Основы Swift

Типы

Создание переменных и констант

Область памяти

var — переменная, значение можно менять
let — константа, значение задается единожды, доступ после присвоения значения

```
var variableName: Type
let variableName: Type
```

Объявление типа

Явное объявление типа

```
var variableName: Type = value
let constantName: Type = value
```

Неявное объявление типа

```
var variableName = value
let constantName = value
```

Строгая типизация в Swift

```
var variableName: Type = value
variableName = otherTypeValue // error
var x, y, z: Float
```

Вложенные типы

Внутри классов и структур объявлять собственные типы.

Можно объявить переменную или константу внутреннего типа, без создания инстанса внешнего типа.

Для доступа к вложенным типам используется дот нотация.

```
class OuterClass {
    class InnerClass {

    }
}

let a: OuterClass.InnerClass
```

Именование констант и переменных

- Можно использовать Unicode символы в том числе эмодзи.
- Нельзя начинать именование с цифр,не должны содержать управляющие символы,пробельные символы, математические символы, стрелки, а также приватные юникод символы.

Хорошие переменные

```
let isPrivate = true
let availableColors: [UIColor]
```

Плохие переменные

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

Целочисленные значения

- Int, Int8, Int16, Int32, Int64 — знаковые
- UInt, UInt8, UInt16, UInt32, UInt64 — беззнаковые

Int и UInt будут использовать 32 или 64 длину в зависимости от платформы.

Точно указываем длину при работе с данными из сети или при работе с железом на низком уровне, если это необходимо.

Int и UInt предпочтительнее.
Меньше ошибок. Легче конвертирование при работе с разными платформами.

Значения с плавающей точкой

Для представления чисел вида
3.1415926, 0.1, 273.5

- Float — 32-разрядные значения с плавающей точкой, минимум 6 знаков после точки
- Double — 64-разрядные значения с плавающей точкой, минимум 15 знаков после точки

Подходит, например, для задания координат.

Неточности значений с плавающей точкой и их сравнение

Float — 32bit. Double — 64bit

0.1 + 0.2 = 0.30000000000000004
0.3 != 0.30000000000000004
FLT_EPSILON = 2 в степени -23

x = 0.3 и y = (0.1 + 0.2)
Вычисляем разность z = (x - y)
Берем модуль от z
Если z <= FLT_EPSILON, значит числа равны

Booleans

Ключевое слово Bool
true или false

В отличие от C или Objective-C
логическое false не является эквивалентом 0,
а любое другое значение — эквивалентом true

Некорректное использование

```
let i = 1
if i {
    // will not compile with error

}
```

Tuple (Кортеж)

```
var coordinate = (x: 0.0, y: 0.0, z: 0.0)
let http404Error = (404, "Not Found")
```

В кортеже можно группировать разные типы.

```
var someTuple = (code: 404, description:
                                "Not Found")
```

Теперь можно не указывать имена при обращении

```
someTuple = (200, "Ok")
```

Optional

Optional используется в тех случаях, когда
у переменной может отсутствовать значение

Конвертируем строку в число

```
let correct = Int("123")
let incorrect = Int("Hello World!")
```

Конструктор возвращает Optional<Int> или Int?

Int? равен Int или nil

nil

- nil — обозначает отсутствующее значение для переменной.
- nil нельзя присвоить обычной переменной, только Optional<Type> тип может принимать значение nil.

Использование if с nil

```
var x: Int?
```

После сравнения используем force unwrap (!) для работы со значением.
Затрудняет чтение и небезопасно.

```
if x != nil {
    return x! + 5
}
```

Лучше if let — эквивалентно проверке с присвоением значения.

```
if let nonOptionalX = x {
    return nonOptionalX + 5
}
```