LITS

iOS. Xcode. Swift

`

Знайомимося

LITS

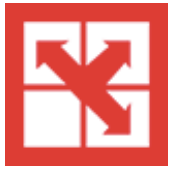
`

Знайомимося

LITS

Формат

Заняття в аудиторії 

Самостійна робота  moodle.lits.com.ua  GitHub

Практичні завдання + Дипломний проект

“Зробіть” = “Зробіть, **не** користуючись stackoverflow.com”

Спілкування 

LITS

Зміст курсу

8.3

3.1

ООП
Паттерни
Git
…

LITS

Чому

iOS-розробка влаштована саме так

LITS

iOS - Основна особливість

App Store

Продукти постачаються користувачам лише через AppStore (*)

LITS

App Store

- Для користувача: Apple гарантує безпеку і що аппа “ОК”
- Для розробника:
  - майданчик для продажу
  - але обмеження
- Apple обмежує можливість зробити продукт, що не відчувається, як рідний

LITS

Обмеження

- Взаємодія з файловою системою та іншими програмами (Sandbox)
- Лише публічне API
- Лише Xcode на зареєстрованому Apple комп'ютері
- Хаки суворо заборонені
- Обмежена багатозадачність
- Декларативна функціональність (iCloud, in-App Purchases, Push Notifications, etc)

LITS

Нема вибору

- ви будете використовувати системне API
  - правильно!
- ви напишете стабільний зручний продукт
- користувач буде сприймати ваш продукт як органічну частину екосистеми Apple

LITS

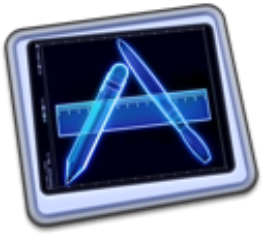
Xcode IDE

Xcode

iOS Simulator

InterfaceBuilder

Playgrounds



Instruments

Project Configuration
Assets Management
Code Signing

Continuous Integration
Testing
Etc…

LITS

Xcode

InterfaceBuilder

LITS

iOS SDK

Cocoa Touch

Media Frameworks

App Frameworks

Core Services

CoreOS

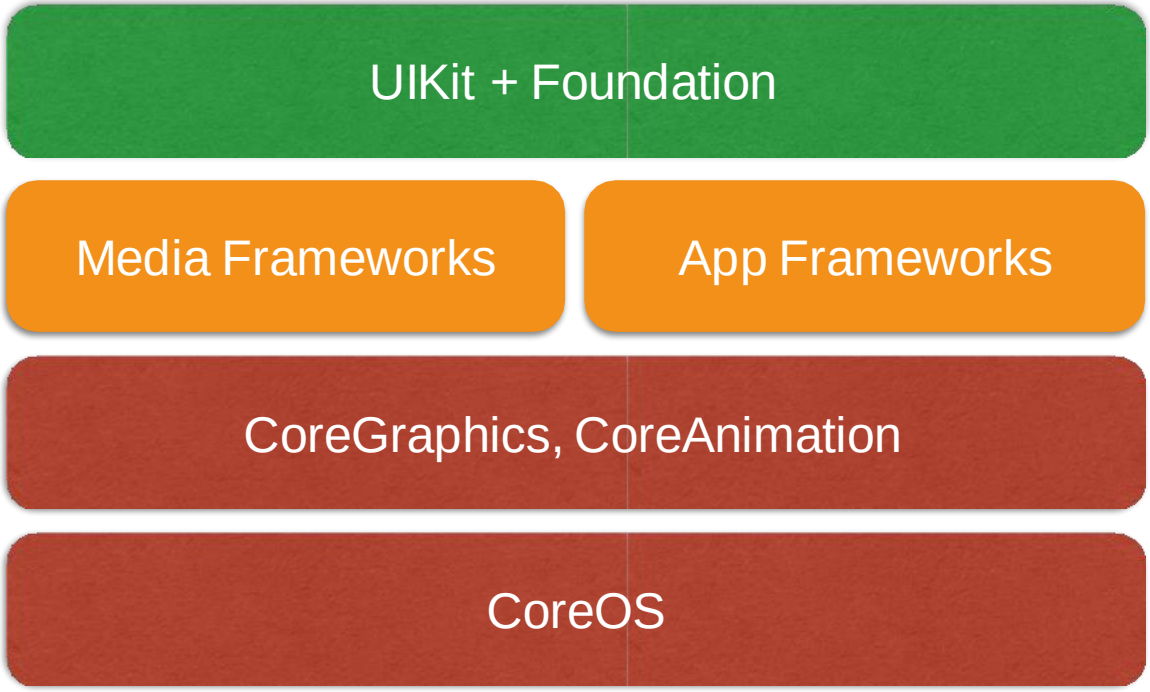
LITS

iOS SDK

UIView, UIColor
NSString, NSArray

MKMapView
EKEvent

CGPoint, CGRect
CALayer



LITS

Objective-C

- Об’єктно-орієнтована мова, дериватив від C
- Окрема лінія розвитку ООП-мов, ніж C++, Java (йде від Smalltalk)
- Динамічний runtime
- Підтримується та покращується Apple (*)

LITS

Objective-C | Swift

LITS

Swift

- Нова мова, з’явилася у 2014 році
- Swift 3.1 - поточна версія в Xcode 8.3
- Все найкраще з Objective-C
  - без C
- Сучасні мовні конструкції
  - вкл. запозичення з функціональних мов програмування
- Open-source (swift.org)

LITS

Swift

```
struct Point {



}
```

```
class Record {



}
```

LITS

Swift

```
struct Point {




}

var point: Point
```

```
class Record {




}

var record: Record
```

LITS

Swift

```
struct Point {




}

var point: Point = Point()
```

```
class Record {




}

var record: Record = Record()
```

LITS

Swift

```
struct Point {



}

var point = Point()
```

```
class Record {



}

var record = Record()
```

Type Inference

LITS

Swift

```
struct Point {
  var x: Int
  var y: Int



}

var point = Point(x: 0, y: 0)
```

```
class Record {



}

var record = Record()
```

LITS

Swift

```
struct Point {
  var x: Int
  var y: Int



}

var point = Point(x: 0, y: 0)
```

```
class Record {
  var name: String = ""
  var lastName: String = ""



}

var record = Record()
```

LITS

Swift

```
struct Point {
   var x: Int
   var y: Int




}

var point = Point(x: 0, y: 0)
```

```
class Record {
   var name: String
   var lastName: String
   init() {
      name = ""
       lastName = ""
   }
}

var record = Record()
```

LITS

Swift

```
struct Point {
  var x: Int
  var y: Int




}

var point = Point(x: 0, y: 0)
point.x = 5
point.y = 10
```

```
class Record {
  var name: String = ""
  var lastName: String = ""




}

var record = Record()
record.name = “John”
record.lastName = “Doe”
```

LITS

Swift

```
struct Point {
   var x: Int = 5
   var y: Int = 10
}

var point = Point()
```

```
class Record {
   var name: String = ""
   var lastName: String = ""

}

var record = Record()
record.name = “John”
record.lastName = “Doe”
```

LITS

Swift

```
struct Point {
  var x: Int = 5
  var y: Int = 10
}

var point = Point()
```

```
class Record {
  var name: String = ""
  var lastName: String = ""

  func fullName()


}

var record = Record()
record.name = “John”
record.lastName = “Doe”
```

LITS

Swift

```
struct Point {
  var x: Int = 5
  var y: Int = 10

}

var point = Point()
```

```
class Record {
  var name: String =""
  var lastName: String = ""

  func fullName() -> String {

  }
}

var record = Record()
record.name = “John”
record.lastName = “Doe”
```

LITS

Swift

```
struct Point {
  var x: Int = 5
  var y: Int = 10

}

var point = Point()
```

```
class Record {
  var name: String = ""
  var lastName: String = ""

  func fullName() -> String {

  }
}

var record = Record()
record.name = “John”
record.lastName = “Doe”

label.text = record.fullName()
```

LITS

Swift

```
struct Point {
  var x: Int = 5
  var y: Int = 10

}

var point = Point()
```

```
class Record {
  var name: String = ""
  var lastName: String = ""

  func fullName() -> String {
    return name + " " + lastName
  }
}

var record = Record()
record.name = "John"
record.lastName = "Doe"

label.text = record.fullName()
```

LITS

Named Parameters

```
struct Point {
  var x: Int = 5
  var y: Int = 10

  func offset(x: Int, y: Int) {

  }
}

var point = Point()
```

```
class Record {
  var name: String = ""
  var lastName: String = ""

  func fullName() -> String {
    return name + " " + lastName
  }
}

var record = Record()
record.name = "John"
record.lastName = "Doe"

label.text = record.fullName()
```

LITS

Named Parameters

```
struct Point {
  var x: Int
  var y: Int

  func offset(x: Int, y: Int) {

  }
}

var point = Point()
point.x = 5
point.y = 10

point.offset(x: 2, y: -1)
```

```
class Record {
  var name: String = ""
  var lastName: String = ""

  func fullName() -> String {
    return name + " " + lastName
  }
}

var record = Record()
record.name = "John"
record.lastName = "Doe"

label.text = record.fullName()
```

LITS

Named Parameters

```
struct Point {
  var x: Int = 5
  var y: Int = 10


  func offset(by x: Int,  y: Int) {


  }
}

var point = Point()

point.offset(by: 2, y: -
1)
```

```
class Record {
  var name: String = ""
  var lastName: String = ""

  func fullName() -> String {
    return name + " " + lastName
  }
}

var record = Record()
record.name = "John"
record.lastName = "Doe"

label.text = record.fullName()
```

LITS

Named Parameters

```
struct Point {
  var x: Int = 5
  var y: Int = 10

  func offset(by x: Int, _ y: Int) {

  }
}

var point = Point()

point.offset(by: 2,-1)
```

```
class Record {
  var name: String
  var lastName: String

  func fullName() -> String {
    return name + " " + lastName
  }
}

var record = Record()
record.name = "John"
record.lastName = "Doe"

label.text = record.fullName()
```

LITS

Class Functions

```
class Record {
  var name: String
  var lastName: String

  static func name() -> String {

    return "Record"
  }
}

label.text = Record.name()
```

LITS

Properties

```
class Record {
  var name: String = ""
  var lastName: String = ""

  var fullName: String {
    get {
       return name + " " + lastName
    }
  }
}
var record = Record()
record.name = “John”
record.lastName = “Doe”
label.text = record.fullName
```

LITS

Properties

```
class Record {
  var name: String = ""
  var lastName: String = ""

  var fullName: String {
    return name + " " + lastName
  }
}

var record = Record()
record.name = “John”
record.lastName = “Doe”
label.text = record.fullName
```

LITS

Properties

```
class Record {
  var name: String = ""
  var lastName: String = ""

  var fullName: String {
    get {
       return name + " " + lastName
    }
    set {
       name = newValue.components(separatedBy: " ").first
       lastName = newValue.components(separatedBy: " ").last
    }

  }
}
```

let record = Record()
record.fullName = "John Doe"

LITS

Instance var vs Param

```
struct Point {
   var x: Int = 5
   var y: Int = 10


   func offset(by x: Int, _ y: Int) {
      x += x
      y += y
   }
}

point.offset(by: 2,-1)
```

LITS

Self

```
struct Point {
   var x: Int = 5
   var y: Int = 10


   func offset(by x: Int, _ y: Int) {
      self.x += x
      self.y += y
   }
}

point.offset(by: 2,-1)
```

LITS

Structs

```
struct Point {
  var x: Int
  var y: Int

  mutating func offset(by x: Int, _ y: Int) {
    self.x += x
    self.y += y
  }
}
```

LITS

Structs vs Classes

```
struct Point {

}

var p1 = Point()
var p2 = Point()

var point = p1
```

Copy

```
class Record {

}

var r1 = Record()
var r2 = Record()

var record = r1
```

Reference

LITS

Structs vs Classes

```
struct Point {

}

var p1 = Point()
var p2 = Point()

var point = p1

let sameObject = p1 === point
```

Copy

false

```
class Record {

}

var r1 = Record()
var r2 = Record()

var record = r1

let sameObject = r1 === record
```

Reference

true

LITS

Structs vs Classes

```
struct Point {

}

var p1 = Point()
var p2 = Point()

let point = p1
point = p2
```

```
class Record {

}

var r1 = Record()
var r2 = Record()

let record = r1
record = r2
```

LITS

Structs vs Classes

```
struct Point {

}

var p1 = Point()
var p2 = Point()

let point = p1
point.x = 10
```

```
class Record {

}

var r1 = Record()
var r2 = Record()

let record = r1
record.name = “Mary”
```

LITS

Tuples

```
func findRecord() -> Record {
  var r: Record

  ...
  return r
}


let record = findRecord()
record.name = “Mary”
```

LITS

Tuples

```
func findRecord() -> (record: Record, index: Int) {
  var r: Record
  var i: Int
  ...
  return (r, i)
}


let pair = findRecord()
pair.record.name = “Mary”
```

LITS

Xcode Playgrounds

LITS

Отже

- Вивчаємо Xcode 8.3 та Swift 3.1
- Читаємо книжку по Swift 3

- Робимо самостійне завдання

  - Ще раз:

  “зробити” = “зорієнтуватися в документації і зробити”