Урок 1

Введение в Swift

Краткая история Swift

После непрерывной работы над улучшениями Objective-C, 2 июня 2014 года на конференции WWDC компания Apple представила новый язык программирования – Swift, вместе с бесплатным руководством по использованию языка объёмом в 500 страниц, доступным на сервисе «iBook Store».

На той конференции Крейг Федериги (старший вице-президент компании Apple по программному обеспечению) сказал «Эффективность и простота Swift дадут молодым программистам стимулы к обучению, к тому же теперь они смогут распространять свои идеи повсюду: от мобильных устройств до облачных систем».

Разработка текущего варианта языка Swift началась в 2010 году. Swift заимствовал идеи из многих языков программирования, таких как Rust, Haskell, Ruby, Python, C# и пр., в т.ч. и Objective-C, который так же является языком программирования от Apple.

Версия Swift 1.0 была выпущена 9 сентября 2014 года, вместе с «Gold Master» версией Xcode 6.0.

8 июня 2015 года компания Apple объявила о выпуске новой версии Swift 2.0, которая получила более высокую производительность, новое API обработки ошибок и улучшения синтаксиса языка, а уже 3 декабря 2015 года была вы-

пущена новая бета-версия Swift 3.0 с поддержкой операционных систем OS X, iOS и Linux!.

Текущая версия Swift вышла 19 сентября 2017 года. Тогда он обновился до версии 4.0. На данный момент уже доступна версия 4.2.

Почему Swift?

Интегрированная среда разработки

Во первых это интегрированная среда разработки Xcode, которая является полноценным инструментом полного цикла для разработки приложения.##### Высокоуровневый язык и ясность кода

Кроме того, Swift это высокоуровневый язык программирования. Это значит, что он более приближен к обычному разговорному английскому языку. Его гораздо легче изучать, т.к. у него простой синтаксис, призванный облегчить чтение и написание кода. От сюда вытекает ясность кода.

Уменьшение количества кода

Количество строк кода, необходимых для реализации той или иной задачи в Swift меньше, чем у его предшественника - Objective C, а это в купе с тем, что Swift строго типизированный язык, в свою очередь позволяет избежать многих досадных ошибок и сохранить ясность кода.

Еще одно улучшение заключается в том, что вы больше не найдете разделения на интерфейс и реализацию. Это позволяет сократить количество файлов в проекте, что в свою очередь упрощает навигацию по нему.

Скорость и безопасность

В настоящее время Apple продолжает улучшать производительность языка, включая скорость выполнения логики приложений. При этом язык достаточно

молодой и находится на стадии своего развития, а значит с развитием самого языка, его производительность будет только расти.

Swift изначально разрабатывался, как безопасный язык. В то время, как Objective-C может вызвать метод, содержащий переменную с указателем на nil, что в свою очередь вызовет падение приложения, Swift благодаря опциональным типам выдаст ошибку ещё на стадии компиляции. Кроме того, Swift обладает самой строгой типизацией переменных по сравнению с другими языками. Предельно строгая типизация требует точного соответствия типам, поэтому пока все переменные не будут приведены к нужным типам, приложение не соберется. Это искусственная защита - создана для того, чтобы не дать возможности разработчику сделать ошибки и, как следствие, повысить качество своего кода.

Динамические iOS библиотеки

Возможно одним из самых больших преимуществ Swift, является отказ от статичных iOS библиотек в пользу динамических. Что это означает? Статические библиотеки обновляются только при крупных обновлениях самой операционной системы, таких как переход с одной версии iOS на другую. Динамические же библиотеки - это фрагменты кода, которые могут быть подключены непосредственно к приложению.

На практике это означает, что обновления самого языка программирования не зависит от обновления iOS. Таким образом, язык сам по себе способен развиваться гораздо быстрее, чем платформа iOS. Это позволяет использовать в своей работе всегда самые современные версии языка.

Использование динамических библиотек позволяет уменьшить начальный размер приложений. Благодаря этому приложения полностью загружаются в память, а весь внешний код выполняется только тогда, когда это необходимо.

Этот шаг позволяет минимизировать начальное время ожидания при запуске приложения.

Управление памятью

Автоматическое и высокопроизводительное управление памятью было проблемой, которую Apple смогли решить и доказали, что это может повысить производительность. Особенно это касается тактильных устройств таких как: iPhone, Apple Watch или IPad (где задержка в работе воспринимается, как разочарование и заставляет пользователя думать, что приложение не работает).

Swift использует automatic reference counting (автоматический подсчет ссылок) для отслеживания и управления памятью вашего приложения. В большинстве случаев это означает, что управление памятью "просто работает"и вам не нужно думать о самостоятельном управлении ею. Automatic reference counting или ARC автоматически освобождает память, которая использовалась экземплярами класса, когда эти экземпляры больше нам не нужны. Это в свою очередь предотвращает обширные утечки памяти при разработке приложений.

Таким образом, управление памятью позволяет разработчикам больше не беспокоится о ней для каждого цифрового объекта, а сосредоточится на основной логике приложения.

Почему надо изучать Swift?

По данным Stack Overflow этот язык получил первое место, как один из самых популярных языков программирования в 2015 году, а так же второе место в 2016-м.

В свою очередь, эта популярность генерирует массу контента, посвященного разработке на Swift. Таким образом, на данный момент существует огромное количество ресурсов, связанных с этой темой, благодаря чему, вы практически всегда сможете найти ответ, на интересующий вас вопрос, если таковой возникнет.

Говоря об инструментах, способствующих более удобному и быстрому написанию кода, стоит отметить, что количество фреймворков, построенных для Swift, растет из года в год.

Playground, как средство программирования с визуальной обратной связью. Кроме того, что это помогает развитию алгоритмов, использующих встроенные визуализации данных, Playground является действительно очень полезным инструментом, особенно для начинающих программистов, когда нужно с чем-то поэкспериментировать, что-то проверить или попробовать.

Swift доступный и полнофункциональный язык программирования для создания iOS и macOS приложений и не только.

Константы и переменные

Переменные

Переменная это объект, который хранит в себе определенные данные. В качестве данных могут выступать какие то значения или другие объекты. Все объекты хранятся в памяти и для каждого из них в ней должна быть выделена соответствующая ячейка. Каждая ячейка памяти уникальна и состоит из набора букв и цифр. Создавая какой либо объект, в нашем случае переменную, программист присваивает ей имя и значение. Далее программа сохраняет созданный объект в определенную ячейку памяти, а для дальнейшего его использова-

ния достаточно обратиться к этой ячейке по имени переменной. По сути, создавая переменную, разработчик присваивает имя для ячейки памяти, в которой будет хранится заданное для этой переменной значение. Переменные объявляются при помощи ключевого слова `var` сокращенного от *variable*.

Особенностью переменной является то, что её можно изменить. Т.е. мы можем изменить нашу переменную, просто, присвоив ей новое значение.

Константы

Единственным но очень важным отличием константы от переменной, является то, что константа не допускает возможности изменить, ранее присвоенное ей значение. В Swift константа объявляется ключевым словом `let`. Лучшей практикой для хранения значений является использование констант, если только это значение не должно изменятся в будущем.

Базовые типы

1. `String` (текстовая строка);
2. `Int` (целое число);
3. `Float` (32-битное число с плавающей точкой);
4. `Double` (64-битное число с плавающей точкой для более длинных переменных);
5. `Bool` (логическое значение «true» или «false»).

Swift является строго типизированным языком программирования. Это значит, что каждое значение переменной или константы или любого другого объекта, который вы создаете имеет свой тип. Соответственно если мы работаем с объектом определенного типа, то и значения которые мы можем присвоить этому

объекту должны быть того же самого типа. В нашем коде не может быть ни одного объекта, который бы не имел определенного типа. Даже если мы явно не говорим какому типу должен принадлежать наш объект, компилятор определяет тип самостоятельно. Поэтому, когда мы присваиваем нашей переменной значение и записываем его в кавычках, компилятор понимает, что данная переменная имеет тип `String`. Если при объявлении нашей переменной мы присвоим ей числовое значение без кавычек и без разделительной запятой, то данная переменная будет иметь тип Int и т.д.

Строковые типы

Строки могут быть как очень длинными, так и короткими, а могу и вовсе быть пустыми. Если мы хотим создать строковую переменную, но при этом без первоначального значения, тогда после имени нашей переменной нужно поставить двоеточие и указать тип переменной: `var myFriend: String`.

Если нужно создать пустую текстовую переменную, но к которой можно обращаться даже, если ей не присвоено ни какого значения, то лучше всегда создавать переменную с пустой строкой. В этом случае тип указывать явно не нужно, компилятор и так поймет, что наша переменная имеет тип `String`: `var myFriend = ""`.

Swift всегда стремится знать заранее, какому типу данных соответствует определение каждой новой переменной и константы. И это хорошо с точки зрения разработчика, ведь в таком случае за него уже реализован механизм устранения возможных ошибок, связанных с несоответствием вводимого значения заданному в определении типу данных.

Числовые типы

Переменные с типом `Int` могут содержать в себе целые числа, т.е. числа без дробной части. Это могут быть, как положительные, так и отрицательные числа, а так же 0: `var age = 25`

Типы `Float` и `Double` в Swift используются для хранения чисел с плавающей запятой. Они отличаются между собой только количеством отображаемых цифр в десятичной части числа. Если вы создадите переменную или константу без явного указания типа и присвоите ей числовое значение с плавающей точкой, то по умолчанию ваша переменная будет иметь тип `Double`.

Основное отличие между `Float` и `Double` заключается в двойной точности хранения и отображения хранимых чисел.

Логический тип

Для хранения логических значений в Swift используются переменные или константы с типом Bool. Это так называемые булево значения, которые могут принимать только `true` или `false`. Выражения сравнения, такие, как например **`x > 0`**, получают логическое значение, зависящее от истинности или ложности выражения и в зависимости от этого определятся дальнейший ход выполнения программы.