

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

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

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 <code>!something</code> нужно писать <code>something == false</code> .		12
	comparison_assignment	При присваивании сравнения, сравнение нужно оборачивать в скобки. <code>let something = (anything == false)</code>		
	required_final	Всегда делаем классы финальными.		
	explicit_failure_calls	Вместо <code>assert(false)</code> используйте <code>assertionFailure()</code> или <code>preconditionFailure()</code> .		
	prohibited_global_constants	Не приватные глобальные константы запрещены. Делайте их приватными, либо используйте <code>enum</code> , если они должны использоваться из других файлов.		

Включенные по умолчанию встроенные правила:

Правило	Описание	Почему?	Кол-во ошибок
weak_computed_property	Добавление <code>weak</code> к вычисляемым полям не имеет никакого смысла.		
block_based_kvo	Используйте KVO API с блоками, а не методами. <i>А лучше вообще не используйте KVO.</i>		
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 <code>compiler-протоколов</code> (<code>ExpressibleByArrayLiteral</code> 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 <code>switch</code>).		
discarded_notification_center_observer	При использовании <code>NotificationCenter</code> 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 (<code>UIDevice</code> , <code>Bundle</code> , ...).		
dynamic_inline	Избегайте одновременного использования <code>dynamic</code> и <code>@inline(__always)</code> .		

empty_enum_arguments	При сравнении по маске enum с ассоциированными значениями, избегайте указания этих значений, если они не используются.		
empty_parameters	Используйте <code>()</code> -> вместо <code>Void</code> ->		
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 <code>for ... where</code> где это возможно (for с одним if внутри).		
force_cast	Не разворачивайте принудительно приведение типа (<code>as!</code>).		
force_try	Не разворачивайте принудительно <code>try</code> (<code>try!</code>).		
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 <code>[a-z0-9]</code> , 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 <code>get</code> .		
inert_defer	Не используйте <code>defer</code> в конце родительской области видимости.		
is_disjoint	Используйте <code>set.isDisjoint</code> вместо <code>set.intersection().isEmpty</code>		
leading_whitespace	Файлы не должны содержать пустых строк в начале.		
legacy_cggeometry_functions	Вместо старых функций типа <code>CGRectGetWidth</code> используйте новые (<code>rect.width</code> , ...).		
legacy_constant	Вместо старых глобальных констант типа <code>CGRectInfinite</code> используйте новые (<code>CGRect.infinite</code>).		
legacy_constructor	Вместо старых конструкторов типа <code>CGPointMake(10, 10)</code> используйте новые (<code>CGPoint(x: 10, y: 10)</code>)		
legacy_hashing	Если вы хотите хэшировать несколько значений, реализуйте метод протокола <code>Hashable</code> <code>hash(into:)</code> вместо написания своего велосипеда в <code>hashValue</code> .		
line_length	Максимальная длина строки: 120		
mark	Комментарии <code>// MARK</code> 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 <code>fallthrough</code> .		
notification_center_detachment	Используйте <code>NotificationCenter.default.removeObserver(self)</code> только в <code>deinit</code> .	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 <code>private</code> вместо <code>fileprivate</code> .		
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 <code>get set</code> .		
redundant_discardable_let	Вместо <code>let _ = foo()</code> пишите <code>_ = foo()</code> .		
redundant_objc_attribute	В некоторых случаях атрибут <code>@objc</code> является лишним. Например, если класс уже объявлен с <code>@objcMembers</code> .		
redundant_optional_initialization	Инициализация <code>Optional</code> -переменной с <code>nil</code> 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 <code>-> Void</code> , если функция ничего не возвращает. <i>Не относится к замыканиям.</i>		
return_arrow_whitespace	Стрелка в указании типа возвращаемого значения должна быть обрамлена пробелами.		
shorthand_operator	Используйте операторы <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> вместо <code>x = x + 1</code> и т.п.		
statement_position	<code>else</code> и <code>catch</code> должны быть на новых строках.		
switch_case_alignment	<code>case</code> должны быть вертикально выровнены со <code>switch</code> .		
syntactic_sugar	Используйте <code>[Int]</code> вместо <code>Array<Int></code> 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 линий <code>break</code> в <code>switch</code> .		
unused_closure_parameter	Неиспользуемые параметры замыканий должны быть заменены на <code>_</code> .		
unused_control_flow_label	Неиспользуемые control-flow метки должны быть удалены.		
unused_enumerated	Используйте <code>collection.enumerated()</code> только там, где вам действительно нужен <code>item</code> и <code>index</code> .		
unused_optional_binding	Используйте <code>!= nil</code> вместо <code>let _ =</code> .		
vertical_whitespace	Можно использовать не более одной пустой строки.		
void_return	Используйте <code>-> Void</code> вместо <code>-> ()</code> .		
weak_delegate	На делегаты всегда нужно ссылаться слабой ссылкой (<code>weak</code>).		
xctfail_message	В тестах все вызовы XCTFail должны содержать сообщение.		

Включенные нами правила:

Правило	Описание	Причина исключения
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 (<code>class</code> , <code>struct</code> , ...) 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.