

iOS Mobile Development

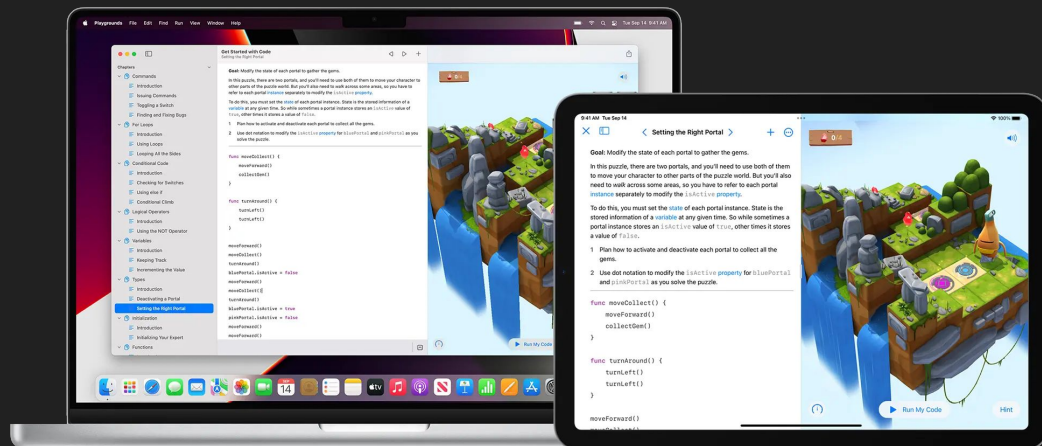
Модуль 1
Мырзакануров А.С.

Содержание

- ★ Знакомство с **Xcode Playground**
- ★ Переменные и константы
- ★ Стандартные типы данных

Xcode Playground

Swift Playgrounds — это революционное приложение для iPad и Mac, которое поможет вам научиться писать код и создавать приложения с помощью **Swift**, языка программирования, который используется для создания приложений для App Store.






Xcode

Xcode — интегрированная среда разработки (IDE) программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple.



Как установить **Xcode**?

1. Открыть **AppStore** на вашем Mac 
2. В поиске написать "Xcode" 
3. Нажать установить ►
4. Готово 

Swift

Swift — это надежный и интуитивно понятный язык программирования от Apple, при помощи которого можно создавать приложения для iOS, Mac, Apple TV и Apple Watch.

Он предоставляет разработчикам небывалую свободу творчества. Благодаря этому простому и удобному языку с открытым кодом вам достаточно просто интересной идеи, чтобы создать нечто невероятное.



Константы и Переменные

- ❑ **Константы** и **переменные** связывают **имя** (например, `maximumOfLoginAttempts` или `welcomeMessage`) со **значением** определенного типа (например, числом 10 или строкой «Hello»).
- ❑ Значение константы не может быть изменено после ее установки, тогда как переменной может быть присвоено другое значение в будущем.

Объявление констант и переменных

- **Константы и переменные** должны быть объявлены (созданы) до того, как они будут использованы
- Вы объявляете (создаете) константы с помощью ключевого слова **let** и переменные с ключевым словом **var**
- Вот пример того, как можно использовать константы и переменные для отслеживания количества попыток входа в систему, предпринятых пользователем:

```
1 let maximumNumberOfLoginAttempts = 10
2 var currentLoginAttempt = 0
```

Этот код можно прочесть как:

«Объявите новую константу с именем maximumNumberOfLoginAttempts и присвойте ей значение 10. Затем объявите новую переменную с именем currentLoginAttempt и присвойте ей начальное значение 0».

Объявление констант и переменных

Вы можете объявить несколько констант или несколько переменных в одной строке, разделив их запятыми:

```
var x = 0.0, y = 0.0, z = 0.0
```

Аннотации типов переменных и констант

- Вы можете предоставить аннотацию (*другими словами определить тип данных для хранения*) типа при объявлении константы или переменной, чтобы было ясно, какие значения может хранить константа или переменная.
- Напишите аннотацию типа, поместив двоеточие после имени константы или переменной, затем пробел и имя используемого типа.
- В этом примере представлена аннотация типа для переменной с именем **welcomeMessage**, чтобы указать, что переменная может хранить **Строки** (**String**)

```
var welcomeMessage: String
```

Присвоение имен константам и переменным

- Имена **констант** и **переменных** могут содержать практически любые символы, включая символы Unicode.
- Имена **констант** и **переменных не могут содержать** пробелы, математические символы, стрелки, скалярные значения Unicode для частного использования или символы рисования линий и прямоугольников. Они также не могут начинаться с цифры, хотя цифры могут быть включены в другое место в имени.
- После того как вы объявили **константу** или **переменную** определенного типа, **вы не можете объявить ее снова с тем же именем или изменить ее для хранения значений другого типа**. Вы также не можете превратить константу в переменную или переменную в константу.

```
1 let π = 3.14159
2 let 你好 = "你好世界"
3 let 🐶🐮 = "dogcow"
```

Присвоение имен константам и переменным

Также существуют негласные правила, что-то вроде Кодекса - программиста, для присвоения имен переменным

Есть несколько типов (техник) для создания имен переменных:

1. camelCase 🐪
2. snake_case 🐍
3. kebab-case 🍡
4. PascalCase 📄
5. UPPER_CASE_SNAKE_CASE 📋

Вывод данных констант и переменных

- Вы можете распечатать текущее значение константы или переменной с помощью функции `print()`
- Функция `print()` — это глобальная функция, которая выводит одно или несколько значений в соответствующий вывод. В Xcode, например, функция `print()` печатает свой вывод на панели «консоли» Xcode.

```
1  var friendlyWelcome = "Hello!"
2  friendlyWelcome = "Bonjour!"
3  // friendlyWelcome is now "Bonjour!"
```

```
1  print(friendlyWelcome)
2  // Prints "Bonjour!"
```

Вывод данных констант и переменных

Swift использует **интерполяцию строк (String Interpolation)**, чтобы включить имя **константы** или **переменной** в качестве заполнителя в более длинную строку и предложить Swift заменить его текущим значением этой константы или переменной. *Заключите имя в круглые скобки и закройте его обратной косой чертой перед открывающей скобкой:*

```
1 print("The current value of friendlyWelcome is \(friendlyWelcome)")
2 // Prints "The current value of friendlyWelcome is Bonjour!"
```

Комментарии

- Используйте **комментарии**, чтобы включить в код **неисполняемый текст** в качестве примечания или напоминания самому себе. Комментарии игнорируются компилятором Swift при компиляции кода.
- **Комментарии** в Swift очень похожи на комментарии в C. Однострочные комментарии начинаются с двух косых черт (//)
- Многострочные комментарии начинаются с косой черты, за которой следует звездочка (/*), и заканчиваются звездочкой, за которой следует косая черта (*/)

```
// This is a comment.
```

```
1 /* This is also a comment
2  but is written over multiple lines. */
```

Точки с запятой (Semicolon)

- В отличие от многих других языков, Swift не требует, чтобы вы ставили **точку с запятой (;)** после каждого оператора в вашем коде, хотя вы можете сделать это, если хотите.
- Однако **точка с запятой** необходима, если вы хотите написать несколько отдельных операторов в одной строке:

```
1 let cat = "🐱"; print(cat)
2 // Prints "🐱"
```


Стандартные типы данных

1. Символы (**Character**)
2. Строки (**Strings**)
3. Целые числа (**Integer**)
4. Числа с плавающей запятой
 - a. **Double** (64 bit)
 - b. **Float** (32 bit)
5. Логический тип данных (**Boolean**)

Строки и Символы (Strings & Characters)

- Строка (String) — это последовательность символов (character), например «привет, мир» или «альбатрос».
- Строки Swift представлены типом String.
- Доступ к содержимому строки можно получить различными способами, в том числе в виде массива значений символов.

```
let someString = "Some string literal value"
```

Строковые литералы (String Literal)

- ❑ Вы можете включать predetermined строковые значения в свой код как **строковые литералы**.
- ❑ **Строковый литерал** — это последовательность символов, заключенная в двойные кавычки ("").
- ❑ Используйте строковый литерал в качестве начального значения для константы или переменной:

```
let someString = "Some string literal value"
```

Многострочные строковые литералы (Multiline String Literal)

- ❏ Если вам нужна строка, занимающая несколько строк, используйте многострочный строковый литерал — последовательность символов, заключенных в **три двойных кавычки**

```
1 let quotation = ""
2 The White Rabbit put on his spectacles. "Where shall I begin,
3 please your Majesty?" he asked.
4
5 "Begin at the beginning," the King said gravely, "and go on
6 till you come to the end; then stop."
7 ""
```

Специальные символы в строковых литералах

Строковые литералы могут включать следующие специальные символы:

- ❑ Экранированные специальные символы: `\0` (нулевой символ), `\\` (обратная косая черта), `\t` (горизонтальная табуляция), `\n` (перевод строки), `\r` (возврат каретки), `\"` (двойная кавычка) и `\'` (одинарная кавычка)
- ❑ Произвольное скалярное значение Unicode, записанное как `\u{n}`, где `n` — шестнадцатеричное число от 1 до 8

Код ниже показывает четыре примера этих специальных символов. Константа `wiseWords` содержит две экранированные двойные кавычки. Константы `DollarSign`, `blackHeart` и `sparklingHeart` демонстрируют скалярный формат Unicode:

```
1 let wiseWords = "\"Imagination is more important than knowledge\" -  
  Einstein"  
2 // "Imagination is more important than knowledge" - Einstein  
3 let dollarSign = "\u{24}"      // $,  Unicode scalar U+0024  
4 let blackHeart = "\u{2665}"   // ♥,  Unicode scalar U+2665  
5 let sparklingHeart = "\u{1F496}" // 💖, Unicode scalar U+1F496
```

Инициализация пустой строки

Чтобы создать **пустое строковое значение** в качестве отправной точки для построения более длинной строки, *либо назначьте пустой строковый литерал переменной, либо инициализируйте новый экземпляр строки с помощью синтаксиса инициализатора*:

```
1  var emptyString = ""           // empty string literal
2  var anotherEmptyString = String() // initializer syntax
3  // these two strings are both empty, and are equivalent to each other
```

Интерполяция строк (String Interpolation)

Интерполяция строк — это способ создания нового значения `String` из сочетания констант, переменных, литералов и выражений путем включения их значений в строковый литерал.

Вы можете использовать интерполяцию строк как в однострочных, так и в многострочных строковых литералах. Каждый элемент, который вы вставляете в строковый литерал, заключается в пару круглых скобок, перед которыми ставится обратная косая черта (`\`):

```
1 let multiplier = 3
2 let message = "\(multiplier) times 2.5 is \((Double(multiplier) * 2.5))"
3 // message is "3 times 2.5 is 7.5"
```

Целые числа (Integers)

- ❑ **Целые числа** — это целые числа без дробной части, например 42 и -23. Целые числа бывают либо **знаковыми** (положительными, нулевыми или отрицательными), либо **беззнаковыми** (положительными или нулевыми).
- ❑ Swift предоставляет целые числа со знаком и без знака в **8, 16, 32 и 64-битных формах**. Эти целые числа следуют соглашению об именах, аналогичному C, в котором 8-разрядное целое число без знака имеет тип UInt8, а 32-разрядное целое число со знаком — тип Int32. Как и все типы в Swift, эти целочисленные типы имеют имена с заглавной буквы.

```
3 let number: Int = 97
```


Целочисленные границы

Вы можете получить доступ к минимальным и максимальным значениям каждого целочисленного типа с его свойствами `min` и `max`:

```
1 let minValue = UInt8.min // minValue is equal to 0, and is of type UInt8
2 let maxValue = UInt8.max // maxValue is equal to 255, and is of type UInt8
```

Числа с плавающей запятой

Числа с плавающей запятой — это числа с дробной частью, такие как 3,14159, 0,1 и -273,15

Типы с плавающей запятой могут представлять гораздо более широкий диапазон значений, чем целые типы, и могут хранить числа, которые намного больше или меньше, чем могут храниться в типе `Int`. Swift предоставляет два типа чисел с плавающей запятой со знаком:

1. `Double` представляет собой 64-битное число с плавающей запятой
2. `Float` представляет собой 32-битное число с плавающей запятой

```
3 let doubleNumber: Double = 3.14159
4 let floatNumber: Float = 0.95
```

Булевы значения (Boolean)

- ❑ В Swift есть базовый **логический тип**, который называется **Bool**
- ❑ Логические значения называются логическими, потому что они могут быть только **истинными** или **ложными**
- ❑ Swift предоставляет два логических постоянных значения, **true** и **false**:

```
1 let orangesAreOrange = true
2 let turnipsAreDelicious = false
```

Безопасность и Определение типов данных

Swift — **типобезопасный (type-safe)** язык. Типобезопасный язык побуждает вас четко понимать, с какими типами значений может работать ваш код. Если часть вашего кода требует **String**, вы не можете по ошибке передать ей **Int**.

Поскольку Swift является **типобезопасным**, он выполняет проверки типов при компиляции вашего кода и помечает любые несоответствующие типы как ошибки. Это позволяет обнаруживать и исправлять ошибки как можно раньше в процессе разработки.

Проверка типов помогает избежать ошибок при работе с различными типами значений. Однако это не означает, что вы должны указывать тип каждой объявляемой константы и переменной. Если вы не укажете тип значения, который вам нужен, Swift использует **вывод типа (type inference)** для определения соответствующего типа. **Вывод типа** позволяет компилятору автоматически определять тип конкретного выражения при компиляции вашего кода, просто изучая предоставленные вами значения.

