Содержание (сводка)

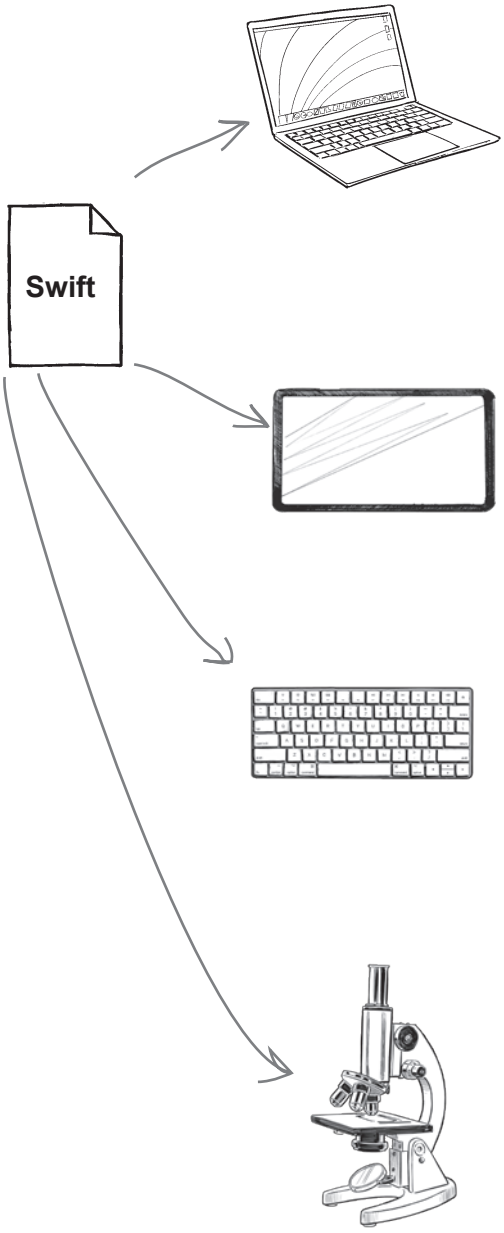
Содержание (настоящее)

Введение

Ваш мозг и Swift. Вы пытаетесь изучить что-то новое, а ваш мозг хочет оказать вам услугу и как можно быстрее забыть выученное. Он думает: «Лучше оставить место для чего-то поважнее, например, от каких диких животных стоит держаться подальше или почему на сноуборде не стоит кататься нагишом». Так как же заставить ваш мозг думать, что ваша жизнь зависит от знания Swift?

1 Знакомство со Swift

Приложения, системы и не только!



Swift — язык программирования, на который можно положиться. Вам не будет стыдно познакомить с ним вашу семью. Он безопасен, надежен, быстр, доступен и несложен. И хотя Swift получил наибольшую известность **как язык программирования для платформ Apple**, таких как iOS, macOS, watchOS и tvOS, проект с открытым кодом Swift также работает в Linux и Windows и постепенно набирает популярность как язык системного программирования, а также как серверный язык. На нем можно строить все что угодно, от мобильных приложений до игр, веб-приложений, фреймворков. Итак, за дело!

2 По имени Swift

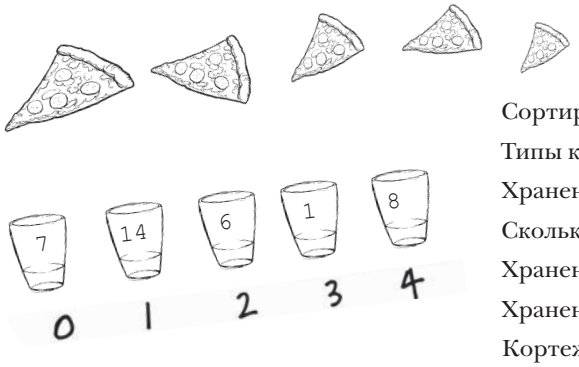
Swift на практике

Вы уже знаете азы Swift. Но пришло время изучить основные элементы языка более подробно. Вы узнали достаточно, чтобы вас воспринимали серьезно, пора употребить новые знания на практике. Мы применяем Playgrounds для написания кода, использования команд, выражений, переменных и констант — основных структурных элементов Swift. В этой главе мы заложим основу вашей будущей карьеры программиста Swift. Вы освоите систему типов Swift и изучите основы представления текста в строковом виде. Не будем терять времени — еще чуть-чуть, и вы начнете писать код Swift.

3 Коллекции и управление

Зацикленные на данных

Вы уже знаете о выражениях, операторах, переменных, константах и типах Swift. Пришло время собрать воедино все, что говорилось ранее, и на этой основе исследовать некоторые более сложные структуры данных и операторы Swift: **коллекции** и **управляющие команды**. В этой главе мы поговорим о сохранении коллекций данных в переменных и константах, о структурировании данных, обработке данных и работе с данными с использованием управляющих команд. Позднее в книге будут рассмотрены другие способы сбора и структурирования данных, а пока начнем с **массивов**, **множеств** и **словарей**.



Функции и перечисления

4 Повторное использование кода

Функции в языке Swift позволяют упаковать некоторое поведение или единицу работы в блок кода, который может вызываться из других частей вашей программы. Функции могут быть **автономными**, а могут определяться как часть **класса**, **структуры** или **перечисления**, где они обычно называются **методами**. При помощи функций можно разбить сложные задачи на меньшие части, более удобные и легкие в тестировании. Функции занимают центральное место в формировании структуры программ в Swift.

5 Замыкания

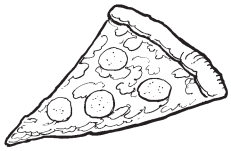
Необычные гибкие функции

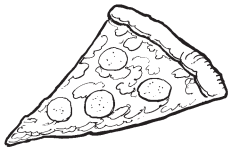
Функции полезны, но иногда нужно больше гибкости. Swift позволяет использовать **функцию как тип** — так же, как вы используете целое число или строку. **Это означает, что вы можете создать функцию и присвоить ее переменной**. После того как функция будет присвоена переменной, ее можно вызывать через эту переменную или передавать другим функциям в параметре. Когда вы создаете и используете функцию подобным образом, это называется **замыканием**. Замыкания очень полезны, потому что они способны сохранять **ссылки** на константы и переменные из контекста, в котором они были определены. Это называется **замыканием по значению**, отсюда и название.

6 Структуры, свойства и методы

Типы, определяемые пользователем, и не только

При работе с данными часто требуется определять собственные виды данных. Структуры, также нередко обозначаемые ключевым словом **struct,** позволяют создавать **типы данных, определяемые пользователем** (подобно тому, как String и Int являются типами данных), посредством **объединения других типов**. Использование структур для представления данных, с которыми работает ваш код Swift, позволяет отступить на шаг и подумать над взаимодействием данных, передаваемых в вашем коде. **Структуры могут содержать переменные и константы** (внутри структур они называются **свойствами**) **и функции** (называемые **методами**). Добавим немного порядка в ваш мир и займемся углубленным изучением структур.





7 Классы, акторы и наследование

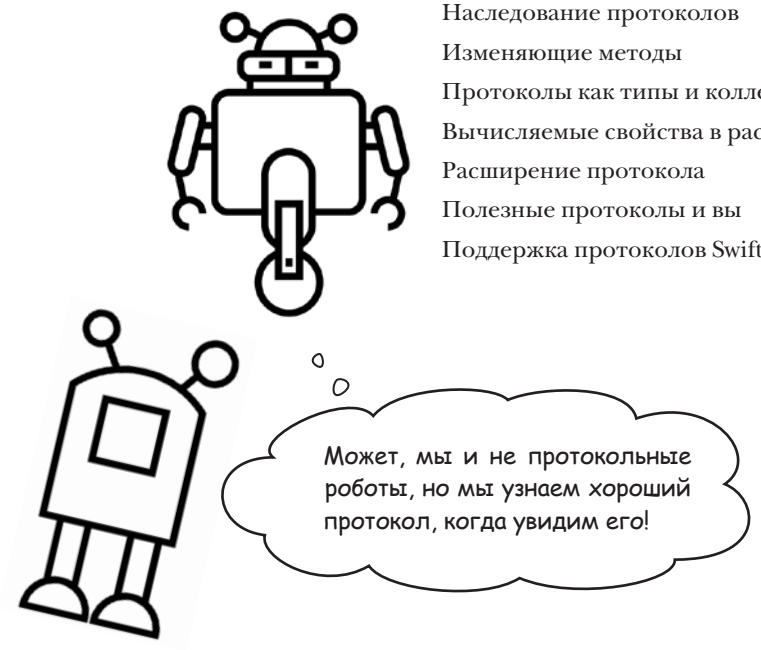
О пользе наследования

Структуры показали, насколько полезным может быть построение типов, определяемых пользователем. Но у Swift в запасе есть и другие средства, включая *классы*. Классы похожи на структуры: они позволяют создавать *новые типы данных, содержащие свойства и методы*. Кроме того что они являются *ссылочными типами,* то есть экземпляры конкретного класса совместно используют одну копию своих данных (в отличие от структур, которые являются типами-значениями, чьи данные копируются), *классы поддерживают наследование*. Наследование позволяет построить один класс на базе другого класса.

8 Протоколы и расширения

Протокольные церемонии

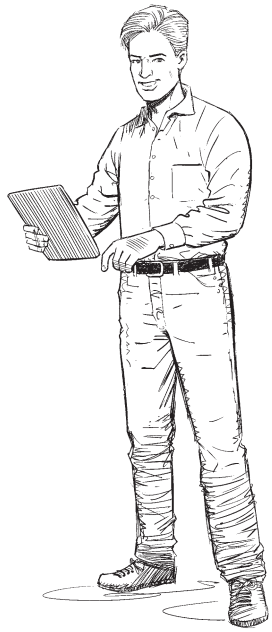
Вы все знаете о классах и наследовании, но у Swift еще есть немало средств структурирования программ. Знакомьтесь: протоколы и расширения. **Протоколы в Swift позволяют определить шаблон с перечнем методов и свойств,** необходимых для некоторой цели или некоторого блока функциональности. **Протокол включается** классом, структурой или перечислением, в которых содержится его фактическая реализация. Типы, которые предоставляют необходимую функциональность, называются **поддерживающими** этот протокол. **Расширения** позволяют легко **добавлять новую функциональность** в существующие типы.



9 Опциональные типы, распаковка, обобщение и другое

Неизбежные опциональные типы

Обработка несуществующих данных может быть весьма непростым делом. К счастью, в Swift для этого существует решение: опциональные типы. В Swift опциональный тип позволяет работать со значением или выполнить действия при отсутствии значения. Это одно из многих проявлений безопасности при проектировании Swift. Ранее вы уже встречались с опциональными типами в коде, а теперь мы изучим их более подробно. Опциональные типы улучшают безопасность Swift, потому что с ними снижается риск написания кода, который перестает работать при отсутствии данных, или возврата значения, которое в действительности значением не является.

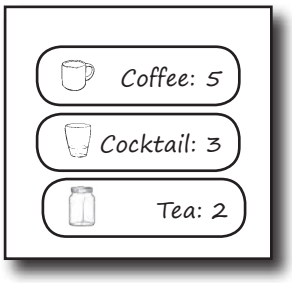


10 Знакомство со SwiftUI

Пользовательские интерфейсы

Пришло время применить на практике все приемы, возможности и компоненты Swift, о которых вы узнали в книге: мы займемся построением пользовательских интерфейсов. В этой главе мы сведем все воедино для построения первого настоящего пользовательского интерфейса. Он будет строиться на основе **SwiftUI, UI-фреймворка для платформ Apple**. Мы по-прежнему будем использовать Playgrounds (по крайней мере на первом этапе), но все, что здесь будет делаться, заложит фундамент для реальных приложений iOS. Приготовьтесь: в этой главе будет много кода и новых концепций. Вдохните поглубже и переверните страницу, чтобы с головой погрузиться в *SwiftUI*.

У вас будет время выпить, но только после того, как вы освоите SwiftUI...



11 Практическое применение SwiftUI

Круги, таймеры, кнопки — выбирайте!

SwiftUI вовсе не ограничивается кнопками и списками. В интерфейсах также можно использовать геометрические фигуры, анимации и многое другое! В этой главе будут рассмотрены некоторые **расширенные возможности построения пользовательских интерфейсов в SwiftUI** и их связывания с источниками данных, содержимое которых не генерируется пользователем (как в случае со списком задач). SwiftUI позволяет строить **пользовательские интерфейсы** с обработкой **событий**, поступающих из разных источников. Мы будем работать в Xcode, среде разработки от компании Apple, а основное внимание будет уделяться приложениям для iOS, но все, что вы узнаете в этой главе, в равной степени применимо к SwiftUI для iPadOS, macOS, watchOS и tvOS. Вперед, глубины SwiftUI ожидают вас!

Создайте свой собственный круг!

12 Приложения, веб-программирование и все такое

Собирая все вместе

Вы значительно продвинулись в изучении Swift. Вы освоили Playgrounds и Xcode. Было понятно, что когда-нибудь нам придется **попрощаться**, и сейчас этот момент настал. Расставаться нелегко, но мы знаем, что вы справитесь. В этой главе — последней, в которой мы будем вместе с вами (в этой книге), — мы еще раз **пройдемся по многим концепциям,** которые вы изучили, и совместно построим несколько приложений. Мы убедимся в том, что ваши навыки Swift закреплены, и дадим некоторые рекомендации относительно того, **что делать дальше,** — своего рода домашнее задание, если хотите. Это будет интересно, и мы расстанемся на высокой ноте.