【文字の装飾タグに関するヒント】

△ストロング△＝強調したいキーワード（太字になります）

▲太字▲＝単なるデザイン目的での装飾で、キーワードではない

本文中に◎Abc("aaa")◎と記載すると、そこは<code>タグで囲まれ、その文字列が□の枠で囲まれて表示されます（GitHubなどでよくあり開発者が慣れていそうなので、最近、クラス名などのコード関連はほとんどこれで囲んでいます）。Markdownだと`Abc("aaa")`です。

ほかにHTML上などでカスタム表現してほしいことがあれば、★これでコメント★を書いていただければ、その部分は制作時に内容を見て手動でHTML化します。※ちなみに斜体は非推奨としています。

Build Insiderオピニオン：小野将之（1）

Swift 3のリリース前に、これまでの進化の変遷をなぞる

Apple発のオープンソースなプログラミング言語「Swift」はこれまでにどのような進化の道のりをたどってきたのか。その道程を追い、その将来に思いはせるコラムが新登場！

小野 将之

2016/08/xx

############################################################

　本連載では、Swift言語の最新動向中心に述べていくが、その前にこの第1回ではこれまでのSwiftの進化の変遷をなぞっていきたい。Swift 3は2016年9月ごろに正式リリースとなりそうだが、今回はまず現最新版の2.2.1までについて触れていく。

# ■Swiftとは

　そもそも、Swiftが当時アップル社（Apple）からどのように発表されたかを今振り返るとなかなか興味深い。Swiftは、2014年6月に開催された◆WWDC 2014◇https://developer.apple.com/videos/wwdc2014/◆のKeynoteで、突如発表された（驚くべきことに事前リーク・予想などもなかった）。

□動画------------------------

2014年に開催されたWWDCでのSwiftの発表

□□----------------------------

<iframe src=" https://www.youtube.com/watch?v=w87fOAG8fjk&t=1h45m"></iframe>

□□□--------------------------

　そこで、以下の4つの特徴を持つ言語として紹介された。

・ **△高速△**

・ **△モダン△**（クロージャー、オプショナル型、ジェネリクス、型推論、タプル……）

・ **△安全△**

・ **△インタラクティブ△**（REPL：対話実行環境、Playground）

　また、Cocoaフレームワーク・ランタイムをObjective-C言語環境と共有しており、Objective-C言語を使った場合と同様に何の制限もなくアプリ開発が可能で、SwiftコードとObjective-Cコードとの混在もできることなどが強調された。

　この発表直後から、Swiftは洗練された言語仕様などから大きな注目を集めたが、そこから現時点で2年ほどの短期間で大きな進化を遂げている。

# ■Swift 2.2までの進化の変遷

　これまで短いスパンでSwiftにはさまざまな改善が繰り返されてきたが、その説明のためにも、現時点での正式版最新の2.2.1までの進化の変遷をざっと見ておこう。

## ●Swiftリリース以前

　WWDC 2014での発表前に水面下でSwiftの開発がなされていた。

・ 2010年半ば： Swiftの開発開始

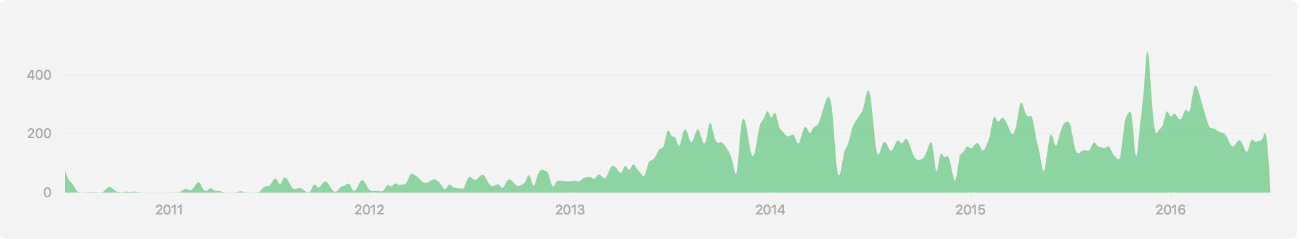
・ 2010年7月17日： ◆Swiftリポジトリへのファーストコミット◇https://github.com/apple/swift/commit/18844bc65229786b96b89a9fc7739c0fc897905e◆

・ 2013年7月にSwiftの開発が開発ツールチームの主要目標となる

　◆Lattner氏◇https://github.com/lattner◆が2010年に△ファーストコミット△した記録が残っている。◆Lattner氏のブログ◇http://nondot.org/sabre/◆には、初めは1～数人程度の小規模で開発が進められ、Objective-C・Rust・Haskell・Ruby・Python・C#・CLUなどの言語が参考にされたことなどが記載されている。

□------------------------------

★画像【01.png】★



□□----------------------------

◆Swiftのコミットグラフ（GitHub）◇https://github.com/apple/swift/graphs/contributors◆

□□□--------------------------

## ●2014年6月2日： WWDC 2014でSwiftの発表、ベータ版公開

　そうして、WWDC 2014でSwiftのベータ版が発表された。同時に丁寧なドキュメントが、アップルのサイトやiBooksなどで公開され、Keynoteやセッションでは紹介されなかった細かい仕様も把握できたことで、素晴らしい言語仕様などが評判を集めた（Objective-Cへの不満が多かった反動も大きい）。この発表の時点で、すぐにでも実際のアプリを書き始められる状態であったものの、当時は少し書くだけでコンパイラーやXcodeがクラッシュしてしまうようなことも頻発しており、開発者はかなり苦労させられた。

　この時期、9月の正式リリースまでは、バグ修正や言語仕様の細かい調整などがなされた。

　以下に示す◆koher◇http://qiita.com/koher◆氏の記事などからは、地味ながらドラスティックな調整がなされた経緯が見受けられ、今振り返っても興味深い。

・ ◆SwiftのArrayがヤバイ◇http://qiita.com/koher/items/6f54eafac59793e7c364◆（△ベータ2△の段階）

・ ◆SwiftのArrayが実はすばらしかった◇http://qiita.com/koher/items/47f444200d4e02b6414e◆（△ベータ3△の段階）

　また、◆△ベータ4△でアクセス修飾子◇https://developer.apple.com/swift/blog/?id=5◆が導入されたことも印象に強く残っている。当初はこの仕様が欠けていたため、Swiftはこれを不要と見なしたのかと筆者は推測したが、単に間に合わなかっただけのようである。このことから、Swiftはまだまだ発展途中の言語であるとあらためて認識させられ、その成長に付き合う必要があり、またそれが楽しみと思うようになった。

## ●2014年9月15日： Swift 1.0／Xcode 6の正式版リリース

　そうしてベータ版のバージョンアップが重なるにつれて、段々とまともに動くようになってきた。そうした中で毎年恒例（9月）のアップル主催イベントでは正式版のリリースが発表され、9月15日に晴れて△Swift 1.0正式版△がリリースされた。

　同年10月15日にはSwift 1.1／Xcode 6.1がリリースされた。◆Failable Initializer（初期化失敗時に◎nil◎が返る）の導入◇https://developer.apple.com/swift/blog/?id=17◆など細かい改善はあったものの、△Swift 1.1△も安定性の向上がメインのアップデートで、言語仕様の目立った変更は少なめとなっていた（とはいえ特にオプショナル型周りの細かい調整などの破壊的変更は繰り返されていたが）。

## ●2015年4月8日： Swift 1.2／Xcode 6.3の正式版リリース

　Swift 1.1系までで、動作がかなり安定してきたこともあり、1.2以降は盛んに仕様改善・機能改善が行われるようになってきた。メジャーアップデートでないにもかかわらず、Swift 1.2では◆たくさんの改善が施された◇https://developer.apple.com/swift/blog/?id=22◆。主要なものを以下に示す。

・ ビルドの高速化：

　　－ ビルドの遅さはSwiftへの不満点として、安定性の次に挙げられていたが、Swift 1.2では差分ビルドの導入などによってかなり改善された

・ ◎if-let◎構文で、複数のオプショナル型の値のアンラップがサポート：

　　－ それまでは値の個数分◎if-let◎のネストが必要だった

・ ダウンキャストの結果がオプショナル型になった

・ ◎let◎で宣言した値のセットが、利用の直前まで許容されるようになった

・ ◎NSString◎型から◎String◎型などへの暗黙キャストの廃止

・ Objective-Cで◎nullable◎（Swiftから見たときにオプショナル型になる）などを宣言可能に

・ クラスのメソッドやプロパティを、◎static◎で宣言可能に

・ ◎@noescape◎属性の追加

・ ◎Set◎データ型の追加：

　　－ セット（集合）を扱う場合、それまでは◎NSSet◎型を使ってしのいできたが、◎Set◎型ではジェネリクス機能を使って型パラメーターに要素の型を指定することでセットをよりSwiftらしく扱えるようになった

　ここでは△Swift 1.2△での改善点を列挙したが、これは個々の項目を把握してほしいというよりは、主要なものをかいつまんだだけでもこれほどのボリュームになることをお分かりいただくためのものだ（詳細については今後の連載で補足していく予定である）。

## ●2015年9月15日： Swift 2.0／Xcode 7の正式版リリース

　2015年6月には、◆WWDC 2015で△Swift 2.0△が発表された◇https://developer.apple.com/swift/blog/?id=29◆。同時にオープンソース化（後述）を年末に予定していることもアナウンスされた。そうしてSwift 2.0は、Swift 1のときと同じく◆同年9月に正式リリースされた◇https://developer.apple.com/swift/blog/?id=32◆。ちょうど1年で初のメジャーアップと、かなり速いスピードで進化を遂げていることが分かる。

　Swift 2.0での主な改善は以下の通りである。

・ ◎try◎、◎throw◎、◎catch◎によるエラーハンドリング：

　　－ それまではこの仕様なしで済んでおり、エラーハンドリングはオプショナル型を利用したいわゆる◎Either◎型などで補うのでは？ と思われていたので意外な仕様追加であった。「◆Swift 2.0 の◎try◎、◎catch◎ファーストインプレッション◇http://qiita.com/koher/items/0c60b13ff0fe93220210◆」に詳しい考察がある

・ ◎#available◎ステートメント：

　　－ OSやそのバージョンでの条件分岐をコンパイラーレベルで確実に行えるように

・ ◎repeat-while◎ステートメント：

　　－ C言語などでおなじみの◎do-while◎と同様の構文

・ Protocol extension：

　　－ それまではプロトコルには必要なプロパティ／メソッドの羅列しかできなかったが、デフォルト実装を与えられるようになった。

　　－ 標準ライブラリもその影響を受け、変更があった

・ ◎defer◎ステートメント：

　　－ Go言語などでおなじみの「スコープを抜ける時に確実に実行される処理」を記述できるようにする構文

・ ◎guard◎ステートメント：

　　－ オプショナル型をアンラップしつつ、◎nil◎だった時に早期◎return◎をする構文

・ パターンマッチングの強化

・ ◎Enum◎型のジェネリクス対応

　そして、同年10月20日には細かい変更を含んだ△Swift 2.1△およびXcode 7.1の正式版がリリースされた。

## ●2015年12月3日： GitHubでSwiftをオープンソースとして公開

　WWDC 2015年で年末と予告されていた通り、同年12月3日に◆Swiftがオープンソースとして公開◇https://github.com/apple/swift/◆された（ライセンスはApache License 2.0）。Swiftリポジトリへのスターが1日で1万個を超えるなど、大変高い注目を集めた。

　Swift 2.0は主に次のリポジトリから構成されていて、多くの機能が含まれていることが分かる。

・ **△コンパイラー△**（◆clang◇https://github.com/apple/swift-clang◆ ＋ ◆llvm◇https://github.com/apple/swift-llvm◆）：

　　－ 参考： ◆Compiler Architecture◇https://swift.org/compiler-stdlib/#compiler-architecture◆

・ ◆**△標準ライブラリ△**◇https://github.com/apple/swift◆：

　　－ ◆SourceKit◇https://github.com/apple/swift/tree/master/tools/SourceKit◆というIDE機能用のフレームワークなども含まれる

・ **△コアライブラリ△**（◆Foundation◇https://github.com/apple/swift-corelibs-foundation◆ ＋ ◆XCTest◇https://github.com/apple/swift-corelibs-xctest◆ ＋ ◆libdispatch◇https://github.com/apple/swift-corelibs-libdispatch◆）

・ ◆**△Swift Evolution△**◇https://github.com/apple/swift-evolution◆：

　　－ 今後のSwift言語の変更・提案などをまとめる場（議論は◆メーリングリスト◇https://lists.swift.org/mailman/listinfo/swift-evolution◆で行われる）

・ ◆**△Swift Package Manager（SPM）△**◇https://github.com/apple/swift-package-manager◆

　その他、細かいことはオープンソース化と同時に公開された◆swift.org◇https://swift.org/◆に掲載されている。

　オープンソース化されたことでmacOS以外でのプラットフォームでのSwiftの動作も可能となり、まずはLinux（Ubuntu）上で動くバイナリが提供された。さらに実際に、2016年2月22日、IBMが◆クラウドサービス「Bluemix」のSwift対応◇https://developer.ibm.com/swift/◆と、SwiftベースのWebフレームワーク「◆Kitura◇https://github.com/IBM-Swift/Kitura◆」を発表・公開し、実際にその成果が活用されつつある。

## ●2016年3月21日： Swift 2.2／Xcode 7.2の正式版リリース

　そして2016年3月21日、オープンソース化後、初となるバージョンアップが行われ、◆△Swift 2.2△がリリースされた◇https://swift.org/blog/swift-2-2-released/◆。その中でも分かりやすいものとしては以下のような変更があった。

・ ◆◎#selector◎式の導入◇https://github.com/apple/swift-evolution/blob/master/proposals/0022-objc-selectors.md◆：

　　－ Selector指定が文字列ではなくタイプセーフに行えるように

・ ◆Swiftバージョンによるビルド切り替えオプションの導入◇https://github.com/apple/swift-evolution/blob/master/proposals/0020-if-swift-version.md◆

・ ◆◎++◎／◎--◎記法が非推奨に◇https://github.com/apple/swift-evolution/blob/master/proposals/0004-remove-pre-post-inc-decrement.md◆

・ ◆Cスタイルの◎for◎ステートメントが非推奨に◇https://github.com/apple/swift-evolution/blob/master/proposals/0007-remove-c-style-for-loops.md◆

・ ◆◎\_\_FILE\_\_◎識別子などが非推奨に（◎#file◎識別子などに変更）◇https://github.com/apple/swift-evolution/blob/e4328889a9643100177aef19f6f428855c5d0cf2/proposals/0028-modernizing-debug-identifiers.md◆

　全てに◆GitHubのProposal◇https://github.com/apple/swift-evolution/tree/master/proposals◆へのリンクを張ってある通り、実際にGitHubを通じたバグ修正・機能追加が反映されていることが分かる。Swiftの開発への貢献の仕方は◆swift.orgのContributingの章◇https://swift.org/contributing/◆（英語）に詳しく記載されている。

　また目立つのが、非推奨になった仕様である。これまでは、バージョンアップのたびにコンパイルエラーが発生するような変更が多かったが、2.2ではバージョン3で予定されている仕様変更の前に緩やかに警告で追従を促すように変わり、Swiftが安定へ向かっていることが伺える。

　その後マイナーアップデートがあり、△2.2.1△が2016年7月現在の最新である。

# ■まとめ

　と、ここまでで実質2年ほどにわたるSwiftの変遷を見てきた。紹介しきれなかった細かい変更点は◆CHANGELOG◇https://github.com/openstack/swift/blob/master/CHANGELOG◆（英語）に全て載っている。

　オープンソース化以前は、Swiftの言語仕様は素晴らしいものの、バージョンアップごとの変更点がありすぎることや、その開発フローがブラックボックスであることなど、Swiftの将来がどうなっていくのか見えにくいところがあった。オープンソース化で期待以上に多くのリソースが公開され、メーリングリストやGitHubを介したオープンな議論が見えるようになったことで、調べさえすればSwiftの今も未来も把握できるようになった。

　オープンソース化のタイミングで△Swift 3△のロードマップが公開され、現在は2016年9月ごろのリリースに向けて着々と開発が進んでいる。第2回ではそのあたりの動向を分かりやすく紹介していく。

また、最後にお知らせとして、8月20日に◆iOSDC Japan 2016 ◆<https://iosdc.jp/2016/◆というiOSとその周辺技術に関するエンジニアのためのカンファレンス>（400人規模）が開催される。3月に渋谷で開催されて世界中から著名なiOS・Swiftエンジニアが集結した◆try!Swif◆thttp://www.tryswiftconf.com◆と同規模のカンファレンスであり、Swiftをきっかけに盛り上がりを見せていることが伺える。筆者も、本連載と関連した内容の◆Swiftのこれまでの動向のまとめと今後のさらなる発展の期待◆<https://iosdc.jp/2016/c/node/155#◆と題した15分のトークをするので>、期待してほしい。

**小野 将之（おの まさゆき）**



入社当時はC／C++を用いた組み込み機器（POS）用のアプリケーション開発に携わる。

10年ほどでサーバーサイドに移り、主にJavaを使用したWebアプリケーション開発に軸足を移す。

2015年夏からクラウドを用いたソリューションのテクニカル・コンサル、PoCを生業としている。

主な著書にJava、Ruby、C言語を用いたものがある。