Follow the Rules: What's New in SwiftLint

Learn about new and lesser-known SwiftLint rules you might have missed.

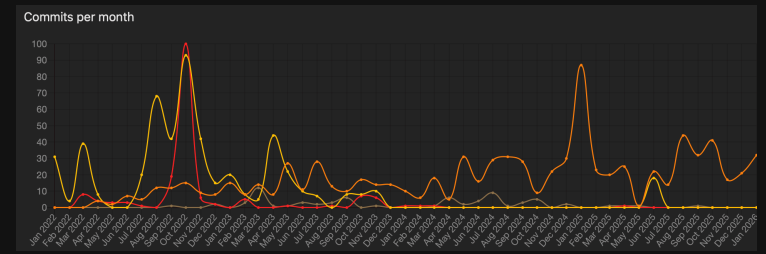
Danny Mösch – SwiftLint Maintainer

☀️ During the Day

- Java
- TypeScript
- C/C++
- Python
- Eclipse IDE
- Maven
- Groovy
- Agentic AI
- Domain-Specific Languages
- Code Generation

🌙 At Night

- Swift
- SwiftLint
- Open Source



SwiftLint

A tool to enforce Swift style and conventions

⭐ 19.400+ stars on GitHub

📕 250 rules built-in

- 96% use SwiftSyntax for parsing, analysis and automatic correction
- 60% are disabled by default
- 36% support automatic correction

🔨 Builds with Swift Package Manager, Xcode and Bazel

👜 Offers Build Tool and Command Plugins

💻 Runs on macOS, Linux (and Windows)

Accessibility Label for Image, Accessibility Trait for Button, Anonymous Argument in Multiline Closure, Array Init, Async Without Await, Attribute Name Spacing, Attributes, Balanced XCTest Life Cycle, Blanket Disable Command, Block Based KVO, Capture Variable, Cases on Newline, Class Declaration in Final Class, Class Delegate Protocol, Closing Brace Spacing, Closure Body Length, Closure End Indentation, Closure Parameter Position, Closure Spacing, Collection Element Alignment, Colon Spacing, Comma Inheritance Rule, Comma Spacing, Comment Spacing, Compiler Protocol Init, Computed Accessors Order, Conditional Returns on Newline, Contains over Filter Count, Contains over Filter is Empty, Contains over First not Nil, Contains over Range Comparison to Nil, Contrasted Opening Brace, Control Statement, Convenience Type, Cyclomatic Complexity, Deployment Target, Direct Return, Discarded Notification Center Observer, Discouraged Assert, Discouraged Direct Initialization, Discouraged None Name, Discouraged Object Literal, Discouraged Optional Boolean, Discouraged Optional Collection, Duplicate Conditions, Duplicate Enum Cases, Duplicate Imports, Duplicated Key in Dictionary Literal, Dynamic Inline, Empty Collection Literal, Empty Count, Empty Enum Arguments, Empty Parameters, Empty Parentheses with Trailing Closure, Empty String, Empty XCTest Method, Enum Case Associated Values Count, Expiring Todo, Explicit ACL, Explicit Enum Raw Value, Explicit Init, Explicit Self, Explicit Top Level ACL, Explicit Type Interface, Extension Access Modifier, Fallthrough, Fatal Error Message, File Header, File Length, File Name no Space, File Name, File Types Order, Final Test Case, First Where, Flat Map over Map Reduce, Force Cast, Force Try, Force Unwrapping, Function Body Length, Function Default Parameter at End, Function Name Whitespace, Function Parameter Count, Generic Type Name, IBInspectable in Extension, Identical Operands, Identifier Name, Implicit Getter, Implicit Optional Initialization, Implicit Return, Implicitly Unwrapped Optional, Inclusive Language, Incompatible Concurrency Annotation, Indentation Width, Invalid SwiftLint Command, Is Disjoint, Joined Default Parameter, Large Tuple, Last Where, Leading Whitespace, Legacy CGGeometry Functions, Legacy Constant, Legacy Constructor, Legacy Hashing, Legacy Multiple, Legacy NSGeometry Functions, Legacy Objective-C Reference Type, Legacy Random, Line Length, Literal Expression End Indentation, Local Doc Comment, Lower ACL than Parent, Mark, Min or Max over Sorted First or Last, Missing Docs, Modifier Order, Multiline Arguments Brackets, Multiline Arguments, Multiline Call Arguments, Multiline Function Chains, Multiline Literal Brackets, Multiline Parameters Brackets, Multiline Parameters, Multiple Closures with Trailing Closure, Nesting, Nimble Operator, No Empty Block, No Extension Access Modifier, No Fallthrough only, No Grouping Extension, No Magic Numbers, No Space in Method Call, Non-optional String -> Data Conversion, Notification Center Detachment, NSLocalizedString Key, NSLocalizedString Require Bundle, NSNumber Init as Function Reference, NSObject Prefer isEqual, Number Separator, Object Literal, One Declaration per File, Opening Brace Spacing, Operator Usage Whitespace, Optional Data -> String Conversion, Optional Enum Case Match, Orphaned Doc Comment, Overridden Method Calls Super, Override in Extension, Pattern Matching Keywords, Period Spacing, Prefer Asset Symbols, Prefer Condition List, Prefer For-Where, Prefer Key Path, Prefer Nimble, Prefer Self in Static References, Prefer Self Type Over Type of Self, Prefer Type Checking, Prefer Zero Over Explicit Init, Prefixed Top-Level Constant, Private Actions, Private Combine Subject, Private Outlets, Private over Fileprivate, Private SwiftUI State Properties, Private Unit Test, Prohibited Calls to Super, Prohibited Interface Builder, Protocol Property Accessors Order, Quick Discouraged Call, Quick Discouraged Focused Test, Quick Discouraged Pending Test, Raw Value for Camel Cased Codable Enum, Reduce Boolean, Reduce into, Redundant @objc Attribute, Redundant Access Control for Setter, Redundant Discardable Let, Redundant Nil Coalescing, Redundant Self, Redundant Sendable, Redundant String Enum Value, Redundant Type Annotation, Redundant Void Return, Required Deinit, Required Enum Case, Return Value from Void Function, Returning Whitespace, Self Binding, Self in Property Initialization, Shorthand Argument, Shorthand Operator, Shorthand Optional Binding, Single Test Class, Sorted Enum Cases, Sorted Imports, Statement Position, Static Operator, Static Over Final Class, Strict Fileprivate, Strong IBOutlet, Superfluous Else, Switch and Case Statement Alignment, Syntactic Sugar, Test Case Accessibility, Todo, Toggle Bool, Trailing Closure, Trailing Comma, Trailing Newline, Trailing Semicolon, Trailing Whitespace, Type Body Length, Type Contents Order, Type Name, Type-safe Array Init, Unavailable Condition, Unavailable Function, Unhandled Throwing Task, Unneeded (Re)Throws Keyword, Unneeded Break in Switch, Unneeded Escaping, Unneeded Overridden Functions, Unneeded Parentheses in Closure Argument, Unneeded Synthesized Initializer, Unowned Variable Capture, Untyped Error in Catch, Unused Closure Parameter, Unused Control Flow Label, Unused Declaration, Unused Enumerated, Unused Import, Unused Optional Binding, Unused Parameter, Unused Setter Value, Valid IBInspectable, Variable Declaration Whitespace, Vertical Parameter Alignment on Call, Vertical Parameter Alignment, Vertical Whitespace after Opening Braces, Vertical Whitespace before Closing Braces, Vertical Whitespace Between Cases, Vertical Whitespace, Void Function in Ternary, Void Return, Weak Delegate, XCTest Specific Matcher, XCTFail Message, Yoda Condition

Async Without Await

A declaration should not be `async` if it does not use `await`.

```
func foo() async {
  bar()
}
```

```
let foo: () async → Void = {
  bar()
}
```

No Violations

```
@concurrent
func foo() async {
  bar()
}
```

```
var foo: (() async → Void)? = nil
```

```
class Child: Parent {
  override func foo() async {}
}
```

```
actor A {
  init() async {
    foo()
    await bar()
  }

  func foo() {}
}

func bar() async {}
```

Blanket Disable Command

Use scoped disable commands instead of blanket disables.

```
// swiftlint:disable unused_parameter
func add(a: Int, b: Int, c: Int) → Int {
  a + b
}

func sub(a: Int, b: Int, c: Int) → Int {
  a - b
}
```

```
// swiftlint:disable force_unwrapping
let v = 1
let w = a!.b
let x = c!
let y = 1
let z = e!.f()
```

No Violations

```
// swiftlint:disable:next blanket_disable_command
// swiftlint:disable implicit_return
func foo() → Int {
  return 42
}
```

```
// swiftlint:disable file_length

// ...
```

Incompatible Concurrency Annotation

Add `@preconcurrency` to public declarations that use `@Sendable`, global actors, or `Sendable` generic parameters so they stay compatible with the Swift 5 language mode.

```
@MyActor
public struct S {}
```

```
public func run(_ block: @Sendable () → Void) {}
```

```
open class S<T> where T: Sendable {}
```

No Violations

```
package @MyActor struct S {}
```

Superfluous Else

Remove `else` after early exists from the current scope.

```
func limit(_ value: Int) → Int {
  if value < 0 {
    return 0
  } else {
    // Positive value
    return value
  }
}
```

```
func result(_ value: Int) throws → String {
  if value == 0 {
    throw MyError.zero
  } else {
    return "ok"
  }
}
```

No Violations

```
func check(_ value: Int) → Int {
  if value < 0 {
    foo()
  } else {
    return value
  }
  return 0
}
```

```
func check(_ i: Int, _ a: Int) → Int {
  if i > 0 {
    if a > 1 {
      return 1
    }
  } else {
    return 3
  }
  return 2
}
```

Unused Parameter

Be intentional with unused parameters.

```
func greet(name: String) {
  print("Hello!")
}
```

```
func log(message: String, verbose: Bool) {
  print(message, verbose: self.verbose)
}
```

No Violations

```
func sum(_ numbers: [Int]) -> Int {
  numbers.reduce(0, +)
}
```

One more thing ...

Changing the Rules

Enable all rules. Disable a few.

```
opt_in_rules:
  - all
disabled_rules:
  - anonymous_argument_in_multiline_closure
  - conditional_returns_on_newline
  - contrasted_opening_brace
  - convenience_type
  - discouraged_optional_collection
  - explicit_acl
  - explicit_enum_raw_value
  - explicit_top_level_acl
  - ...
```

One keyword to rule them `all` ... 👑

Thank You!

@SimplyDanny