

# **iOS projects code formattings**

**Eduard Panasiuk**

# Tools

- ClangFormat
- SwiftFormat
- SwiftLint

# ClangFormat

# ClangFormat

<https://clang.llvm.org/docs/ClangFormat.html>

## Instalation:

```
brew install clang-format
```

## Usage:

```
clang-format -style=file ./MyClass.m
```

# ClangFormat options

<https://clang.llvm.org/docs/ClangFormatStyleOptions.html>

```
---
# We'll use defaults from the LLVM style, but with 4 columns indentation.
BasedOnStyle: LLVM
IndentWidth: 4
---
Language: Cpp
# Force pointers to the type for C++.
DerivePointerAlignment: false
PointerAlignment: Left
---
Language: JavaScript
# Use 100 columns for JS.
ColumnLimit: 100
---
Language: Proto
# Don't format .proto files.
DisableFormat: true
...
```

# ClangFormat predefined styles

LLVM, Google, Chromium, Mozilla, WebKit

**Dump predefined style:**

```
clang-format -style=llvm -dump-config > .clang-format
```

# ClangFormat options

<https://clangformat.com/>

## .clang-format

An interactive guide and builder

```
AccessModifierOffset: 0
AlignEscapedNewlinesLeft: true
AlignTrailingComments: false
AllowAllParametersOfDeclarationOnNextLine: false
AllowShortFunctionsOnASingleLine: false
AllowShortIfStatementsOnASingleLine: true
AllowShortLoopsOnASingleLine: false
AlwaysBreakBeforeMultilineStrings: false
AlwaysBreakTemplateDeclarations: false
BinPackParameters: false
BreakBeforeBinaryOperators: false
BreakBeforeBraces: Allman
BreakBeforeTernaryOperators: false
```

### AlignEscapedNewlinesLeft (bool)

If `true`, aligns escaped newlines as far left as possible. Otherwise puts them into the right-most column.

This does not necessarily mean flushing lines to the left but, rather, attempting to align the current line's left margin with the previous line's left margin.

### Example

☒ true ☐ false

```
if (foo && // Some comment
bar) {
    baz();
}

void foo() {
    someFunction();
    someOtherFunction();
}
```

# ClangFormat usage

## Git hook

Create `.git/hooks/pre-commit`

```
#!/bin/bash
```

```
CLANG_FORMAT=$(which clang-format)
```

```
git diff-index --cached --diff-filter=ACMR --name-only $against -- | while read file;
do
    "$CLANG_FORMAT" -i -style=file "$file"
    git add "$file"
done
```



# SwiftFormat

# SwiftFormat

<https://github.com/nicklockwood/SwiftFormat>

## Instalation:

```
brew install swiftformat
```

# SwiftFormat usage

## Terminal

Run in project directory:

```
swiftformat .
```

# SwiftFormat options

<https://github.com/nicklockwood/SwiftFormat>

<code>--allman</code>	use allman indentation style. "true" or "false" (default)
<code>--binarygrouping</code>	binary grouping, threshold or "none", "ignore". default: 4,8
<code>--commas</code>	commas in collection literals. "always" (default) or "inline"
<code>--comments</code>	indenting of comment bodies. "indent" (default) or "ignore"
<code>--decimalgrouping</code>	decimal grouping, threshold or "none", "ignore". default: 3,6
<code>--elseposition</code>	placement of else/catch. "same-line" (default) or "next-line"
<code>--empty</code>	how empty values are represented. "void" (default) or "tuple"
<code>--experimental</code>	experimental rules. "enabled" or "disabled" (default)
<code>--exponentcase</code>	case of 'e' in numbers. "lowercase" or "uppercase" (default)
<code>--header</code>	header comments. "strip", "ignore", or the text you wish use
<code>--hexgrouping</code>	hex grouping, threshold or "none", "ignore". default: 4,8
<code>--hexliteralcase</code>	casing for hex literals. "uppercase" (default) or "lowercase"
<code>--ifdef</code>	#if indenting. "indent" (default), "noindent" or "outdent"
<code>--indent</code>	number of spaces to indent, or "tab" to use tabs
<code>--indentcase</code>	indent cases inside a switch. "true" or "false" (default)
<code>--linebreaks</code>	linebreak character to use. "cr", "crlf" or "lf" (default)
<code>--octalgrouping</code>	octal grouping, threshold or "none", "ignore". default: 4,8
<code>--operatorfunc</code>	spacing for operator funcs. "spaced" (default) or "nospace"
<code>--patternlet</code>	let/var placement in patterns. "hoist" (default) or "inline"
<code>--ranges</code>	spacing for ranges. "spaced" (default) or "nospace"
<code>--semicolons</code>	allow semicolons. "never" or "inline" (default)
<code>--self</code>	use self for member variables. "remove" (default) or "insert"
<code>--stripunusedargs</code>	"closure-only", "unnamed-only" or "always" (default)
<code>--trimwhitespace</code>	trim trailing space. "always" (default) or "nonblank-lines"
<code>--wraparguments</code>	wrap function args. "beforefirst", "afterfirst", "disabled"
<code>--wrapelements</code>	wrap array/dict. "beforefirst", "afterfirst", "disabled"

# SwiftFormat options

## Enable/Disable rules for concrete file

```
// swiftformat:disable <rule1> [<rule2> [rule<3> ...]]
```

```
// swiftformat:enable <rule1> [<rule2> [rule<3> ...]]
```

# SwiftFormat usage

## Xcode build phase

Install pod:

```
pod 'SwiftFormat/CLI'
```

Add build phase:

```
"${PODS_ROOT}/SwiftFormat/CommandLineTool/swiftformat"
```

# SwiftFormat usage

## Git hook

Create `.git/hooks/pre-commit`

```
#!/bin/bash
git diff --staged --name-only | grep -e '\(.*\)\.swift$' | while read file; do
    swiftformat ${file};
    git add $file;
done
```

# Swift linters

Tailor - Cross-platform static analyzer and linter for Swift

SwiftLint - Swift linter



# SwiftLint

# SwiftLint

## Instalation:

```
brew install swiftlint
```

```
pod 'SwiftLint'
```

# SwiftLint usage

## Xcode build phase

Install pod:

```
pod 'SwiftLint'
```

Add build phase:

```
"${PODS_ROOT}/SwiftLint/swiftlint autocorrect"
```

# SwiftLint usage

## Fastlane

```
swiftlint(  
  mode: :lint, # SwiftLint mode: :lint (default) or :autocorrect  
  executable: "Pods/SwiftLint/swiftlint", # The SwiftLint binary path (optional). Important if you've installed it via CocoaPods  
  path: "/path/to/lint", # Specify path to lint (optional)  
  output_file: "swiftlint.result.json", # The path of the output file (optional)  
  reporter: "json", # The custom reporter to use (optional)  
  config_file: ".swiftlint-ci.yml", # The path of the configuration file (optional)  
  files: [  
    "AppDelegate.swift",  
    "path/to/project/Model.swift"  
  ],  
  ignore_exit_status: true, # Allow fastlane to continue even if SwiftLint returns a non-zero exit status (Default: false)  
  quiet: true, # Don't print status logs like 'Linting ' & 'Done linting' (Default: false)  
  strict: true # Fail on warnings? (Default: false)  
)
```

# SwiftLint

## Supported rules

`.swiftlint.yml` in project directory

Supports nested configurations (`.swiftlint.yml` in directory structure)

# SwiftLint

## Custom rules

```
custom_rules:  
  pirates_beat_ninjas: # rule identifier  
    included: ".*\\.swift" # regex that defines paths to include during linting. optional.  
    excluded: ".*Test\\.swift" # regex that defines paths to exclude during linting. optional  
    name: "Pirates Beat Ninjas" # rule name. optional.  
    regex: "([n,N]inja)" # matching pattern  
    match_kinds: # SyntaxKinds to match. optional.  
      - comment  
      - identifier  
    message: "Pirates are better than ninjas." # violation message. optional.  
    severity: error # violation severity. optional.  
  no_hiding_in_strings:  
    regex: "([n,N]inja)"  
    match_kinds: string
```

# **iOS projects code formattings**

**Eduard Panasiuk**