Основы Swift

Extensions

Что могут добавлять Extensions

- Вычисляемые свойства
- Новые методы
- Создавать инициализаторы
- Добавлять сабскрипты
- Определять и использовать вложенные типы
- Добавлять поддержку протоколов

Объявление Extension

```
extension SomeType: SomeProtocol,
AnotherProtocol {
    //
    // Код
    //
}
```

Расширяем тип String

```
extension String {
    var upperCaseDescription: String {
        return "Upper case: " + self.uppercased()
    }
}

print("Hello world".upperCaseDescription)
// Upper case: HELLO WORLD
```

Создаем инициализатор

```
extension String {
    init(left: String, right: String) {
        self = left + right
    }
}

let helloWorld = String.init(left: "Hello",
right: "World")
print(helloWorld)
// HelloWorld
```

Инициализатор в extenstion

```
struct Display {
    var width: Float = 0.0
    var horizontalResolution: Float = 0.0
    var ppi: Float = 0.0
}

extension Display {
    init(width: Float, horizontalResolution: Float) {
        var ppi = horizontalResolution / width
        self.init(width: width, horizontalResolution:
        horizontalResolution, ppi: ppi)
    }
}

let display = Display(width: 15.0, horizontalResolution: 1920)
print("PPI: ", display.ppi)
// PPI: 128.0
```

Методы в extension

```
extension String {
    func makeMePickle() {
        print("I'm pickle", self)
    }

    mutating func doubleString() {
        self = self + " " + self
    }
}

var name = "Rick"
name.doubleString()
name.makeMePickle()
// I'm pickle Rick Rick
```

subscript в extension

```
extension Int {
    subscript(digitIndex: Int) -> Int {
        var decimalBase = 1
        for _ in 0..<digitIndex {
            decimalBase *= 10
        }
        return (self / decimalBase) % 10
    }
}

print(4815162342[4])
// 6
```

nested type в extension

```
struct Display {
    var width = 0.0
    var height = 0.0
    var ppi = 0.0
}

extension Display {
    struct Resolution {
        var horizontal = 0.0
        var vertical = 0.0
    }

    var resolution: Resolution {
        return Resolution(horizontal: width * ppi,
        vertical: height * ppi)
    }
}

var resolution = Display.Resolution(horizontal: 1920,
vertical: 1080)
var display = Display(width: 16, height: 9, ppi: 320.0)
print(display.resolution)
// Resolution(horizontal: 5120.0, vertical: 2880.0)
```

Группировка в private extension

```
struct Display {
    var width = 0.0
    var height = 0.0
    var ppi = 0.0
}

private extension Display {
    func diagonalSize() -> Double {
        return sqrt(width * width + height * height)
    }
}

var display = Display(width: 16, height: 9,
ppi: 320.0)>
print("\(display.diagonalSize())")
// Diagonal size: 18.3575597506858
```

Группировка схожих по функционалу методов

```
extension String {
    func pringString(nTimes: Int) {
        var resultingString = ""
        for _ in 1...nTimes {
            resultingString.append(self)
        }
        print(resultingString)
    }
}

extension String {
    func printDoubleString() {
        pringString(nTimes: 2)
    }

    func printTrippleString() {
        pringString(nTimes: 3)
    }
}
```

```
let name = "Snufkin"

name.printDoubleString()
// SnufkinSnufkin

name.printTrippleString()
// SnufkinSnufkinSnufkin
```

Реализация протокола в extension

```
class MainController:
UIViewController {
    var model = [0, 1, 2, 3, 4, 5]

    override func viewDidLoad() {
        let tableView = UITableView()
        tableView.dataSource = self
    }
}
```

```
extension MainController:
UITableViewDataSource {
    func numberOfSections(in
    tableView: UITableView) -> Int
    {
        return 1
    }

    func tableView(_ tableView:
    UITableView,
    numberOfRowsInSection section:
    Int) -> Int {
        return model.count
    }

    func tableView(_ tableView:
    UITableView, cellForRowAt
    indexPath: IndexPath) ->
    UITableViewCell {
        return UITableViewCell()
    }
}
```

Computed properties в extension

```
struct Display {
    var width = 0.0
    var height = 0.0
    var ppi = 0.0
}

extension Display {
    var diagonalSize: Double {
        return sqrt(width * width + height * height)
    }
}
```