**Swift API Design Guidelines**

**Fundamentals**

* 명쾌함이 핵심이다.
* 짧은 것보다 명쾌함이 중요하다.
* 주석을 사용해라.

디테일

* 가능한한 엔티티에 관해 모든 것을 알 수 있도록 서머리를 작성해라.
* 서머리는 가장 중요한 부분이다. 훌륭한 문서화는 훌륭한 서머리와 다르지 않다.
* 가능하면 완전한 문장보다 온점으로 끝나는 구를 활용하라.
* 함수나 메소드가 무슨 일을 ‘**하는지’**와 ‘**반환하는지’**를표현해라.
* 섭스크립트가 **무엇**에 접근하는지 표현해라.
* 생성자가 **무엇**을 생성하는지 표현해라.

**Naming**

명쾌한 사용성을 지향해라.

* 가능한한 모든 모호함을 피해라.

예를 들어 콜렉션 안에서 특정 인자를 삭제하는 함수를 선언한다면

extension List {

public mutating func remove(at position: Index) -> Element

}

employees.remove(at: x) // 좋음

employees.remove(x) // 나쁨

간단한 사례이지만 레이블이 달린 코드는 x의 위치에 해당하는 인자를 삭제하라는 맥락을 전달한다. 하지만 후자는 캐릭터 ‘x’와 일치하는 값을 삭제하라는 건지 인덱스를 삭제하라는 건지 중의적으로 해석될 수 있다.

따라서 레이블을 활용해 메소드가 정확히 무슨 일을 하는지 구체적으로알려주는 게 좋다.

* 쓸데없는 말은 빼라

간혹 메소드 이름이 해당 메소드가 무엇을 대하는지를 중복적으로 드러내는 경우가 있다. 자바에서는 그런 긴 메소드 이름이 일반적이지만 스위프트에는 아규먼트 레이블과 파라미터가 존재한다. 따라서 메소드 이름이 표현할 영역과 레이블이 표현할 영역을 구분하는 게 좋다. 레이블이 인자 타입의 성격을 충분히 드러낸다면, 메소드에서는 타입에 관한 설명을 하지 않아도 좋다.

* 변수, 파라미터, associatedtype 등을 타입이 아닌 **역할**에 맞춰 이름 붙여라.

타입은 이름 뒤에 타입으로 표현되기 때문에 이름에 타입을 표현하면 중복되는 경우가 많다. 타입 자체가 역할과 강하게 연결된 경우에는 어쩔 수 없지만 가능하면 타입보다는 **역할**을 강조해라.

* 타입이 광범위한 경우에는 어쩔 수 없지만 메소드 차원에서 타입의 역할을 이를 강조해라

타입이 원시 타입이나 NSObject, Any, AnyObject 등인 경우에는 타입이 전하는 정보만으로 메소드가 **무엇**을 다루는지 알기 힘들 수 있다. 그럴 때는 단순히 짧은 메소드 명보다 메소드가 다루는 대상을 포함한 명쾌한 메소드 명을 사용해라.

**원활한 가독성을 위해 노력하기**

* 함수명과 레이블을 정할 때 가능한한 영어 문법적으로 옳은 방향을 지향해라.

x.insert(y, at: z) “x, insert y at z”

x.subViews(havingColor: y) “x's subviews having color y”

x.capitalizingNouns() “x, capitalizing nouns”

x.insert(y, position: z)

x.subViews(color: y)

x.nounCapitalize()

위 세 줄은 레이블에 문법적 맥락이 반영됐고 아래 세 줄은 문법적 맥락 없이 명사가 쓰였다. 스위프트에서는 영어의 문법적 맥락을 고려하는 전자의 방식이 더 좋은 방식이다.