Extracted from:

Swift Style, Second Edition

An Opinionated Guide to an Opinionated Language

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Swift Style

Second Edition

An Opinionated Guide to an Opinionated Language

Erica Sadun

edited by Brian MacDonald

Swift Style, Second Edition

An Opinionated Guide to an Opinionated Language

Erica Sadun

The Pragmatic Bookshelf

Raleigh, North Carolina

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Susan Conant
Development Editor: Brian MacDonald
Copy Editor: Nicole Abramowitz

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-627-3
Book version: P1.0—March 2019

This book is dedicated to the Swift community both inside and outside Apple.

Selecting Colon Styles

The Ash Rule (named for developer luminary Mike Ash) adopts left-hugging colons in all uses except operator declarations and ternary forms. His rule applies to dictionary references, protocols, and so forth:

```
let dict = ["a": 1, "b": 2] // Ash Rule
let dict = ["a" : 1, "b" : 2] // Commonly used
```

Here are a few examples of magnetic-left (that is, left-hugging) colons in common use cases:

```
let x: [String: String] = ["key": "value"]
let y = foo(param1: value1, param2: value2)
func bar<T: Hashable>(a: T) -> Void {}
```

Moving to this style felt odd for a while, but I've now grown used to it. This approach is consistent and prioritizes the role of the left item. It is, in my opinion, quite readable. Compare and contrast with my prior style, which I call the Full Monty spacing approach:

```
let x : [String : String] = ["key" : "value"]
let y = foo(param1 : value1, param2 : value2)
func bar<T : Hashable>(a : T) -> Void {}
```

The Ash Rule puts you in step with more than half of Apple source code and out of step with the rest. Colon magnetism tends to vary by group, but it is growing as a trend.

Some people use magnetic left except for protocol conformance and inheritance clauses. Adding spaces before colons elevates prominence. For example, in `extension AnyHashable : Equatable`, the spacing makes colons stand out more. In theory, this enables readers to better identify the colon's use point.

Here's an old example from Swift's source code that showcases this style before this type moved to magnetic-left style. Notice the Full Monty spacing for the generic type parameter and protocol conformance:

```
public struct EnumeratedIterator<
  Base : IteratorProtocol
> : IteratorProtocol, Sequence {
  internal var _base: Base
  internal var _count: Int
```

The current implementation updates both wrapping and colon use. (It also drops conformance to `IteratorProtocol` and `Sequence`.) I think it's easier to read now:

```
public struct EnumeratedIterator<Base: IteratorProtocol> {
  internal var _base: Base
  internal var _count: Int
```

Apple Developer Publications uses magnetic-left colons. (See this tweet,[3] for example.) This style is simple and readable:

```
class SimpleClass: ExampleProtocol {
    var simpleDescription: String = "A very simple class."
    var anotherProperty: Int = 69105
    func adjust() {
        simpleDescription += "  Now 100% adjusted."
    }
}
```

Skip left-hugging colons in ternary expressions because they unnaturally prioritize the left value over the right:

```
let result = booleanCondition ? value1: value2 // no
let result = booleanCondition ? value1 : value2 // yes
```

In operator declarations, conjoined colons aren't legal. They confuse the compiler because Swift assumes the colon is part of the operator name:

```
infix operator &&: LogicalConjunctionPrecedence // no
infix operator && : LogicalConjunctionPrecedence // yes
```

Common Colon Styles

When it comes to colon style, there aren't any naturally right or wrong answers. Styles break down into the following common categories:

Label declarations

func f(x: T), f(x y: T)

Most commonly magnetic left.

Constant and variable typing

let x: T

Most commonly magnetic left.

Dictionary declarations

[key: value]

Consensus is split between full spacing and magnetic left with magnetic left becoming prevalent.

Empty dictionaries

[:] and not [:]

3. https://twitter.com/_jackhl/status/646723367576276992

Universal consensus: no spaces.

Cases

`case a:`, `case a, b:`

Universal consensus: magnetic left.

Attribute

`@attribute(key: value)`—for example, `@available(*, unavailable, renamed: “MyRenamed”)`
`@available(swift, deprecated: 4.1, obsoleted: 5.0.0, renamed: "copyMemory(from:byteCount:)")`

Consensus: magnetic left.

Inheritance

`Derived: Parent`

Consensus is split between full spacing and magnetic left, with magnetic left becoming increasingly more common.

Conformance

`<T: A>`, `<T: A & B>`

Consensus is split between full spacing and magnetic left, with magnetic left becoming increasingly more common.

Operator conformance

`public func <= <T>...`

Add a space between the operator symbol and its generic declaration.

Ternary

`A ? B : C`

Always use fully spaced ternary expressions.

Some coders blaspheme and use no spaces at all between labels and argument values. Although this offers a slight advantage (Xcode wraps the two together on long lines), it’s an uncommon style and possibly a sin of commission.

```
unsafeBitCast(type, to: Int.self) // yes
String(validatingUTF8: ptr) // yes
unsafeBitCast(type, to:Int.self) // no
String(validatingUTF8:ptr) // no
unsafeBitCast(type,to:Int.self) // no
```

Using Type-Magnetic Colons

As with any style, there are always a few outlier users. Leung-style typing (named for Kenny Leung) uses type-magnetic colons, as in the following example:

```
var soccer :Ball = newBall
```

Here, the colon hugs the type rather than being tied to the symbol or using full spacing. Under this paradigm, a colon introduces the type using a magnetic-right style. While this approach will never be my personal choice, if used consistently, it offers a rational coding alternative to the majority Swift convention.

- Colons are as worthy of style consideration as any other language member.
- Spaces communicate prominence.
- Adopt consistent styling, regardless of language construct.
- Type-magnetic colons are just plain weird. I avoid them.