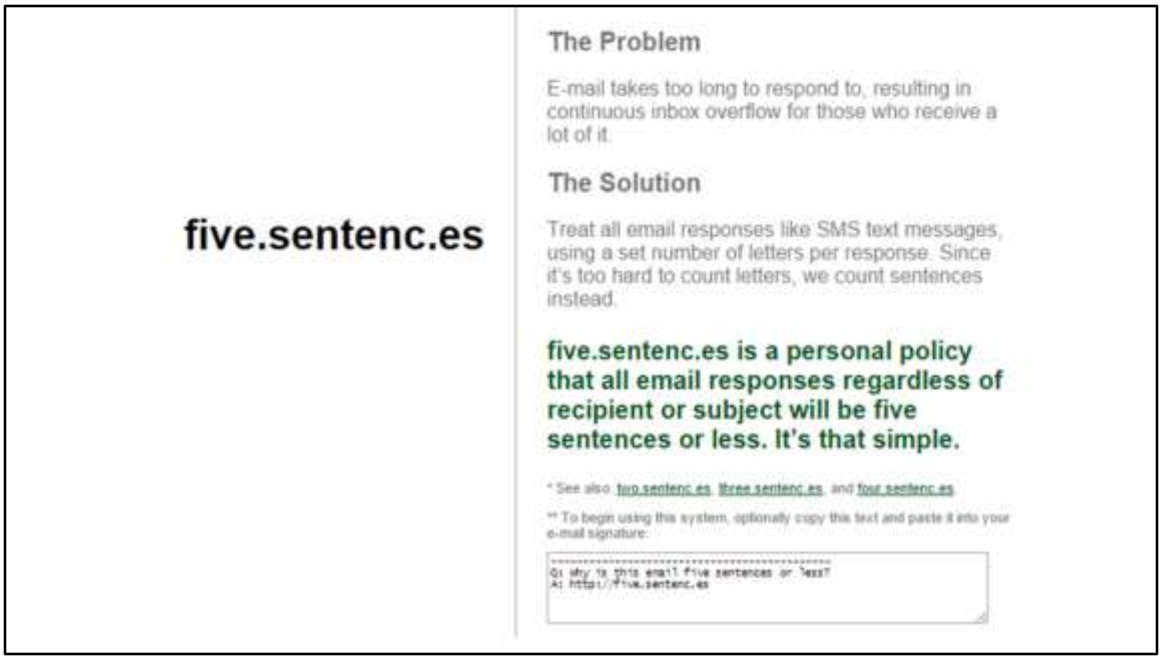
#10
It's both what you say and how you say it

Your boss comes out of a meeting with a frazzled look on her face.
Not a good time to ask for day off.

You've just made a mistake that cost the company a million dollars.
Not a good time to ask for a raise.

Pick your time and place.
Pick your medium.
Be clear with your message.

Image credit: sharman @ Flickr


five.sentenc.es

The Problem

E-mail takes too long to respond to, resulting in continuous inbox overflow for those who receive a lot of it.

The Solution

Treat all email responses like SMS text messages, using a set number of letters per response. Since it's too hard to count letters, we count sentences instead.

five.sentenc.es is a personal policy that all email responses regardless of recipient or subject will be five sentences or less. It's that simple.

* See also: two.sentenc.es, three.sentenc.es, and four.sentenc.es

** To begin using this system, optionally copy this text and paste it into your e-mail signature:

```
----------------------------------------------
Q: Why is this email five sentences or less?
A: http://five.sentenc.es
```

five.sentenc.es

Image credit: sharman @ Flickr

#67 Treat English as just another programming language

Being precise and clear is a requirement for communicating with the computer.
It’s also a great idea for communicating with people.

- Poor spelling
- Bad grammar
- Lack of clarity

If you don’t place enough value on your message to present it well
Why should the reader place enough value to read it?

#68 Build Documentation in, don’t bolt it on

Documentation is important – but keeping it up to date is hard.

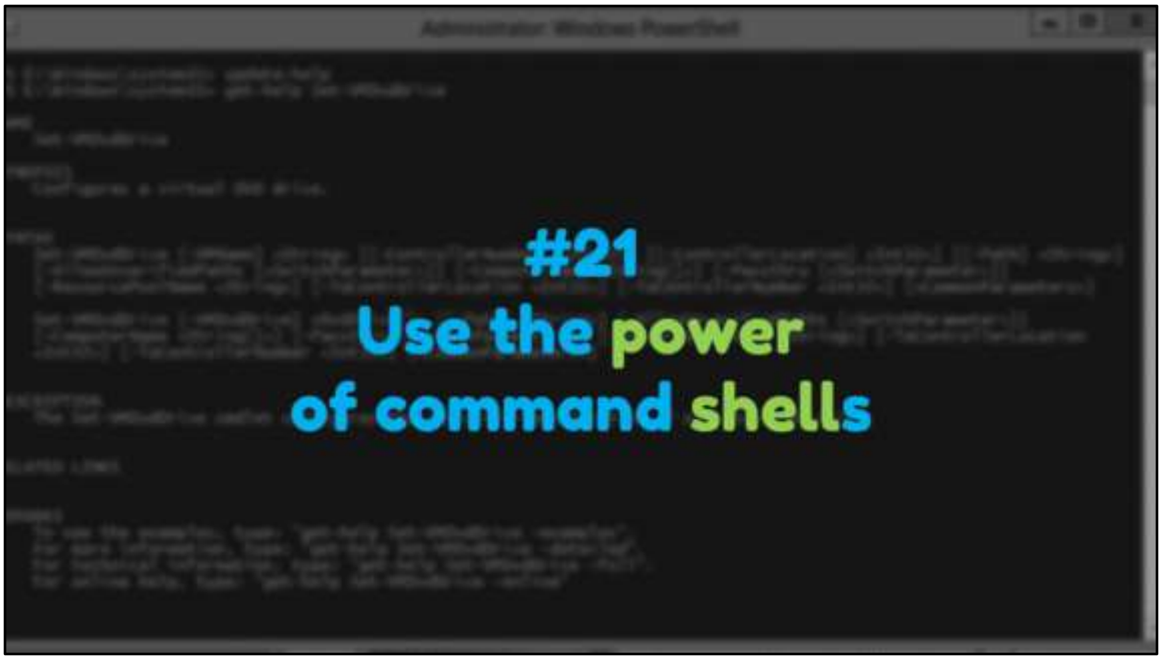
Keep the documentation as close to the code as possible
to make it as easy as possible to update.

Sparse documentation that is up to date
beats complex documentation that’s stale

Where you can, generate docs from artifacts you already have.
e.g. C# xml comments => API documentation Sandcastle, SharpDox
e.g. Rule definitions => code generation + documentation

Image credit: featheredtar @ Flickr

PRAGMATIC TOOLING



If you’re not using PowerShell, why not?

Fundamental tool to server administration – all of Microsoft’s projects are based on PowerShell for admin, with GUI tools over the top.

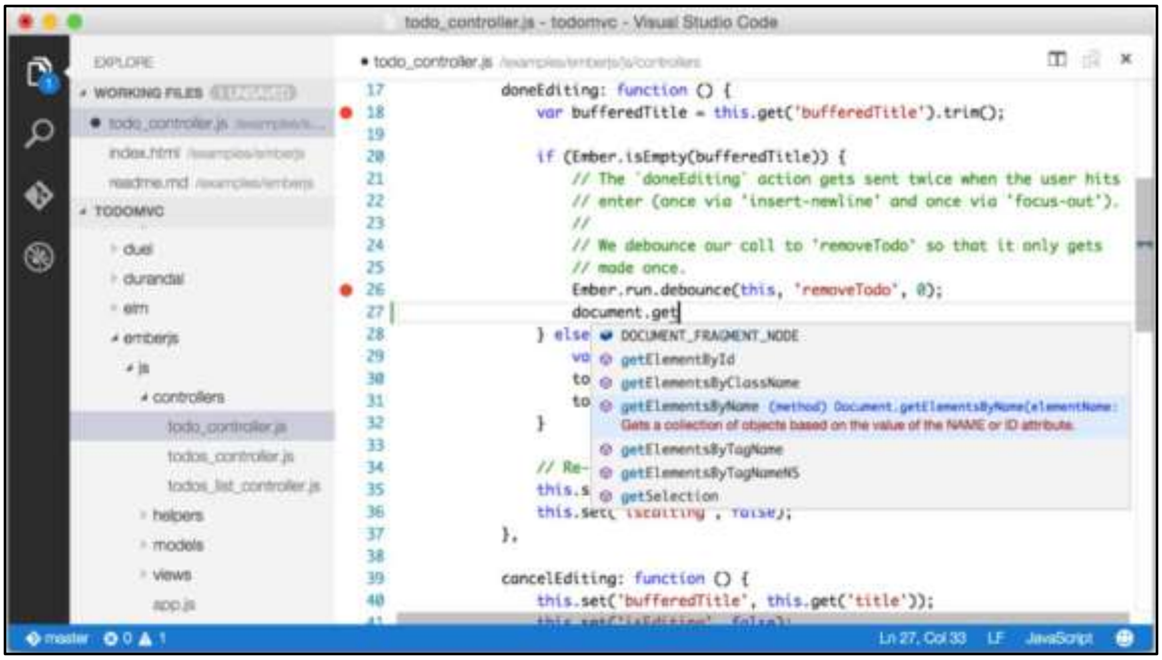
Combine PowerShell with PSake for task automation.

Don’t limit yourself to using just Notepad.

Pick a competent tool that does a lot – Visual Studio, Notepad++, UltraText, Sublime – and learn to make it sing.

Learn all the hot keys, how to use column mode, macro modes.

You have a finite number of keystrokes left – make them count.



```
doneEditing: function () {
    var bufferedTitle = this.get('bufferedTitle').trim();

    if (Ember.isEmpty(bufferedTitle)) {
        // The `doneEditing` action gets sent twice when the user hits
        // enter (once via 'insert-newline' and once via 'focus-out').
        //
        // We debounce our call to 'removeTodo' so that it only gets
        // made once.
        Ember.run.debounce(this, 'removeTodo', 0);
        document.get
    } else
        va
        to
        to
    }

    // Re-
    this.s
    this.set('isEditing', false);
},

cancelEditing: function () {
    this.set('bufferedTitle', this.get('title'));
```

There’s a new kid on the block – **Visual Studio Code**.
Not an ide, but a code editor that includes intellisense and debugging.
Cross Platform – Windows, Macintosh, Linux

This is kinda obvious.

The dog ate my homework wasn’t really acceptable at age 10, certainly not now.

No longer need to set up a server, use a **git** and a **local** repository.
Even for a solo effort. Works better with text content.

CRASHPLAN™

Automatic Online Backup

Make sure your machine is backed up – I use **CrashPlan**

THANKS

@UNREPENTANTGEEK
WWW.NICHESOFTWARE.CO.NZ
BEVAN@NICHESOFTWARE.CO.NZ