# Правила SwiftLint

Используемая версия SwiftLint: 0.34.0

При изменении правил или обновлении SwiftLint не забывайте актуализировать этот документ.

Подробное описание встроенных в SwiftLint правил: https://github.com/realm/SwiftLint/blob/master/Rules.md

Также посмотреть актуальные правила и их конфигурацию можно, введя `$ swiftlint rules` в терминале в директории с файлом `.swiftlint`. Предварительно лучше сделать окно терминала шире.

## Встроенные opt-in правила:

| Приоритет | Правило | Описание | Почему? | Кол-во ошибок |
|---|---|---|---|---|
| | fatal_error_message | Вызов fatalError должен всегда содержать сообщение об ошибке. | | |
| | force_unwrapping | Избегайте принудительного разворачивание Optional-значений. | | |
| | identical_operands | Сравнение двух одинаковых выражений приведет к предупреждению. | | |
| | implicitly_unwrapped_optional | Избегайте принудительно разворачиваемых Optional-объявлений. | | |
| | modifier_order | Порядок модификаторов в объявлениях должен быть строго определен (`public override final class var foo: String`). | | |
| | explicit_self | К переменным и методам экземпляра нужно всегда обращаться, явно указывая `self`. | Для удобства во время ревью, так как стандартные средства Bitbucket и GitLab не подсвечивают члены класса. | 3611 |
| | unavailable_function | Методы без реализации должны быть объявлены как `@available(*, unavailable)`. | | |
| | redundant_type_annotation | Не указывайте тип там, где он может быть автоматически выведен. | | |
| | multiline_function_chains | Вызовы функций в цепочке вызовов должны быть либо все на одной строке, либо каждый на отдельной. | | |
| | no_extension_access_modifier | Указывайте уровень доступа для каждого члена `extension` отдельно, вместо общего указания на весь `extension`. **Пока не актуально.** | Для удобства во время ревью, чтобы сразу было видно уровень доступа.<br><br>Также удобно во время разработки, если весь экстеншн не помещается на экране, и не видно уровня доступа. | 217 |
| | unused_declaration | Предупреждение, если приватный метод или свойство не используются. | | |
| | unused_import | Предупреждение, если найден неиспользуемый `import`. | | |
| | vertical_whitespace_closing_braces | Не оставляйте пустые строки перед закрывающими скобками. | | |
| | multiline_parameters | Параметры при определении функции должны быть либо все на одной строке, либо каждый на отдельной. | | |
| | multiline_arguments | Аргументы при вызове функций должны быть либо все на одной строке, либо каждый на отдельной. | | |
| | trailing_closure | Где возможно, используйте синтаксис trailing closure (`collection.map { ... }` вместо `collection.map({ ... })` и т.п.). *Включено только для методов, принимающих 1 аргумент без лейбла.* | | |
| | closure_spacing | Выражение внутри замыкания должно быть обрамлено пробелами (один после открывающей скобки, один перед закрывающей). | | |
| | literal_expression_end_indentation | Закрывающая скобка массива или словаря должна иметь такие же отступы, как открывающая. | | 13 |

| | | | | |
|---|---|---|---|---|
| | anyobject_protocol | Используйте `AnyObject` вместо `class` в class-only протоколах. | Использование `class` будет deprecated в будущем. | |
| | closure_end_indentation | Закрывающая фигурная скобка замыкания должна иметь такие же отступы, как открывающая. | | |
| | convenience_type | Типы, содержащие только статические методы, должны быть объявлены как `enum` без `case`. | Для избежания инициализации таких типов. | |
| | explicit_init | Избегайте явного вызова `.init`. | | |
| | let_var_whitespace | `let` и `var` должны быть отделены от остальных выражений пустыми строками. | | |
| | multiline_literal_brackets | Открывающая и закрывающая скобки в многострочных литералах (массивы, словари) должны быть на новых строках. | | |
| | prohibited_super_call | Предупреждение, если вызвали `super` там, где это НЕ нужно. | | |
| | overridden_super_call | Предупреждение, если забыли вызвать `super` там, где это нужно. | | |
| | unneeded_parentheses_in_closure_argument | Не обязательно использовать круглые скобки при объявлении аргументов замыкания. | | |
| | untyped_error_in_catch | `catch` должен явно указывать тип ошибки, которую отлавливает. | Ловить все исключения – плохая практика. | |
| | collection_alignment | Все элементы коллекций должны иметь одинаковое количество отступов. | | |
| | yoda_condition | Используйте `some == nil` вместо `nil == some`. | | |
| | operator_usage_whitespace | Операторы должны быть обрамлены одним пробелом. | | |
| | contains_over_first_not_nil | Используйте `collection.contains` вместо `collection.first(where:) != nil`. | | |
| | empty_string | Для проверки строк на пустоту, используйте `string.isEmpty` вместо `string.count == ""`. | | |
| | first_where | Используйте `collection.first(where:)` вместо `collection.filter { }.first` в коллекциях. | | |
| | function_default_parameter_at_end | Объявляйте параметры со значениями по умолчанию в конце списка параметров. | | 22 |
| | last_where | Используйте `collection.last(where:)` вместо `collection.filter { }.last` в коллекциях. | | |
| | legacy_random | Используйте `type.random(in:)` вместо устаревших функций (`arc4random`). | | |
| | lower_acl_than_parent | Объявления должны иметь более низкий уровень доступа, чем их родительский элемент. | | |
| | prohibited_interface_builder | Использование `@IBAction` и `@IBOutlet` запрещено. | Мы не используем Interface Builder. | |
| | redundant_nil_coalescing | Выражения типа `option ?? nil` не имеют смысла. | | |
| | sorted_first_last | Используйте `min()` и `max()` вместо `sorted().first` и `sorted().last`. | | |
| | static_operator | Операторы должны быть объявлены как статические функции а не свободные. | Теоретически должно давать компилятору больше возможностей для оптимизации. | |
| | trailing_comma | После последнего элемента в многострочном литерале (массиве, словаре) должна быть запятая. | При добавлении нового элемента в литерал в диффе не будет затронута строчка с добавленной запятой. Таким образом в диффе будет только элемент, который добавили. | |
| | toggle_bool | Используйте `bool.toggle()` вместо `bool = !bool`. | | |
| | reduce_boolean | Используйте `.allSatisfy()` или `.contains()` вместо `.reduce(true)` и `.reduce(false)` | | |

## Кастомные правила:

| Приоритет | Правило | Описание | Почему? | Кол-во ошибок |
|---|---|---|---|---|
| | prohibited_cyrillic | Использование кириллицы в именах идентификаторов и типов запрещено. | Можно случайно написать кириллическую С вместо латинской. | |
| | tabs_for_indentation | Для отступов используйте табы вместо пробелов. | | |
| | opening_brace_statement | Открывающая скобка управляющих конструкций должна быть на той же строке. | | |
| | opening_brace_func | Открывающая скобка объявлений функций должна быть на той же строке. | | |
| | opening_brace_types | Открывающая скобка объявлений типов должна быть на новой строке. | | |
| | spacing_before_parenthesis | Перед открывающей круглой скобкой в объявлении функции не должно быть пробелов. | | |
| | todo_fixme | Комментарии TODO и FIXME должны содержать номер задачи формата QIS-<task>. | | |
| | explicit_false | Вместо `!something` нужно писать `something == false`. | | 12 |
| | comparison_assignment | При присваивании сравнения, сравнение нужно оборачивать в скобки. `let something = (anything == false)` | | |
| | required_final | Всегда делаем классы финальными. | | |
| | explicit_failure_calls | Вместо `assert(false)` используйте `assertionFailure()` или `preconditionFailure()`. | | |
| | prohibited_global_constants | Не приватные глобальные константы запрещены. Делайте их приватными, либо используйте `enum`, если они должны использоваться из других файлов. | | |

## Включенные по умолчанию встроенные правила:

| Правило | Описание | Почему? | Кол-во ошибок |
|---|---|---|---|
| weak_computed_property | Добавление weak к вычисляемым полям не имеет никакого смысла. | | |
| block_based_kvo | Используйте KVO API с блоками, а не методами. *А лучше вообще не используйте KVO.* | | |
| class_delegate_protocol | Протоколы-делегаты должны быть class-only, чтобы на них можно было ссылаться по слабой ссылке. | | |
| closing_brace | Между закрывающей фигурной скобкой и круглой скобкой не должно быть пробелов. | | |
| closure_parameter_position | Параметры замыкания должны быть на той же строке, что и открывающая скобка замыкания. | | |
| colon | Перед двоеточием не должно быть пробелов, после двоеточия должен быть только один пробел. | | |
| comma | Перед запятой не должно быть пробелов, после запятой должен быть только один пробел. | | |
| compiler_protocol_init | Инициализаторы compiler-протоколов (`ExpressibleByArrayLiteral` и прочие) не должны явно вызываться. | | |
| control_statement | Не нужно использовать круглые скобки в управляющих операторах. | | |
| cyclomatic_complexity | Сложность функций должна быть ограничена (правило выключено для `switch`). | | |
| discarded_notification_center_observer | При использовании NotificationCenter с использованием блока в качестве слушателя, возвращаемое значение должно быть сохранено для последующей отписки от нотификаций. | | |
| discouraged_direct_init | Для некоторых типов запрещено вызывать явные инициализаторы (UIDevice, Bundle, ...). | | |
| dynamic_inline | Избегайте одновременного использования `dynamic` и `@inline(__always)`. | | |

| | | | |
|---|---|---|---|
| empty_enum_arguments | При сравнении по маске enum с ассоциированными значениями, избегайте указания этих значений, если они не используются. | | |
| empty_parameters | Используйте `() ->` вместо `Void ->` | | |
| empty_parentheses_with_trailing_closure | При использовании trailing-замыканий, избегайте использования пустых круглых скобок после вызова метода. | | |
| file_length | Максимальное количество строк в файле: **400** | | |
| for_where | Используйте `for ... where` где это возможно (for с одним if внутри). | | |
| force_cast | Не разворачивайте принудительно приведение типа (`as!`). | | |
| force_try | Не разворачивайте принудительно try (`try!`). | | |
| function_body_length | Максимальное количество строк в теле функции: **40** | | |
| function_parameter_count | Максимальное количество параметров функции: **5** | | |
| generic_type_name | Имя дженериков должно состоять только из символов `[a-z0-9]`, начинаться с большой буквы и быть не длиннее 20 символов. | | |
| identifier_name | Аналогичное правило для имен идентификаторов. Минимальная длина: **2** | | |
| implicit_getter | Вычисляемые read-only свойства не должны использовать ключевое слово `get`. | | |
| inert_defer | Не используйте `defer` в конце родительской области видимости. | | |
| is_disjoint | Используйте `set.isDisjoint` вместо `set.intersection().isEmpty` | | |
| leading_whitespace | Файлы не должны содержать пустых строк в начале. | | |
| legacy_cggeometry_functions | Вместо старых функций типа `CGRectGetWidth` используйте новые (`rect.width`, ...). | | |
| legacy_constant | Вместо старых глобальных констант типа `CGRectInfinite` используйте новые (`CGRect.infinite`). | | |
| legacy_constructor | Вместо старых конструкторов типа `CGPointMake(10, 10)` используйте новые (`CGPoint(x: 10, y: 10)`) | | |
| legacy_hashing | Если вы хотите хэшировать несколько значений, реализуйте метод протокола Hashable `hash(into:)` вместо написания своего велосипеда в `hashValue`. | | |
| line_length | Максимальная длина строки: **120** | | |
| mark | Комментарии `// MARK` должны иметь правильный формат. | | |
| multiple_closures_with_trailing_closure | Синтаксис trailing-closure не должен использоваться, если в функцию передается несколько замыканий. | | |
| nesting | Максимальный уровень вложенности для типов: **1**. Для всего другого: **5**. | | |
| no_fallthrough_only | Избегайте case-ов, содержащих только fallthrough. | | |
| notification_center_detachment | Используйте `NotificationCenter.default.removeObserver(self)` только в `deinit`. | Использование в другом месте может привести к отписке от нотификаций, слушаемых дочерним классом. | |
| operator_whitespace | При объявлении новых операторов, они должны быть обрамлены пробелом с каждой стороны. | | |
| private_over_fileprivate | Используйте `private` вместо `fileprivate`. | | |
| private_unit_test | Не пишите приватные Unit-тесты. | Они не будут выполнены, и IDE об этом ничего не скажет. | |
| protocol_property_accessors_order | При объявлении свойств протокола, порядок модификаторов должен быть `get set`. | | |
| redundant_discardable_let | Вместо `let _ = foo()` пишите `_ = foo()`. | | |
| redundant_objc_attribute | В некоторых случаях атрибут `@objc` является лишним. Например, если класс уже объявлен с @objcMembers. | | |
| redundant_optional_initialization | Инициализация Optional-переменной с nil избыточна. | | |

| | | | |
|---|---|---|---|
| redundant_set_access_control | Не указывайте явно уровень доступа сеттера, если он соответствует уровню доступа геттера. | | |
| redundant_string_enum_value | Не указывайте явно строковое значение `enum`, если оно равно названию `case`. | | |
| redundant_void_return | Не используйте `-> Void`, если функция ничего не возвращает. *Не относится к замыканиям.* | | |
| return_arrow_whitespace | Стрелка в указании типа возвращаемого значения должна быть обрамлена пробелами. | | |
| shorthand_operator | Используйте операторы `+=, -=, *=, /=` вместо `x = x + 1` и т.п. | | |
| statement_position | `else` и `catch` должны быть на новых строках. | | |
| switch_case_alignment | `case` должны быть вертикально выровнены со `switch`. | | |
| syntactic_sugar | Используйте `[Int]` вместо `Array<Int>` и т.п. | | |
| trailing_newline | В конце файла должна быть новая строка. | | |
| trailing_semicolon | Не используйте точку с запятой в конце строк. | | |
| trailing_whitespace | В конце строк не должно быть пробелов. | | |
| type_body_length | Максимальное количество строк в определении типа: **200** | | |
| type_name | Максимальная длина имени типа: **3-40** | | |
| unneeded_break_in_switch | Избегайте линих `break` в `switch`. | | |
| unused_closure_parameter | Неиспользуемые параметры замыканий должны быть заменены на _. | | |
| unused_control_flow_label | Неиспользуемые control-flow метки должны быть удалены. | | |
| unused_enumerated | Используйте `collection.enumerated()` только там, где вам действительно нужен `item` и `index`. | | |
| unused_optional_binding | Используйте `!= nil` вместо `let _ =`. | | |
| vertical_whitespace | Можно использовать не более одной пустой строки. | | |
| void_return | Используйте `-> Void` вместо `-> ()`. | | |
| weak_delegate | На делегаты всегда нужно ссылаться слабой ссылкой (`weak`). | | |
| xctfail_message | В тестах все вызовы XCTFail должны содержать сообщение. | | |

## Выключенные нами правила:

| Правило | Описание | Причина выключения |
|---|---|---|
| large_tuple | Ограничивает максимальное количество элементов в кортежах. | Кортеж на то и кортеж. |
| opening_brace | Открывающая фигурная скобка должна быть на той же строке, что и определение. | Для определений типов (class, struct, ...) мы переносим открывающую скобку на следующую строку. |
| todo | Все TODO и FIXME кидают предупреждение. | Мы можем оставлять TODO и FIXME, но только указывая номер задачи в Jira. |
| vertical_parameter_alignment | Параметры функции в определении должны быть выровнены вертикально. | Из-за использования нами табов, SwiftLint выдает слишком много ложноположительных предупреждений. |