Lessons Learned
from
Refactors and Rearchitectures

Josh Goldberg

@ JoshuaKGoldberg /.com

Every three slides, if I don’t take a sip of water, yell “hydrate!”.

@ JoshuaKGoldberg /.com

1

2

3

@ JoshuaKGoldberg /.com

Why do experienced devs still make big mistakes?

@ JoshuaKGoldberg /.com

Five Whys

@ JoshuaKGoldberg /.com

How do we set systems up to prevent big mistakes?

@ JoshuaKGoldberg /.com

@ JoshuaKGoldberg /.com

@ JoshuaKGoldberg /.com

@ JoshuaKGoldberg /.com

Situation 1: Buggy Release

@ JoshuaKGoldberg /.com

Why was the release so buggy?

Lesson 1.1:
Test

@ JoshuaKGoldberg /.com

Lesson 1.2:
Test before release

@ JoshuaKGoldberg /.com

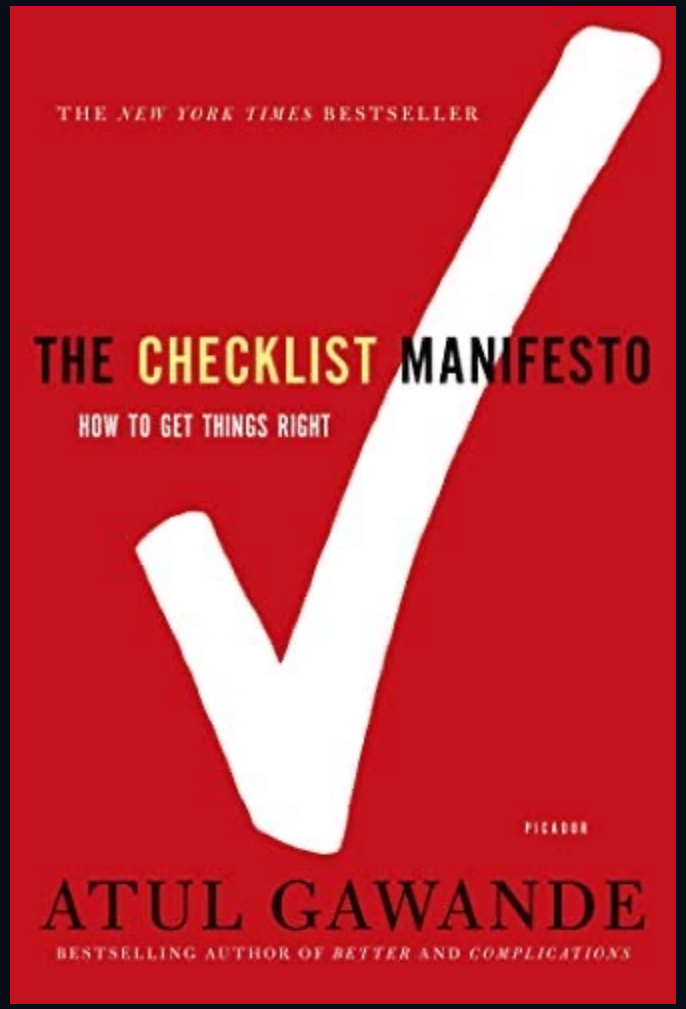
Lesson 1.3:
Don’t punt P0s

@ JoshuaKGoldberg /.com

Lesson 1.4:
Release checklists

@ JoshuaKGoldberg /.com

Situation 1: Buggy Release



@ JoshuaKGoldberg /.com

@ JoshuaKGoldberg /.com

Situation 2: Bus Person

@ JoshuaKGoldberg /.com

Why was our expertise concentrated in one person?

Situation 2: Bus Person

Lesson 2.1:
Include knowledge sharing time

@ JoshuaKGoldberg /.com

Lesson 2.2:
Schedule 1:1 pairings

@ JoshuaKGoldberg /.com

Situation 2: Bus Person

Lesson 2.3:
Schedule mob sessions

@ JoshuaKGoldberg /.com

Situation 2: Bus Person

Lesson 2.4:
Schedule documentation

@ JoshuaKGoldberg /.com

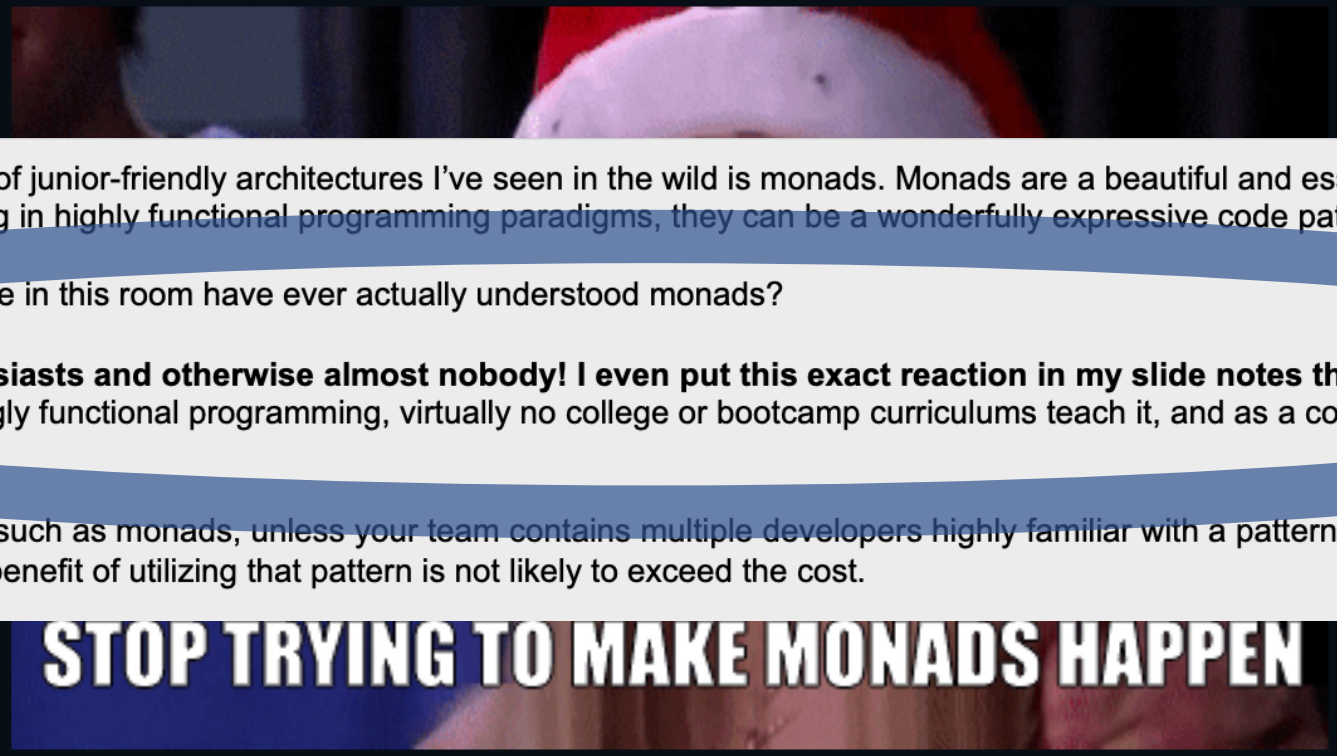
Situation 2: Bus Person

Lesson 2.5:
Junior-friendly architectures

@ JoshuaKGoldberg /.com


Code
Complexity
Developer Skill

Situation 2: Bus Person



One of the most common violations of junior-friendly architectures I've seen in the wild is monads. Monads are a beautiful and essential part of pure functional programming. If your team is working in highly functional programming paradigms, they can be a wonderfully expressive code pattern.

But show of hands, how many people in this room have ever actually understood monads?

That's right – a few jubilant enthusiasts and otherwise almost nobody! I even put this exact reaction in my slide notes that almost nobody uses monads – because almost no teams use strongly functional programming, virtually no college or bootcamp curriculums teach it, and as a concept it often takes a few tries to understand.

For non-essential software patterns such as monads, unless your team contains multiple developers highly familiar with a pattern and is well positioned to onboard new developers to that pattern, the benefit of utilizing that pattern is not likely to exceed the cost.

@ JoshuaKGoldberg /.com

Situation 3: Jerk Seniority

@ JoshuaKGoldberg /.com

Situation 3: Jerk Seniority

Lesson 3.1:
Respect best practices

@ JoshuaKGoldberg /.com

Lesson 3.2:
Be humble and listen

@ JoshuaKGoldberg /.com

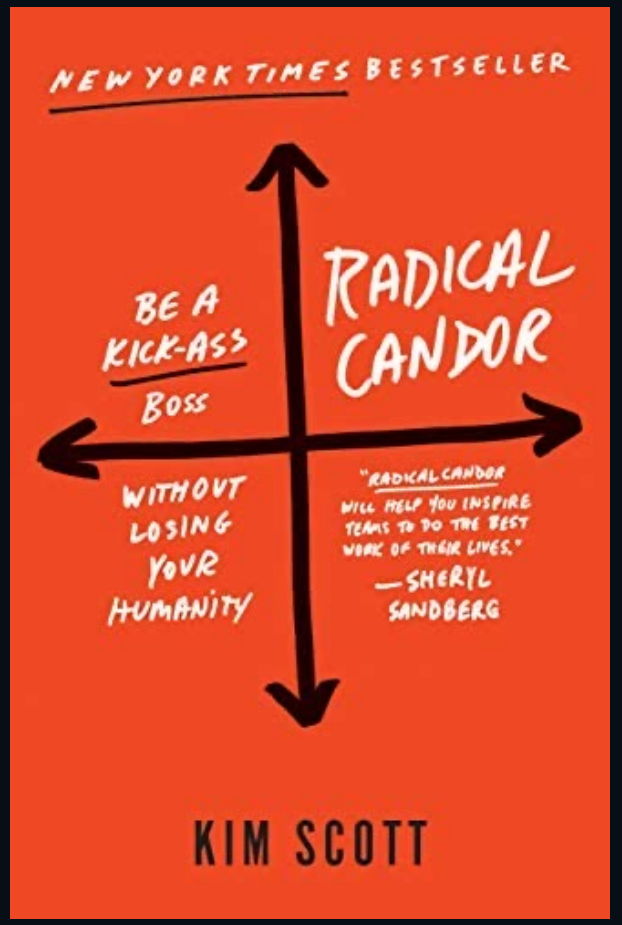
Lesson 3.3:
Prioritize learning

@ JoshuaKGoldberg /.com

Lesson 3.4:
Involve management when needed

@ JoshuaKGoldberg /.com

Situation 3: Jerk Seniority



@ JoshuaKGoldberg /.com

Situation 3: Jerk Seniority

Lesson 3.5:
Dev 🤝 PM

@ JoshuaKGoldberg /.com

@ JoshuaKGoldberg /.com

Situation 4: No Seniority

Lesson 4.1:
Nothing beats real experience

@ JoshuaKGoldberg /.com

Lesson 4.2:
Give first-timers support

@ JoshuaKGoldberg /.com

Situation 3: Jerk Seniority

Rule of Three:

1. Observe
2. Pair
3. Lead

@ JoshuaKGoldberg /.com

Lesson 4.3:
Underload seniors

@ JoshuaKGoldberg /.com

Situation 4: No Seniority

Many teams have a hard time underloading seniors because they themselves are overloaded. And by “overloaded” I mean “given project time roughly equal to their capacity”.

By show of hands who here frequently works in projects that are completed on time? Anybody?
Basically nobody.
Again, high confidence in this one.

Assume your team will overpromise and underdeliver. Life happens. The world is collapsing around us. Underask and overcompensate.

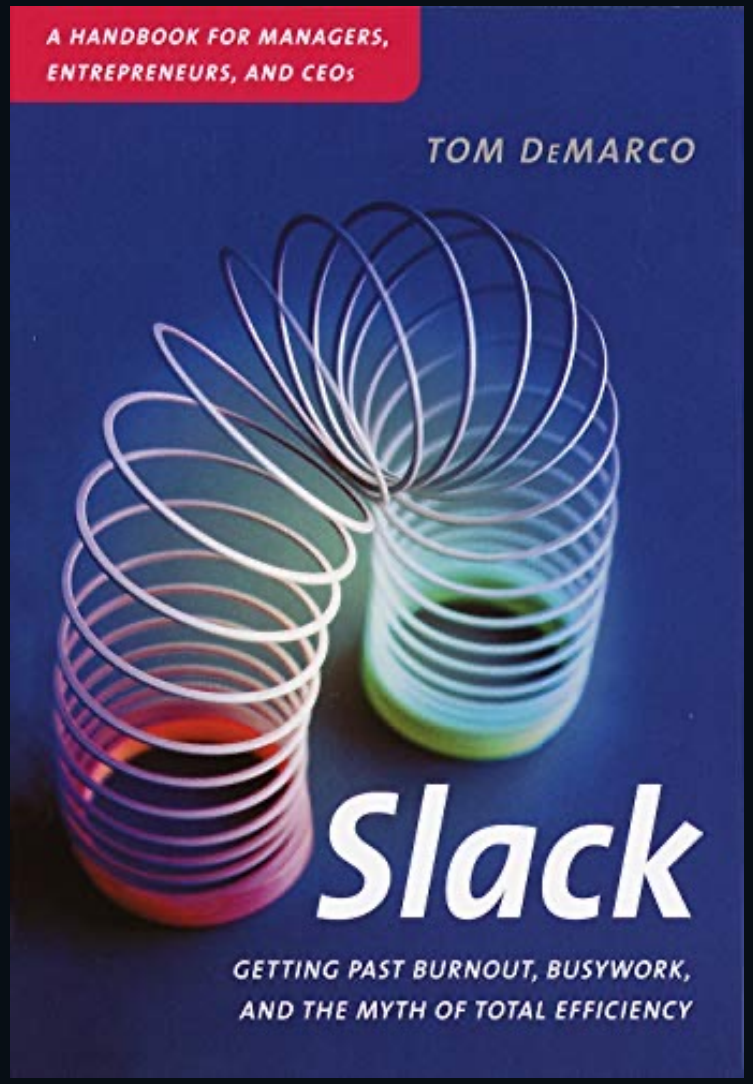
Lesson 4.4:
Underload OKRs

@ JoshuaKGoldberg /.com

Situation 4: No Seniority

@ JoshuaKGoldberg /.com

Situation 4: No Seniority



@ JoshuaKGoldberg /.com

@ JoshuaKGoldberg /.com

Situation 5: Great Success

@ JoshuaKGoldberg /.com

Situation 5: Great Success

@ JoshuaKGoldberg /.com

Lesson 5.1:
Celebrate your successes 🎉

@ JoshuaKGoldberg /.com

Situation 5: Great Success

@ JoshuaKGoldberg /.com

Lesson 5.2:
Quantify benefits

@ JoshuaKGoldberg /.com

Situation 5: Great Success

Adding a monolith page

1 config/routes.rb

```ruby
get '/example/page' => 'example#page'
```

2 app/controllers/example_controller.rb

```ruby
class ExampleController < ApplicationController
  def page
    render_portal(title: 'My Page')
  end
end
```

3 webpack/assets/portal/routes.ts

```ts
const ExamplePage = loadable(() => import('~/portal/scenes/ExamplePage'));

  // (within routesMap)
  [`${routeActions.examplePage}`]: '/example/page',

  // (within routesMeta)
  [`${routeActions.examplePage}`]: {
    scene: ExamplePage,
    pageName: 'example_page',
  },
```

4 webpack/assets/portal/state/location/actions.tsx

```ts
export const examplePage = createAction('location/EXAMPLE_PAGE');
```

5 webpack/assets/portal/scenes/ExamplePage/index.tsx

```tsx
export const ExamplePage: React.FC = () => {
  return <div>Hello, world!</div>
};

export default ExamplePage;
```

codecademy

Adding a Next.js page

1 src/pages/example/page.tsx

```tsx
export const ExamplePage: React.FC = () => {
  return <div>Hello, world!</div>;
};
```

codecademy

@ JoshuaKGoldberg /.com

Lesson 5.3:
Quantify penalties

@ JoshuaKGoldberg /.com

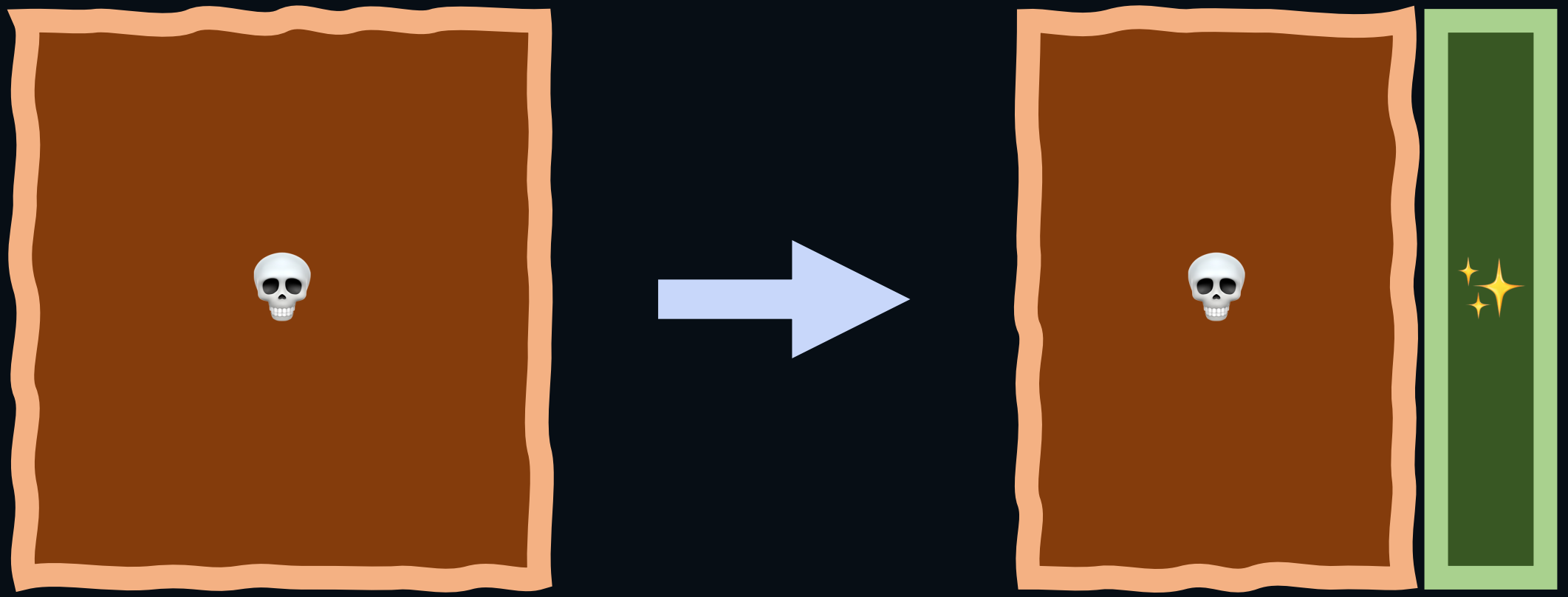
Lesson 5.4:
Incremental victories

@ JoshuaKGoldberg /.com

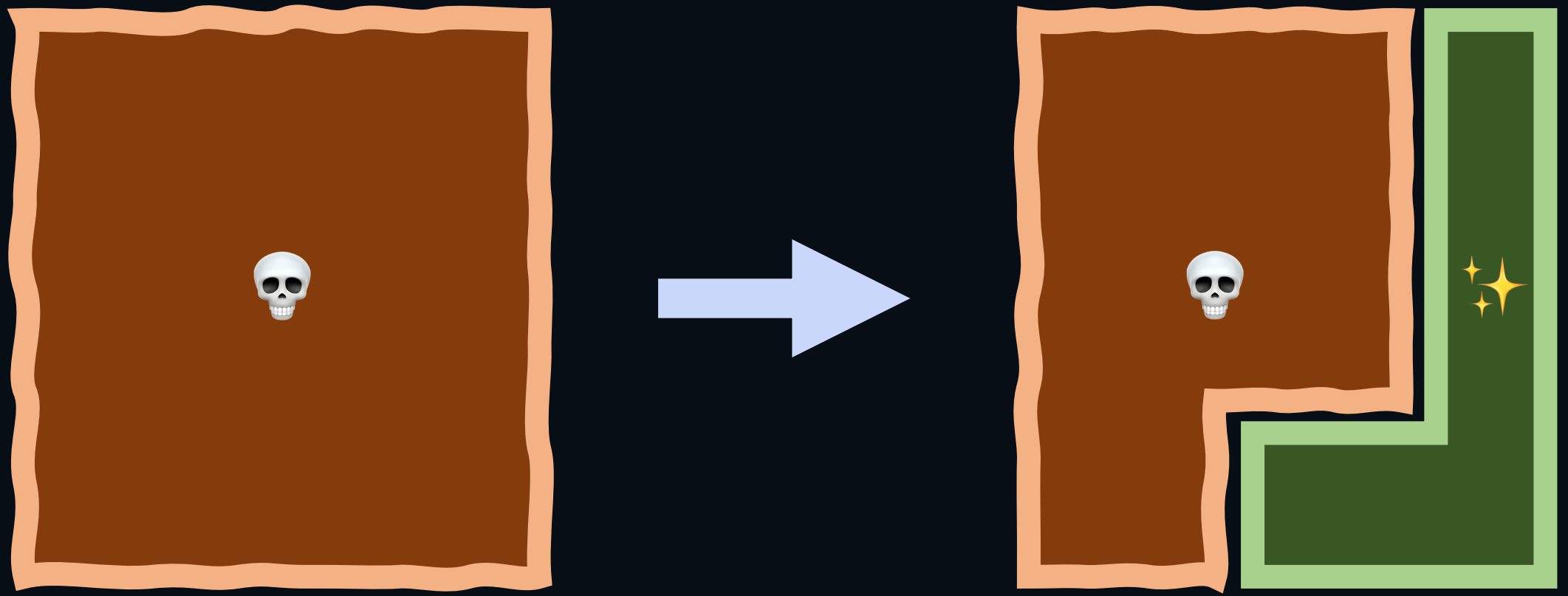
Lesson 5.5:
Incremental refactors

@ JoshuaKGoldberg /.com

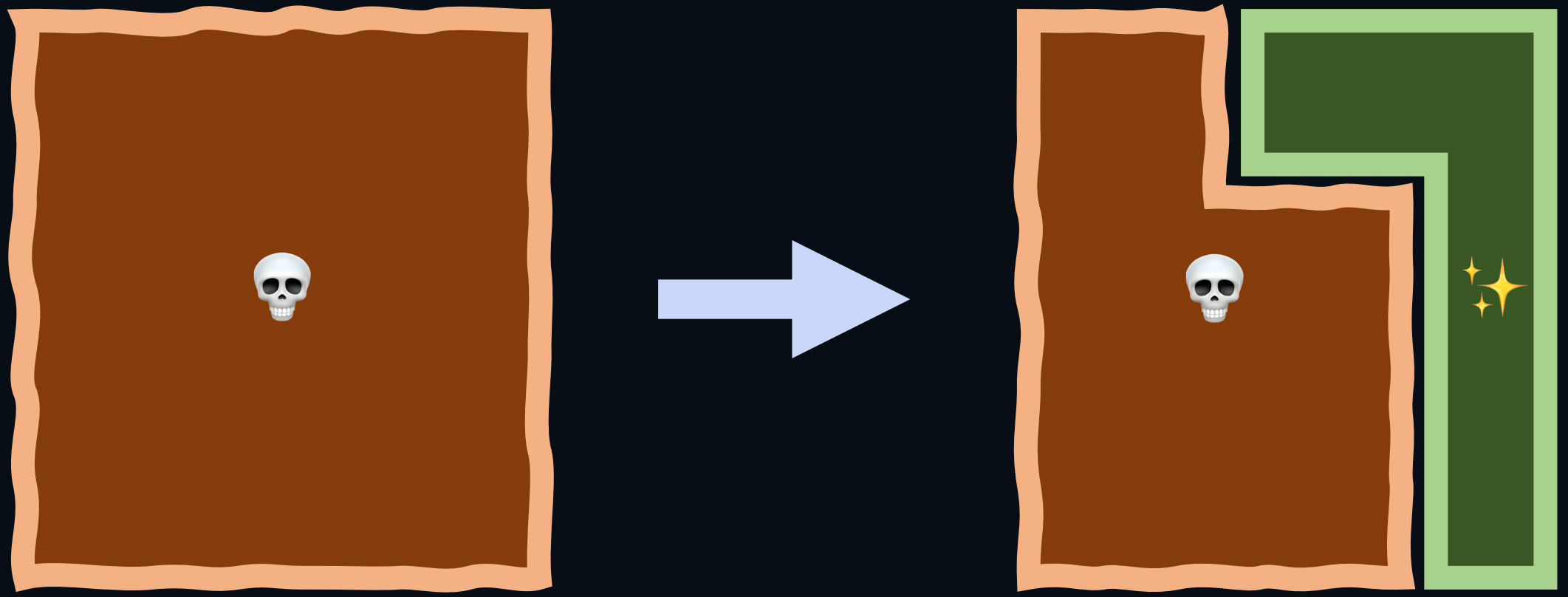
Keep Both Alive



Wrap the New

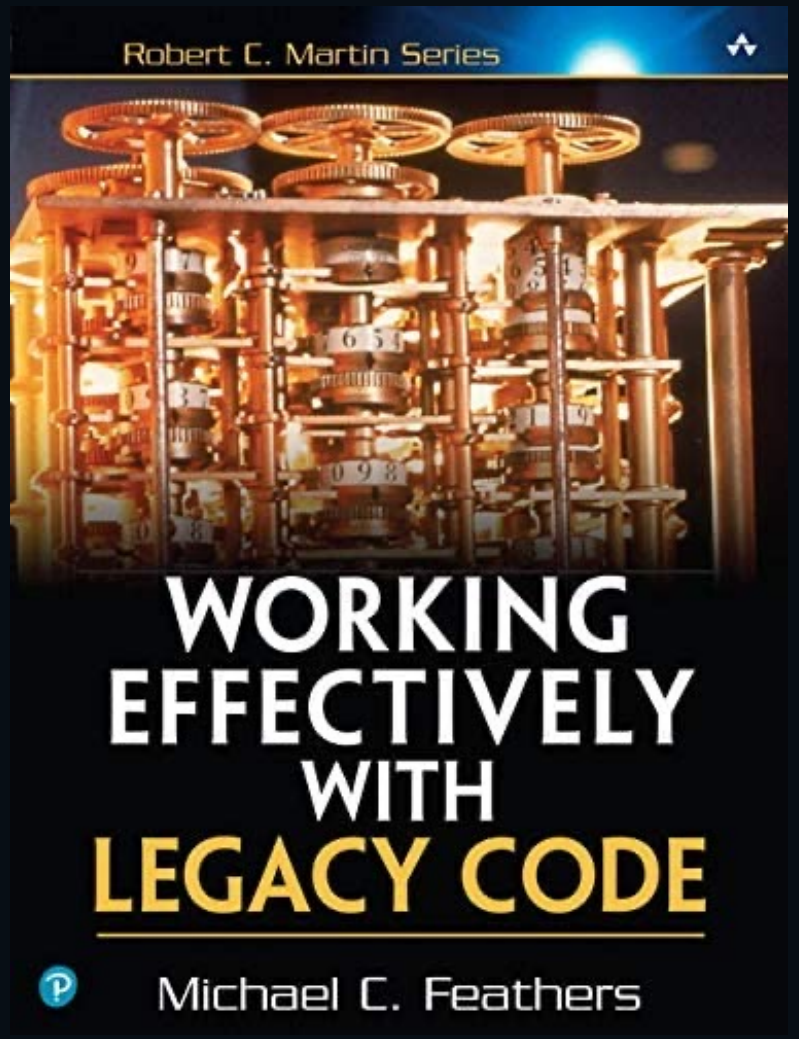


Wrap the Old



@ JoshuaKGoldberg /.com

Situation 5: Great Success



@ JoshuaKGoldberg /.com

Fin

@ JoshuaKGoldberg /.com

About Me

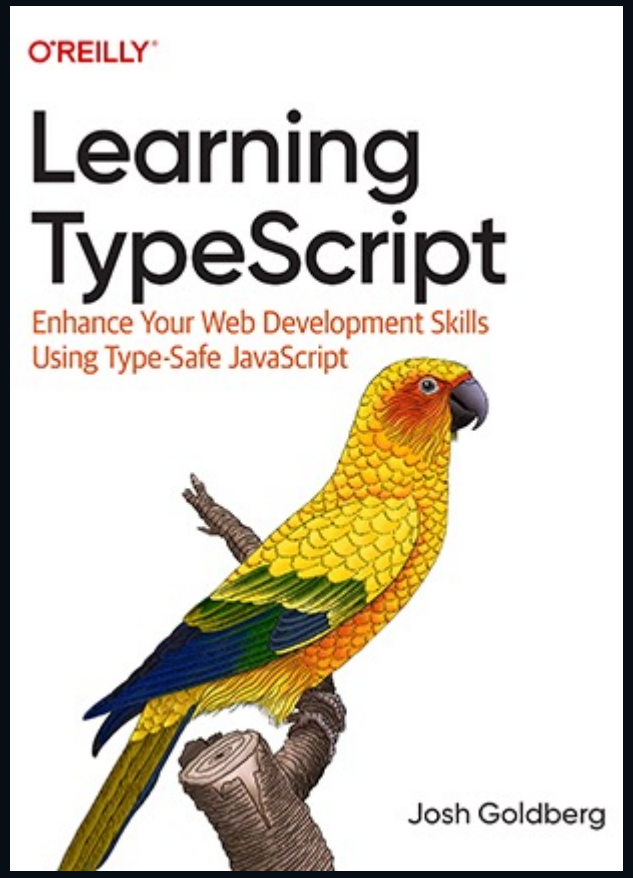
Open Source Maintainer

I focus on static analysis tooling around JavaScript & TypeScript.

Author, Learning TypeScript (O'Reilly).

@ JoshuaKGoldberg /.com

Support Me 💙



learningtypescript.com

github.com/sponsors/JoshuaKGoldberg

@ JoshuaKGoldberg /.com

Thank you!

@ JoshuaKGoldberg /.com