Основы Swift и введение в разработку на iOS

Основы Swift

Swift - это мощный и интуитивно понятный язык программирования, разработанный Apple для создания приложений для iOS, macOS, watchOS и tvOS. Swift сочетает в себе лучшие элементы современных языков программирования и обладает мощной системой типов и управления памятью.

Основные особенности Swift:

- **Безопасность типов.**
- **Управление памятью без использования указателей.**
- **Функциональное программирование.**
- **Современный синтаксис.**

Swift определяет большие классы распространенных ошибок программирования, принимая современные шаблоны программирования:

- **Переменные всегда инициализуются перед использованием.**
- **Индексы массива проверяются на наличие ошибок вне граничных.**
- **Неполные числа проверяются на переполнение.**
- **Опционально гарантирует, что nil значения обрабатываются явно.**
- **Память управляется автоматически.**
- **Обработка ошибок позволяет контролировать восстановление после неожиданных сбоев.**

Swift является результатом последних исследований языков программирования в сочетании с многолетним опытом создания платформ Apple. Именованные параметры выражены в чистом синтаксисе, что делает API в Swift еще более простыми в чтении и обслуживании. Еще лучше, вам даже не нужно набирать точку с запятой. Выведенные типы делают код чище и менее склонен к ошибкам, в то время как модули устраняют заголовки и предоставляют пространства имен. Чтобы наилучшим образом поддерживать международные языки и эмодзи, строки являются неправильными Unicode и используют кодировку на основе UTF-8 для оптимизации производительности для широкого спектра сценариев использования. Память управляется автоматически с помощью жесткого, детерминированного подсчета ссылок, сводя использование памяти к минимуму без накладных расходов на сбор мусора. Вы даже можете писать параллельный код с помощью простых встроенных ключевых слов, которые определяют асинхронное поведение, делая ваш код более читабельным и менее подверженным ошибкам.

```
struct Player {
    var name: String
    var highScore: Int = 0
    var history: [Int] = []

    init(_ name: String) {
        self.name = name
    }
}

var player = Player("Tomas")
```

Объявите новые типы с помощью современного, простого синтаксиса. Предоставьте значения по умолчанию для свойств экземпляра и определите пользовательские инициализаторы.

```swift
extension Player {
    mutating func updateScore(_ newScore: Int) {
        history.append(newScore)
        if highScore < newScore {
            print("\(newScore)! A new high score for \(name)! 🎉")
            highScore = newScore
        }
    }
}

player.updateScore(50)
// Prints "50! A new high score for Tomas! 🎉"
// player.highScore == 50
```

Добавьте функциональность к существующим типам с помощью расширений и сократите шаблонный код с помощью пользовательских строковых интерполяции.

```swift
extension Player: Codable, Equatable {}

import Foundation
let encoder = JSONEncoder()
try encoder.encode(player)

print(player)
// Prints "Player(name: "Tomas", highScore: 50, history: [50])”
```

Быстро расширяйте свои пользовательские типы, чтобы воспользоваться мощными языковыми функциями, такими как автоматическое кодирование и декодирование JSON.

```swift
extension Player: Codable, Equatable {}

import Foundation
let encoder = JSONEncoder()
try encoder.encode(player)

print(player)
// Prints "Player(name: "Tomas", highScore: 50, history: [50])”
```

Быстро расширяйте свои пользовательские типы, чтобы воспользоваться мощными языковыми функциями, такими как автоматическое кодирование и декодирование JSON.

Эти дальновидные концепции приводят к тому, что язык является веселым и простым в использовании.
Swift имеет много других функций, чтобы сделать ваш код более выразительным:

- **Дженерики, которые мощные и простые в использовании**
- **Расширения протокола, которые делают написание общего кода еще проще**
- **Первоклассные функции и легкий синтаксис закрытия**
- **Быстрая и краткая итерация по диапазону или коллекции**
- **Выемки и несколько возвращаемых значений**
- **Структуры, поддерживающие методы, расширения и протоколы**
- **Перемки могут иметь полезную нагрузку и поддерживать сопоставление шаблонов**
- **Функциональные шаблоны программирования, например, карта и фильтр**
- **Встроенная обработка ошибок с помощью try / catch / throw**

Разработано для безопасности

Swift устраняет целые классы небезопасного кода. Переменные всегда инициализируются перед использованием, массивы и целые числа проверяются на переполнение, память управляется автоматически, а эксклюзивный доступ к памяти предохраняет множество ошибок программирования. Синтаксис настроен, чтобы было легко определить ваше намерение - например, простые трехсимвольные ключевые слова определяют переменную (var) или константу (let). И Swift в значительной степени использует типы значений, особенно для часто используемых типов, таких как массивы и словари. Это означает, что когда вы делаете копию чего-то такого типа, вы знаете, что это не будет изменено в другом месте.

Еще одна функция безопасности заключается в том, что по умолчанию объекты Swift никогда не могут быть нулевыми. Фактически, компилятор Swift остановит вас от попытки создать или использовать нулевой объект с ошибкой во время компиляции. Это делает написание кода намного чище и безопаснее, а также предотвращает огромную категорию сбоев во время выполнения в ваших приложениях. Тем не менее, есть случаи, когда ноль является действительным и уместным. Для этих ситуаций Swift имеет инновационную функцию, известную как опции. Необязательный может содержать ноль, но синтаксис Swift заставляет вас безопасно справляться с ним, используя ? синтаксис, чтобы указать компилятору, что вы понимаете поведение и будете безопасно с ним обращаться.

```swift
let players = getPlayers()

// Sort players, with best high scores first
let ranked = players.sorted(by: { player1, player2 in
    player1.highScore > player2.highScore
})

// Create an array with only the players' names
let rankedNames = ranked.map { $0.name }
// ["Erin", "Rosana", "Tomas"]
```

Выполняйте мощные пользовательские преобразования с помощью оптимизированных затворов.

```swift
if let bestPlayer = players.highestScoringPlayer() {
    recordHolder = """
        The record holder is \(bestPlayer.name),\
        with a high score of \(bestPlayer.highScore)!
        """
} else {
    recordHolder = "No games have been played yet."
}
print(recordHolder)
// The record holder is Erin, with a high score of 271!

let highestScore = players.highestScoringPlayer()?.highScore ?? 0
// highestScore == 271
```

Такие функции, как опциональное связывание, опциональное связывание и нулевое слияние, позволяют вам работать безопасно и эффективно с дополнительными значениями.

Быстрый и мощный

С самого начала своего создания Swift был создан, чтобы быть быстрым. Используя невероятно высокопроизводительную технологию компилятора LLVM, код Swift преобразуется в оптимизированный машинный код, который получает максимальную отдачу от современного оборудования. Синтаксис и стандартная библиотека также были настроены таким образом, чтобы самый очевидный способ написания вашего кода также работал наилучшим образом, независимо от того, работает ли он в часах на вашем запястье или на кластере серверов.

Swift является преемником языков C, C++ и Objective-C. Он включает в себя низкоуровневые примитивы, такие как типы, управление потоком и операторы. Он также предоставляет объектно-ориентированные функции, такие как классы, протоколы и дженерики, предоставляя разработчикам Cocoa и Cocoa Touch производительность и мощность, которые они требуют.

Open Source

Swift разрабатывается открыто на Swift.org, с исходным кодом, трекером ошибок, форумами и регулярными сборками разработки, доступными для всех. Это широкое сообщество разработчиков, как внутри Apple, так и сотни внешних участников, работает вместе, чтобы сделать Swift еще более удивительным. Существует еще более широкий спектр блогов, подкастов, конференций и встреч, где разработчики в сообществе делятся своим опытом о том, как реализовать большой потенциал Swift.

Кроссплатформенный

Swift уже поддерживает все платформы Apple и Linux, а члены сообщества активно работают над переносом на еще большее количество платформ. С помощью SourceKit-LSP сообщество также работает над интеграцией поддержки Swift в широкий спектр инструментов для разработчиков. Мы рады увидеть больше способов, которыми Swift делает программное обеспечение более безопасным и быстрым, а также делает программирование более веселым.

Swift для сервера

В то время как Swift питает множество новых приложений на платформах Apple, он также используется для нового класса современных серверных приложений. Swift идеально подходит для использования в серверных приложениях, которые нуждаются в безопасности во время выполнения, производительности компиляции и небольшом объеме памяти. Чтобы направить направление Swift для разработки и развертывания серверных приложений, сообщество сформировалоСервер Swiftрабочая группа. Первым продуктом этих усилий был SwiftNIO, кросс-платформенная асинхронная платформа сетевых приложений, управляемая событиями, для высокопроизводительных серверов и клиентов протоколов. Он служит основой для создания дополнительных серверно-ориентированных инструментов и технологий, включая ведение журнала, метрики и драйверы баз данных, которые находятся в активной разработке.
Чтобы узнать больше о сообществе Swift с открытым исходным кодом иСервер Swiftрабочая группа, посетите Swift.org.

Установка и настройка Xcode

Xcode - это интегрированная среда разработки (IDE) от Apple, предназначенная для создания приложений для всех продуктов Apple.

Шаги установки Xcode:

1. **Перейдите в App Store на вашем Mac.**
2. **В поисковой строке введите "Xcode" и нажмите Enter.**
3. **Нажмите "Установить".**
4. **После установки запустите Xcode и следуйте инструкциям на экране для настройки разработческой среды.**

Структура iOS-приложения

Каждое iOS-приложение имеет определенную структуру. Основные компоненты:

1. **AppDelegate:** Это точка входа в ваше приложение. Он отвечает за инициализацию основных компонентов приложения.
2. **Storyboard:** Графический интерфейс для создания пользовательского интерфейса вашего приложения.
3. **View Controllers:** Контроллеры управляют содержимым экрана, обрабатывают взаимодействия пользователя и управляют переходами между экранами.
4. **Models:** Объекты или структуры данных, которые представляют информацию в вашем приложении.
5. **Assets:** Ресурсы, такие как изображения и звуки.

Игровые площадки и Read-Eval-Print-Loop (REPL)

Как и Swift Playgrounds для iPad и Mac, игровые площадки в Xcode делают написание кода Swift невероятно простым и веселым. Введите строку кода, и результат появится немедленно. Затем вы можете быстро просмотреть результат сбоку вашего кода или закрепить этот результат прямо ниже. Представление результатов может отображать графику, списки результатов или графики значения с течением времени. Вы можете открыть помощник временной шкалы, чтобы наблюдать, как развивается и анимируется сложное представление, что отлично подходит для экспериментов с новым кодом пользовательского интерфейса или для воспроизведения анимированной сцены SpriteKit во время кодирования. Когда вы усовершенствовали свой код на игровой площадке, просто переместите этот код в свой проект. Swift также является интерактивным, когда вы используете его в терминале или в консоли отладки Xcode LLDB. Используйте синтаксис Swift для оценки и взаимодействия с запущенным приложением или напишите новый код, чтобы увидеть, как он работает в среде, похожей на скрипты.

Менеджер пакетов

Swift Package Manager - это единый кросс-платформенный инструмент для создания, запуска, тестирования и упаковки ваших библиотек и исполняемых файлов Swift. Пакеты Swift - это лучший способ распространения библиотек и исходного кода среди сообщества Swift. Конфигурация пакетов написана на самом Swift, что упрощает настройку целей, объявление продуктов и управление зависимостями пакетов. Пакеты Swift также могут включать пользовательские команды, которые помогают создавать ваши проекты и предоставляют дополнительные инструменты. Сам менеджер пакетов Swift на самом деле создан с помощью Swift и включен в проект с открытым исходным кодом Swift в качестве пакета.

Совместимость Objective-C и C++

Вы можете создать совершенно новое приложение с помощью Swift сегодня или начать использовать код Swift для реализации новых функций и функциональности в вашем приложении. Код Swift сосуществует вместе с вашими существующими файлами Objective-C и C++ в одном проекте, с доступом к вашим API Objective-C и C++, что упрощает его принятие.

Задание

- **Изучите документацию Apple по основам Swift. https://www.apple.com/ru/swift/**

- **The Swift Programming Language (5.9) https://docs.swift.org/swift-book/documentation/the-swift-programming-language/**

Контрольные вопросы

- **Что такое Swift?**
- **Чем Swift отличается от Objective-C?**
- **Что такое безопасность типов в Swift?**
- **Что такое Optionals в Swift?**
- **Какие основные структуры данных поддерживаются в Swift?**
- **Как в Swift обрабатываются ошибки?**
- **Что такое замыкания (closures) в Swift?**
- **Что такое extensions в Swift?**
- **Каковы основные особенности управления памятью в Swift?**
- **Что такое протоколы в Swift и как они используются?**