Making Existing Code Testable

JIM WEAVER
SOFTWARE DEVELOPER

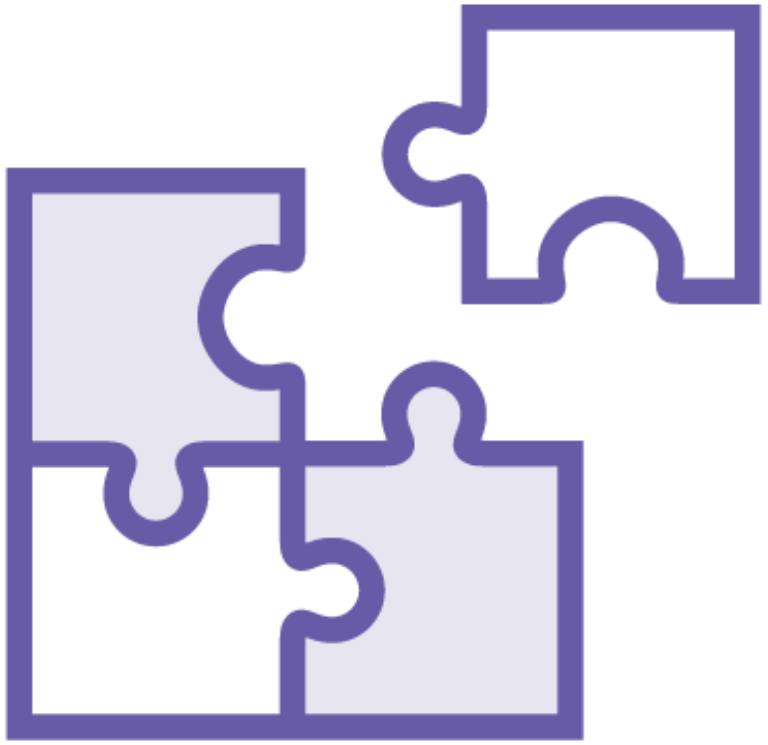
@tenoxtweets

Module Overview

Learn how to tackle two issues that make code difficult to test

- Mixed concerns
- Problematic dependencies

Single Responsibility Principle

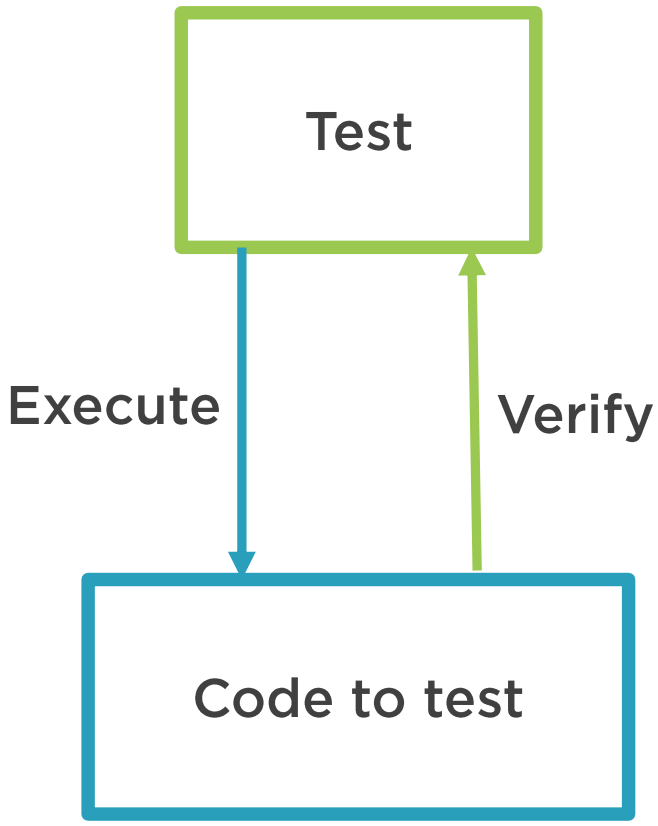


A unit of code does one thing

Related to cohesion, separation of concerns

It has one reason to change

Finding a Seam to Test Target Code

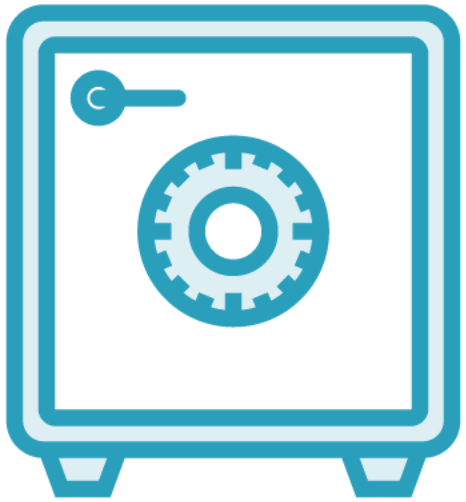


Provide inputs to target code

Validate outputs of target code

Verify target code fulfills its single responsibility

Mixed Concerns Makes Code Hard to Test

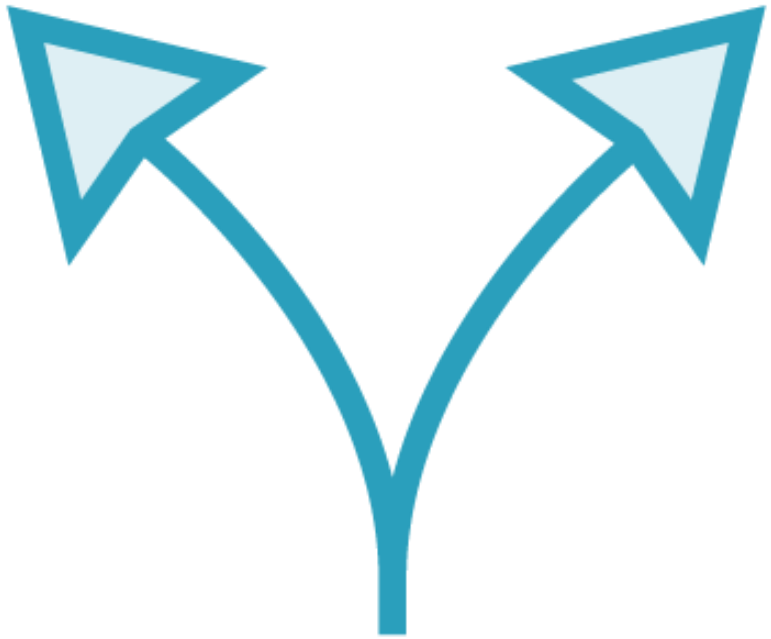


Inaccessible system output

Inaccessible inputs

Undesirable side effects

Extraction – Separate Code to Test



Separate concerns by moving code

Code can move to new methods, classes, and functions

Conservative, IDE supported extraction best for code that is not tested yet

Demo

Extracting code to make it testable

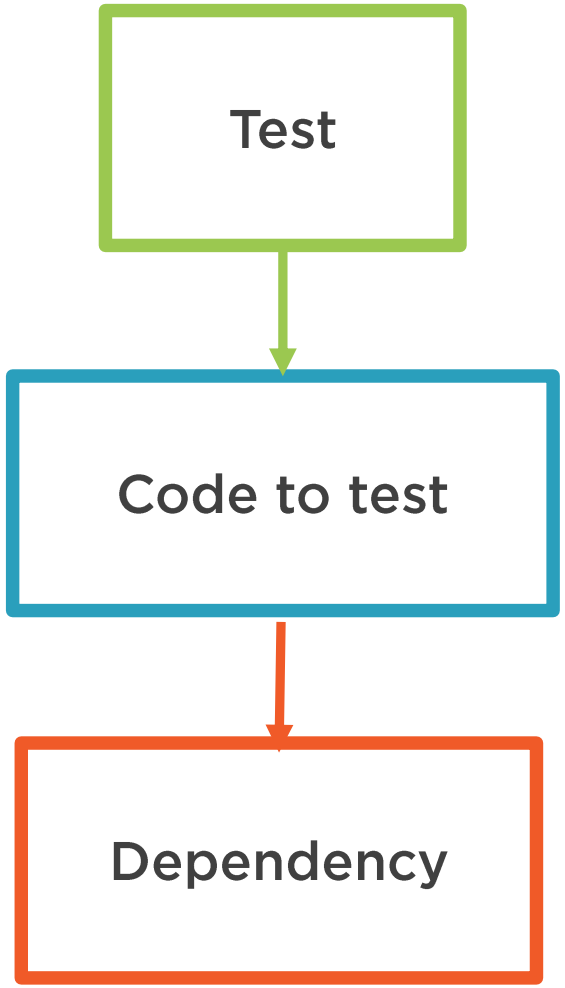
BMI Calculation

$$BMI = \frac{(\text{weight in pounds} * 703)}{\text{height in inches}^2}$$

Test scenarios:

130 pounds, 69 inches = BMI 19.2
150 pounds, 70 inches = BMI 21.52

Dependencies



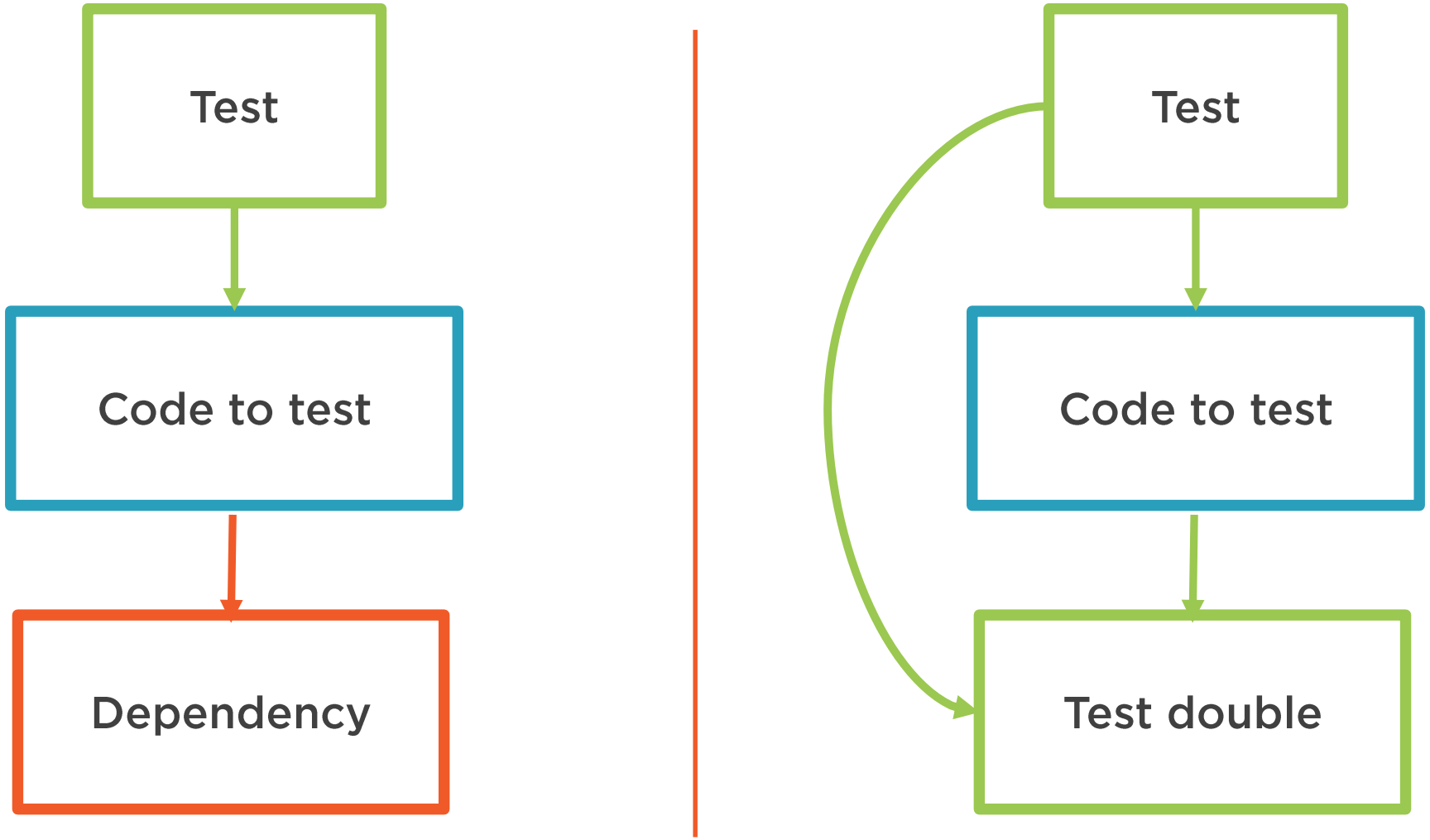
Code rarely works in isolation – code calls other code

Such dependencies may be application code or library code

Testing code that calls other code can be difficult

- Side effects
- Talks to remote services that are not always present
- Inconsistent behavior

Understanding Test Doubles



Demo

Using a test double

Module Summary

- Extracted business logic to test away from UI code
- Created a test double for an email notifier
- Verified business logic for appointment notifications using the test double