Об авторах

Пэрис Баттфилд-Эддисон

Джон Мэннинг

Пэрис Баттфилд-Эддисон и **Джон Мэннинг** — соучредители студии разработки Secret Lab в Хобарте (штат Тасмания, Австралия).

И Пэрис, и Джон имеют кандидатскую степень в области компьютерных наук, а за прошедшие годы они написали более 30 книг. Они вместе работали во влиятельном стартапе эпохи «Web 2.0» Meebo и входят в команду одной из самых давних конференций разработчиков Apple AUC /dev/world.

В Secret Lab Пэрис и Джон совместно работали над тысячами приложений и игр. Они наиболее известны своей приключенческой игрой Night in the Woods, получившей награды Independent Game Festival и BAFTA, а также популярным проектом с открытым кодом Spinner (*https://yarnspinner.dev*), лежащим в основе тысяч повествовательных видеоигр.

Пэрис и Джон живут и работают в Хобарте, они увлекаются фотографией, кулинарией и выступлениями на многочисленных конференциях. С Пэрисом можно связаться на сайте *https://paris.id.au*, с Джоном на сайте *https://desplesda.net*, а с Secret Lab на сайте *https://secretlab.games*.

Студия Secret Lab

Содержание (сводка)

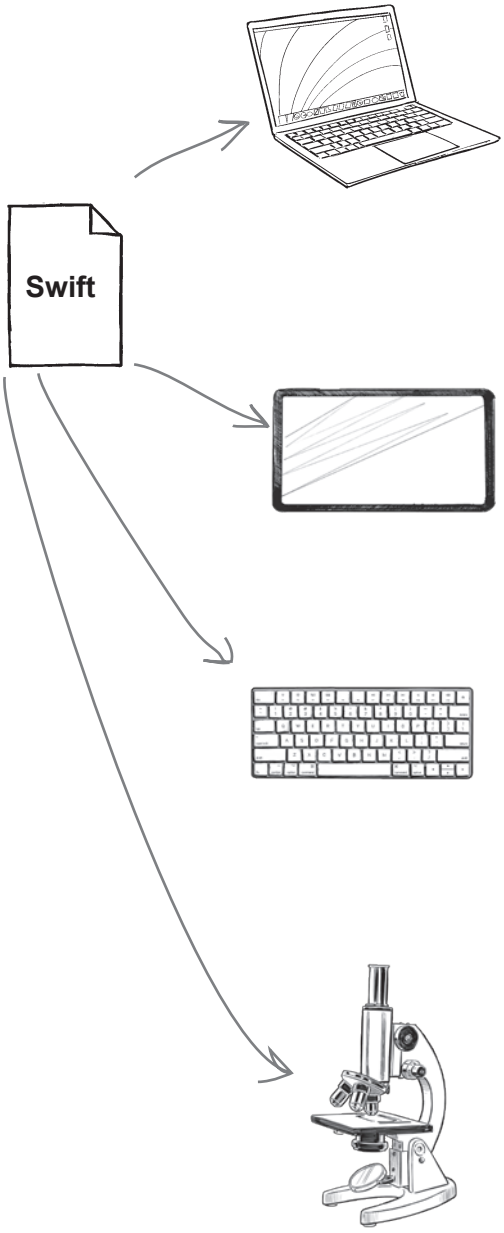
Содержание (настоящее)

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

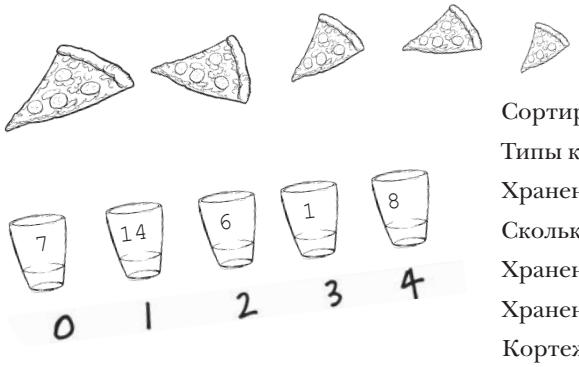
Swift на практике

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

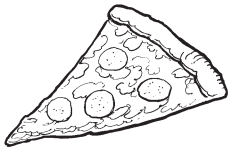
Необычные гибкие функции

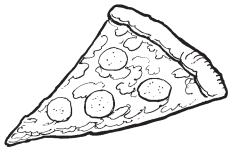
Функции полезны, но иногда нужно больше гибкости. Swift позволяет использовать **функцию как тип** — так же, как вы используете целое число или строку. **Это означает, что вы можете создать функцию и присвоить ее переменной**. После того как функция будет присвоена переменной, ее можно вызывать через 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

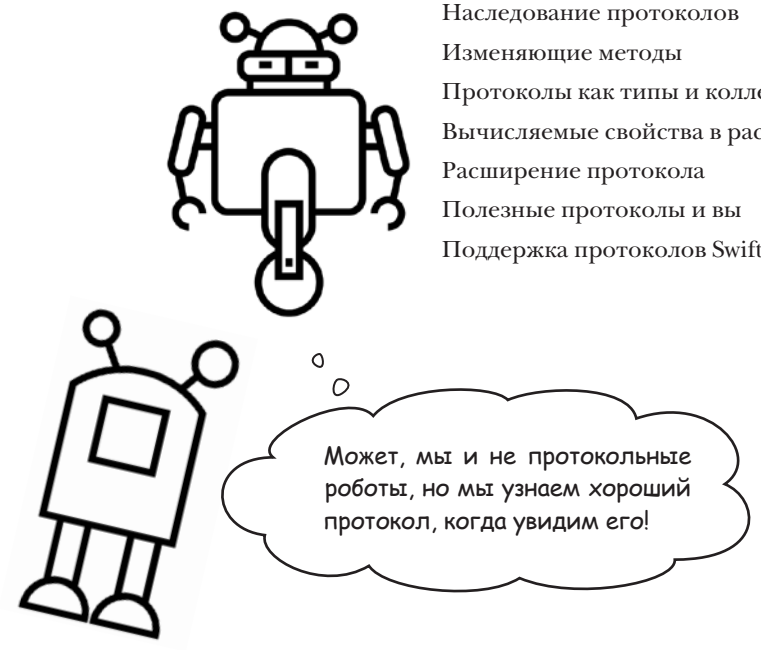
О пользе наследования

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

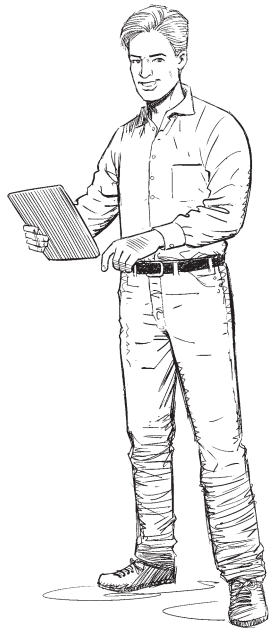
Вы все знаете о классах и наследовании, но у Swift еще есть немало средств структурирования программ. Знакомьтесь: протоколы и расширения. **Протоколы в Swift позволяют определить шаблон с перечнем методов и свойств,** необходимых для некоторой цели или некоторого блока функциональности. **Протокол включается** 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.

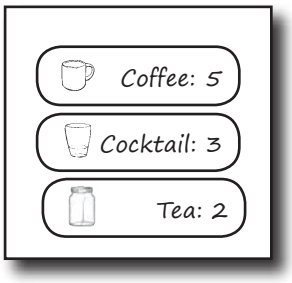


10 Знакомство со SwiftUI

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.

Введение

В этом разделе мы ответим на важный вопрос:
«Так почему они включили ТАКОЕ в книгу о Swift?»

Для кого написана эта книга?

Если на вопросы:

1. В вашем распоряжении есть устройство macOS или iPadOS, на котором работают последние общедоступные версии этих операционных систем?
2. Вы хотите изучить принципы программирования на примере языка Swift, чтобы потом продолжить свое путешествие в мире Swift?
3. Вы хотите в один прекрасный день заняться разработкой приложений для iPhone или любых других устройств в экосистеме Apple или изучить перспективный язык для написания веб-приложений?

вы отвечаете положительно, то эта книга для вас.

Кому эта книга не подойдет?

Если вы ответите «да» на любой из следующих вопросов...

1. Вы отличный разработчик с опытом программирования на macOS, iOS или Swift, которому нужен справочник?
2. Вы не хотите быть программистом и не хотите учиться программировать?
3. Вам не нравится пицца, еда, напитки или неуклюжие шутки?

...эта книга не для вас.

[Замечание от отдела продаж:
«Вообще-то эта книга для любого, у кого есть деньги».]

Мы знаем, о чем вы думаете

«Разве серьезные книги по программированию на Swift *такие?*»

«И почему здесь столько рисунков?»

«Можно ли *так* чему-нибудь научиться?»

И мы знаем, о чем думает ваш мозг

Мозг жаждет новых впечатлений. Он постоянно ищет, анализирует, *ожидает* чего-то необычного. Он так устроен, и это помогает нам выжить.

В наши дни вы вряд ли попадете на обед к тигру. Но наш мозг постоянно остается настороже. Просто мы об этом не знаем.

Как же наш мозг поступает со всеми обычными, повседневными вещами? Он *всеми силами* пытается оградиться от них, чтобы они не мешали его *настоящей* работе — запоминанию того, что действительно *важно*. Мозг не считает нужным сохранять скучную информацию. Она не проходит через фильтр, отсекающий «очевидно несущественное».

Но как же мозг *узнает*, что важно? Представьте, что вы отправились на прогулку, и вдруг прямо перед вами появляется тигр. Что происходит в вашей голове и в теле?

Активизируются нейроны. Вспыхивают эмоции. Происходят химические реакции.

И тогда ваш мозг понимает...

Конечно, это важно! Не забывать!

А теперь представьте, что вы находитесь дома или в библиотеке, в теплом, уютном месте, где тигры не водятся. Вы учитесь — готовитесь к экзамену. Или пытаетесь освоить сложную техническую тему, на которую вам выделили неделю... максимум десять дней.

И тут возникает проблема: ваш мозг пытается оказать вам услугу. Он старается сделать так, чтобы на эту *очевидно* несущественную информацию не тратились драгоценные ресурсы. Их лучше потратить на что-нибудь *важное*. На тигров, например. Или на то, что ни в коем случае нельзя вывешивать фото с этой вечеринки на своей страничке в соцсетях.

Нет простого способа сказать своему мозгу: «Послушай, мозг, я тебе, конечно, благодарен, но какой бы скучной ни была эта книга и пусть мой датчик эмоций сейчас на нуле, я *хочу* запомнить то, что здесь написано».

Эта книга для тех, кто хочет учиться

Как мы что-то *узнаем*? Сначала нужно это «что-то» *понять*, а потом *не забыть*. Затолкать в голову побольше фактов недостаточно. Согласно новейшим исследованиям в области когнитивистики, нейробиологии и психологии обучения, для *усвоения* материала требуется нечто большее, чем просто прочитать текст.

Мы знаем, как заставить ваш мозг работать.

Основные принципы серии «Head First»:

Наглядность. Графика запоминается гораздо лучше, чем обычный текст, и значительно повышает эффективность восприятия информации (до 89% по данным исследований). Кроме того, материал становится более понятным. **Текст размещается на рисунках, к которым он относится**, а не под ними или на соседней странице.

Разговорный стиль изложения. Недавние исследования показали, что при разговорном стиле изложения материала (вместо формальных лекций) улучшение результатов на итоговом тестировании составляло до 40%. Рассказывайте историю, вместо того чтобы читать лекцию. Не относитесь к себе слишком серьезно. Что скорее привлечет ваше внимание: занимательная беседа за столом или лекция?

Активное участие читателя. Пока вы не начнете напрягать извилины, в вашей голове ничего не произойдет. Читатель должен быть заинтересован в результате; он должен решать задачи, формулировать выводы и овладевать новыми знаниями. А для этого необходимы упражнения и каверзные вопросы, в решении которых задействованы оба полушария.

Привлечение (и сохранение) внимания читателя. Ситуация, знакомая каждому: «Я очень хочу изучить это, но засыпаю на первой же странице». Мозг обращает внимание на интересное, странное, притягательное, неожиданное. Изучение сложной технической темы не обязано быть скучным. Интересное узнается намного быстрее.

Обращение к эмоциям. Известно, что наша способность запоминать в значительной мере зависит от эмоционального сопереживания. Мы запоминаем то, что нам небезразлично. Мы запоминаем, когда что-то чувствуем. Нет, сантименты здесь ни при чем: речь идет о таких эмоциях, как удивление, любопытство, интерес и чувство «Да я крут!» при решении задачи, которую окружающие считают сложной, — или когда вы понимаете, что разбираетесь в теме лучше, чем всезнайка Боб из технического отдела.

Метапознание: наука о мышлении

Если вы действительно хотите быстрее и глубже усваивать новые знания, задумайтесь над тем, как вы задумываетесь. Учитесь учиться.

Мало кто из нас изучает теорию метапознания во время учебы. Нам *положено* учиться, но нас редко этому *учат*.

Но раз вы читаете эту книгу, то вероятно, вы хотите узнать, как программировать на Swift, и по возможности быстрее. Вы хотите запомнить прочитанное и применять новую информацию на практике. Чтобы извлечь максимум пользы из учебного процесса, нужно заставить ваш мозг воспринимать новый материал как Нечто Важное. Критичное для вашего существования. Такое же важное, как тигр. Иначе вам предстоит бесконечная борьба с вашим мозгом, который всеми силами уклоняется от запоминания новой информации.

Как же УБЕДИТЬ мозг, что программирование на Swift так же важно, как и тигр?

Есть способ медленный и скучный, а есть быстрый и эффективный. Первый основан на тупом повторении. Всем известно, что даже самую скучную информацию *можно* запомнить, если повторять ее снова и снова. При достаточном количестве повторений ваш мозг прикидывает: «*Вроде бы* несущественно, но раз одно и то же повторяется *столько раз*... Ладно, уговорил».

Быстрый способ основан на ***повышении активности мозга***, и особенно на сочетании разных ее *видов*. Доказано, что все факторы, перечисленные на предыдущей странице, помогают вашему мозгу работать на вас. Например, исследования показали, что размещение слов *внутри* рисунков (а не в подписях, в основном тексте и т. д.) заставляет мозг анализировать связи между текстом и графикой, а это приводит к активизации большего количества нейронов. Больше нейронов — выше вероятность того, что информация будет сочтена важной и достойной запоминания.

Разговорный стиль тоже важен: обычно люди проявляют больше внимания, когда они участвуют в разговоре, так как им приходится следить за ходом беседы и высказывать свое мнение. Причем мозг совершенно не интересует, что вы «разговариваете» с книгой! С другой стороны, если текст сух и формален, то мозг чувствует то же, что чувствуете вы на скучной лекции в роли пассивного участника. Его клонит в сон.

Но рисунки и разговорный стиль — это только начало.

Вот что сделали МЫ

Мы использовали **рисунки**, потому что мозг лучше приспособлен для восприятия графики, чем текста. С точки зрения мозга рисунок сто́ит тысячи слов. А когда текст комбинируется с графикой, мы внедряем текст прямо в рисунки, потому что мозг при этом работает эффективнее.

Мы используем **избыточность**: повторяем одно и то же несколько раз, применяя *разные* средства передачи информации, обращаемся к разным чувствам — и все для повышения вероятности того, что материал будет закодирован в нескольких областях вашего мозга.

Мы используем концепции и рисунки несколько ***неожиданным*** образом, потому что мозг лучше воспринимает новую информацию. Кроме того, рисунки и идеи обычно имеют ***эмоциональное*** *содержание*, потому что мозг обращает внимание на биохимию эмоций. То, что заставляет нас *чувствовать*, лучше запоминается, будь то ***шутка***, ***удивление*** или ***интерес***.

Мы используем *разговорный стиль*, потому что мозг лучше воспринимает информацию, когда вы участвуете в разговоре, а не пассивно слушаете лекцию. Это происходит и при *чтении*.

В книгу включено более 80 ***упражнений***, потому что мозг лучше запоминает, когда вы ***работаете*** самостоятельно. Мы постарались сделать их непростыми, но интересными — то, что предпочитает большинство читателей.

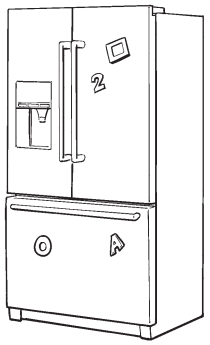
Мы совместили ***несколько стилей обучения***, потому что одни читатели любят пошаговые описания, другие стремятся сначала представить «общую картину», а третьим хватает фрагмента кода. Независимо от ваших личных предпочтений, полезно видеть несколько вариантов представления одного и того же материала.

Мы постарались задействовать ***оба полушария вашего мозга***: это повышает вероятность усвоения материала. Пока одно полушарие мозга работает, другое часто имеет возможность отдохнуть; это повышает эффективность обучения в течение продолжительного времени.

А еще в книгу включены ***истории*** и упражнения, отражающие ***другие точки зрения***. Мозг качественнее усваивает информацию, когда ему приходится оценивать и выносить суждения.

В книге часто встречаются вопросы, на которые не всегда можно дать простой ответ, потому что мозг быстрее учится и запоминает, когда ему приходится что-то ***делать***. Невозможно накачать мышцы, ***наблюдая*** за тем, как занимаются другие. Однако мы позаботились о том, чтобы усилия читателей были приложены в ***верном*** направлении. Вам не придется ***ломать голову*** над невразумительными примерами или разбираться в сложном, перенасыщенном техническим жаргоном или слишком лаконичном тексте.

В историях, примерах, на картинках использованы антропоморфные образы. Ведь ***вы*** человек. И ваш мозг уделяет больше внимания ***людям***, а не ***вещам***.



Вырежьте и прикрепите на холодильник.

Что можете сделать ВЫ, чтобы заставить свой мозг повиноваться

Мы свое дело сделали. Остальное за вами. Эти советы станут отправной точкой; прислушайтесь к своему мозгу и определите, что вам подходит, а что не подходит. Пробуйте новое.

1 Не торопитесь. Чем больше вы поймете, тем меньше придется запоминать.

Просто читать недостаточно. Когда книга задает вам вопрос, не переходите к ответу. Представьте, что кто-то *действительно* задает вам вопрос. Чем глубже ваш мозг будет мыслить, тем скорее вы поймете и запомните материал.

2 Выполняйте упражнения, делайте заметки.

Мы включили упражнения в книгу, но выполнять их за вас не собираемся. И не *разглядывайте* упражнения. **Берите карандаш** и пишите. Физические действия *во время* учения повышают его эффективность.

3 Читайте врезки.

Это значит: читайте всё. ***Врезки – часть основного материала***! Не пропускайте их.

4 Не читайте другие книги после этой перед сном.

Часть обучения (особенно перенос информации в долгосрочную память) происходит *после* того, как вы откладываете книгу. Ваш мозг не сразу усваивает информацию. Если во время обработки поступит новая информация, часть того, что вы узнали ранее, может быть потеряна.

5 Говорите вслух.

Речь активизирует другие участки мозга. Если вы пытаетесь что-то понять или запомнить, произнесите вслух. А еще лучше – попробуйте объяснить кому-нибудь другому. Вы быстрее усвоите материал и, возможно, откроете что-то новое.

6 Пейте воду. И побольше.

Мозгу нужна влага, так он лучше работает. Дегидратация (которая может наступить еще до того, как вы почувствуете жажду) снижает когнитивные функции.

7 Прислушивайтесь к своему мозгу.

Следите за тем, когда ваш мозг начинает уставать. Если вы стали поверхностно воспринимать материал или забываете только что прочитанное – пора сделать перерыв.

8 Чувствуйте!

Ваш мозг должен знать, что материал книги действительно ***важен***. Переживайте за героев наших историй. Придумывайте собственные подписи к фотографиям. Поморщиться над неудачной шуткой все равно лучше, чем не почувствовать ничего.

9 Пишите программы!

Научиться программировать можно только одним способом: **писать код**. Именно этим вам предстоит заняться, читая книгу. Подобные навыки лучше всего закрепляются практикой. В каждой главе вы найдете упражнения. Не пропускайте их. Не бойтесь **подсмотреть** в решение задачи, если не знаете, что делать дальше! (Иногда можно застрять на элементарном.) Но все равно пытайтесь решать задачи самостоятельно. Пока ваш код не начнет работать, не стоит переходить к следующим страницам книги.

Примите к сведению

Это учебник, а не справочник. Мы намеренно убрали из книги все, что могло бы помешать изучению материала, над которым вы работаете. И при первом чтении книги начинать следует с самого начала, потому что книга предполагает наличие у читателя определенных знаний и опыта.

Мы начинаем с изложения основных концепций Swift и сведем все воедино только после того, как будет заложен надежный фундамент.

Невозможно написать приложение для iPhone, не зная, как работают переменные и константы (и многое другое). По этой причине мы начнем с азов, прежде чем переходить к основному материалу.

Мы не пытаемся рассказать всё во всех подробностях.

Swift — достаточно обширная область, и в мире найдется немало хороших книг (в том числе и написанных нами!), в которых Swift рассматривается на разных уровнях. Было бы неразумно обсуждать все аспекты Swift в одной книге. Мы рассматриваем то, что вам необходимо знать, чтобы уверенно чувствовать себя на начальной стадии.

Мы отобрали самые полезные темы.

Существует много способов построения пользовательских интерфейсов на базе Swift, от AppKit до UIKit и SwiftUI. В книге мы решили изложить азы работы со SwiftUI и не рассматривать остальные. Но поскольку в книге вы узнаете обо всех структурных элементах, вам будет гораздо проще изучать AppKit в будущем.

Упражнения обязательны к выполнению.

Упражнения и задачи являются частью основного содержания книги, а не дополнительным материалом. Некоторые помогают запомнить новую информацию, некоторые — лучше понять ее, а некоторые — научиться применять ее на практике. *Не пропускайте упражнения.* Необязательными являются только ребусы «У бассейна», но следует помнить, что они хорошо развивают логическое мышление.

Повторение применяется намеренно.

У книг этой серии есть одна принципиальная особенность: мы хотим, чтобы вы действительно хорошо усвоили материал. И чтобы вы запомнили все, что узнали. Большинство справочников не ставит своей целью успешное запоминание, но это не справочник, а учебник, поэтому некоторые концепции излагаются в книге по несколько раз.

Мы постарались сделать примеры по возможности компактными.

Наши читатели говорят, что им не хочется просматривать 200-страничный листинг в поисках двух строк, которые нужно понять. Многие примеры в книге приводятся в минимально возможном контексте, чтобы часть, которую вы изучаете, была простой и понятной. Не ожидайте, что все примеры будут полностью защищенными от ошибок или хотя бы полными, — они написаны в учебных целях и не всегда полностью функциональны.

Мы разместили большой объем кода в интернете, чтобы вы могли копировать его в Playgrounds и Xcode. Код доступен по адресу *https://secretlab.com.au/books/head-first-swift*.

Упражнения «Мозговой штурм» не имеют ответов.

В некоторых из них правильного ответа вообще нет, в других вы должны сами решить, насколько правильны ваши ответы (это является частью процесса обучения). В некоторых упражнениях «Мозговой штурм» приводятся подсказки, которые помогут вам найти нужное направление.

Научные редакторы

Тим Нагент

Ник Сейрс

Ишмаэл Шабаз

Огромное спасибо всем, кто помог нам разобраться с технологическими сторонами этой книги. Они потратили много времени на проверку материала и его качества, а также на оповещения нас о том, что мы сделали что-то неразумное. Мы не всегда воспринимали ваши комментарии буквально, но они неизменно помогали нам улучшить книгу.

Нескольких человек мы хотим поблагодарить особо: это **Тим Нагент**, **Ник Сейрс** и **Ишмаэл Шабаз.**

Благодарности

Наш редактор:

Нам не удалось бы написать эту книгу без поддержки **Мишель Кронин**. Мы бы непременно написали нечто достойное ее и разместили здесь, если бы это вообще было возможно. За прошедшие годы мы написали много книг, но эта, пожалуй, была самой сложной, а работа над ней заняла больше всего времени. Нам доводилось работать с разными редакторами, пока мы не встретились с Мишель (кстати, все они были замечательными), но только с участием Мишель книга стала принимать правильную форму. Она постоянно помогала нам, с ней было интересно, и наше общение во время многих, многих встреч было просто фантастическим. Надеемся поработать с тобой над другими проектами. И еще раз: без тебя у нас ничего бы не вышло.

Мишель Кронин

Команда издательства O'Reilly:

Огромное и искреннее спасибо **Кристоферу Фошеру** — выпускающему редактору этой книги, без которого нам не удалось бы так четко собрать книгу в единое целое. Сожалеем, что у нас все так плохо с InDesign.

Также спасибо **Кристен Браун**, которая помогла нам отшлифовать материал, и **Рэчел Хед** за потрясающе качественную редактуру (как обычно) и поддержку.

Благодарим **Зэна Маккуэйда**, который не только был невероятно интересным и веселым собеседником на всех наших встречах, но и терпел наши жалобы на InDesign, Apple и все между ними.

Рэчел Румелиотис

Еще хотим поблагодарить нашего друга и (одного из) редакторов O'Reilly: **Рэчел Румелиотис**. Нам не хватает наших встреч на конференциях каждые несколько месяцев, и мы надеемся, что все начнется снова, когда вот это всё *широкий жест руками* наконец-то закончится.

Спасибо **Брайану Макдональду** — одному из редакторов, работавших с нами в течение долгого времени, а также человеку, который уговорил нас заняться написанием книг, — Нилу Голдстейну.

Также нам ужасно жалко, что мы всегда используем австралийский диалект, приятель.

Спасибо всему коллективу O'Reilly Media в целом. Они лучшие... буквально. Другой такой команды не существует, и каждый новый человек из O'Reilly, с которым мы знакомились, был приятным в общении, высококлассным специалистом и в целом интереснейшей личностью.

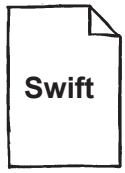
1. Знакомство со Swift

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!

Swift — универсальный язык

На Swift можно создавать программы для **macOS**, **Linux и Windows**. В системе macOS можно строить полнофункциональные графические приложения при помощи UI-фреймворка, к которому мы вернемся позднее.



Swift для iPad, iPhone и iPod позволяет строить **приложения iOS и iPadOS. Кроме того, существует возможность строить приложения** tvOS для Apple TV и приложения watchOS для Apple Watch.

Swift также может использоваться для создания **веб-приложений**. Для веб-сайтов используются серверные части или статически генерируемые страницы, а для приложений — серверные API, что упрощает синхронизацию.

Swift также распространяется с открытым кодом. Проект с открытым кодом Swift энергично развивается, эффективно управляется, а процесс привлечения новых участников отличается ясностью и доброжелательностью.

Swift также может использоваться в области **обработки данных** и **машинного обучения**, а также в области **системного программирования** для решения разнообразных задач: научных, моделирования и т. д.

О том, как присоединиться к работе над проектом, будет рассказано ближе к концу книги.

Что бы вы ни хотели сделать, это можно сделать на Swift.

Эволюция Swift

Язык Swift постоянно развивается, в нем ежегодно появляются новые дополнения: от скромного старта в Swift 1 до дня сегодняшнего и в будущее.

В каждом ***крупном*** обновлении Swift добавлялись новые **языковые средства**, а некоторые элементы **синтаксиса** устаревали. С первых объявлений и выпуска в 2014 г. Swift стал одним из наиболее быстро растущих и самых популярных языков в истории программирования.

В последние годы Swift регулярно занимает место в **десятке самых популярных языков программирования**, а количество пользователей и проектов, использующих этот язык, продолжает расти. Навыки программирования на Swift также пользуются высоким спросом, так что не забудьте добавить Swift в свое резюме, когда завершите работу над книгой!

Их можно подсчитывать разными способами, причем некоторые способы логичнее других. Как бы то ни было, Swift занимает высокие места во всех списках!

Swift создавался как язык, предназначенный исключительно для платформ Apple: iOS, macOS, tvOS, watchOS, но с момента его перехода на распространение с открытым кодом в 2015 г. он вырос, а его возможности расширились. Он прекрасно подходит для системного программирования, научных задач, веб-программирования и многих других областей.

Что бы вы ни собирались создавать, при выборе языка программирования Swift станет отличным кандидатом.

В книге мы не будем рассматривать все без исключения грани Swift, но вы будете знать все необходимое для дальнейшего самостоятельного изучения!

Ключевые моменты

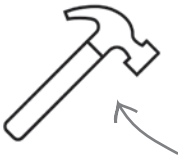
- **Средства языка** — такие вещи, как структуры, протоколы и представления SwiftUI.
- **Синтаксис** подразумевает конкретное размещение и использование таких элементов, как ! и ?, квадратные скобки и тому подобное.

Стремительная эволюция Swift

2014

Swift 1

Появление Swift было анонсировано (ко всеобщему удивлению и под фанфары) на большой конференции разработчиков Apple (WWDC) в июне 2014 года. В то время Swift был запатентованным языком, который мог использоваться только для платформ Apple, и он распространялся с закрытым кодом.



Первые выпуски Swift поддерживали только Xcode для macOS.

2015

Язык Swift был переведен на модель с открытым кодом в декабре 2015 г., в версии 2.2. С тех пор разработка Swift ведется у всех на виду по адресу https://github.com/apple/swift.

Swift 2

Версия Swift 2 была представлена на конференции WWDC в 2015 г. Многие элементы синтаксиса изменились на основании обратной связи с сообществом, а некоторые языковые средства эволюционировали. Swift продолжал набирать популярность и признание в сообществе.

2016

Swift 3

В версию Swift 3, выпущенную в сентябре 2016 г., был включен ряд изменений синтаксиса, также основанных на обратной связи от сообщества. Выход Swift 3 ознаменовал начало новой эпохи стабильности для Swift: было обещано, что в будущем синтаксис не будет изменяться так часто.

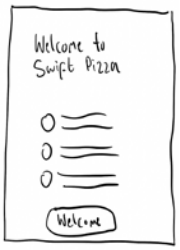
Версия Swift Playgrounds была выпущена для iPad одновременно с выпуском Swift 3.

Изображение молотка от Iconic из проекта Noun Project.

Стремление в будущее

Swift 4

Версия Swift 4 вышла в сентябре 2017 г. В ней появились новые возможности, но с сохранением более высокого уровня стабильности между версиями, чем прежде. Проект с открытым кодом Swift продолжал завоевывать новые позиции, а популярность языка продолжала расти.



SwiftUI, UI-фреймворк на базе Swift, был запущен в середине 2019 г.

Светлое будущее

Swift ждет светлое будущее. На горизонте появляется версия Swift 6; она сохранит синтаксис и функциональность Swift 5 и дополнит их рядом новых интересных возможностей. В Swift 5 появились новые средства (например, акторы), которые будут рассмотрены позднее в этой книге.

2017

2018

2019

2020

2021

2022

При проектировании в Swift были включены лучшие элементы других языков, включая Objective-C, Rust, Haskell, Ruby, Python, C# и т. д.

Swift 5

Swift 5 — основная версия, которая рассматривается в книге, — была выпущена в марте 2019 г.; миру была представлена полностью стабильная версия Swift.

Версия Swift Playgrounds была выпущена в виде приложения macOS в феврале 2020 г.

Как Вы будете писать код Swift

Да, именно вы!

Каждому языку программирования необходим инструмент для написания кода и запуска написанного кода. Чтобы сделать что-то полезное со Swift, вам неизбежно придется выполнить код Swift.

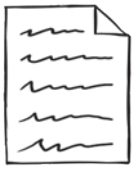
Как же это делается? В большей части книги мы будем использовать приложение **Playgrounds,** разработанное компанией Apple!

Playgrounds представляет собой нечто вроде (коллекции) больших традиционных текстовых файлов, но оказывается, вы также можете выполнить каждую написанную строку в компиляторе Swift и просмотреть результат.

Playgrounds будет использоваться в большей части кода Swift, написанного в книге.

...Но не во всем. Мы объясним чуть позднее.

Работа со Swift в Playgrounds проходит в три стадии:

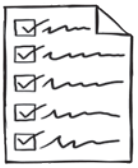


Написание кода

❶

Вы пишете свой код Swift в Playgrounds, который представляет собой **редактор для программистов** — такой же, какой может использоваться при работе с другими языками. Вы можете сохранить весь свой код Swift в одном файле или же разбить его на *несколько файлов*. Выбирайте по своему усмотрению.

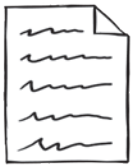
О том, как это делается, будет рассказано позднее.



Выполнение

❷

Затем вы выполняете свой код Swift — либо *весь сразу*, либо *строку за строкой*, в Playgrounds. Вы видите ошибки, а также *вывод* вашего кода рядом с каждой строкой и в консоли под областью редактирования кода.



Настройка

❸

Playgrounds формирует настолько быстрый цикл «написание кода/запуск», что вы можете легко внести изменения, доработать свой код и привести его в форму, необходимую для достижения ваших целей. Скорость разработки Swift — одна из самых сильных сторон языка. Пользуйтесь ею разумно!

Использование Playgrounds — самый быстрый способ переключения между выполнением и модификацией кода, чтобы вы получили хорошее представление о работе программы.