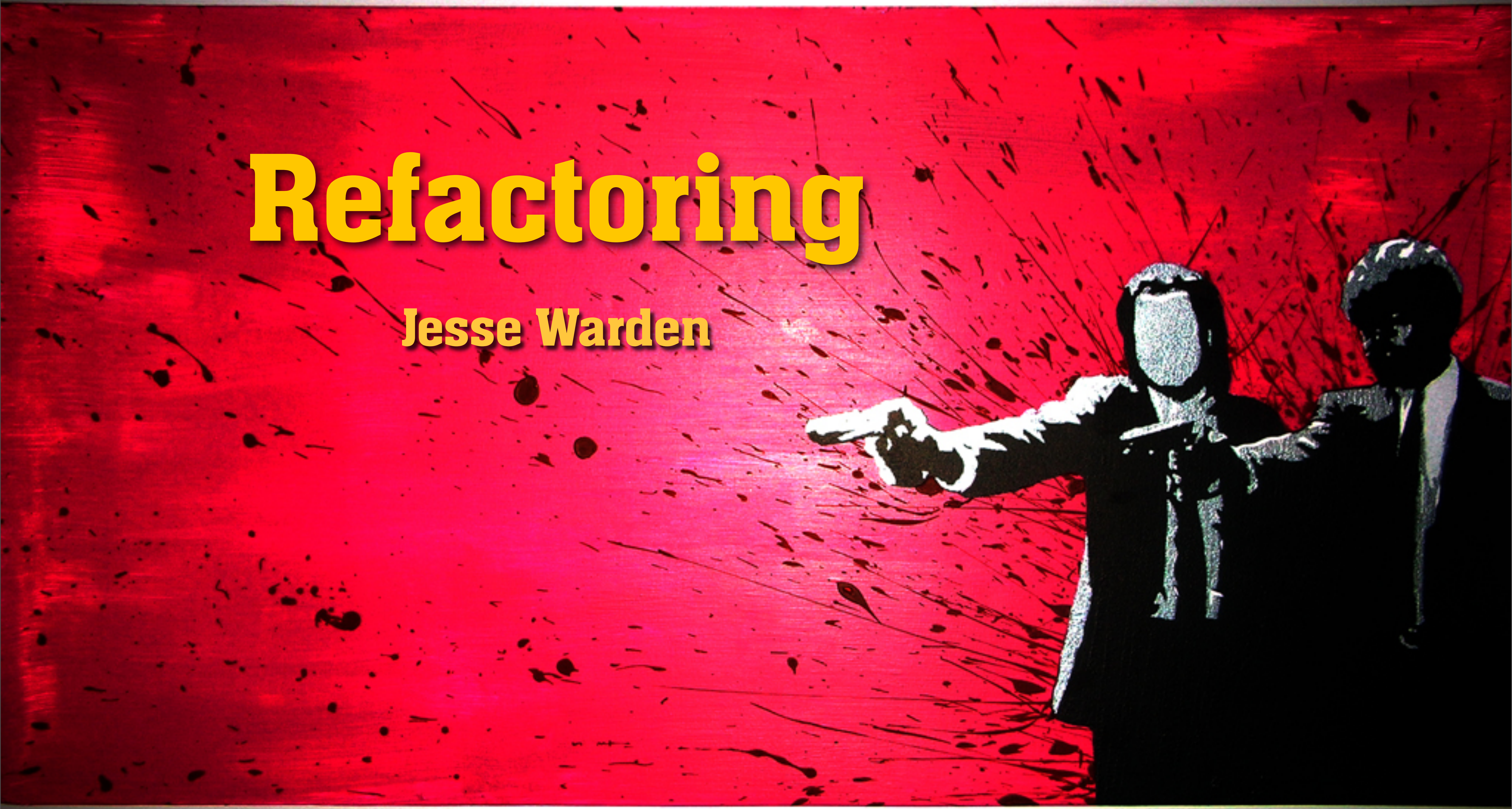
Refactoring

Jesse Warden



Who?

Who?

- Jesse Warden

Who?

- Jesse Warden
- Web App Solution

Who?

- Jesse Warden
- Web App Solution
- Flex, Java, and BlazeDS Consulting

What?

What?

- Basic Refactoring

What?

- Basic Refactoring
- Priority Pyramid

What?

- Basic Refactoring
- Priority Pyramid
- Stories

What?

- Basic Refactoring
- Priority Pyramid
- Stories
- Leadership

Why?

Why?

- Coding is more fun

Why?

- Coding is more fun
- People like working with you

Why?

- Coding is more fun
- People like working with you
- Grow in your career

Why?

- Coding is more fun
- People like working with you
- Grow in your career
- Better Software

No seriously...

No seriously...

- I loathe it

No seriously...

- I loathe it
- People have said articles helped them

No seriously...

- I loathe it
- People have said articles helped them
- Mentor Leadership / Stewardship

Basics

Basics

- Exceptions

Basics

- Exceptions
- Architecture

Basics

- Exceptions
- Architecture
- Refactoring Paths

Exceptions

Exceptions

- Synchronous: 100%

Exceptions

- Synchronous: 100%
- Asynchronous: 99%

Exceptions

- Synchronous: 100%
- Asynchronous: 99%
- Null Pointers

Synchronous

Synchronous

- throw new Error(“b00m”);

Synchronous

- throw new Error(“b00m”);
- try/catch

Synchronous

- throw new Error(“b00m”);
- try/catch
- “Road to hell is paved with good intentions.”

Asynchronous

Asynchronous

- dispatchEvent(new ErrorEvent(“b00m”));

Asynchronous

- dispatchEvent(new ErrorEvent(“b00m”));
- addEventListener(“b00m”, onHandle);

Asynchronous

- dispatchEvent(new ErrorEvent(“b00m”));
- addEventListener(“b00m”, onHandle);
- Same as before, don’t swallow.

Uncaught

Uncaught

- loaderInfo.uncaughtErrorEvents

Uncaught

- loaderInfo.uncaughtErrorEvents
- won't always get a stack trace

Uncatchable

Uncatchable

- Error #2044: Unhandled IOErrorEvent:. text=Error #2036: Load Never Completed.

Null Pointers

Null Pointers

- Sir Charles Antony Richard Hoare

Null Pointers

- Sir Charles Antony Richard Hoare
- British Computer Scientist

Null Pointers

- Sir Charles Antony Richard Hoare
- British Computer Scientist
- "I call it my billion-dollar mistake"

Null Pointers

- Sir Charles Antony Richard Hoare
- British Computer Scientist
- "I call it my billion-dollar mistake"
- Programmer Enemy #1

Null Pointers

Null Pointers

- Loose Typing

Null Pointers

- Loose Typing
- External Data

ExternalInterface

ExternalInterface

- Marshall Exceptions

ExternalInterface

- Marshall Exceptions
- set to true

Exceptions

Exceptions

- Crashes

Exceptions

- Crashes
- Unexpected/Unrelated Errors

Exceptions

- Crashes
- Unexpected/Unrelated Errors
- Breed Insecurity

Exceptions

- Crashes
- Unexpected/Unrelated Errors
- Breed Insecurity
- Cause Fire Drills

Exceptions

Exceptions

- Best Defense is a Good Offense

Exceptions

- Best Defense is a Good Offense
- Broken Windows

Fighting Back

Fighting Back

- try/catch

Fighting Back

- try/catch
- log the catch

Fighting Back

- try/catch
- log the catch
- log uncaughtErrors

Fighting Back

- try/catch
- log the catch
- log uncaughtErrors
- Code Review

Loose Typing

Loose Typing

- Object

Loose Typing

- Object
- Dictionary

Loose Typing

- Object
- Dictionary
- *

Loose Typing

- Object
- Dictionary
- *
- Dynamic

Loose Typing

- Object
- Dictionary
- *
- Dynamic
- Interfaces

Object

Object

- Protoyping

Object

- Protoyping
- JSON

Object

- Protoyping
- JSON
- Teaching

Object

- Protoyping
- JSON
- Teaching
- delete slots

Object Solution

Object Solution

- Don’t Use Object

Object Solution

- Don't Use Object
- Ensure casts aren't null

Object Solution

- Don’t Use Object
- Ensure casts aren’t null
- Same for Dynamic & Interface

Dictionary

Dictionary

- Easier lookups vs. ByteArray

Dictionary

- Easier lookups vs. ByteArray
- Strict Equality on Keys

Dictionary Solution

Dictionary Solution

- Wrap

Star

Star

- *

Star Solution

Star Solution

- Don't Use It

Star Solution

- Don’t Use It
- Log It

Star Solution

- Don’t Use It
- Log It
- Ensure Consumers Guard

External Data

External Data

- Factories

External Data

- Factories
- Value Object / Data Transfer Object

External Data

- Factories
- Value Object / Data Transfer Object
- Parsing

Parsing

Parsing

- Poison Entire System

Parsing

- Poison Entire System
- When in doubt, return null

Parsing

- Poison Entire System
- When in doubt, return null
- Proactive: throw meaningful error

Parsing

- Poison Entire System
- When in doubt, return null
- Proactive: throw meaningful error
- Uber Proactive: Unit Test

Parsing

Parsing

- **Holy S$*(%&: Null Pattern**

Architecture

Architecture

- Encapsulation

Architecture

- Encapsulation
- Testability

Encapsulation

Encapsulation

- Black Box

Encapsulation

- Black Box
- Manages its own internal state

Bad Encapsulation

Bad Encapsulation

- Global Variables / Singletons

Bad Encapsulation

- Global Variables / Singletons
- No Accessors/API for Data Models

Bad Encapsulation

- Global Variables / Singletons
- No Accessors/API for Data Models
- No API so you dig

Testability

Testability

- Easy to test

Testability

- Easy to test
- not welded together

Testability

- Easy to test
- not welded together
- not spaghetti code

Services

Services

- Service Inside the Proxy

Services

- Service Inside the Proxy
- Not a service, just a loader

Services

- Service Inside the Proxy
- Not a service, just a loader
- No centralized error reporting

Service Solutions

Service Solutions

- Create a base class

Service Solutions

- Create a base class
- log all errors

Service Solutions

- Create a base class
- log all errors
- create an integration suite

Big Ole' Commands

Big Ole' Commands

- switch

Refactoring Paths

Refactoring Paths

- What do you build now?

Refactoring Paths

- What do you build now?
- What do you change to later?

Refactoring Paths

- What do you build now?
- What do you change to later?
- When?

Refactoring Paths

- What do you build now?
- What do you change to later?
- When?
- Not Architecting

Refactoring Path

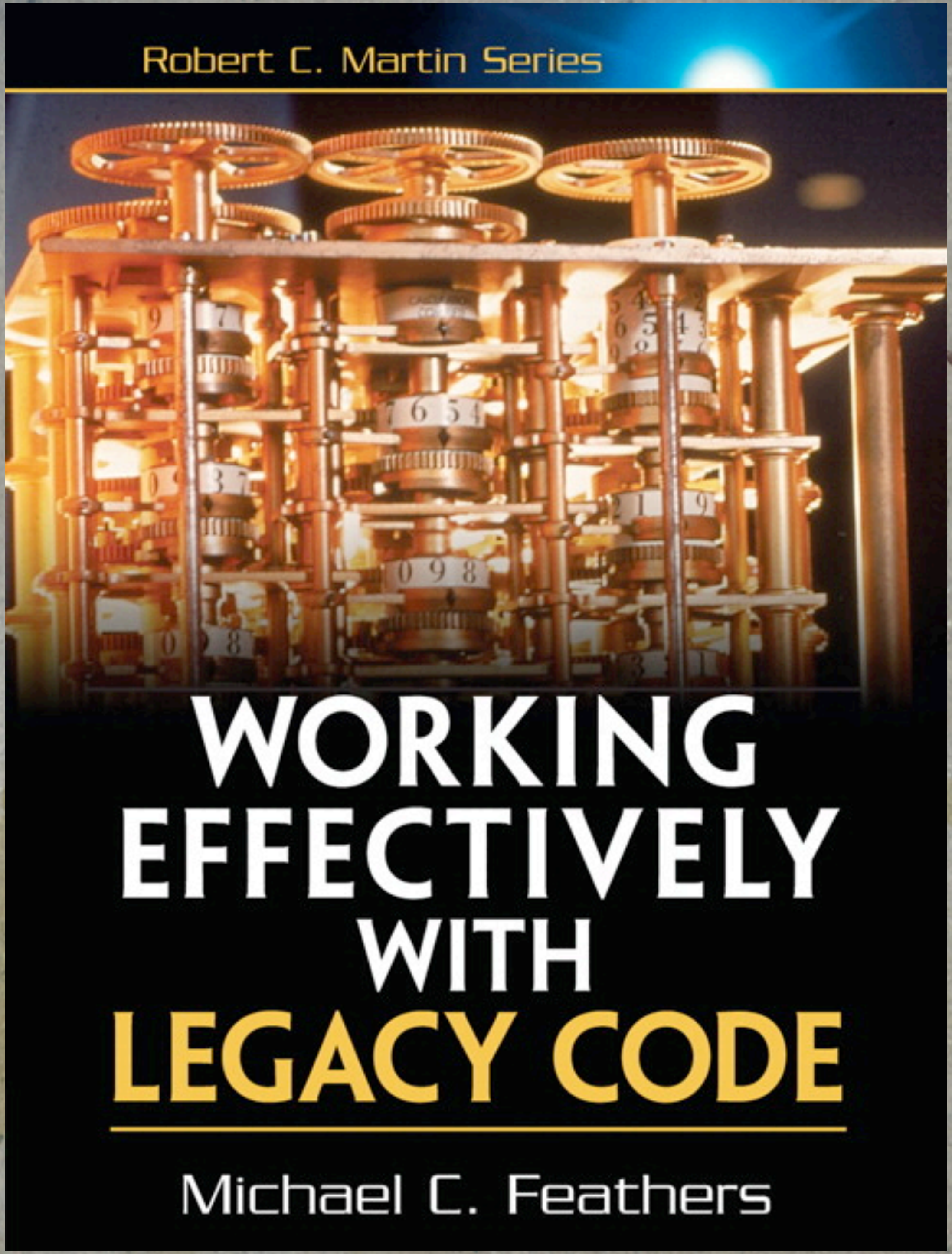
Refactoring Path

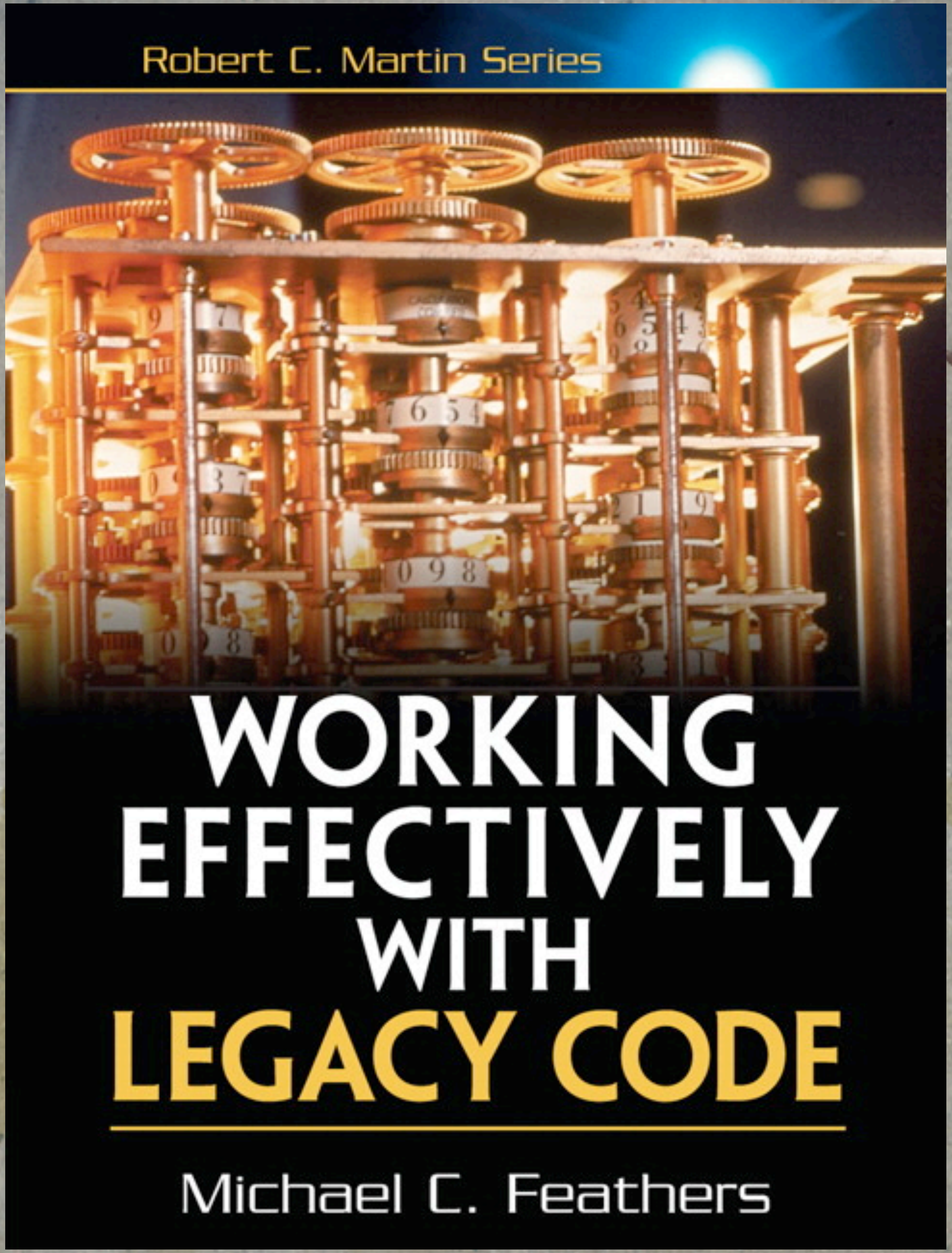
Ability to code now, satisfy non-programmers for trust earning, with assurance you haven't coded yourself into a corner.

Marriage

Marriage

Compromise


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers

Good Book

It’s Ok. It’ll Be Ok.

It’s Ok. It’ll Be Ok.

- Lose some battles...

It’s Ok. It’ll Be Ok.

- Lose some battles...
- ...as long as you win the war.

Detractors

Detractors

- Don’t have their Poetic License

Detractors

- Don’t have their Poetic License
- Fear

Detractors

- Don't have their Poetic License
- Fear
- Understand the Rules & Ramifications

Detractors

Detractors

- Ivory Zealots

Detractors

- Ivory Zealots
- Scars

Detractors

- Ivory Zealots
- Scars
- Don’t Believe in their Refactoring Ability

Detractors

- Ivory Zealots
- Scars
- Don’t Believe in their Refactoring Ability
- or Yours

Refactoring Paths

Refactoring Paths

- Refactoring is Temporal

Refactoring Paths

- Refactoring is Temporal
- Architecture is not

Refactoring Paths

- Refactoring is Temporal
- Architecture is not
- Architecture is Plan

Refactoring Paths

- Refactoring is Temporal
- Architecture is not
- Architecture is Plan
- Refactoring is a Strategy

Refactoring Path

Refactoring Path

- Know the rule(s) ur breaking

Refactoring Path

- Know the rule(s) ur breaking
- do it

Refactoring Path

- Know the rule(s) ur breaking
- do it
- document what needs to change

Refactoring Path

- Know the rule(s) ur breaking
- do it
- document what needs to change
- when you have time, fix it

Refactoring Paths

Refactoring Paths

- Use a *

Refactoring Paths

- Use a *
- Cast It

Refactoring Paths

Refactoring Paths

- Make an Object

Refactoring Paths

- Make an Object
- Make a ValueObject

Refactoring Path

Refactoring Path

- Event with public var data:*;

Refactoring Path

Refactoring Path

- Scouting

Refactoring Path

- Scouting
- Bear Bait

Refactoring Path

- Scouting
- Bear Bait
- Intentional Parsing Throws

Refactoring Path

- Scouting
- Bear Bait
- Intentional Parsing Throws
- Intentional Grenades

Refactoring Path

Refactoring Path

- if if if if if if

Refactoring Path

- if if if if if if
- then then then then

Refactoring Path

- if if if if if if
- then then then then
- badger badger badger badger

Refactoring Path

- if if if if if if
- then then then then
- badger badger badger badger
- make a switch

Refactoring Path

- if if if if if if
- then then then then
- badger badger badger badger
- make a switch
- make functions

Refactoring Paths

Refactoring Paths

- <s:Component> vs Item Renderer

Refactoring Paths

- <s:Component> vs Item Renderer
- Passive

Priority Pyramid

Priority Pyramid

1. Report

Priority Pyramid

1. Report
2. Understanding

Priority Pyramid

1. Report
2. Understanding
3. Trust

Priority Pyramid

1. Report
2. Understanding
3. Trust
4. Lead

Priority Pyramid

Priority Pyramid

5. Build

Priority Pyramid

5. Build
6. Explosions

Priority Pyramid

5. Build
6. Explosions
7. Diagnostics

Priority Pyramid

5. Build
6. Explosions
7. Diagnostics
8. Architecture

Priority Pyramid

5. Build
6. Explosions
7. Diagnostics
8. Architecture
9. Workflow

Pyramid: Report

Pyramid: Report

- Who is my boss?

Pyramid: Report

- Who is my boss?
- What will make you happy when I leave?

Pyramid: Understanding

Pyramid: Understanding

- Listen

Pyramid: Understanding

- Listen
- What's the True Goal?

Pyramid: Understanding

- Listen
- What's the True Goal?
- What are the problems?

Pyramid: Understanding

- Listen
- What's the True Goal?
- What are the problems?
- What are the challenges?

Pyramid: Understanding

Pyramid: Understanding

- Explore & Document

Pyramid: Understanding

- Explore & Document
  - Learn Data Model

Pyramid: Understanding

- Explore & Document
  - Learn Data Model
  - Learn Framework

Pyramid: Understanding

- Explore & Document
  - Learn Data Model
  - Learn Framework
  - Understanding the Story

Pyramid: Understanding

Pyramid: Understanding

- Looking for Red Flags

Pyramid: Understanding

- Looking for Red Flags
- Looking for Mines

Pyramid: Understanding

- Looking for Red Flags
- Looking for Mines
- Looking for Validation

Pyramid: Understanding

Pyramid: Understanding

- Dissect People

Pyramid: Understanding

- Dissect People
- What makes them tick?

Pyramid: Understanding

- Dissect People
- What makes them tick?
- Who listens to who?

Pyramid: Trust

Pyramid: Trust

- Make Friends

Pyramid: Trust

- Make Friends
- Provide Immediate Value

Pyramid: Trust

Pyramid: Trust

- Check in code into source control 1st week

Pyramid: Trust

- Check in code into source control lst week
- fix a problem

Pyramid: Trust

- Check in code into source control 1st week
- fix a problem
- Only get 1 change at 1st impression

Pyramid: Trust

- Check in code into source control 1st week
- fix a problem
- Only get 1 change at 1st impression
- Under Promise, Over Deliver

Pyramid: Trust

Pyramid: Trust

- Stop Fire Drills

Pyramid: Trust

- Stop Fire Drills
- Empower Through Diagnostic Tools

Pyramid: Trust

- Stop Fire Drills
- Empower Through Diagnostic Tools
- Easy to use, visible logger

Pyramid: Trust

- Stop Fire Drills
- Empower Through Diagnostic Tools
- Easy to use, visible logger
- Easy to use, visible environment info

Pyramid: Trust

Pyramid: Trust

- Update on Direction

Pyramid: Trust

- Update on Direction
- Provide Transparency

Pyramid: Trust

- Update on Direction
- Provide Transparency
- What, why, and when

Pyramid: Trust

- Update on Direction
- Provide Transparency
- What, why, and when
- Plan A, Plan B

Pyramid: Lead

Pyramid: Lead

- Be Positive

Pyramid: Lead

- Be Positive
- Short Term Goals

Pyramid: Lead

- Be Positive
- Short Term Goals
- Long Term Goals

Pyramid: Lead

Pyramid: Lead

- Plan

Pyramid: Lead

- Plan
- 1 thing at a time

Pyramid: Lead

- Plan
- 1 thing at a time
- Assess Problems

Pyramid: Lead

- Plan
- 1 thing at a time
- Assess Problems
- Document

Pyramid: Lead

Pyramid: Lead

- Re-prioritize as needed

Pyramid: Lead

Pyramid: Lead

- Delegate

Pyramid: Lead

- Delegate
- Help

Pyramid: Lead

- Delegate
- Help
- Divide & Conquer

Pyramid: Lead

- Delegate
- Help
- Divide & Conquer
- Short Term Branches: 3 days

Experiences

Hearts & Minds

Fight Only to Win

People on Pedestals

Idle Hands

Winston Wolfe

[show video]
[dissect]

Conclusions

Conclusions

- Exceptions: Catch & Log

Conclusions

- Exceptions: Catch & Log
- Refactoring: Remember The Paths

Conclusions

- Exceptions: Catch & Log
- Refactoring: Remember The Paths
- Listen, Build Trust, and Lead

Thank You! Questions?

Thank You! Questions?

Jesse Warden

Thank You! Questions?

Jesse Warden

webappsolution.com

Thank You! Questions?

Jesse Warden

webappsolution.com

jessewarden.com

Thank You! Questions?

Jesse Warden

webappsolution.com

jessewarden.com

@jesterxl