Learning Rust with Humility

and Three Ideas

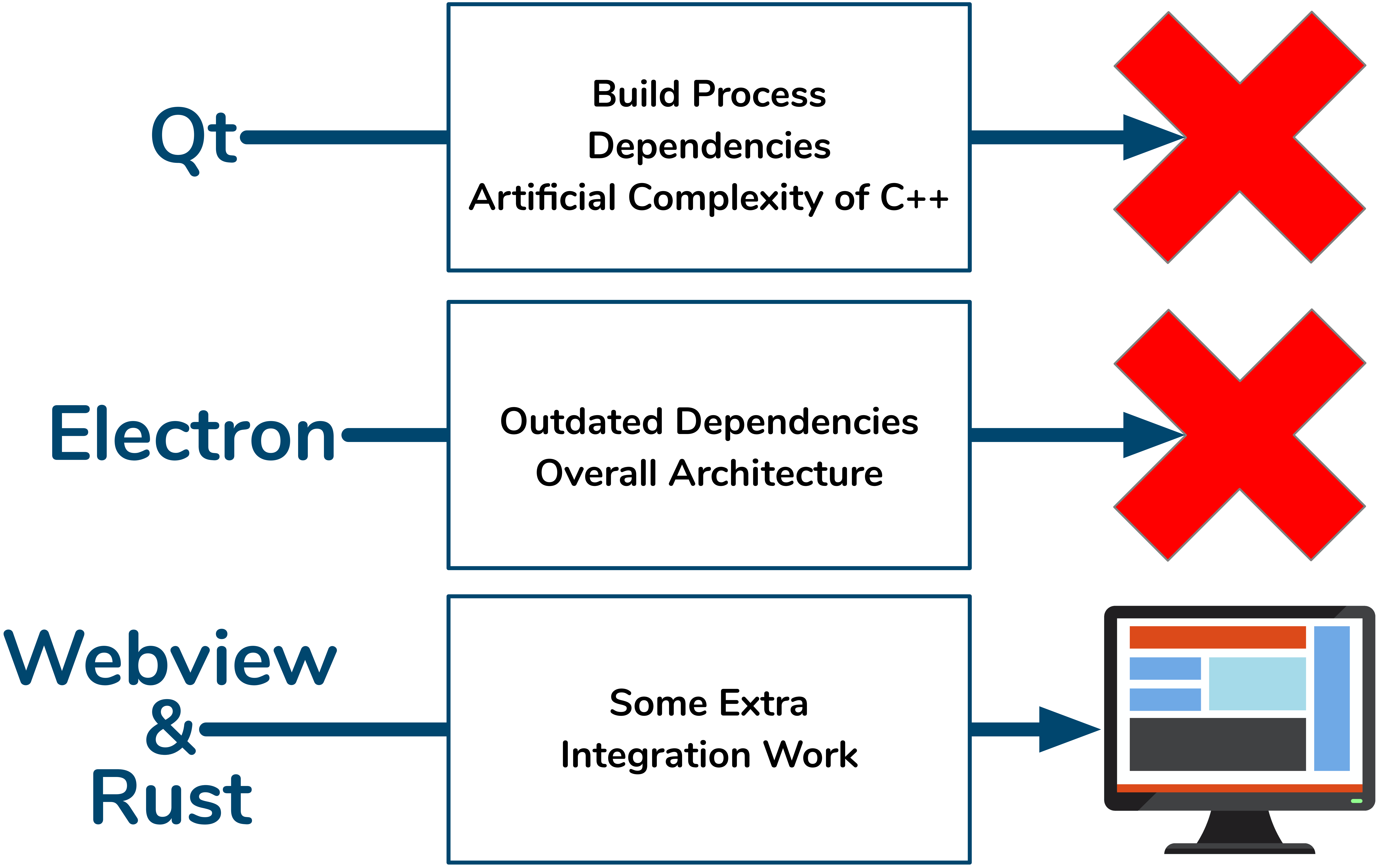
Stefan Baerisch, stefan@stbaer.com, 2020-11-07

kid image: freepik/colorfuelstudio

Idea 1

Know the Challenges

Building a Desktop App



Qt

Build Process
Dependencies
Artificial Complexity of C++

Electron

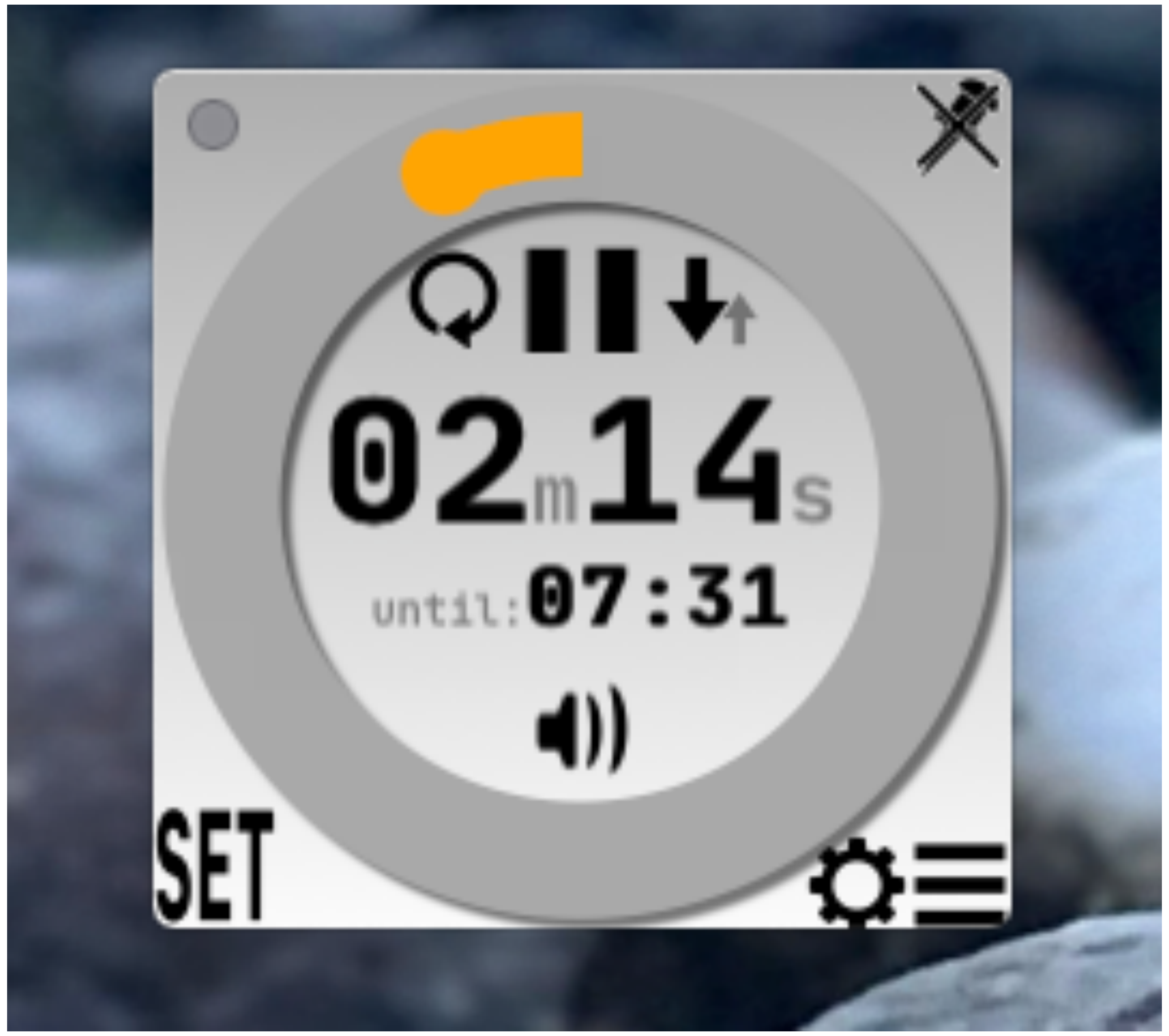
Outdated Dependencies
Overall Architecture

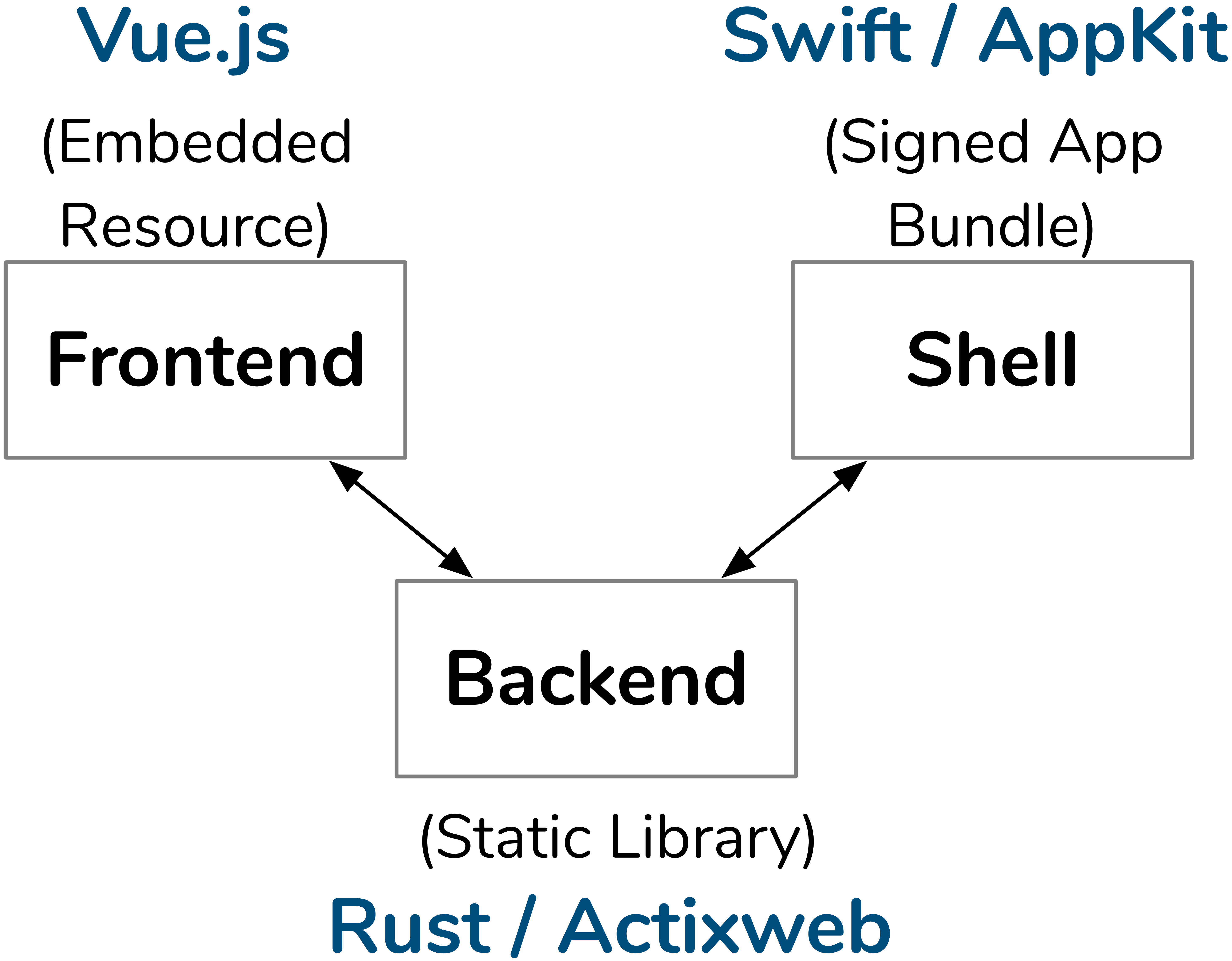
Webview
&
Rust

Some Extra
Integration Work

image: freepics / macrovector

The App





Vue.js

(Embedded Resource)

Frontend

Swift / AppKit

(Signed App Bundle)

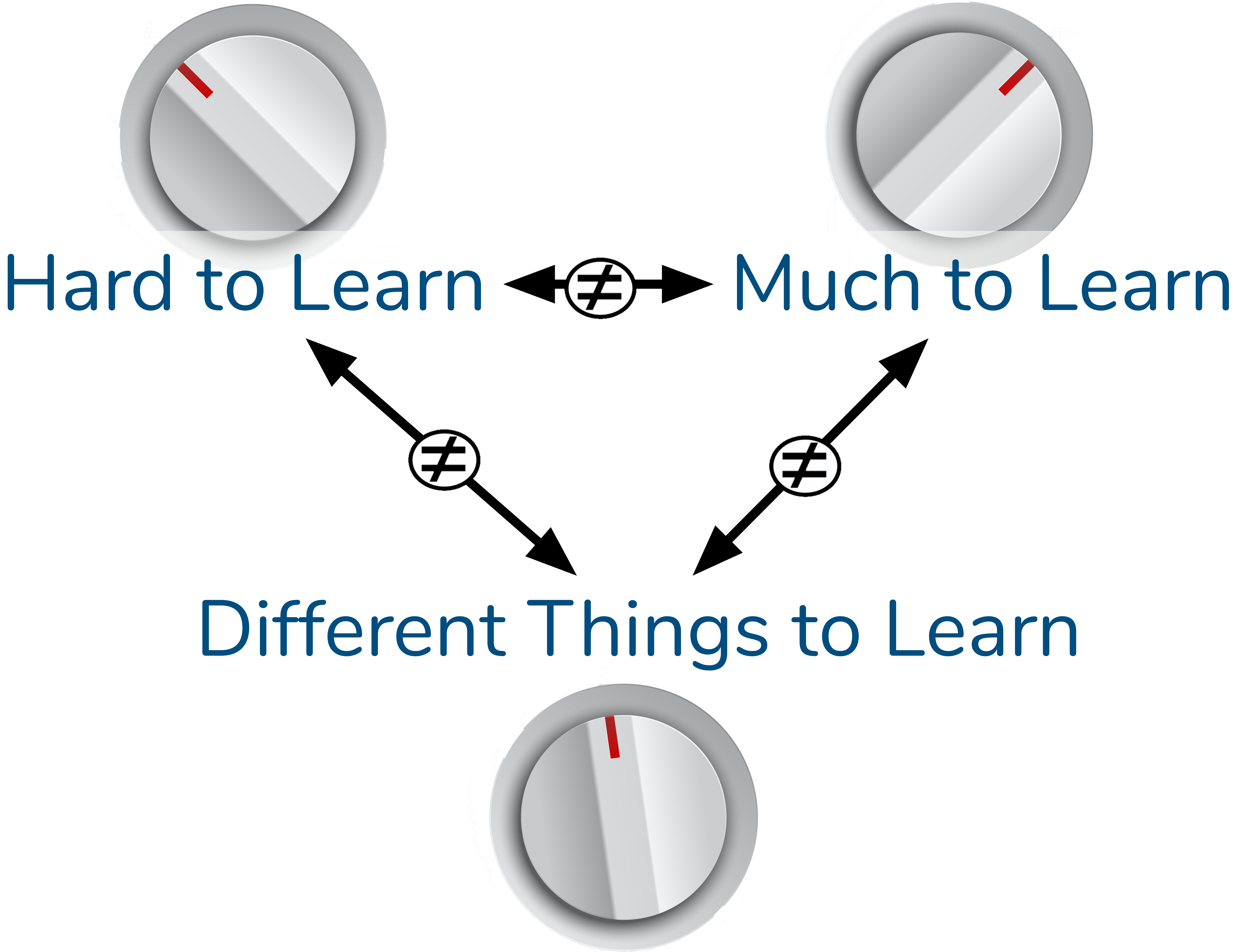
Shell

Backend

(Static Library)

Rust / Actixweb

Lessons Learned from Learning



Hard to Learn ≠ Much to Learn

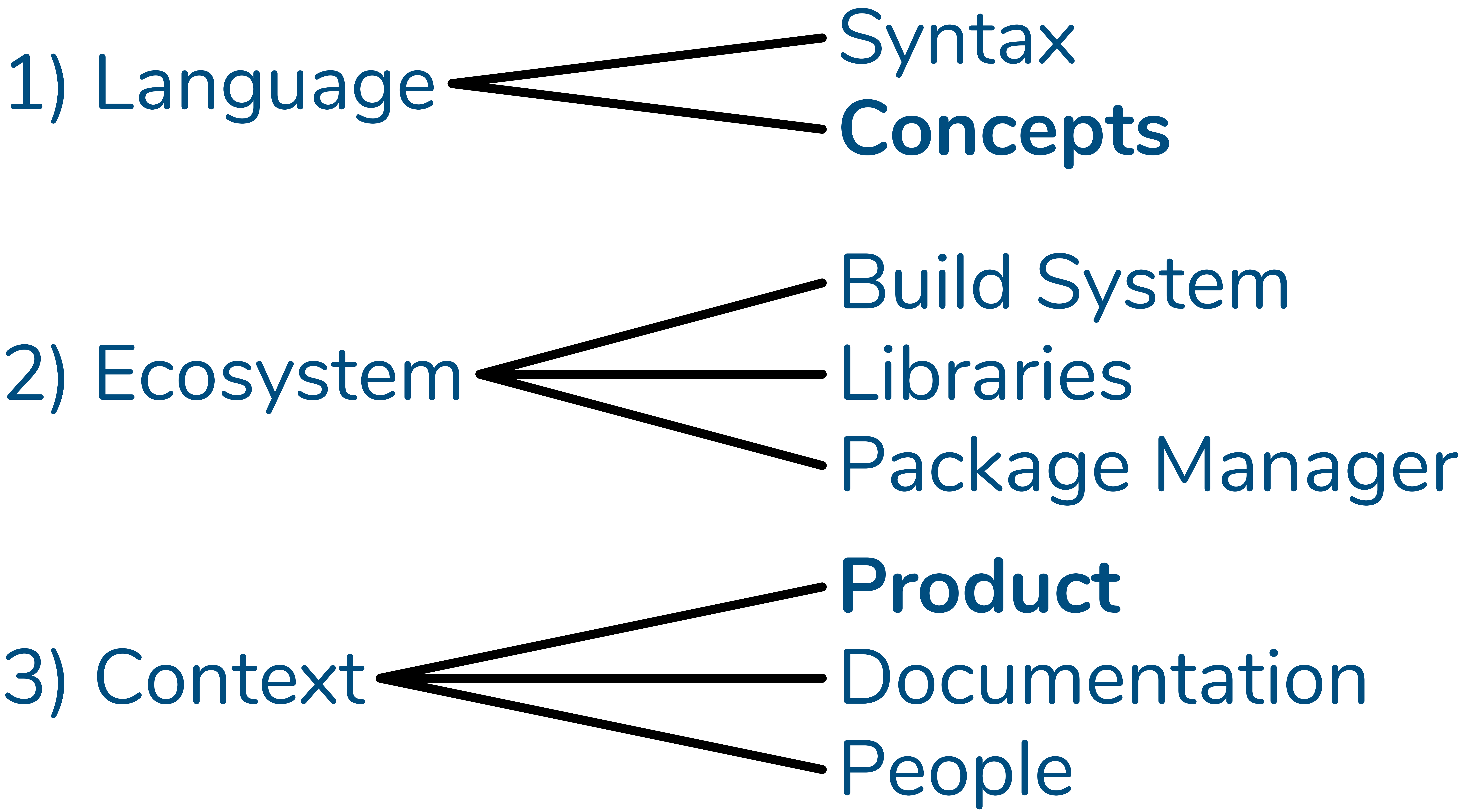
≠

≠

Different Things to Learn

image: freepik/macrovector

What makes Languages Hard to Learn



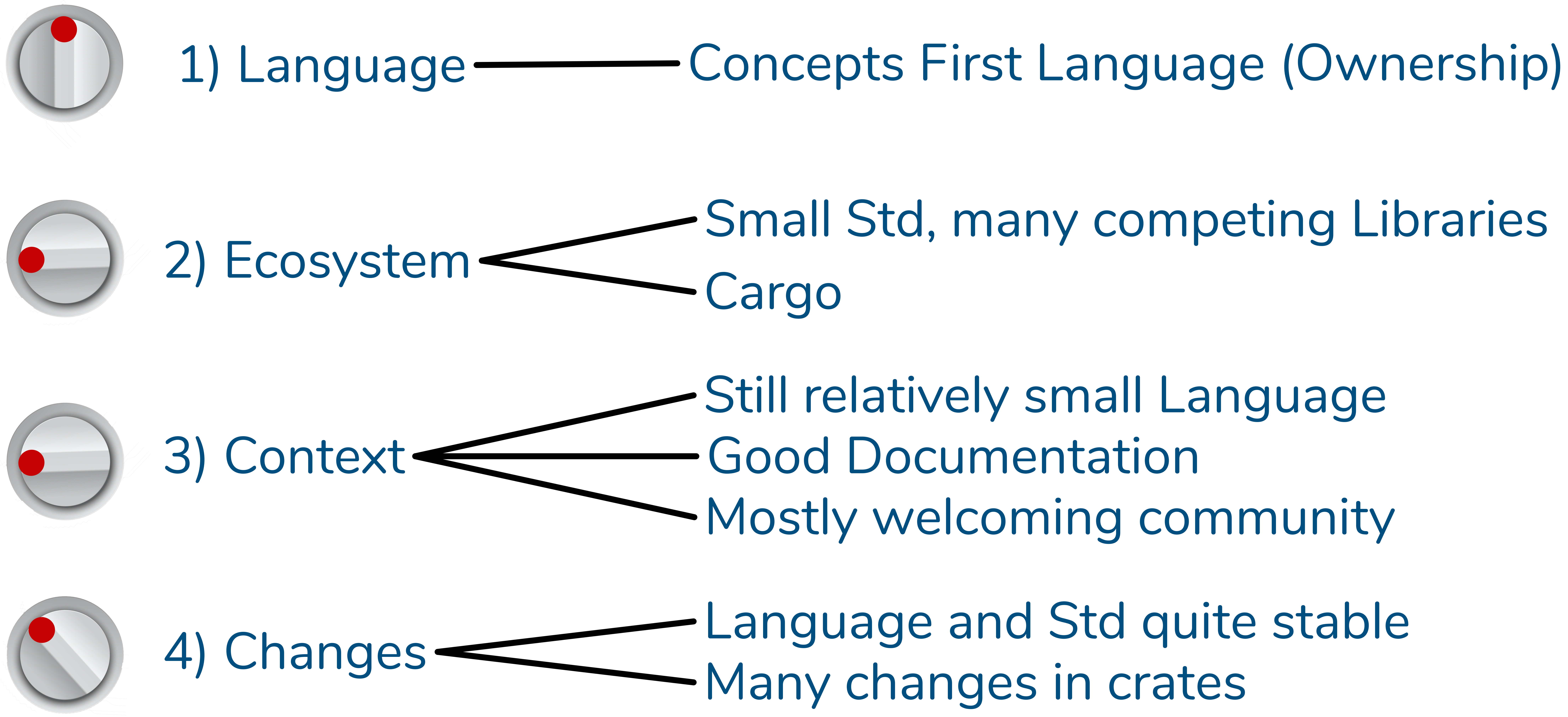
1) Language
- Syntax
- **Concepts**

2) Ecosystem
- Build System
- Libraries
- Package Manager

3) Context
- **Product**
- Documentation
- People

4) Changes

Rust as a Programming Language to Learn



1) Language — Concepts First Language (Ownership)

2) Ecosystem
- Small Std, many competing Libraries
- Cargo

3) Context
- Still relatively small Language
- Good Documentation
- Mostly welcoming community

4) Changes
- Language and Std quite stable
- Many changes in crates

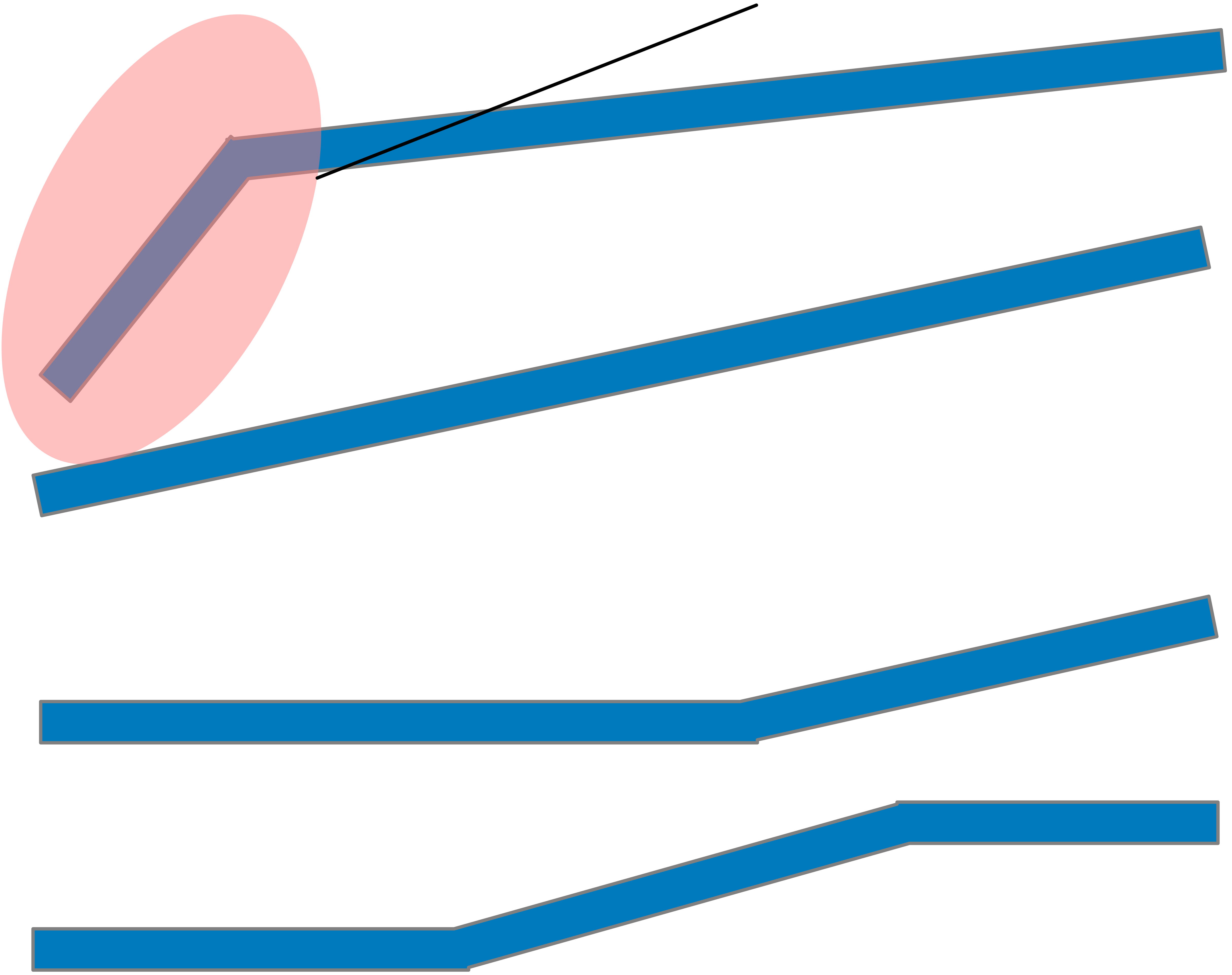
Learning Rust

1) Language

2) Ecosystem

3) Context

4) Changes

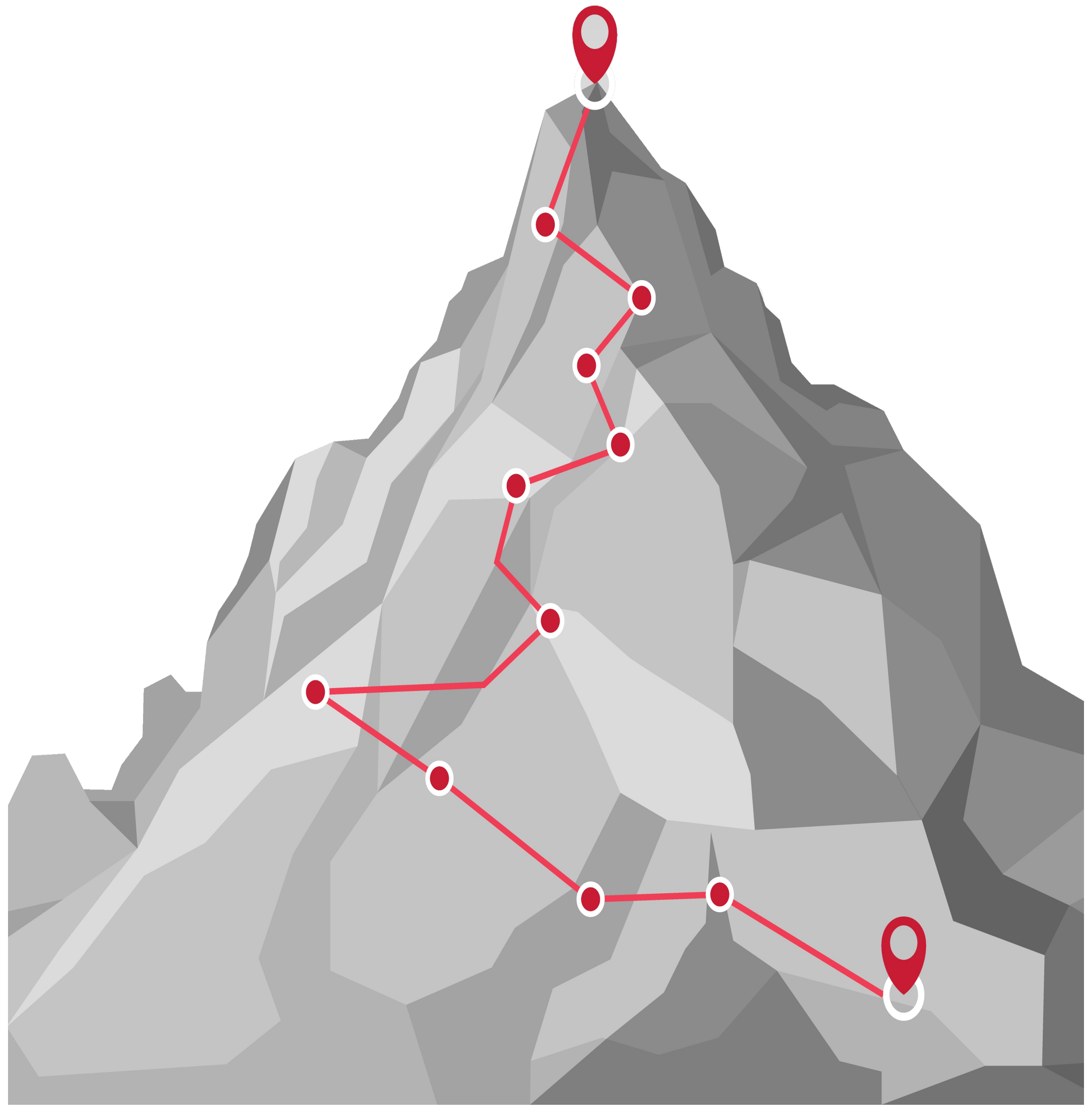


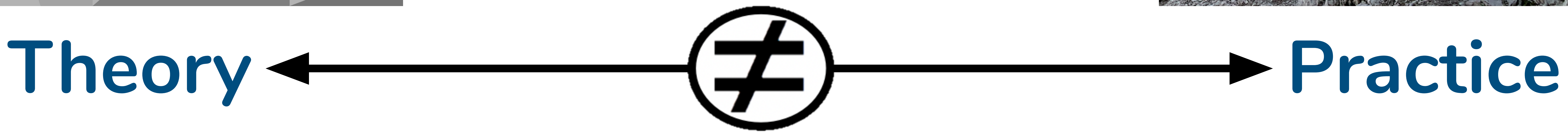
Rust’s Learning Curve is Frontloaded

Idea 2

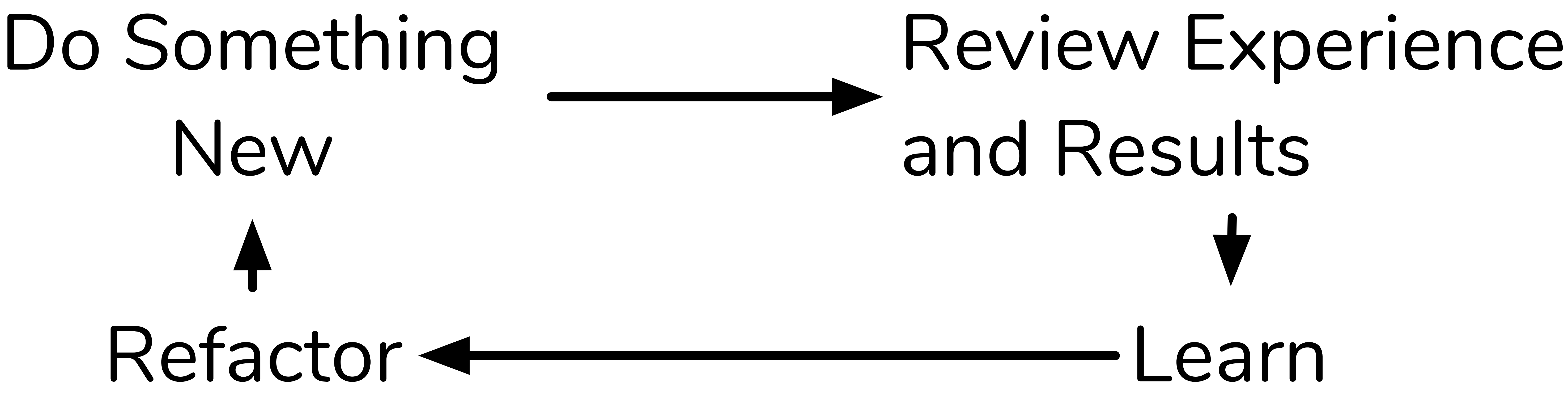
Practice

Lessons Learned from Learning





Theory ≠ Practice



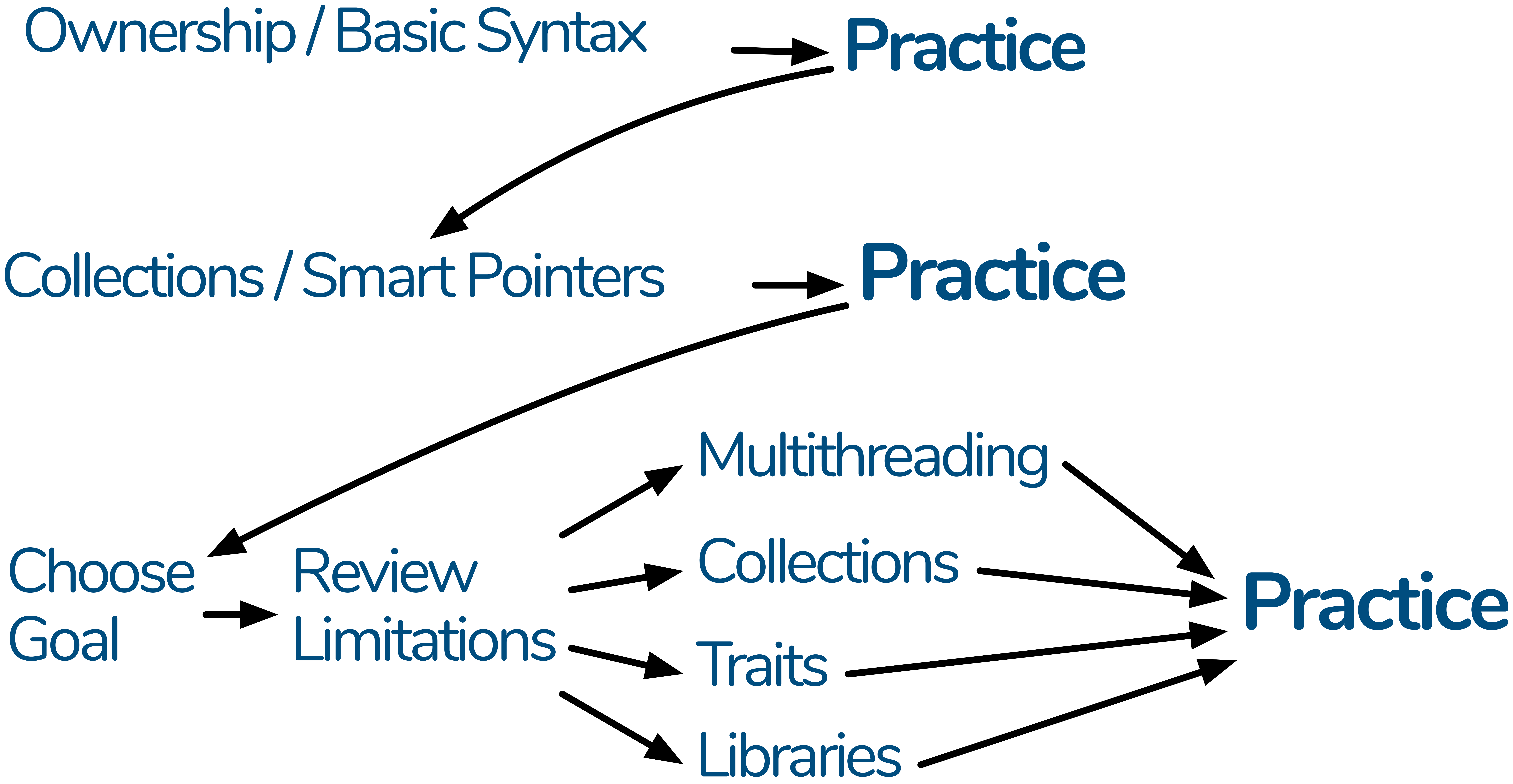
Do Something New

Review Experience and Results

Learn

Refactor

Rust Learning Roadmap



Ownership / Basic Syntax

Practice

Collections / Smart Pointers

Practice

Choose Goal

Review Limitations

Multithreading

Collections

Traits

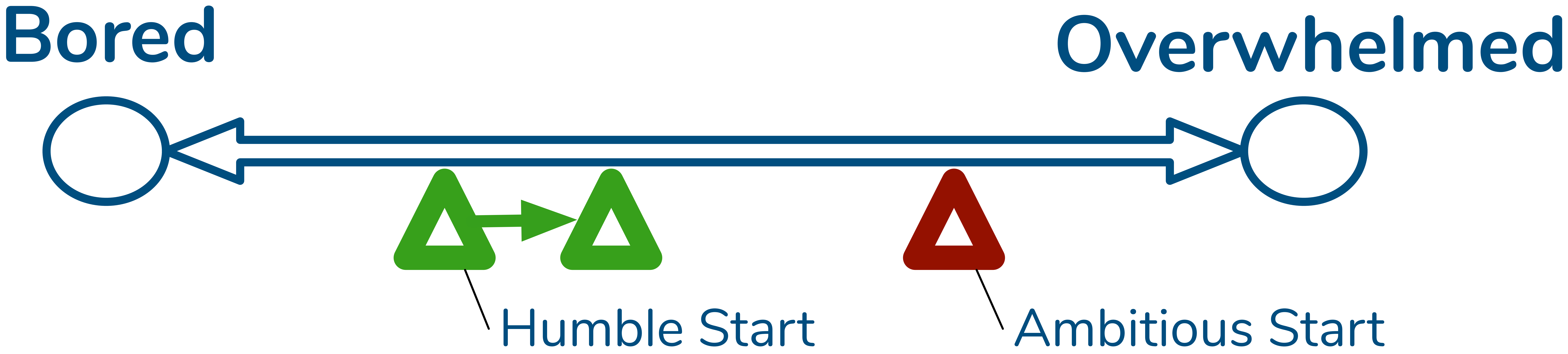
Libraries

Practice

Idea 3

Humility

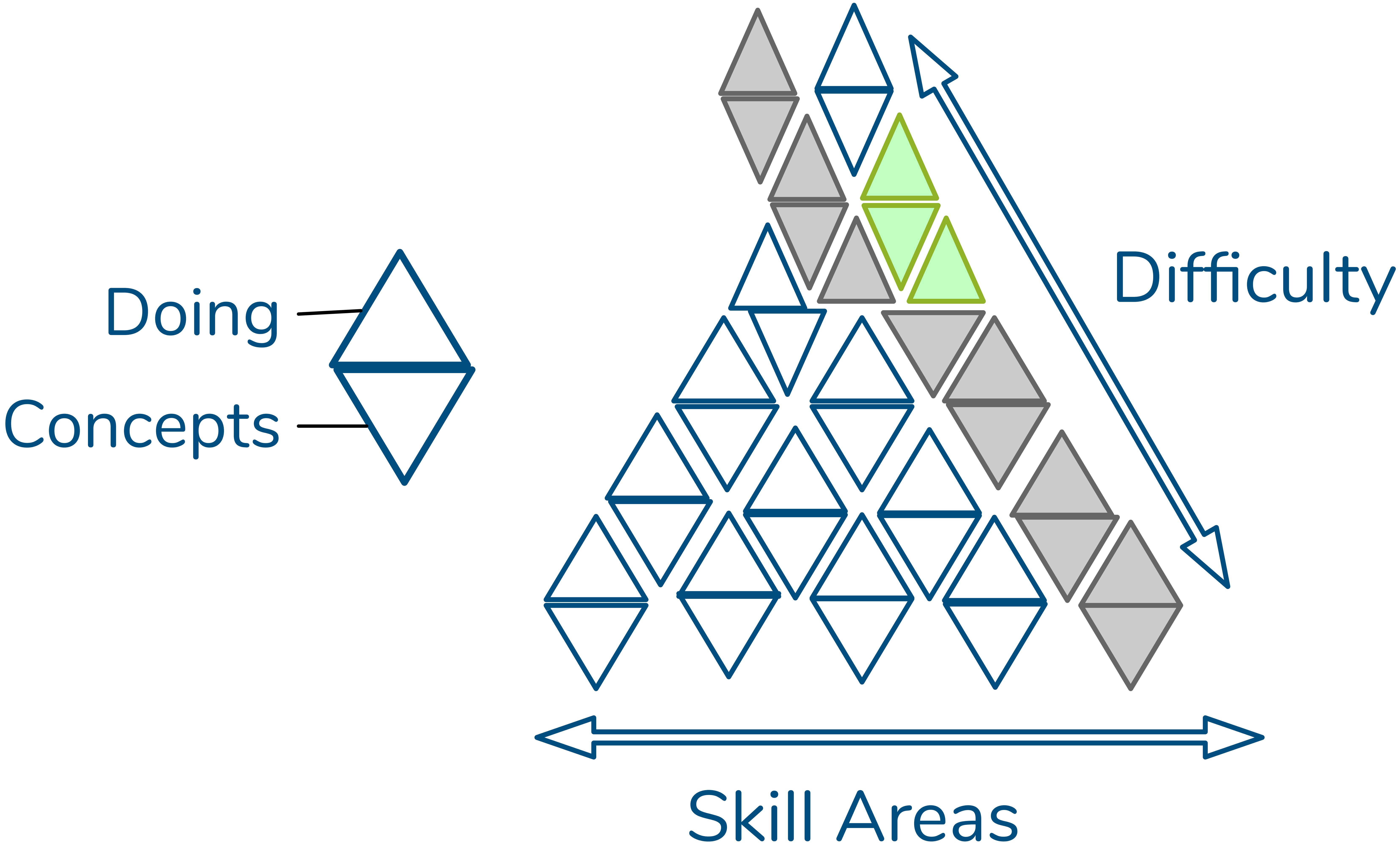
Why Learning (with) Humility



It is more productive start humble, even if it is potentially less productive

Why Learning (with) Humility

Humble, slow approach



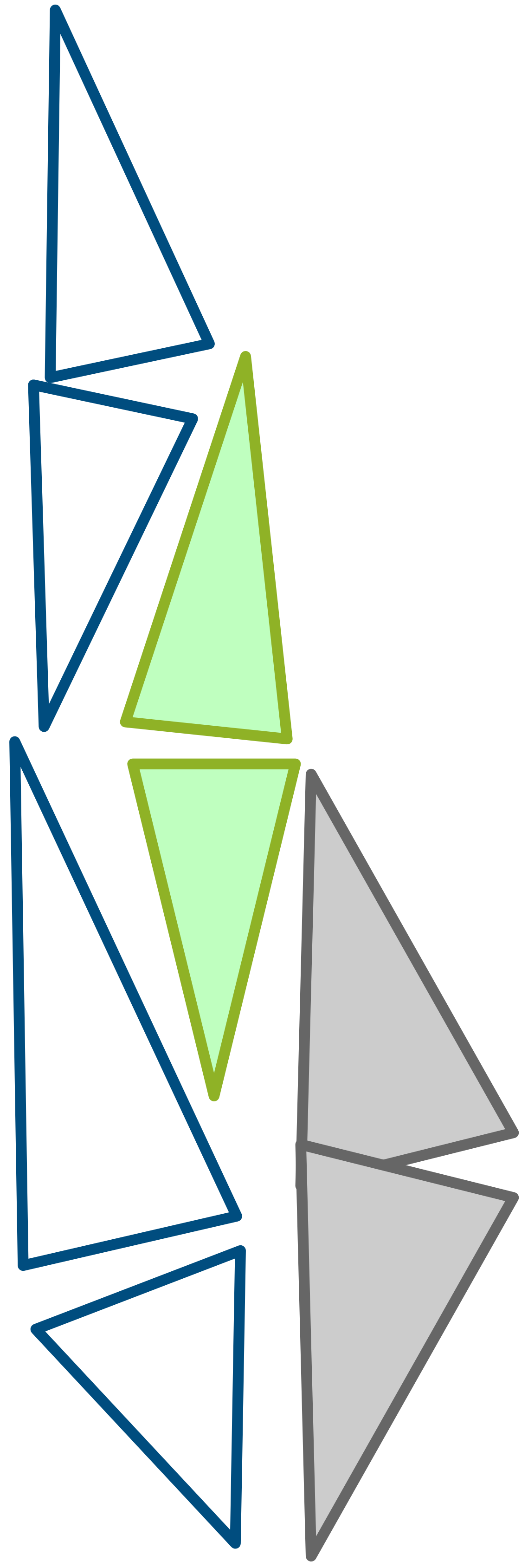
Doing

Concepts

Difficulty

Skill Areas

Risky, potentially fast approach



Humbly Learning Rust (1/2)

Simple Code is Ok (to start with)

if / else before match

write complex programs

Limited Code is Ok (to start with)

std before crate

Inefficient Code is Ok (to start with)

clone() sync I/O

Humbly Learning Rust (2/2)

Unsafe Code is not Ok (to start with)

would make it harder to reason about the code

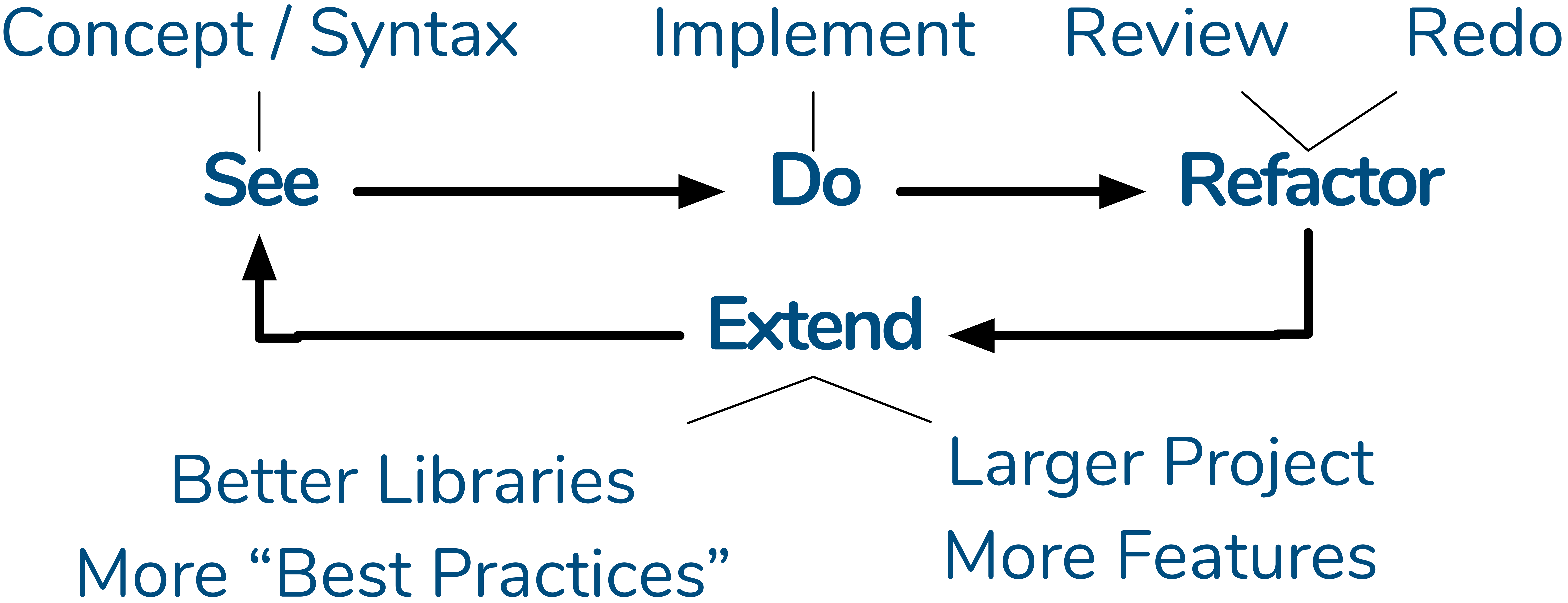
Boring is Ok (to start with)

no non-std macros
no nightly

Small Steps are Ok (always)

gives a working state to step back to
keeps mental load low

Summary



Concept / Syntax

Implement

Review

Redo

See

Do

Refactor

Extend

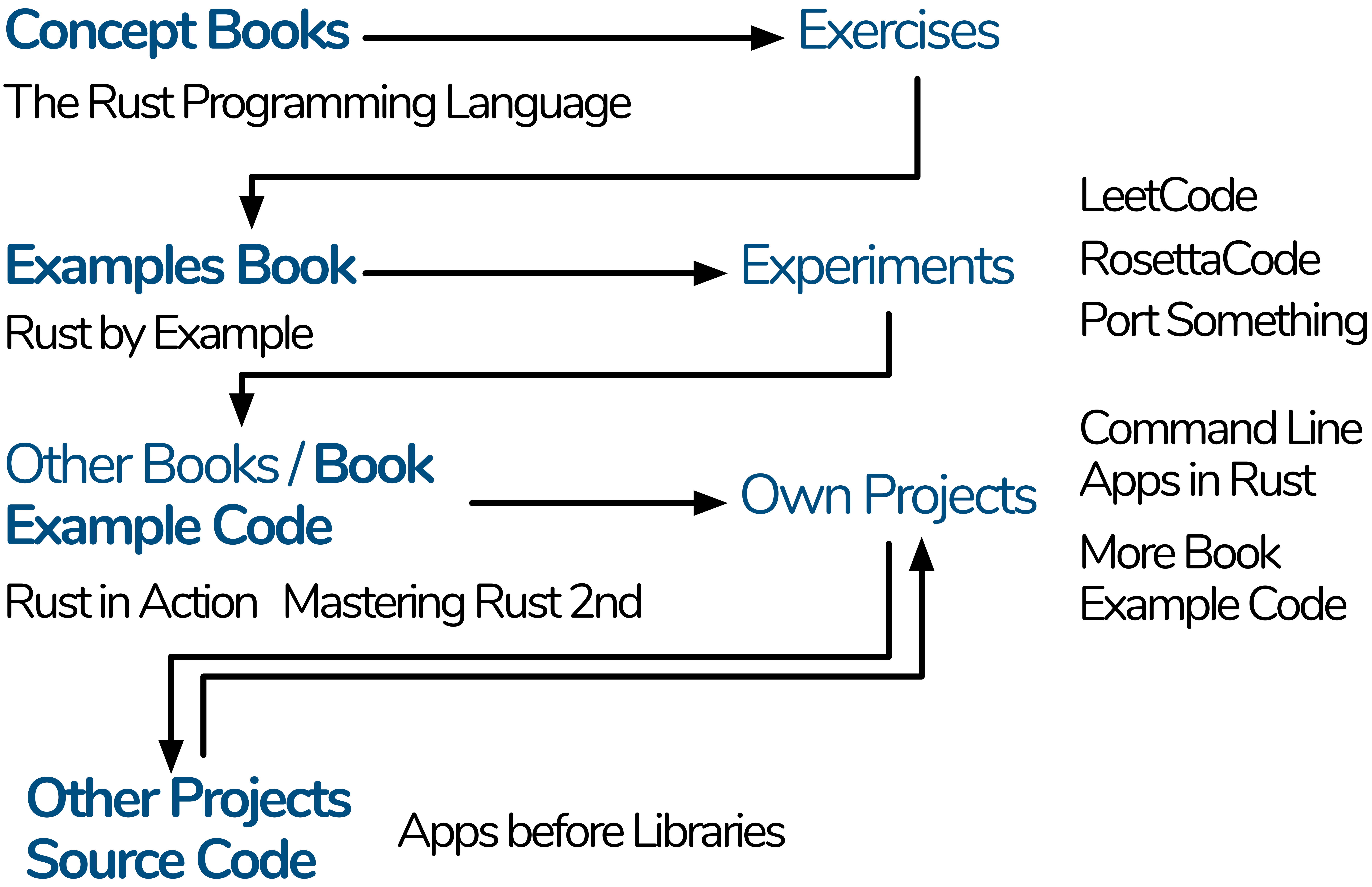
Better Libraries

More "Best Practices"

Larger Project

More Features

Rust Learning Resources



Concept Books

Exercises

The Rust Programming Language

Examples Book

Experiments

Rust by Example

LeetCode

RosettaCode

Port Something

Other Books / **Book Example Code**

Own Projects

Rust in Action   Mastering Rust 2nd

Command Line Apps in Rust

More Book Example Code

Other Projects Source Code

Apps before Libraries

Thank you!

Stefan Baerisch, stefan@stbaer.com, 2020-11-07