самоучитель

Айк Харазян

Язык Swift

Простые типы данных в Swift
Операторы и типы коллекций
Условные конструкции и циклы
Функции и замыкания
Перечисления и кортежи
Классы и структуры
ООП в Swift
Расширения и протоколы
Обобщенные типы

bhv®

УДК 004.438 Swift
ББК 32.973.26-018.1
Х20

Харазян А. А.

Х20 Язык Swift. Самоучитель. — СПб.: БХВ-Петербург, 2016. — 176 с.: ил. — (Самоучитель)

ISBN 978-5-9775-3572-4

Книга предназначена для самостоятельного изучения Swift — нового языка программирования для iOS и OS X. Описана версия Swift 2.0. Материал построен по принципу от более легкого к сложному, изложение сопровождается большим количеством листингов кода, для тестирования и отладки используется новая среда быстрой разработки Playground. Объяснены основы Swift, синтаксис языка и его особенности. Описаны типы данных, условные выражения, циклы, массивы, функции, кортежи, базовые операторы и другие стандартные конструкции. Кратко даны основы объектно-ориентированного программирования. Подробно рассмотрены более сложные или специфические для Swift конструкции: перечисления, замыкания, опциональные типы, классы, структуры, встроенные и обобщенные типы, расширения, протоколы, расширенные операторы и др.

Для программистов

УДК 004.438 Swift
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Марины Дамбиевой*

Подписано в печать 31.07.15.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 14,19.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3572-4

Оглавление

ГЛАВА 1

Swift: как он появился и с чего начать?

1.1. Как появился Swift?

Язык для человека является универсальным средством коммуникации и обмена информацией. С его помощью мы можем высказать наши мысли друзьям, членам семьи или коллегам по работе. И компьютерные языки программирования в этом плане также не являются исключением. Принцип выражения своих мыслей в них тот же. Только на этот раз мы высказываем их компьютеру.

Языки программирования, подобно человеческим языкам, существуют в разных формах и со временем эволюционируют и совершенствуются. Главной задачей языков программирования изначально являлась необходимость взаимодействовать с инструкциями компьютера, чтобы задать набор определенных действий, которые мы хотим от него получить.

Опытные пионеры компьютерной эпохи знают, что центральный процессор компьютера понимает только набор инструкций, состоящий из нулей и единиц. Чтобы разработчики могли разрабатывать свой код на более понятном им языке, чем нули и единицы, существуют программы-компиляторы, которые переводят код, написанный на определенном языке программирования, в инструкции для процессора, состоящие из тех самых нулей и единиц.

Компромисс между возможностями языка программирования и его производительностью обеспечили такие языки программирования, как C и C++. Но пока языки C и C++ распространялись на все большее количество платформ, Apple решила пойти иным путем и разработала язык Objective-C. Построенный на основе C, Objective-C сочетал в себе мощь широко используемого языка программирования с объектно-ориентированными методологиями. На протяжении многих лет Objective-C являлся основой для разработки программ для компьютеров Macintosh и iOS-устройств.

Однако, несмотря на производительность и простоту, Objective-C носил с собой весь багаж, наследованный от C. Для разработчиков, которые разбираются в C, это не вызывает трудностей. Но новички жаловались, что Objective-C труден для понимания и разработки.

Apple прислушалась к мольбам начинающих разработчиков и снизила для них входной барьер, выпустив язык программирования Swift. Теперь писать программы получается намного легче и понятнее.

1.2. Что нужно, чтобы начать разрабатывать на Swift?

Прежде всего, мы настоятельно рекомендуем писать листинги кода вместе с нами. Это позволяет лучше запомнить конструкции языка Swift. Для того чтобы вместе с нами писать и запускать код Swift, вам понадобится компьютер Macintosh с операционной системой не ниже OS X 10.9 Mavericks или 10.10 Yosemite. Вам также потребуется Xcode 7, в котором содержится компилятор Swift и его среда разработки.

1.3. Playground

Во время разработки какого-либо приложения нам часто хочется иметь возможность быстро протестировать несколько строчек кода. Обычно для этого приходится создавать новый проект, который формирует новую папку, генерирует конфигурационные файлы, пишет классы по умолчанию и т. д. И все это делается лишь для того, чтобы мы могли протестировать несколько кодовых строк.

В Swift нам этого делать не придется — чтобы протестировать свои алгоритмы мы можем воспользоваться уже упомянутым во *введении* Playground — новым типом документа Xcode, представляющим собой самостоятельный файл, в котором можно написать несколько строк кода для тестирования и не беспокоиться более о создании новых проектов.

Быстрое создание «площадки для тестирования» — не единственное преимущество Playground. Главным его отличием от проектов является немедленное выполнение любой написанной строки кода с выводом результатов выполнения этой строки. Они автоматически обновляются и выполняются по мере изменения документа.

Из сказанного можно сделать вывод, что Playground является очень удобным средством для углубления в Swift. Используйте его для достижения лучших и быстрых результатов в процессе изучения этого языка.

1.4. Как создать новый документ Playground?

Чтобы создать новый файл Playground, нам нужно запустить Xcode 7. Сразу после запуска откроется окно приветствия, содержащее меню, первым пунктом которого будет: **Get started with a playground** (рис. 1.1). Нам нужно выбрать именно этот пункт.

В результате откроется окно с предложением выбора имени для нового документа и типа платформы для разработки: OS X или iOS (рис. 1.2).

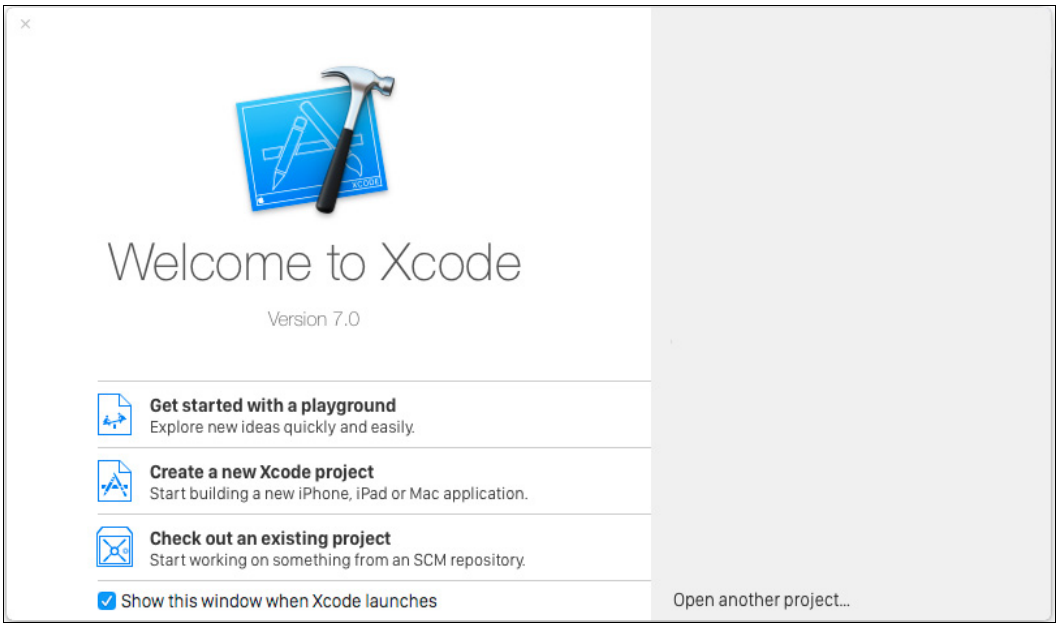

Рис. 1.1. Окно приветствия Xcode

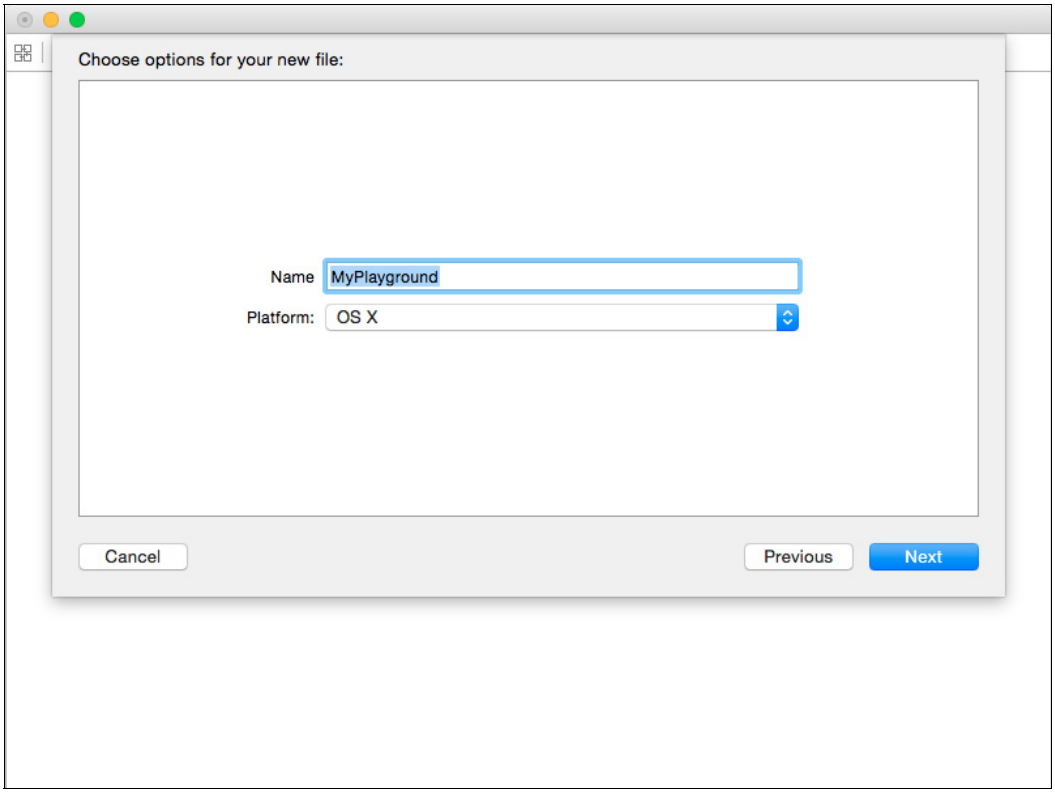

Рис. 1.2. Окно выбора имени и типа платформы

Указав имя и платформу, мы можем нажать кнопку **Next** и переместиться в диалоговое окно выбора места для сохранения нашего Playground. Выберите нужное местонахождение и нажмите на кнопку **Create**, чтобы окончательно его создать.

Если мы все правильно сделали, то должны увидеть новое окно Playground (рис. 1.3).

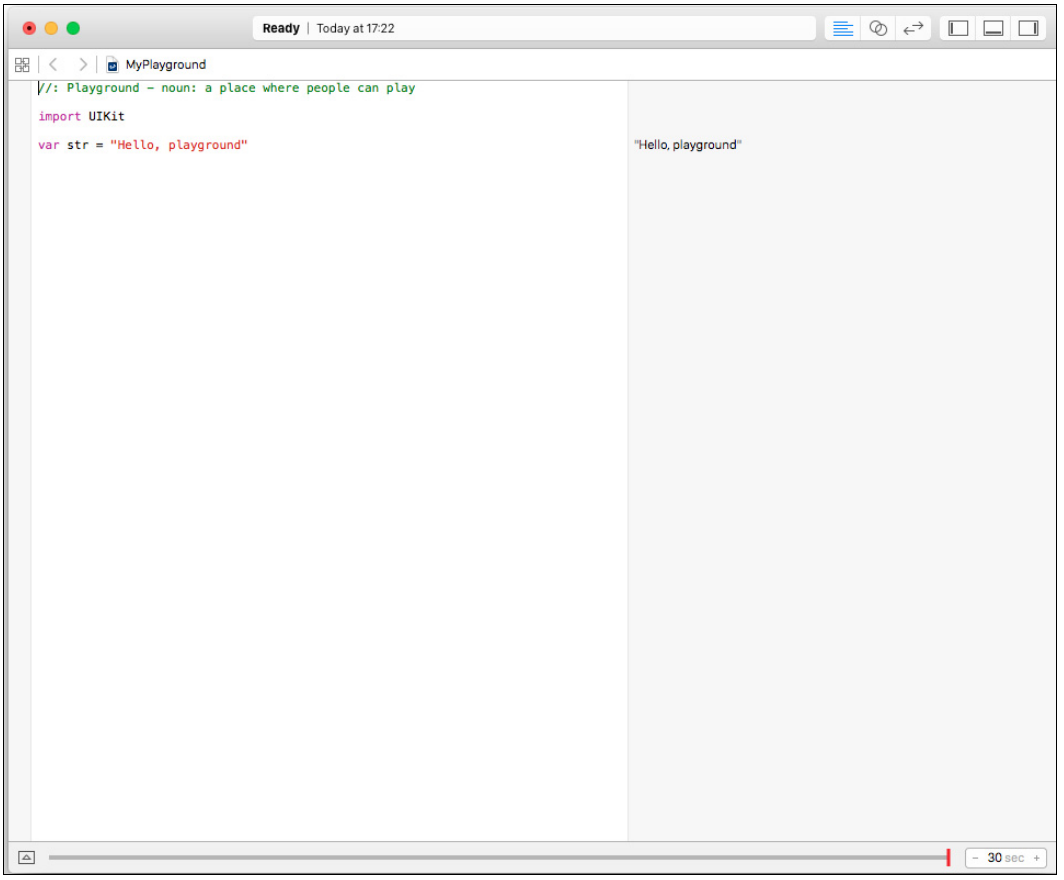

Рис. 1.3. Новое окно Playground

Теперь мы можем писать свои первые строчки кода на Swift. Левая область окна Playground предназначена для нашего кода, а правая — отображает результат выполнения этого кода.

В правой области отображается также любая полезная информация о значениях, содержащихся в переменных, о количествах итераций циклов, о возвращаемых значениях функций. Эти значения носят информативный характер и не совпадают со значениями, которые выводит наша программа. Для того чтобы узнать, какие данные выводит программа, нужно в правом верхнем углу окна найти три кнопки  , отвечающие за дополнительные панели с левой, с нижней и правой

его сторон. Нажать нужно на центральную кнопку, отвечающую за консольный вывод (рис. 1.4).

Теперь мы можем приступить к изучению Swift. Пишите и тестируйте листинги кода вместе с нами!

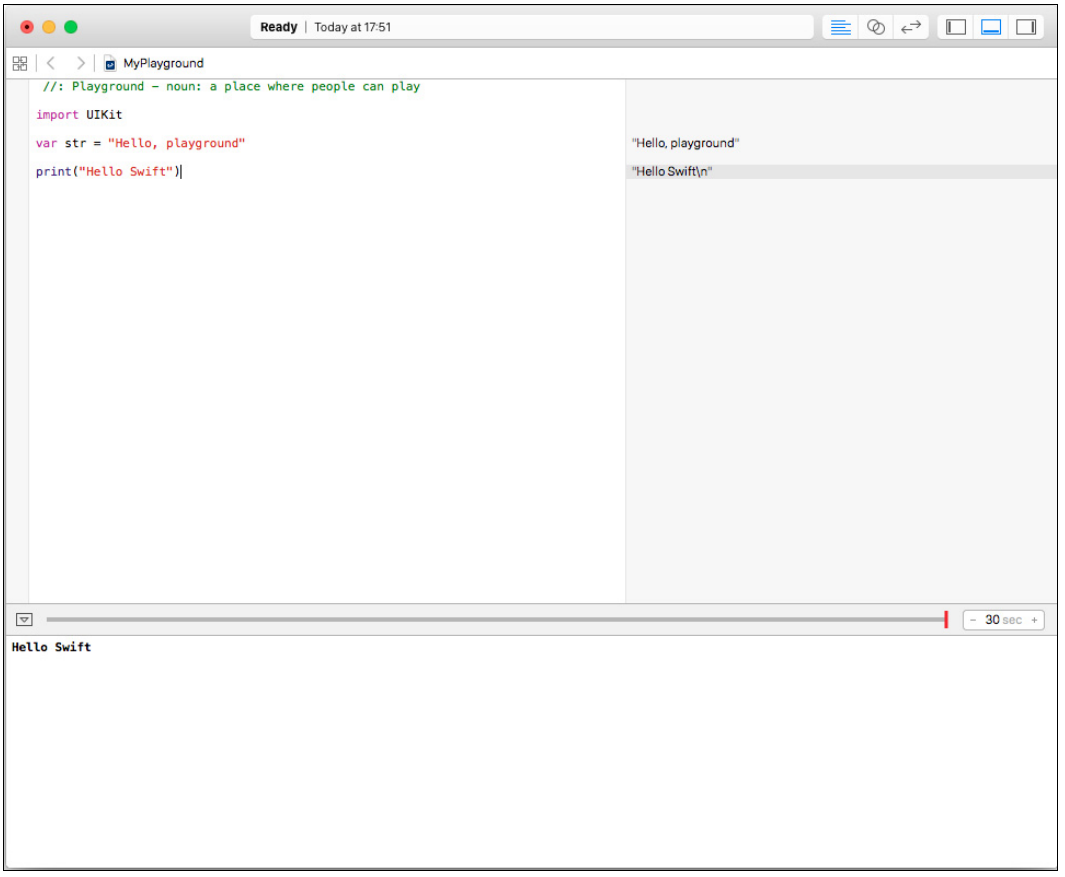

Рис. 1.4. Окно Playground с областью консольного вывода

ГЛАВА 2

Особенности синтаксиса Swift

Синтаксис языка программирования — это набор правил, определяющих, как пишутся программы на этом языке. Он определяет, какие символы будут употреблены для обозначения переменных, как нужно разделять инструкции друг от друга, какие конструкции употребляются в том или ином случае и т. д.

2.1. Swift — это C-подобный язык

Первой особенностью синтаксиса языка Swift является его принадлежность к группе C-подобных языков программирования. Разработчики, которые уже знакомы с C-подобными языками, найдут в Swift много знакомых конструкций. Блоки кода все также разделяются парой фигурных скобок, а циклы, условия и функции содержат знакомые нам ключевые слова `if`, `else`, `for`, `return`. Но, в отличие от Objective-C, мы не можем взять код, написанный на C, и вставить его в программу, написанную на Swift. Если в Objective-C мы могли так сделать, и все бы работало, то со Swift это не пройдет. Это означает, что Swift — заново написанный язык, а не просто надстройка над языком C.

2.2. Отсутствие заголовочных файлов

Следующей особенностью синтаксиса Swift является отсутствие заголовочных файлов. Программа, написанная на Swift, теперь не нуждается в отдельно написанных для него заголовочных файлах. Исполняемые файлы программы теперь имеют только один формат: .swift.

2.3. Точки с запятой ставить не обязательно

Одной из особенностей языка Swift является отсутствие необходимости ставить точку с запятой после каждого выражения. Теперь компилятор сам понимает, когда выражение закончилось. Но возможность использовать точки с запятой есть. Так,

в тех случаях, когда нам нужно записать несколько выражений в одну строку, мы должны после каждого выражения поставить точку с запятой. Либо, если вы в прошлом программировали на каком-нибудь C-подобном языке и по привычке ставите после каждого выражения точку с запятой, то можете не отказываться от этой привычки. Это означает, что вставлять точки с запятой не обязательно, но при желании вы их использовать можете.

2.4. Набор символов

При написании программ в Swift можно использовать символы Unicode. Но, в отличие от других языков программирования, где символы Unicode допускается употреблять только для строк и символов, в Swift символы Unicode можно использовать в названиях переменных, функций и других объектов. То есть, например, писать названия переменных на русском языке не возбраняется.

Кроме стандартных символов UTF-8, Swift также поддерживает UTF-16 и UTF-32. Этот факт сильно расширяет набор допустимых символов. Мы можем писать названия для переменных даже символами «эмодзи»[1].

Выводы

- Swift — заново написанный язык программирования, который относится к семейству C-подобных языков.
- Исходный код, написанный на Swift, теперь имеет только один тип: .swift, и не нуждается в заголовочных файлах.
- В конце каждой инструкции не обязательно ставить точку с запятой.
- Переменные, функции и другие объекты можно писать любыми символами Unicode.

[1] Эмодзи — язык идеограмм и смайликов, используемый в электронных сообщениях и на веб-страницах.

ГЛАВА 3

Простые типы данных, переменные и константы

Самым простым понятием в языках программирования являются *переменные*. Они позволяют хранить нужные нам значения в памяти компьютера. В Swift, наряду с переменными, существуют еще и *константы*. Если в переменных значения могут быть изменены, то значения констант после первого присвоения изменить невозможно.

Употребление констант в Swift сильно отличается от их употребления в других языках программирования. Если в других языках мы редко применяли константы, то в Swift константы используются так же часто, как и переменные. Apple рекомендует употреблять константы везде, где это возможно. Например, если нам нужно хранить информацию о дате рождения человека, то, по рекомендациям Apple, это значение лучше всего хранить внутри константы. Поскольку дата рождения человека не должна никогда изменяться, то использование константы позволит обезопасить это значение от случайного изменения. Кроме того, с использованием констант компилятор позволяет генерировать более оптимизированный код.

В этой главе мы рассмотрим, как создаются переменные и константы. Познакомимся с тем, что собой представляет механизм вывода типов, и где полезно явно указывать типы. После этого мы детально изучим каждый из простых встроенных типов значений в Swift. А в конце главы познакомимся с псевдонимами типов.

3.1. Переменные и константы

Давайте начнем с простой конструкции присвоения переменной значения. В Swift переменные объявляются через ключевое слово `var`. Название переменной мы можем придумать любое, но оно не должно начинаться с цифры:

```
var myHome = "Earth"
```

Эта строчка кода является законченной программой. Больше не нужно подключать дополнительные библиотеки и писать всю программу внутри глобальной функции `main`, как мы это делали в Objective-C.

В приведенном примере мы решили использовать название переменной `myHome`. Apple советует при написании имен для переменных, констант и других объектов придерживаться *горбатой нотации*. Согласно ее принципам, мы должны следовать правилу: если название состоит из нескольких слов, мы записываем всех их вместе, но каждое новое слово пишем с большой буквы. По-английски эту нотацию называют CamelCase (верблюжий стиль). Она перекочевала в Swift из Objective-C, где мы в таком же стиле писали названия переменных, функций и т. д. Следует заметить, что горбатая нотация не обязательна для соблюдения, но Apple рекомендует это делать, чтобы код был одинаково читаем для всех разработчиков.

Как мы отмечали в предыдущей главе, при задании имен переменных и других объектов у Swift, по сравнению с Objective-C, имеется одна интересная особенность — имена переменных могут состоять из любых Unicode-символов. А это означает, что названия переменных в Swift мы можем писать также и на русском языке. Да что там говорить — даже символ торговой марки (®) является в Swift приемлемым названием для переменной! А поскольку Swift поддерживает не только UTF-8, но еще UTF-16 и UTF-32, то мы можем использовать в названии переменных и символы «эмодзи»:

```
var 🏠 = "Earth"
```

Если вы заметили, то после приведенного выражения мы не поставили в конце точку с запятой. Действительно, в Swift, как уже и отмечалось ранее, не обязательно ставить точку с запятой после каждой строчки. Единственный случай, когда точку с запятой все же нужно ставить, — это когда мы пишем несколько выражений в одну строчку, например:

```
var myHome = "Earth"; var myShip = "Enterprise";
```

В Swift, наряду с переменными, существуют и константы. Они представляют собой те же переменные, только с постоянным значением. При попытке изменить значение константы, которой уже присвоили значение, Swift выведет ошибку. В остальном константы полностью идентичны переменным. Константы объявляются через ключевое слово `let`:

```
let myName = "James Kirk"
```

В отличие от других языков программирования, в Swift константы используются так же часто, как и переменные. Суть констант проста: когда у нас есть значение, которое никогда не будет изменяться (например, чья-то дата рождения), то настоятельно рекомендуется объявить ее в виде константы.

На первый взгляд вам может показаться непонятным, почему в Swift присутствуют и константы, и переменные? Разве переменная не предоставляет больше возможностей, чем константа? Это хороший вопрос. Ответ на него скрыт в основе работы компилятора. Компилятор в Swift может лучше оптимизировать код, когда знает, что значение, записанное в памяти, никогда не будет изменяться. Поэтому всегда используйте константы там, где вы точно знаете, что значение меняться не будет.

По мере разработки приложений на Swift вы все чаще будете использовать константы. Грубо говоря, с константами меньше головной боли. Мы тем самым защищаем наши значения от внезапного изменения в коде.