Nouvelles technologies du web

LI385

Olivier Pitton

Backend

Cloud, web, DevOps, etc.

Olivier Pitton

Backend

Cloud, web, DevOps, etc.

Adrien Humilière

Frontend

iOS development, Swift

About me

Adrien Humilière

Mobile team lead @ Trainline

DANT 2011/2012


trainline

About me

adhumi+dant@gmail.com

Development tools

Swift 3

User interfaces

iOS SDK

Development tools

Swift 3

User interfaces

iOS SDK

Organisation

7 lessons (1h30)

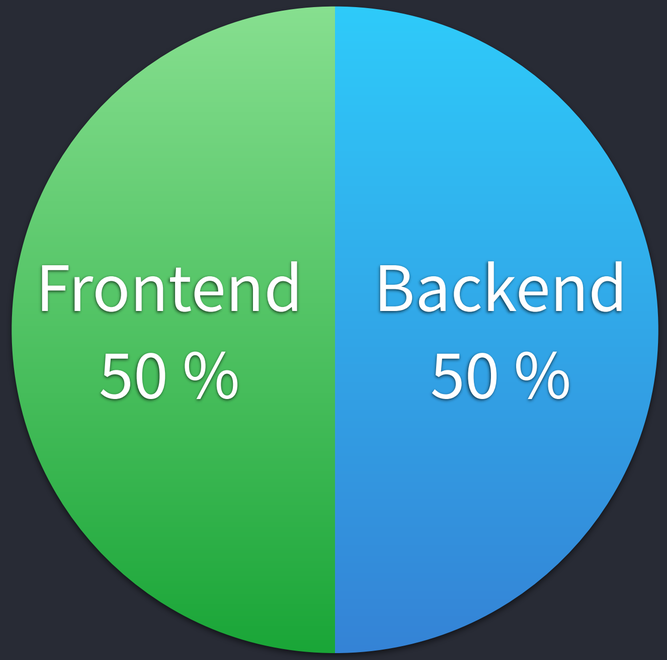
9 labs (3h)

3 project sessions (3h)

Notation

Notation

Notation



Notation

Project

application + backend

Practice at home

Have a mac ? Install Xcode.

Swift code can be written and built
on Mac, Linux, iPad, and web.

Developper account (free) on developer.apple.com
needed to build on device.

Practice at university

Salle 14-15, 409

available for you (if not in use)

Introduction to iOS development with Swift

Lesson 1

Adrien Humilière
Trainline

adhumi+dant@gmail.com

- Swift and playgrounds
- Constants, Variables, and Data Types
- Operators
- Control Flow
- Interface Builder

- Swift and playgrounds
- Constants, Variables, and Data Types
- Operators
- Control Flow
- Interface Builder

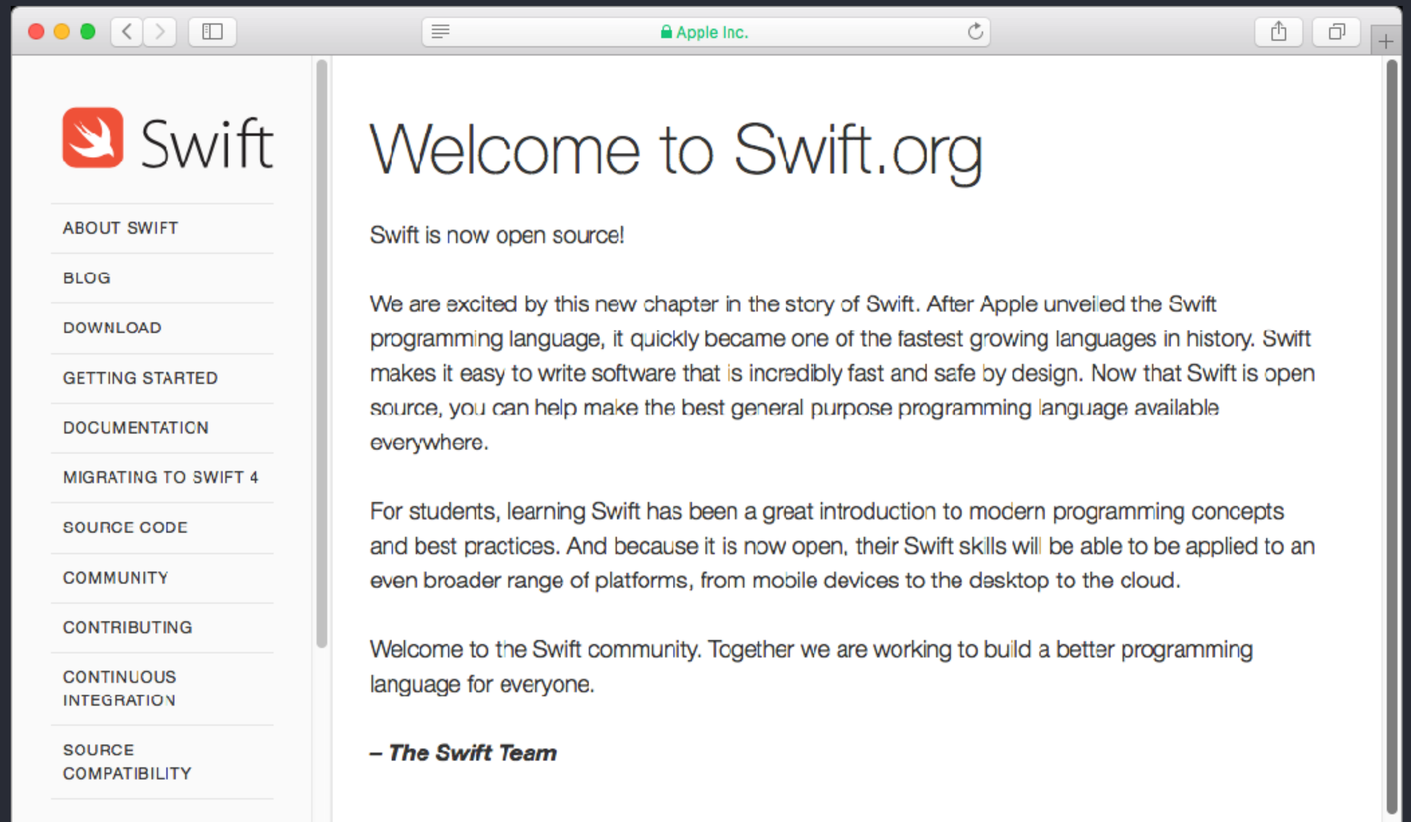
Swift and playgrounds

A **modern** language

A **safe** language

- → Explicit object « types »
- → Type inference
- → Optionals
- → Error handling

Open Source


Apple Inc.
Swift
ABOUT SWIFT
BLOG
DOWNLOAD
GETTING STARTED
DOCUMENTATION
MIGRATING TO SWIFT 4
SOURCE CODE
COMMUNITY
CONTRIBUTING
CONTINUOUS INTEGRATION
SOURCE COMPATIBILITY
Welcome to Swift.org
Swift is now open source!
We are excited by this new chapter in the story of Swift. After Apple unveiled the Swift programming language, it quickly became one of the fastest growing languages in history. Swift makes it easy to write software that is incredibly fast and safe by design. Now that Swift is open source, you can help make the best general purpose programming language available everywhere.
For students, learning Swift has been a great introduction to modern programming concepts and best practices. And because it is now open, their Swift skills will be able to be applied to an even broader range of platforms, from mobile devices to the desktop to the cloud.
Welcome to the Swift community. Together we are working to build a better programming language for everyone.
– The Swift Team

Hello, world!

```
print("Hello, world!")
```

Playgrounds

Constants, Variables, and Data Types

Constants

- Defined using the let keyword

```
let name = "John"
```

- Defined using the let keyword

```
let pi = 3.14159
```

- Can’t assign a constant a new value

```
let name = "John"
name = "James"
```

Variables

- Defined using the var keyword

```
var age = 29
```

- Can assign a new value to a variable

```
var age = 29
age = 30
```

Naming constants and variables

- No mathematical symbols
- No spaces
- Can’t begin with a number

```
let π = 3.14159
let 一百 = 100
let 🎲 = 6
let mañana = "Tomorrow"
let anzahlDerBücher = 15 //numberOfBooks
```

Naming constants and variables

- Clear and descriptive
- camelCase if multiple words

Types

```
struct Person {
  let firstName: String
  let lastName: String

  func sayHello() {
    print("Hello there! My name is \(firstName) \(lastName).")
  }
}
```

Most common types

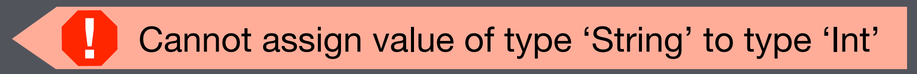
Type safety

```
let playerName: String = "Julian"
var playerScore: Int = 1000
var gameOver: Bool = false
playerScore = playerName
```

```
var wholeNumber:Int = 30
var numberWithDecimals: Double = 17.5
wholeNumber = numberWithDecimals
```

Type safety

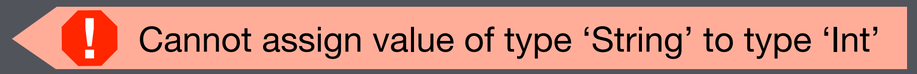
```
let playerName: String = "Julian"
var playerScore: Int = 1000
var gameOver: Bool = false
playerScore = playerName
```



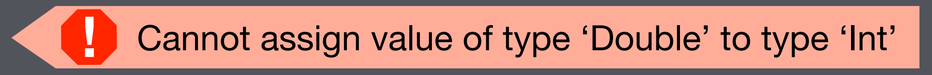

```
var wholeNumber:Int = 30
var numberWithDecimals: Double = 17.5
wholeNumber = numberWithDecimals
```

Type safety

```
let playerName: String = "Julian"
var playerScore: Int = 1000
var gameOver: Bool = false
playerScore = playerName
```




```
var wholeNumber:Int = 30
var numberWithDecimals: Double = 17.5
wholeNumber = numberWithDecimals
```



Type inference

```
let cityName = "San Francisco"
let pi = 3.1415927
```

Type annotation

```
let cityName: String = "San Francisco"
let pi: Double = 3.1415927
```

```
let number: Double = 3
print(number) // ~> 3.0
```

Mandatory type annotation

- When you create a constant or variable before assigning it a value

```
let firstName: String
//...
firstName = "Layne"
```

Mandatory type annotation

- When you create a constant or variable that could be inferred as two or more different types

```
let middleInitial: Character = "J"
var remainingDistance: Float = 30
```

Mandatory type annotation

- When you add properties to a type definition

```
struct Car {
    let make: String
    let model: String
    let year: Int
}
```

Operators

Assign a value

- Use the = operator to assign a value

```
var favoritePerson = "Luke"
```

- Use the = operator to modify or reassign a value

```
var shoeSize = 8
shoeSize = 9
```

Basic arithmetic

- You can use the +, -, *, and / operators to perform basic math functions

```
var opponentScore = 3 * 8
var myScore = 100 / 4
```

Basic arithmetic

- Use Double values for decimal precision

```
let totalDistance = 3.9
var distanceTravelled = 1.2
var remainingDistance = totalDistance - distanceTravelled
print(remainingDistance) // ~> 2.7
```

Basic arithmetic

```
let x = 51
let y = 4
let z = x / y
print(z) // ~> 12
```

Basic arithmetic

```
let x: Double = 51
let y: Double = 4
let z = x / y
print(z) // ~> 12.75
```

Compound assignment

```
var myScore = 10
myScore = myScore + 3

myScore += 3
myScore -= 5
myScore *= 2
myScore /= 2
```

Numeric type conversion

```
let x = 3
let y = 0.1415927
let pi = x + y
```

Numeric type conversion

```
let x = 3
let y = 0.1415927
let pi = x + y
```

Numeric type conversion

```
let x = 3
let y = 0.1415927
let pi = x + y
```

❗ Binary operator ‘+’ cannot be applied to operands of type ‘Int’ and ‘Double’

```
let x = 3
let y = 0.1415927
let pi = Double(x) + y
```

✅

Control Flow

Logical operators

==	Two items must be equal
!=	The values must not be equal to each other
>	Value on the left must be greater than the value on the right
>=	Value on the left must be greater than or equal to the value on the right
<	Value on the left must be less than the value on the right
<=	Value on the left must be less than or equal to the value on the right
&&	AND—The conditional statement on the left and right must be true
||	OR—The conditional statement on the left or right must be true
!	Returns the opposite of the conditional statement immediately following the operator

if statements

```
if condition {
  code
}
```

```
let temperature = 100
if temperature >= 100 {
  print("The water is boiling.")
}
```

if-else statements

```
if condition {
  code
} else {
  code
}
```

```
let temperature = 100
if temperature >= 100 {
  print("The water is boiling.")
} else {
  print("The water is not boiling.")
}
```

switch statement

```
switch value {
case n:
   code
case n:
   code
case n:
   code
default:
   code
}
```

switch statement

```
let numberOfWheels = 2
switch numberOfWheels {
case 1:
    print("Unicycle")
case 2:
    print("Bicycle")
case 3:
    print("Tricycle")
case 4:
    print("Quadcycle")
default:
    print("That's a lot of wheels!")
}
```

switch statement

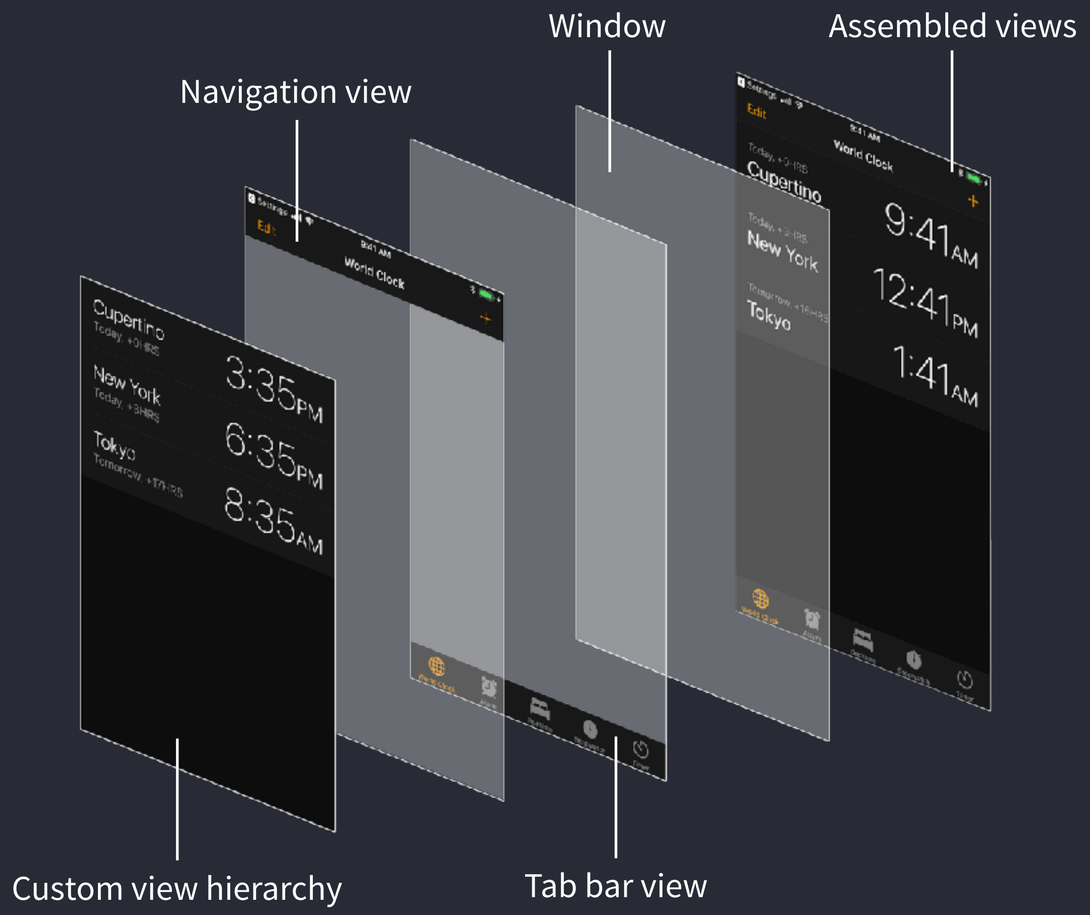
```
let character = "z"

switch character {
case "a", "e", "i", "o", "u", "y":
    print("This character is a vowel.")
default:
    print("This character is not a vowel.")
}
```

switch statement

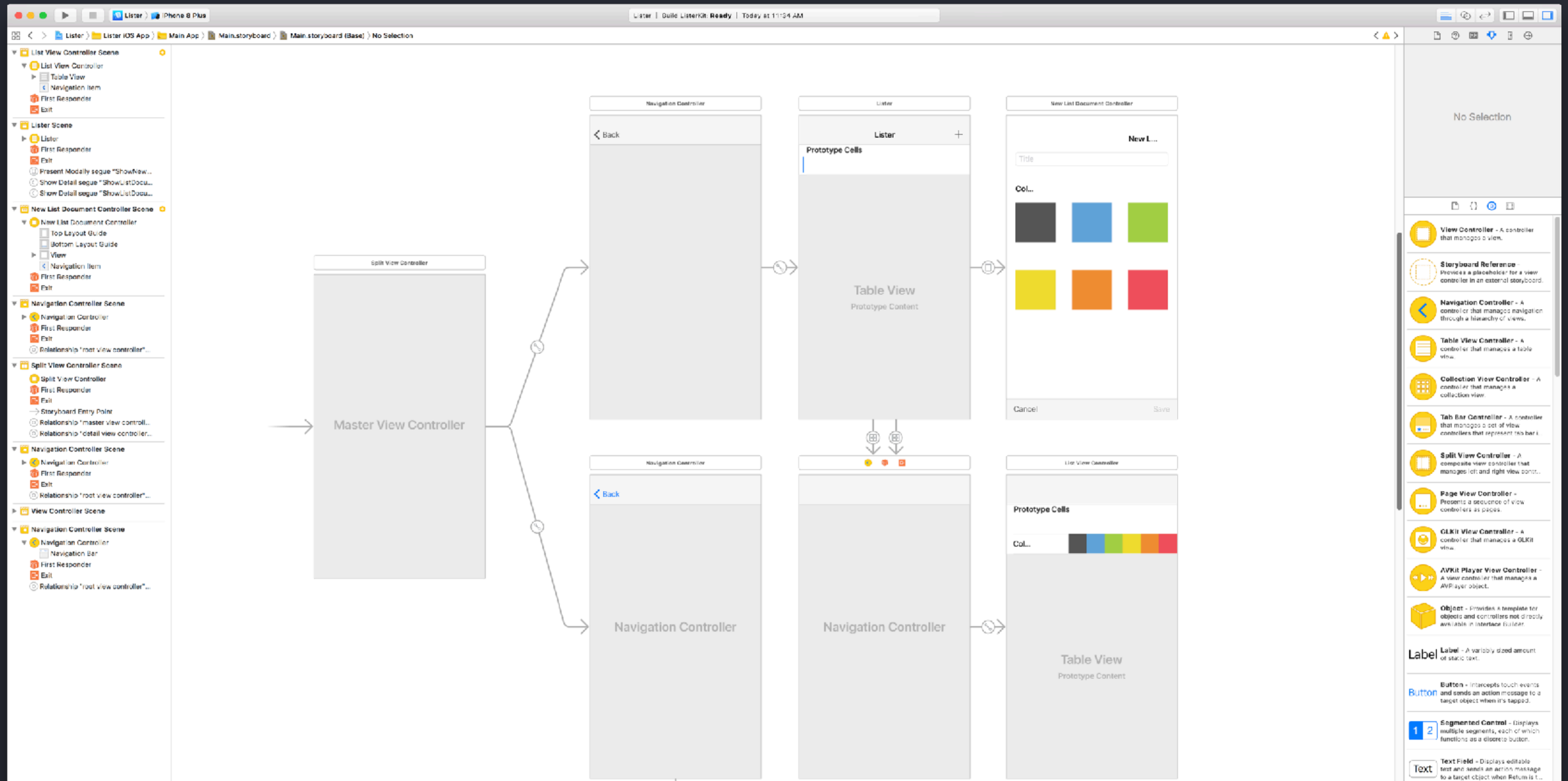
```
switch distance {
case 0...9:
    print("Your destination is close.")
case 10...99:
    print("Your destination is a medium distance from here.")
case 100...999:
    print("Your destination is far from here.")
default:
    print("Are you sure you want to travel this far?")
}
```

Interface Builder

Common system views



Interface builder



The End.