Практическая работа № 13

Приведение типов,Any и AnyObject

Цель работы:

1. Изучить и научиться работать с типами Any и AnyObjectв Swift.
2. Изучить особенности применения.
3. Научиться использовать приведение типовв программе.

Краткие теоретические сведения

**Приведение типов** - это способ проверить тип экземпляра и/или способ обращения к экземпляру так, как если бы он был экземпляром суперкласса или подкласса откуда-либо из своей собственной классовой иерархии.

Приведение типов в Swift реализуется с помощью операторов is и as. Эти два оператора предоставляют простой и выразительный способ проверки типа значения или преобразование значения к другому типу.

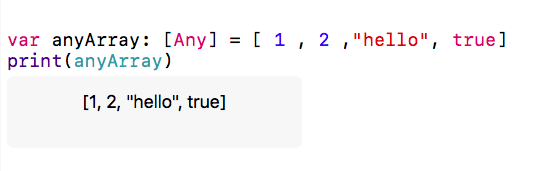
Any и AnyObject

**Any** и **AnyObject** - это два специальных типа в Swift, которые используются для работы с неспецифическими типами.

Согласно документации Apple Swift,

1. ***Any****может представлять экземпляр любого типа вообще, включая типы функций и дополнительные типы.*
2. ***AnyObject****может представлять экземпляр любого типа класса.*

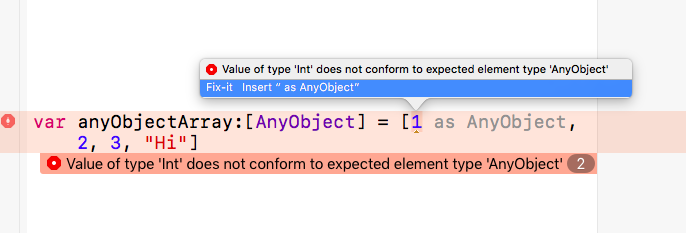
Поработаем с ними playgraund.



***Any*пример**

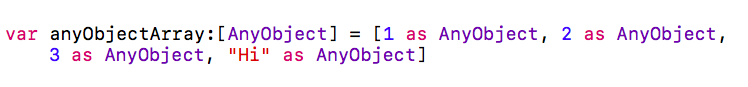
***Any***позволяет работать со смесью разных типов, включая функциональные и неклассовые типы, такие как Int, String и Bool. Согласно документации, элементы в этом массиве являются структурами, которые являются типами значений, поэтому теоретически **AnyObjec** t не должен работать в этих случаях.

Включим Strings и Ints, которые являются типами значений в Swift, используя **AnyObject** .



**Ошибка AnyObjectArray при включении типов структур**

Как и ожидалось, компилятор выдает ошибку, говоря, что элементы не соответствуют типу **AnyObject** в массиве.



**Элементы, приведенные в AnyObject**

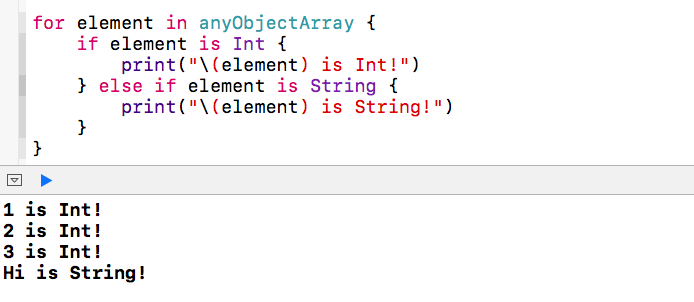
Можно использовать **AnyObject** для Ints и Strings, явно **приведя** каждый элемент к **AnyObject**.



**Печать anyObjectArray**

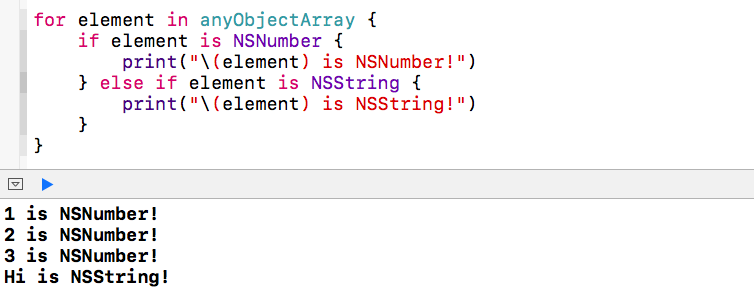
Элемент **Hi** явно выглядит как строка, но в нем нет кавычек, как у обычного значения String в Swift!

Распечатаем каждый элемент, используя цикл for-in, чтобы проверить его фактический тип, а не приведенный тип **AnyObject** .



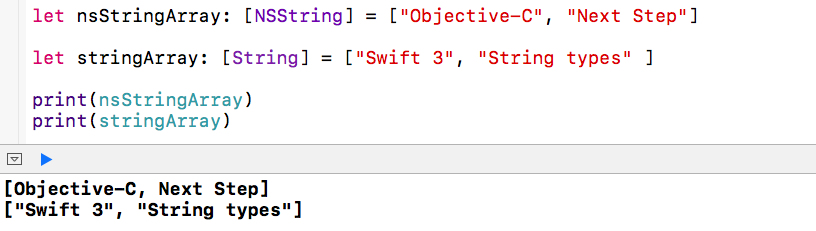
**Проверка типов элементов в anyObjectArray 1**

Это тип String! Тогда как его передать в **AnyObject** ? Опять же, строки в Swift - это структуры, а не типы классов. Таким образом, теоретически я не могу использовать их как **AnyObject** .

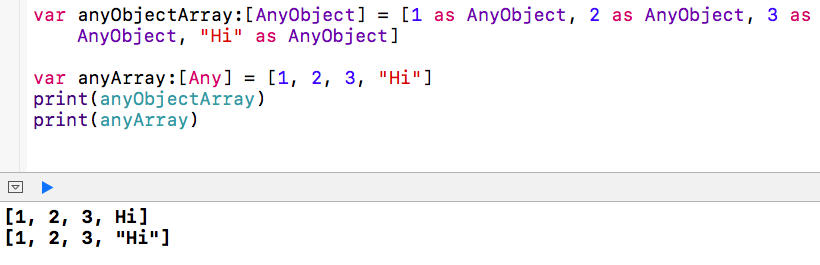


**Проверка типов элементов в anyObjectArray 2**

**Hi** - это тоже NSString, а числовые элементы - это NSNumber! И… это ссылочные типы в Objective-C!



**Печать массива NSString и массива строк**



**Привет, без кавычек в консоли как NSString**

Элементы, приведенные к **AnyObject** в массиве, теперь являются типами классов Objective-C: NSString и NSNumber.

В рамках взаимодействия с Objective-C Swift предлагает удобные и эффективные способы работы с фреймворками Cocoa. Swift автоматически преобразует некоторые типы Objective-C в типы Swift, а некоторые типы Swift - в типы Objective-C.Типы, которые можно преобразовать между Objective-C и Swift, называются **мостовые типы**.

**Везде, где вы можете использовать мостовой ссылочный тип Objective-C, вы можете использовать вместо него тип значения Swift.**Это позволяет вам использовать функциональные возможности, доступные в реализации ссылочного типа, естественным образом для кода Swift. По этой причине вам почти никогда не нужно использовать мостовой ссылочный тип непосредственно в вашем собственном коде. Фактически, когда код Swift импортирует API-интерфейсы Objective-C, импортер заменяет ссылочные типы Objective-C их соответствующими типами значений. Аналогичным образом, когда код Objective-C импортирует API Swift, импортер также заменяет типы значений Swift их соответствующими ссылочными типами Objective-C ».

Другими словами, компилятор делает все возможное, чтобы быть гибким в обработке таких типов за счет автоматического преобразования и соединения, предотвращая при этом легкое падение нашего приложения.

Таккогда мы на самом деле используем **AnyObject** ? Как указано в документации Apple, **AnyObject** можно использовать для работы с объектами, которые являются производными от Class, но не имеют общего корневого класса.

Но обязательно ли использовать это в нашем коде?**нет** .

Apple говорит:

[В Swift 3](https://developer.apple.com/swift/blog/?id=39) тип id в Objective-C теперь сопоставляется с типом Any в Swift, который описывает значение любого типа, будь то класс, перечисление, структура или любой другой тип Swift. **Это изменение делает API-интерфейсы Objective-C более гибкими в Swift, поскольку типы значений, определенные в Swift, могут передаваться в API-интерфейсы Objective-C и извлекаться как типы Swift, устраняя необходимость в ручных «блочных» типах.**

Эти преимущества также распространяются на коллекции: **типы** коллекций **Objective-C NSArray, NSDictionary и NSSet, которые ранее принимали только элементы AnyObject, теперь могут содержать элементы любого типа.**Для хешированных контейнеров, таких как Dictionary и Set, есть новый тип AnyHashable, который может содержать значение любого типа, соответствующего протоколу Swift Hashable.

Кажется, что один **Any** отлично работает в соединении этих двух языков в Swift 3 без необходимости использования **AnyObject** !

Так что же было главной причиной этих изменений?

Своими словами Apple поясняет:

Swift 3 взаимодействует с API Objective-C более мощным способом, чем предыдущие версии. Например, Swift 2 сопоставил тип id в Objective-C с типом AnyObject в Swift, который обычно может содержать только значения типов классов. Swift 2 также предоставил неявные преобразования в AnyObject для некоторых типов значений моста, таких как String, Array, Dictionary, Set и некоторых чисел, в качестве удобства, чтобы собственные типы Swift можно было легко использовать с API-интерфейсами Какао, которые ожидали NSString, NSArray, или другие классы-контейнеры из Foundation. [**Эти преобразования несовместимы с остальным языком, что затрудняет понимание того, что именно можно использовать в качестве AnyObject, что приводит к ошибкам**](https://developer.apple.com/swift/blog/?id=39)**.**

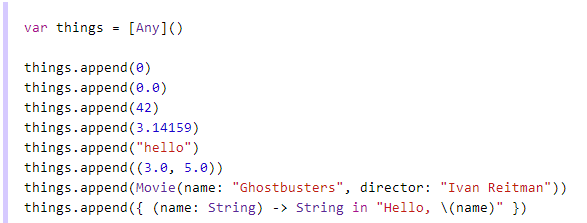
Однако можно было бы настаивать на том, что мы, разработчики iOS, всегда должны быть как можно более конкретными с точки зрения использования типов в коде.

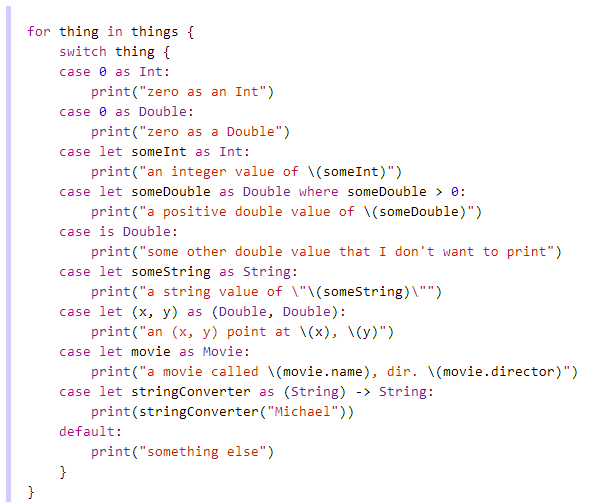
Действительно, Apple рекомендует:

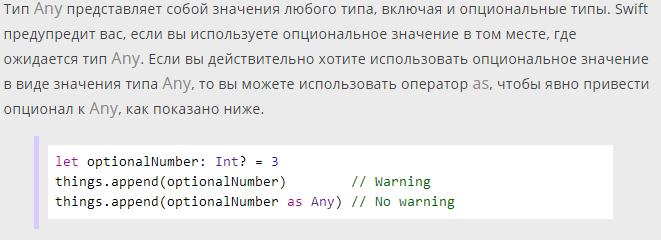
Используйте Any и AnyObject только тогда, когда вам явно необходимы поведение и возможности, которые они предоставляют. [**Всегда лучше конкретизировать типы, с которыми вы ожидаете работать в своем коде** .](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/TypeCasting.html)

Подумайте об этом сценарии: мы работаем с числом 12,5 в Swift. В этом случае мы бы специально заявили, что это тип Double или Float, а не объявляли бы его типа **Any** . Таким образом, мы можем получить удобный доступ к различным свойствам или методам, доступным для этого конкретного типа. В этом контексте мы бы использовали **AnyObject** для типов классов, потому что они немного более специфичны, чем **Any** . Но опять же, использование **AnyObject** - это всего лишь вариант.

**Пример использования Any с различными типами**







Задание для самостоятельного выполнения

1. Создайте на основе примера выше массив с информацией об одном экземпляре из Вашего задания
2. Числовое значение увеличьте на 10
3. Название и прочие характеристики напечатайте в консоле
4. Отдельно выведите весь массив

Варианты

1. Автобус (номер маршрута, название начальной и конечной остановки, количество остановок по маршруту)
2. Книга (название книги, автор, издательство, количество страниц)
3. Файл (имя, время создания, дата создания, размер)
4. Спортсмен (ФИО, вид спорта, сколько лет занимается)
5. Студент (ФИО, группа, средний балл зачетки)
6. Фирма (название, вид услуг, годовой оборот)
7. Компьютер (производитель, частота процессора, количество ядер, объем памяти)
8. Спорт (вид спорта, количество занимающихся в городе, средняя стоимость комплекта инвентаря)
9. Автомобиль (марка, фирма-производитель,год выпуска, пробег)
10. Статья (название, автор, название журнала, год и месяц издания)