



**SWIFTCRAFT**  
21-24 May 2024

# Swift Platform State of the Union for Software Crafters

**Jon Reid**



# Software Crafting



**Continuous  
improvement**



**Language/platform  
agnostic**

# Extreme Programming

Anything worth doing  
is worth doing continuously



Fast  
feedback

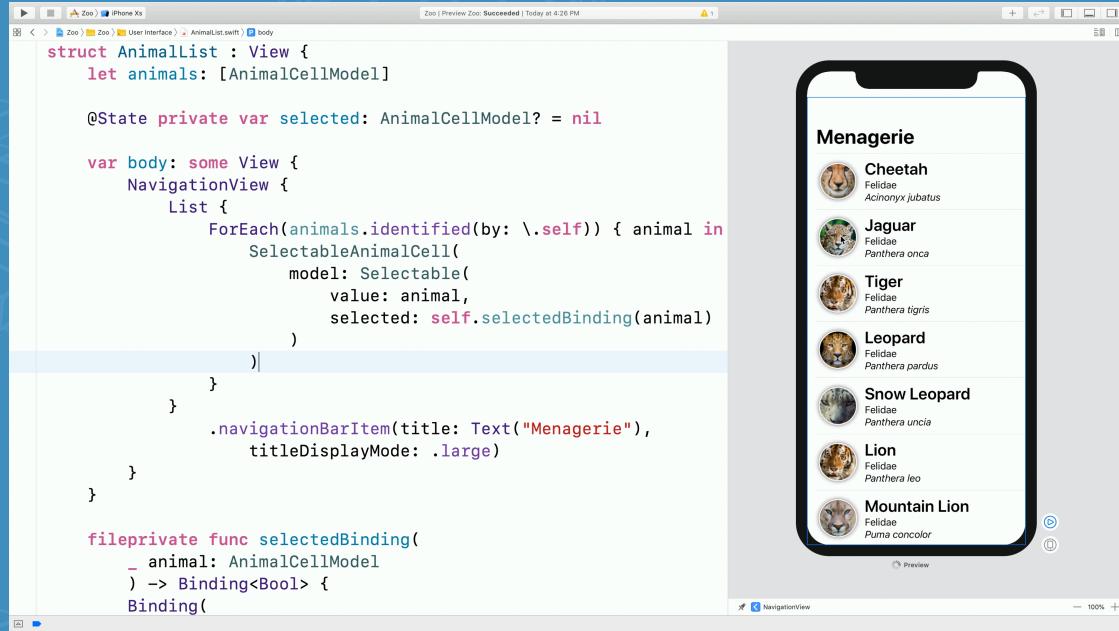


Work  
together



# **Principles ⇒ Practices ⇒ Tools**

# Does Apple Own the Entire Road?



The image shows a screenshot of an Xcode workspace. The main editor window displays a Swift file named `AnimalList.swift`. The code defines a `struct AnimalList : View` containing a list of animals. Each animal is represented by a `SelectableAnimalCell`, which is a `Selectable` view with a `value` of the animal and a `selected` binding. The `body` of the view includes a `NavigationView` with a large title "Menagerie". A preview window on the right shows the resulting mobile application interface, titled "Menagerie", displaying a list of animals with their names, families, and scientific names:

Animal	Family	Scientific Name
Cheetah	Felidae	<i>Acinonyx jubatus</i>
Jaguar	Felidae	<i>Panthera onca</i>
Tiger	Felidae	<i>Panthera tigris</i>
Leopard	Felidae	<i>Panthera pardus</i>
Snow Leopard	Felidae	<i>Panthera uncia</i>
Lion	Felidae	<i>Panthera leo</i>
Mountain Lion	Felidae	<i>Puma concolor</i>

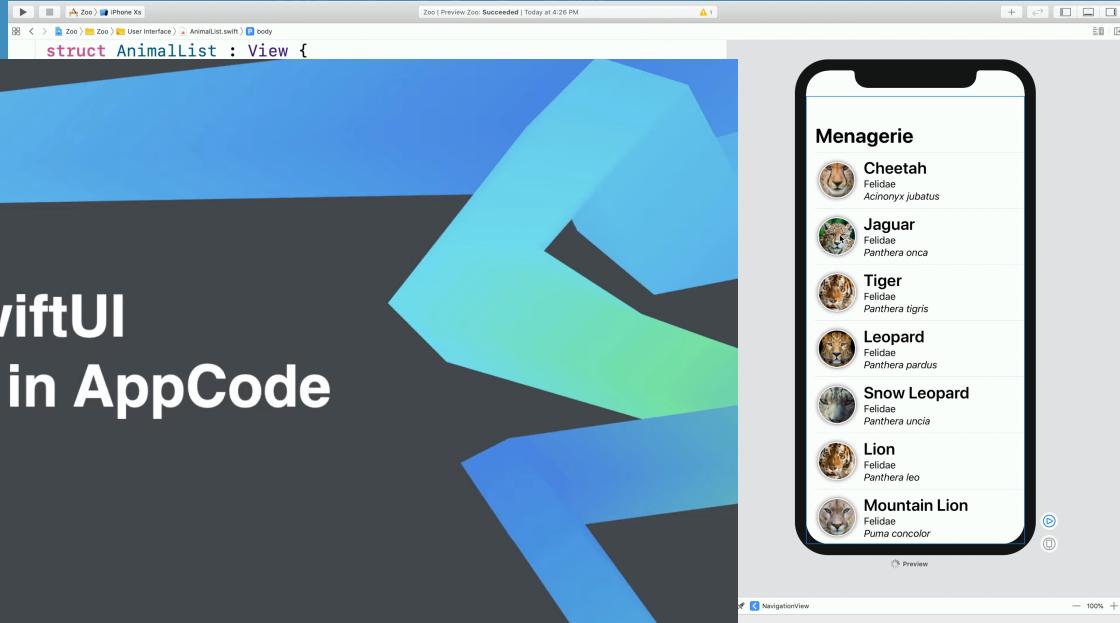
# Does Apple Own the Entire Road?

AC AppCode

## Create a SwiftUI application in AppCode

Stanislav Dombrovsky  
@s\_dombrovsky

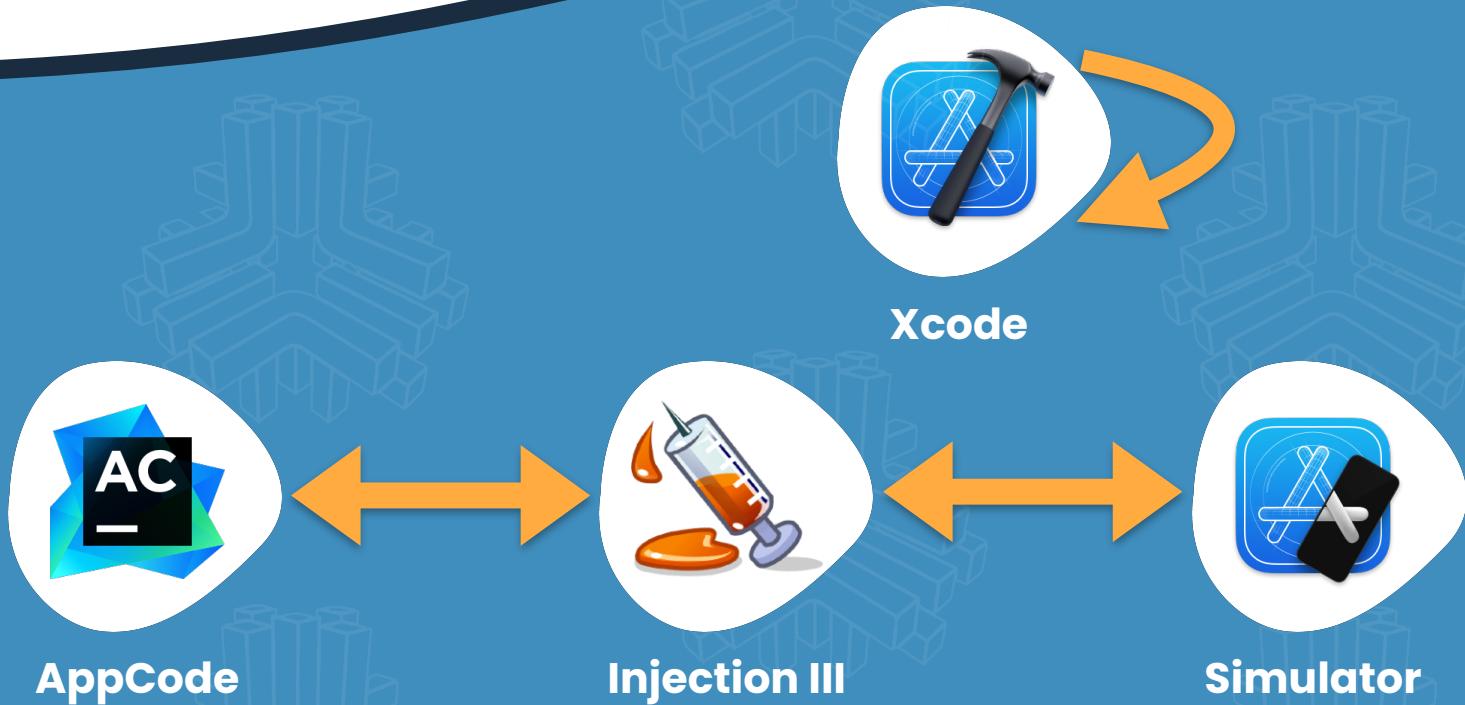
© JetBrains. All rights reserved



# Does Apple Own the Entire Road?

Xcode

# Does Apple Own the Entire Road?



**It was the best of times...**

## A Tale of 3 IDEs

It's good to stretch beyond  
familiar tools.



**Xcode**



**AppCode**



**IntelliJ**

# Crafting Tools for Swift

1

## Dependency Analysis

Tools to visualize dependencies: what depends on what?

2

## Unit Testing

Meanwhile, elsewhere in the xUnit world: what do they have?

3

## Continuous Integration

Do our build systems encourage small, frequent commits?

4

## Test-Driven Development

What tooling supports TDD patterns?

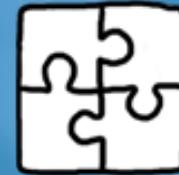
5

## Refactoring

Are we out of options? I don't think so.

## 1: DEPENDENCY ANALYSIS

Tools to visualize  
dependencies:  
What depends on what?

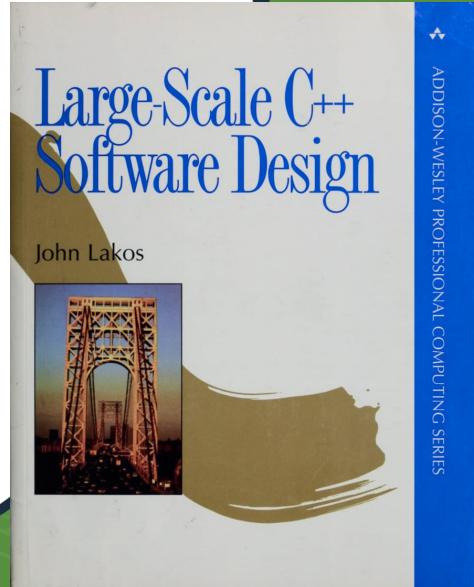


Dependency Analysis

# Physical Design

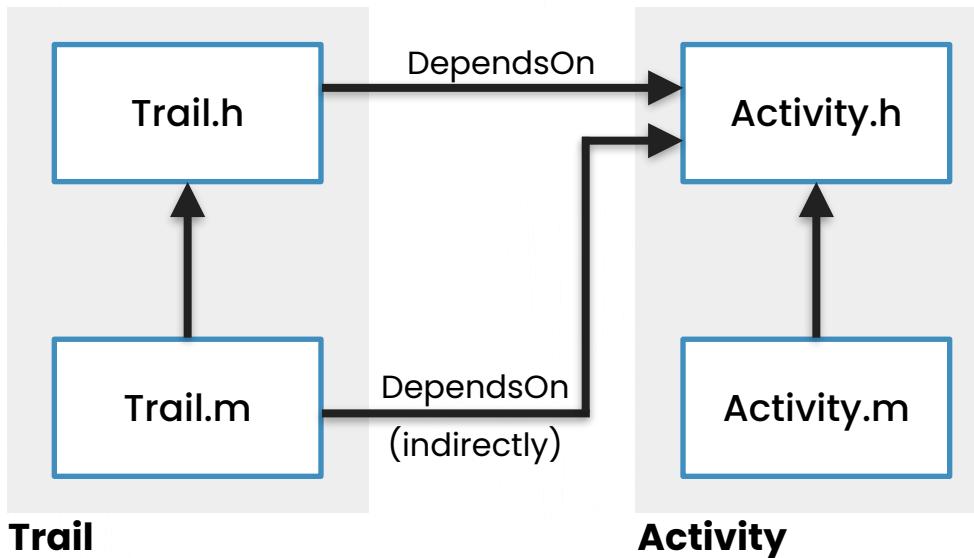
**Logical design** addresses only *architectural* issues.

**Physical design** addresses *organizational* issues.

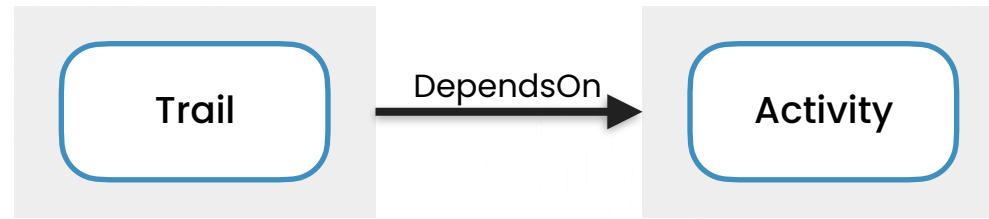


# Compile-Time Dependency

```
#import  
"Activity.h"
```



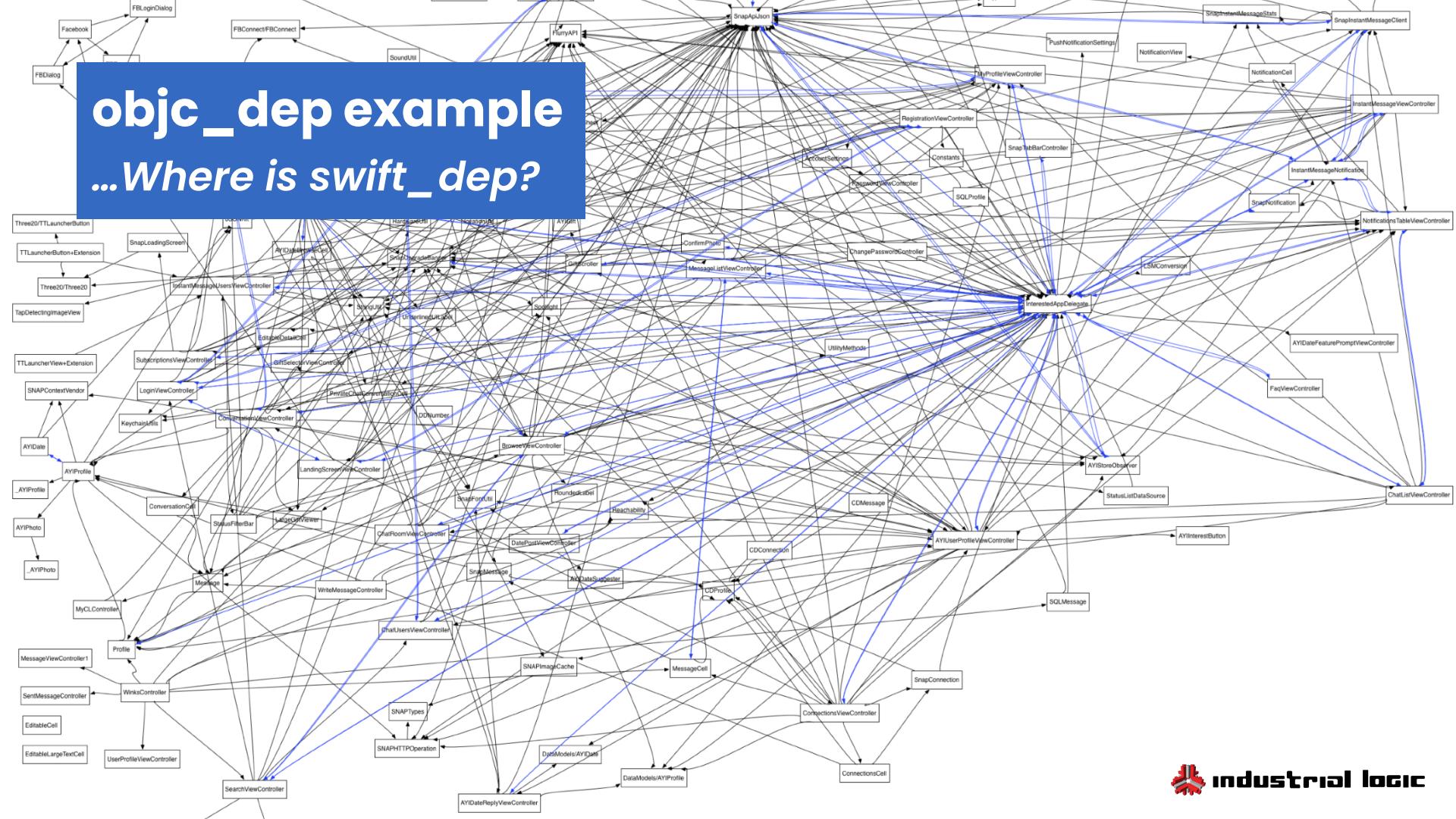
# Behavioral Dependency



**objc\_dep**

# objc\_dep example

*...Where is swift\_dep?*



## 2: UNIT TESTING

Meanwhile, elsewhere  
in the xUnit world:  
What do they have?



Unit Testing

# Basic Parts of a Testing Framework

**Test Description**

**Test Runner**

# Test Description: XCTest

## *iOS Unit Testing by Example* page 6:

- It lives within a subclass of XCTestCase.
- It isn't declared **private**.
- Its name starts with **test**.
- It takes no parameters.
- It has no return value.

```
class AssertYourSelfTests: XCTestCase {  
  
    func testSomethingUsuallyALongName() {  
        XCTAssertEqual(a, b)  
    }  
}
```

# Test Description: JUnit

```
class AuthorNameNormalizerTests {  
  
    @Test  
    fun `initializes middle name`() {  
        val normalizer = AuthorNameNormalizer()  
  
        assertThat(  
            normalizer.normalize("Henry David Thoreau"),  
            equalTo("Thoreau, Henry D.")  
        )  
    }  
}
```

# Test Description: JUnit

```
class AuthorNameNormalizerTests {  
  
    @Test  
    fun `initializes middle name`() {  
        val normalizer = AuthorNameNormalizer()  
  
        assertThat(  
            normalizer.normalize("Henry David Thoreau"),  
            equalTo("Thoreau, Henry D.")  
        )  
    }  
}
```

No subclass requirement

# Test Description: JUnit

```
class AuthorNameNormalizerTests {  
  
    @Test ←  
    fun `initializes middle name`() {  
        val normalizer = AuthorNameNormalizer()  
  
        assertThat(  
            normalizer.normalize("Henry David Thoreau"),  
            equalTo("Thoreau, Henry D.")  
        )  
    }  
}
```

- No subclass requirement
- Test annotation (there are others)

# Test Description: JUnit

```
class AuthorNameNormalizerTests {  
    @Test  
    fun `initializes middle name`() {  
        val normalizer = AuthorNameNormalizer()  
  
        assertThat(  
            normalizer.normalize("Henry David Thoreau"),  
            equalTo("Thoreau, Henry D.")  
    }  
}
```

- No subclass requirement
- Test annotation (there are others)
- Natural language description

# Test Description: JUnit

```
class AuthorNameNormalizerTests {  
  
    @Test  
    fun `initializes middle name`() {  
        val normalizer = AuthorNameNormalizer()  
  
        assertThat(  
            normalizer.normalize("Henry David Thoreau"),  
            equalTo("Thoreau, Henry D."))  
    }  
}
```

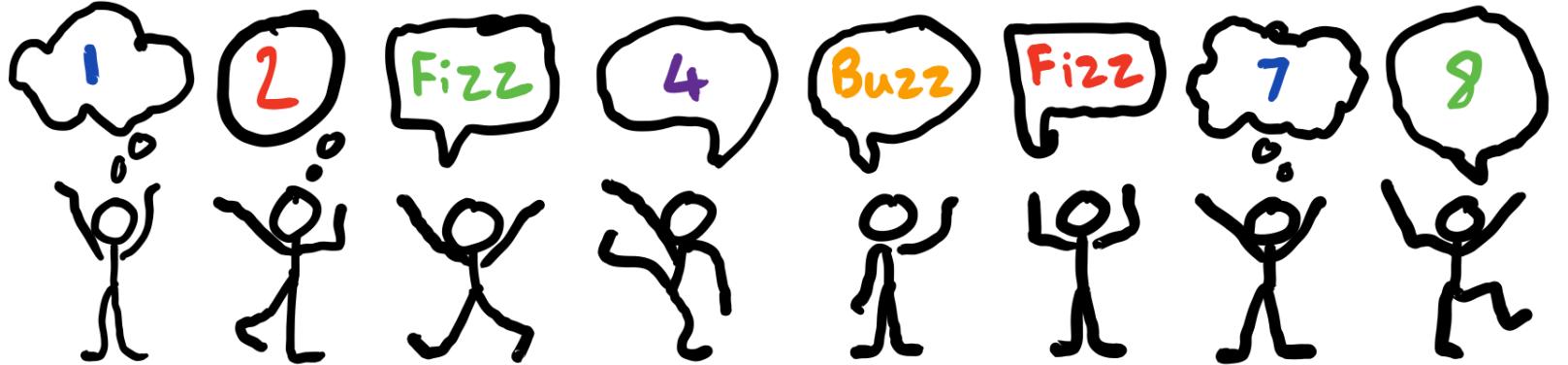
- No subclass requirement
- Test annotation (there are others)
- Natural language description
- Assertion without baggage

# Test Description: swift-testing

```
@Suite("AuthorNameNormalizer Tests")
struct AuthorNameNormalizerTests {

    @Test("Initializes middle name")
    func initializesMiddleName() {
        let normalizer = AuthorNameNormalizer()
        let actual = normalizer.normalize("Henry David Thoreau")
        #expect(actual == "Thoreau, Henry D.")
    }
}
```

- No subclass requirement
- Test annotation (with traits!)
- Natural language description
- Assertion without baggage



# Test Description

## Parameterized Tests **NUnit**

```
[TestCase(3)]  
[TestCase(6)]  
[TestCase(9)]  
public void FizzWhenDivisibleBy3(int input)  
{  
    var fizzBuzz = new FizzBuzz()  
    var actual = fizzBuzz.evaluate(input)  
    Assert.AreEqual("Fizz", actual)  
}
```

Since May 2009

# Test Description

## Parameterized Tests **NUnit**

```
[TestCase(1, "1")]
[TestCase(2, "2")]
[TestCase(3, "Fizz")]
[TestCase(6, "Fizz")]
[TestCase(5, "Buzz")]
[TestCase(10, "Buzz")]
[TestCase(15, "FizzBuzz")]
[TestCase(30, "FizzBuzz")]
public void TestFizzBuzz(int input, string expected)
{
    var fizzBuzz = new FizzBuzz()
    var actual = fizzBuzz.evaluate(input)
    Assert.AreEqual(expected, actual)
}
```

# Test Description

## Parameterized Tests **EGTest**

```
func testFizzBuzz() {  
    check([  
        eg((1), expect: "1"),  
        eg((2), expect: "2"),  
        eg((3), expect: "Fizz"),  
        eg((6), expect: "Fizz"),  
        eg((5), expect: "Buzz"),  
        eg((10), expect: "Buzz"),  
        eg((15), expect: "FizzBuzz"),  
        eg((30), expect: "FizzBuzz"),  
    ]) { example in  
        let fizzBuzz = FizzBuzz()  
        let actual = fizzBuzz.evaluate(example.input.1)  
        EGAssertEqual(actual, example)  
    }  
}
```

# Test Description

Parameterized Tests **XCTestParametrizedMacro**

```
@Parametrize(input: [3, 6, 9])
func testFizz(input: Int) {
    let fizzBuzz = FizzBuzz()
    let actual = fizzBuzz.evaluate(input)
    XCTAssertEqual(actual, "Fizz")
}
```

# Test Description

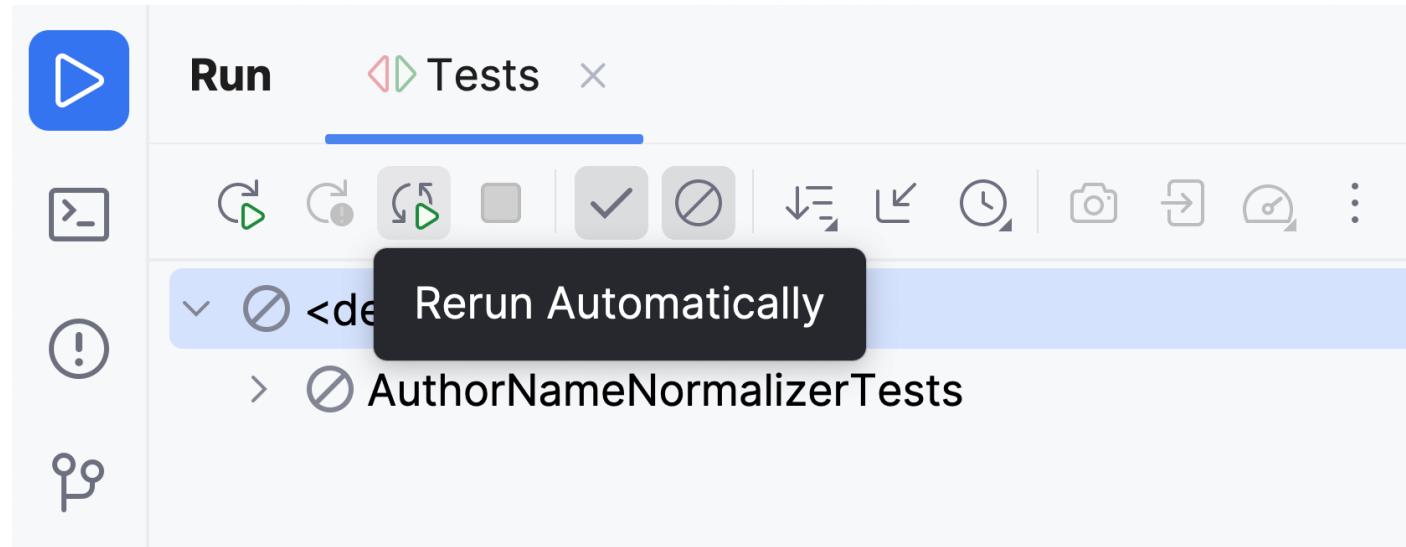
## Parameterized Tests **swift-testing**

```
@Parametrize(input: [3, 6, 9])
func testFizz(input: Int) {
    let fizzBuzz = FizzBuzz()
    let actual = fizzBuzz.evaluate(input)
    XCTAssertEqual(actual, "Fizz")
}
```

```
@Test(arguments: [3, 6, 9])
func fizz(input: Int) {
    let fizzBuzz = FizzBuzz()
    let actual = fizzBuzz.evaluate(input)
    #expect(actual = "Fizz")
}
```

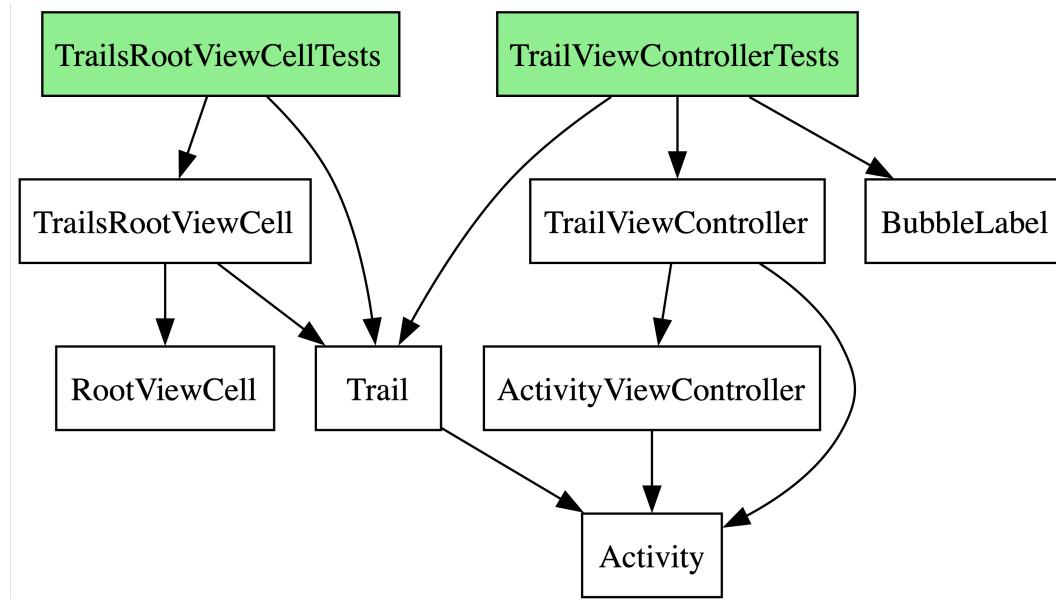
# Test Runner

Continuous Testing **JetBrains IDEs**



# Test Runner

If we had dependency analysis...



# Test Runner

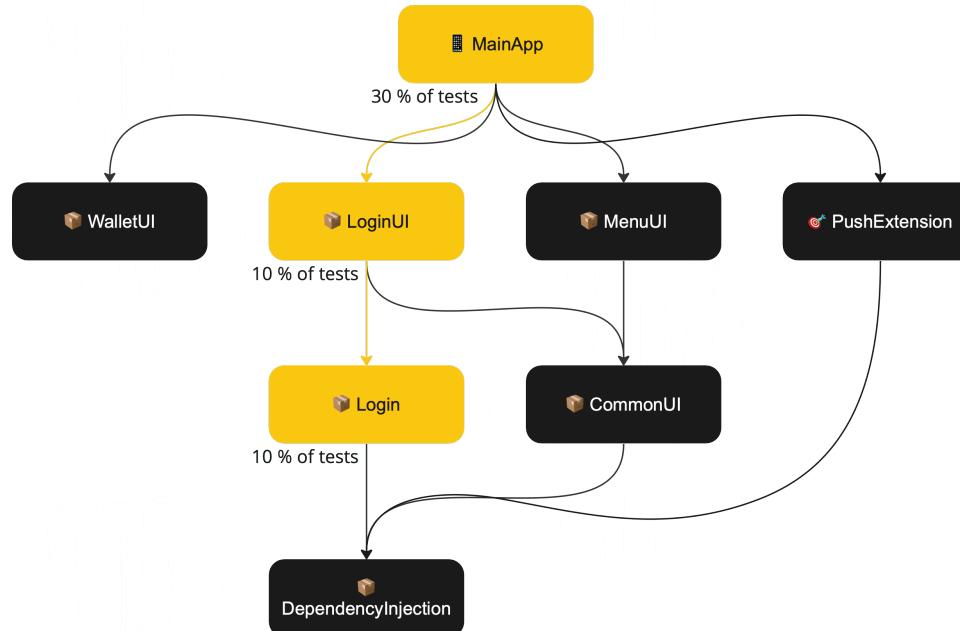
Continuous Minimal Testing **Infinitest**



Every time you change some code  
and IntelliJ recompiles it,  
infinitest computes *which tests*  
*need to be executed* and runs them.

# Test Runner

## Module Minimal Testing XcodeSelectiveTesting



### Wishlist:

- File minimal
- Continuous

# Test Runner

Faster Feedback **Jest**

Run any **failing tests first!**

# Test Runner

Random Execution Order **XCTest**

## ▼ Test Execution

Execution Order

✓ Alphabetical

Random

Test Timeouts

Default Test Execution Time Allowance (s)

600

Working in RSpec since Jan 2012

### **3: CONTINUOUS INTEGRATION**

Do our build systems  
encourage small,  
frequent commits?

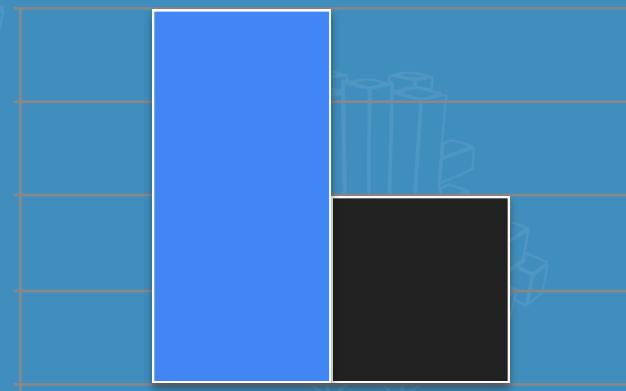


**WAITING FOR  
CI BUILD...**



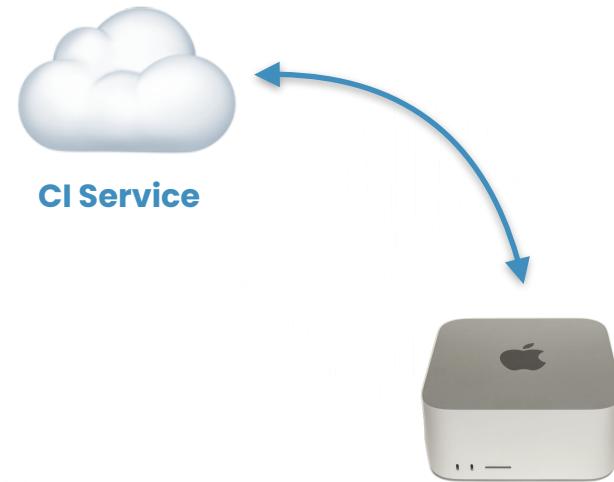
# Build Speed Discrepancy

 Local Build Speed  
 Remote Build Speed



Continuous Integration

# Use Your Own Build Hardware



**Fastest Mac you  
can buy**

**STILL WAITING...**



**CI Is a Practice** *Not a Tool*

**Principles ⇒ Practices ⇒ Tools**

# Continuous Integration

Martin Fowler

We pull from the central store to ensure our local copy is up-to-date



Local environment



Central repo and environment

# Continuous Integration

Martin Fowler

We make our changes to  
the codebase, until we're  
ready and tests are green...



Local environment

...meanwhile, colleagues  
update the central repo  
with their commits



Teammates



# Continuous Integration

Martin Fowler

We pull collaborator's changes  
from the central store and  
combine them with our changes



Local environment



Central repo and  
environment

Teammates



# Continuous Integration

Martin Fowler

We rebuild and check  
tests are green



Local environment



Central repo and  
environment

# Continuous Integration

Martin Fowler

We push our changes  
to the central repo



Local environment



Central repo and  
environment



Teammates

# Continuous Integration

Martin Fowler

A service builds the product  
on a central environment



Local environment



Central repo and  
environment



Teammates

# Continuous Integration

Martin Fowler

The central system  
notifies us that all was well  
with the central build



Local environment

Our colleagues will combine their  
changes when they prepare to  
push their next changes



Teammates



Central repo and  
environment

## 4: TEST-DRIVEN DEVELOPMENT

What tooling supports  
TDD patterns?





# Frame First

**by Jon Reid**

Copyright © 2022 Industrial Logic, Inc. All Rights Reserved.

# Caller Creates

Click on undefined type to create it.

Click on undefined method to create it.



Stay focused on  
caller's point-of-view

Available as Eclipse "Quick Fix" since... 2002?

## 5: REFACTORING

Are we out of options?

*I don't think so.*



ExpectToEventuallyEqual  
main

ExpectToEventuallyEqual > My Mac Test Completed +

ExpectToEventuallyEqual

ExpectToEventuallyEqual > Sources > ExpectToEven > ExpectToEventuallyEqual > expectToEventuallyEqual(actual:expected:timeout:file:line:fail:)

```
1 // ExpectToEventuallyEqual by Jon Reid (https://qualitycoding.org) and Steven Baker (https://stevenrbaker.com)
2 // Copyright 2023 Jonathan M. Reid. https://github.com/jonreid/ExpectToEventuallyEqual/blob/main/LICENSE
3 // SPDX-License-Identifier: MIT
4
5 import XCTest
6
7 public func expectToEventually<T: Equatable>(
8     actual: () throws -> T,
9     expected: T,
10    timeout: TimeInterval = 1.0,
11    file: StaticString = #filePath,
12    line: UInt = #line,
13    fail: (String, StaticString, UInt) -> Void = XCTFail
14) throws {
15    let runLoop = RunLoop.current
16    let timeoutDate = Date(timeIntervalSinceNow: timeout)
17
18    var lastActual = try actual()
19    var tryCount = 0
20    repeat {
21        tryCount += 1
22        if lastActual == expected {
23            return
24        }
25        runLoop.run(until: Date(timeIntervalSinceNow: 0.01))
26        lastActual = try actual()
27    } while Date().compare(timeoutDate) == .orderedAscending
28
29    fail(
30        "\u{2022}(describeMismatch(T.self, expected: expected, actual: lastActual)) after \(tryCount) tries, tim
```

# Recommended Key Bindings for Xcode

⌘⌫N

## Rename

Conflicts with “New Workspace” so delete that.

⌘⌫V

## Extract All Occurrences

Don’t map to “Extract to Variable” which extracts only one occurrence.

⌘⌥M

## Extract to Method

Often doesn’t work, but now you can try it quickly to see.

⌫☒

## Delete Line

Conflicts with “Delete to Beginning of Line” but easier than triple-clicking.

# Automated Refactoring

## Xcode

- Rename
- Extract to Function
- Extract to Method
- Extract to Variable
- Extract All Occurrences

## AppCode (Swift)

- Rename
- Change Signature
- Extract Variable
- Extract Closure
- Extract Function (Method)
- Inline
- Safe Delete

# Automated Refactoring

## AppCode (Swift)

- Rename
- Change Signature
- Extract Variable
- Extract Closure
- Extract Function (Method)
- Inline
- Safe Delete

## AppCode (additional for Objective-C)

- Convert to Method
- Convert to Block
- Convert to Property
- Convert to Instance Variable
- Move Member
- Extract Block
- Extract Category
- Extract Constant
- Extract Define
- Extract Instance Variable
- Extract Parameter
- Extract Property
- Extract Protocol
- Extract Subclass
- Extract Superclass
- Extract Typedef
- Pull Members Up
- Push Members Down

# AppCode Intentions

- ✓ Swift
  - ✓ Add 'catch'
  - ✓ Add digit separators
  - ✓ Add documentation comment
  - ✓ Add explicit qualifier
  - ✓ Add explicit type
  - ✓ Convert to binary
  - ✓ Convert to closure argument
  - ✓ Convert to decimal
  - ✓ Convert to hex
  - ✓ Convert to multiline string literal
  - ✓ Convert to octal
  - ✓ Convert to single-line string literal
  - ✓ Convert to trailing closure
  - ✓ DeMorgan's Law
  - ✓ Decrease string escape level
  - ✓ Extract Conformance to Extension
  - ✓ Extract members to extension
  - ✓ Extract members to its original type declaration
  - ✓ Flip binary expression

- ✓ Flip comparison
- ✓ Increase string escape level
- ✓ Invert 'if' statement
- ✓ Join declaration and assignment
- ✓ Localize string
- ✓ Merge 'else if'
- ✓ Merge conditions
- ✓ Merge nested 'if's
- ✓ Negate comparison
- ✓ Remove digit separators
- ✓ Remove explicit type
- ✓ Replace 'guard' with 'if'
- ✓ Replace 'if' with 'guard'
- ✓ Simplify if-else
- ✓ Split 'else if'
- ✓ Split into declaration and assignment
- ✓ Split into nested 'if's
- ✓ Split into separate conditions
- ✓ Split into separate declarations

# DeMorgan's Law

$$!(a \parallel b)$$

$$!a \And !b$$

The screenshot shows the Xcode IDE interface. The top menu bar includes Apple, Code, File, Edit, Selection, View, Go, Run, Terminal, Window, and Help. On the right side, there are icons for search, file, and system status. The main window has a sidebar on the left labeled "TESTING" with a "Filter (e.g. text, !exclude, @tag)" field and a count of "11/11". Below the filter are several test cases under "ExpectToEventuallyEqualTests" and "DescribeMismatchTests". The main editor area displays the source code for "ExpectToEventuallyEqual.swift". The code implements a function to check if an actual value eventually matches an expected value within a specified timeout.

```
// ExpectToEventuallyEqual by Jon Reid (https://qualitycoding.org) and Steven Baker (https://stevenrbaker.com)
// Copyright 2023 Jonathan M. Reid. https://github.com/jonreid/ExpectToEventuallyEqual/blob/main/LICENSE.txt
// SPDX-License-Identifier: MIT

import XCTest

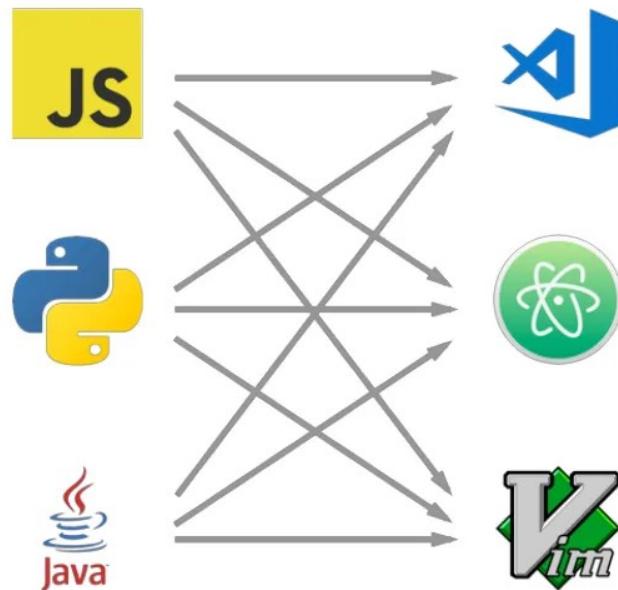
public func expectToEventuallyEqual<T: Equatable>(
    actual: () throws -> T,
    expected: T,
    timeout: TimeInterval = 1.0,
    file: StaticString = #filePath,
    line: UInt = #line,
    fail: (String, StaticString, UInt) -> Void = XCTFail
) throws {
    let runLoop: RunLoop = RunLoop.current
    let timeoutDate: Date = Date(timeIntervalSinceNow: timeout)

    var lastActual: T = try actual()
    var tryCount: Int = 0
    repeat {
        tryCount += 1
        if lastActual == expected {
            return
        }
        runLoop.run(until: Date(timeIntervalSinceNow: 0.01))
        lastActual = try actual()
    } while Date().compare(timeoutDate) == .orderedAscending

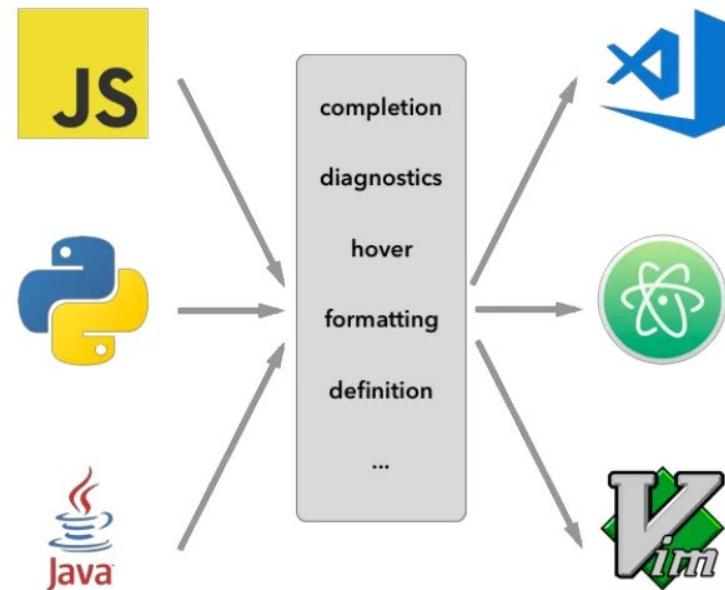
    fail(
        "\u{descriptionMismatch}(T.self, expected: expected, actual: lastActual) after \(tryCount) tries, timing out"
    )
}
```

# Language Server Protocol

NO LSP



LSP





EXTENSIONS ⌂ ...

Search Extensions in Marketplace ⌂ ⚙

INSTALLED 9

- Installed | pair programmer  
GitHub
- GitHub Copilot Chat ⌂ 33ms  
AI chat features powered by Copilot  
GitHub
- Pylance  
A performant, feature-rich language ser...  
Microsoft
- Python  
Python language support with extensio...  
Microsoft
- Python Debugger  
Python Debugger extension using debu...  
Microsoft
- Swift ⌂ 16ms  
Swift Language Support for Visual Studi...  
Swift Server Work Group
- Test Adapter Converter ⌂ 2ms  
Converter extension from the Test Adap...  
Microsoft
- Test Explorer UI ⌂ 13ms  
Run your tests in the Sidebar of Visual ...  
Holger Benl

> RECOMMENDED 6

main ⌂ 0 1 1 ⌂ 0 0 Coverage: Off

← → ⌂ ExpectToEventuallyEqual

ExpectToEventuallyEqual.swift

Extension: Swift X



## Swift v1.9.0

Swift Server Work Group ⌂ 256,447 ⌂ ★★★★★(24)

Swift Language Support for Visual Studio Code.

Disable ⌂ Uninstall ⌂ ⚙

This extension is enabled globally.

DETAILS FEATURES CHANGELOG DEPENDENCIES

## Swift for Visual Studio Code

This extension adds language support for Swift to Visual Studio Code. It supports:

- Code completion
- Jump to definition, peek definition, find all references, symbol search
- Error annotations and apply suggestions from errors
- Automatic generation of launch configurations for debugging with [CodeLLDB](#)
- Automatic task creation
- Package dependency view
- Test Explorer view

Swift support uses [SourceKit LSP](#) for the [language server](#) to power code completion and [LLDB](#) to enable debugging.

The extension is developed by members of the Swift Community and maintained by the [SSWG](#). The aim is to provide a first-class, feature complete extension to make developing Swift applications on all platforms a seamless experience.

### Categories

Programming Languages

Debuggers

### Resources

Marketplace Issues Repository License Swift Server Work Group

### More Info

Published 2021-12-23,

17:36:51

Last released 2024-04-17,

02:27:29



# A Ray of Hope!



Six s Over Texas  
@cfi@mastodon.social

This is a really exciting addition to Swift's LSP support:  
refactoring actions based on SwiftSyntax!  
[github.com/apple/sourcekit-lsp...](https://github.com/apple/sourcekit-lsp...)

I've wanted this for a very long time and we've finally got all the foundations laid right. If you have a favorite refactoring action you want LSP to support, you can build it entirely in Swift and turn it around as fast as you can put up a PR!



GitHub

**Add local refactoring code actions based on ...**

By DougGregor

Apr 05, 2024 at 15:12 · Edited Apr 05 at 15:13 ▾ · 0 · 14 · ★ 2

# Add local refactoring code actions based on swift-syntax #1169

 Closed

DougGregor wants to merge 6 commits into [apple:main](#) from [DougGregor:syntax-refactoring-actions](#) 

 Conversation 100

 Commits 6

 Checks 0

 Files changed 11



DougGregor commented on Apr 5 · edited

Member 

Introduce new local refactoring code actions based on the Swift syntax tree, without going through SourceKit.

This change includes a number of new refactoring code actions. Most of them adapt the syntax refactorings from the [SwiftRefactor module of swift-syntax](#):

- Add digit separators to an integer literal, e.g., `100000 -> 1_000_000`.
- Remove digit separators from an integer literal, e.g., `1_000_000 -> 1000000`. (from SwiftRefactor)
- Format a raw string literal, e.g., `"Hello \#(world)" -> ##"Hello \#(world)"##`
- Migrate to new if let syntax, e.g., `if let x = x { ... } -> if let x { ... }`
- Replace opaque parameters with generic parameters, e.g., `func f(p: some P) --> func f<T1: P>(p: T1)`.

This is generally easy to do, requiring one conformance to provide a name for the refactoring:

```
extension AddSeparatorsToIntegerLiteral: SyntaxRefactoringCodeActionProvider {
    public static var title: String { "Add digit separators" }
}
```



There are also a few new refactoring code actions implemented here. These could be sunk down into SwiftRefactor to be made more widely available, or stay here if they're too IDE-centric to live in swift-refactor:

- Convert an integer literal from one base to another (e.g., `172387 -> 0x2a163`).
- Convert JSON into a Codable struct.
- Apply Demorgan's law, e.g., `!(a || b) --> !a && !b`.

There is a small amount of overlap between the refactoring code actions added here and [those SourceKit provides](#). For example,

The screenshot shows the Xcode IDE interface. The title bar displays "TestPackage". The left sidebar contains various icons for navigation and project management. The main area shows the file structure of a Swift package:

- EXPLORER**: Shows the project structure:
  - TESTPACKAGE**:
    - .build
    - .vscode
    - Sources / TestPackage**:
      - TestPackage.swift** (selected)
      - Tests
      - .gitignore
      - default.proraw
      - Package.swift
- OUTLINE**
- TIMELINE**
- PACKAGE DEPENDENCIES (TESTPACKAGE)**

The right pane displays the content of **TestPackage.swift**:

```
1 // The Swift Programming Language
2 // https://docs.swift.org/swift-book
3
4
```

The status bar at the bottom shows: **x 0**, **⚠ 0**, **⌚ 0**, **Coverage: Off**, **Ln 4, Col 1**, **Spaces: 4**, **UTF-8 LF {} Swift**.

# How to Refactor Without Tools?

**Principles  $\Rightarrow$  Practices  $\Rightarrow$  Tools**

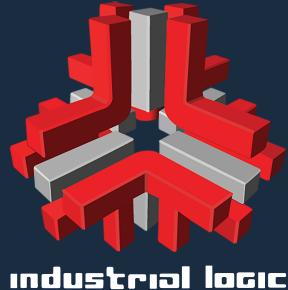
# How to Refactor Without Tools?



# About *Jon Reid*

Senior Consultant, Industrial Logic

**Better Software Sooner**

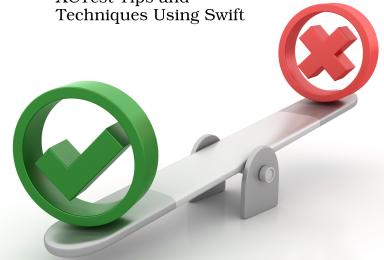


Copyright © 2024, Industrial Logic, All Rights Reserved.

The  
Pragmatic  
Programmers

## iOS Unit Testing by Example

XCTest Tips and  
Techniques Using Swift



Jon Reid  
edited by Michael Swaine



# Quality Coding

# IDEs: End of Story?



AppCode



Xcode

# Tools Outside Apple



SwiftLint



SwiftFormat



Tuist



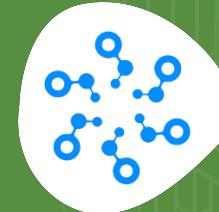
Muter



Sourcery



Injection III



Periphery



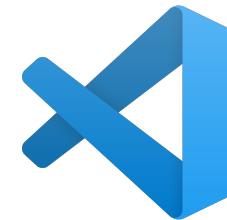
ViewInspector



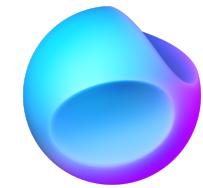
ApprovalTests

# Keep an Eye on These New IDEs

Check the Fleet Features Matrix for Swift



Visual  
Studio Code



JetBrains  
Fleet

# What Can You Do?

1

## Craft the Tools You Need

You may start solo, but it will thrive as a community effort.

2

## Try Other Environments

Learn a new language/IDE combination (especially by JetBrains).

3

## Connect with Crafters

Join XP or Software Crafting meet-ups. Or make your own!

# Connect with Crafters

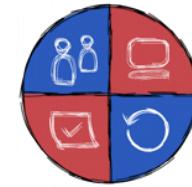


- London Software Craftsmanship
- PhillyXP
- Seattle Software Crafters

*...Far too many to list!*



Form your own  
book club



SoCraTes  
BE, DE, FR, IT, UK

**Principles ⇒ Practices ⇒ Tools**

