Développement natif sous iOS

Cours n°2 - Swift et Xcode

iOS

Swift

- Langage créé par Apple (2014)
- Spécifiquement pour iOS au départ
  - Objective-C étant plutôt complexe et bas niveau
- Aujourd’hui, le langage peut être utilisé pour faire du web, des scripts systèmes, etc.
  - Le compilateur Swift est disponible sous Linux

Swift

- Langage compilé
- Typage statique et fort
- Paradigme impératif (comme C ou Java) avec quelques concepts de programmation fonctionnelle

Hello, World!

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

Hello, World!

```
$ swiftc hello_world.swift
```

```
$ ./hello_world
Hello, World!
```

REPL

```
$ swift
Welcome to Apple Swift version 5.1.2
(swiftlang-1100.0.278 clang-1100.0.33.9).
Type :help for assistance.
  1>
```

Variables

```
var nom = "Jacques"
var nom2 : String = "Jacques"

var age : Int
```

Constantes

```
let nom = "Paul"

let nom2 : String = "Jacques"
```

Conditions

```
if age < 18 {

} else if age == 18 {

} else {

}
```

Tableaux

```swift
var noms = ["Pierre", "Paul", "Mouloud"]

var notes : [Int] = []
var ages: Array<Int>

// Accéder à un élément
noms[0]

// Ajouter un élément
notes.append(18)
```

Dictionnaires

```swift
var notes = ["Anglais": 19, "Maths": 12]

var notes : [ String : Int ] = [:]
var notes : Dictionary<String, Int>

// Accéder à un élément
notes["Anglais"]

// Définir un élément
notes["Physique"] = 13
```

Boucles

```swift
// Boucle "pour"
for i in 1...10 {

}
```

```swift
// Boucle "tant que"
var i = 1

while i <= 10 {
    i = i + 1
}
```

Boucles

```swift
// Tableaux
let noms = ["Pierre", "Karim"]

for nom in noms {

}

// Dictionnaires
let notes = ["SVT": 12, "Français": 15]

for (matiere, note) in notes {

}
```

Fonctions

```
func bonjour(nom: String) {
    print("Bonjour", nom)
}
```

```
func addition(a: Int, b: Int) -> Int {
    return a + b
}

bonjour(nom: "John")
addition(a: 1, b: 2)
```

Fonctions

```
func addition(_ a: Int, _ b: Int) -> Int {
    a + b
}

addition(1, 2)
```

Classes

```
class Voiture {
    private var immat : String

    init(immat: String) {
        self.immat = immat
    }
}

var voiture = Voiture(immat: "AA-001-BB")
```

Encapsulation

```
class Voiture {
    // Lecture et écriture
    public var immat : String

    // Lecture seule
    public private(set) var essence : Double
}
```

Méthodes

```
class Voiture {
    func rouler() {
        print("Vroum")
    }
}

let voiture = Voiture()

voiture.rouler()
```

Héritage

```
class Voiture {
    init(proprietaire: Personne) {
        self.proprietaire = proprietaire
    }
}

class Jaguar : Voiture {
    init(proprietaire: Personne, modele: String) {
        super.init(proprietaire: proprietaire)

        self.modele = modele
    }
}
```

Héritage

```
class Voiture {
    func plein() {
        self.proprietaire.porteMonnaie -= 50
    }
}

class Jaguar : Voiture {
    override func plein() {
        self.proprietaire.porteMonnaie -= 200
    }
}
```

Protocoles

```
protocol Vehicule {
    func avancer()
    func entretenir()
}
```

```
class Voiture : Vehicule {
    func avancer() {
        // ...
    }

    func entretenir() {
        // ...
    }
}
```

Énumérations

```
enum Unites {
    case Gramme
    case Kilogramme
    case Tonne
}

var uniteFarine = Unites.Gramme

var unitePommes : Unites
unitePommes = .Kilogramme
```

Erreurs

```
enum PleinError : Error {
    case caisseEnPanne
    case plusDargent(manquant: Int)
}

throw PleinError.caisseEnPanne
throw PleinError.plusDargent(manquant: 42)
```

Erreurs

```
func plein() throws {
    guard proprietaire.porteMonnaie > 0 else {
        throw PleinError.plusDargent(manquant: ...)
    }
    // ...
}

do {
    try voiture.plein()
} catch PleinError.plusDargent(let manquant) {
    print("Il vous manque \(manquant) euros")
}
```

Closures

- Aussi appelées « fonctions anonymes »

- Permettent de passer des fonctions en paramètres d’autres fonctions

- Très utilisées pour réaliser des traitements sur des collections

Closures

```
let maClosure = {element in
    print(element)
}

maClosure(10)
```

Closures

```
let maClosure = { print($0) }

maClosure(10)
```

Closures

```
let maClosure = { print($0) }
let etudiants = ["Pierre", "Paul", "Jacques"]

etudiants.forEach(maClosure)

etudiants.forEach({
    print($0)
})
```

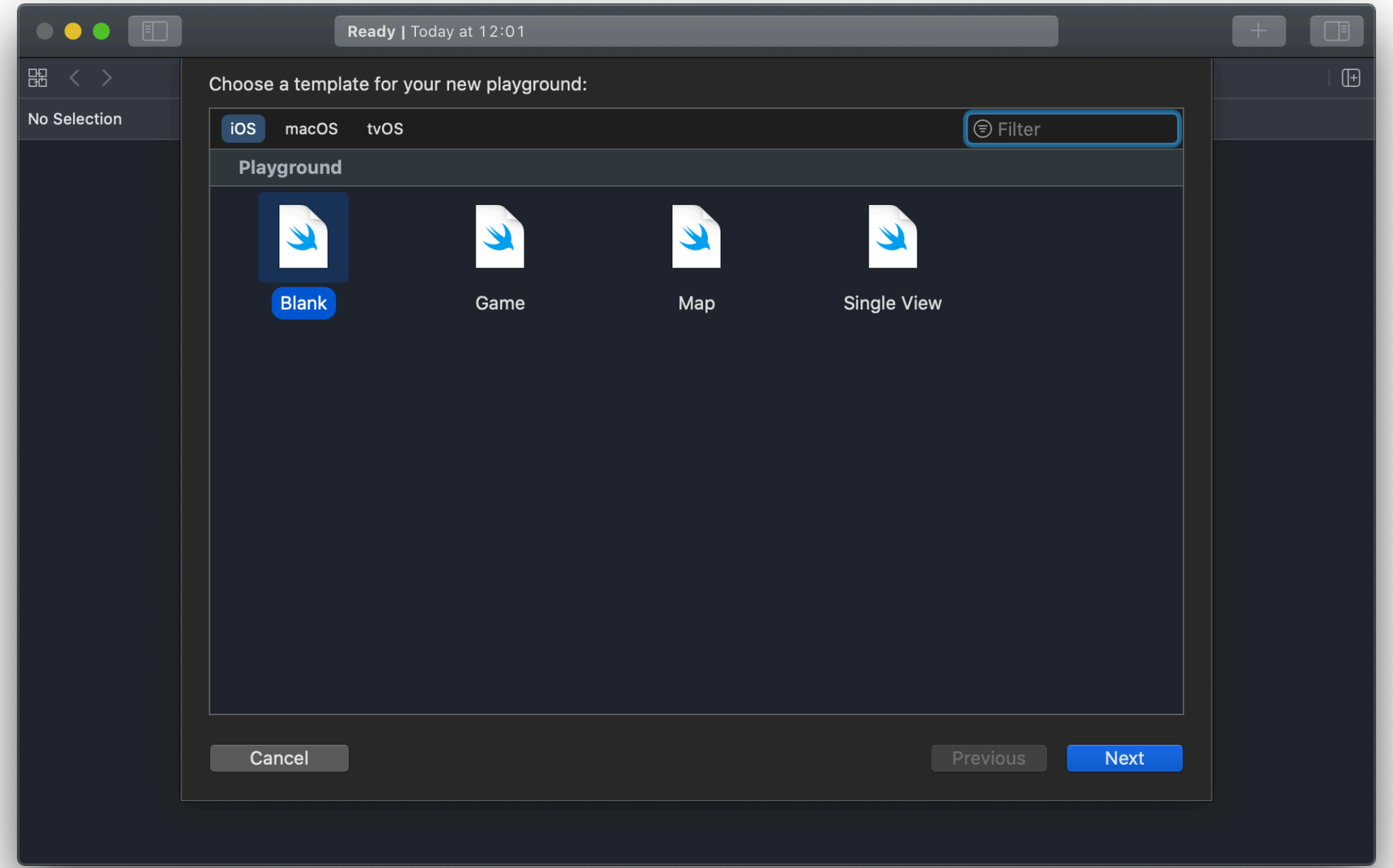
Xcode

- IDE fourni par Apple
- Permet de faire du C, C++, Objective-C, Objective-C++ et Swift
- Permet de réaliser des applications pour iOS, iPadOS, watchOS, etc.

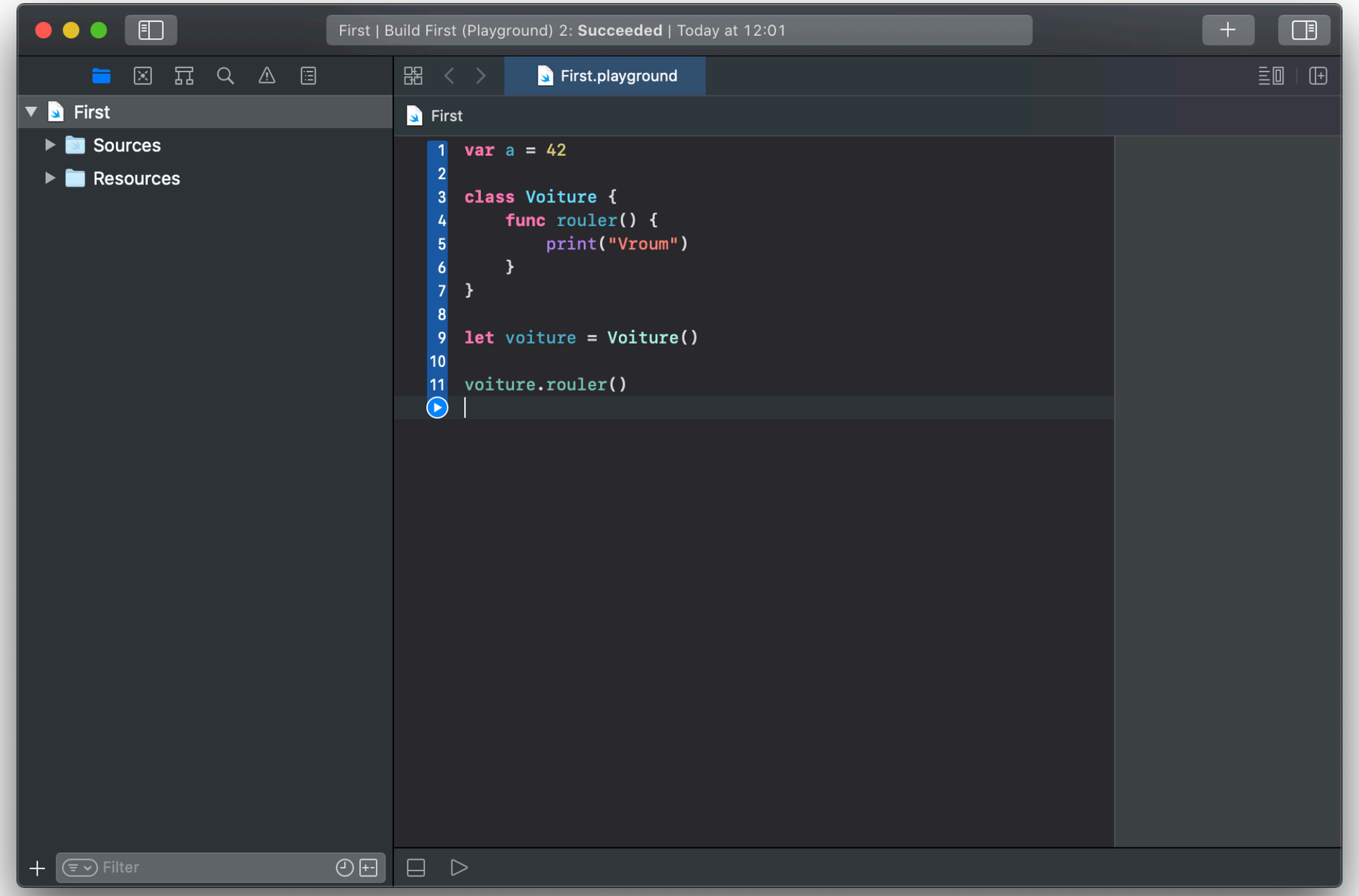
Xcode

Pour créer un nouveau « Playground » (Bac à sable), aller dans le menu « File » > « New » > « Playground »

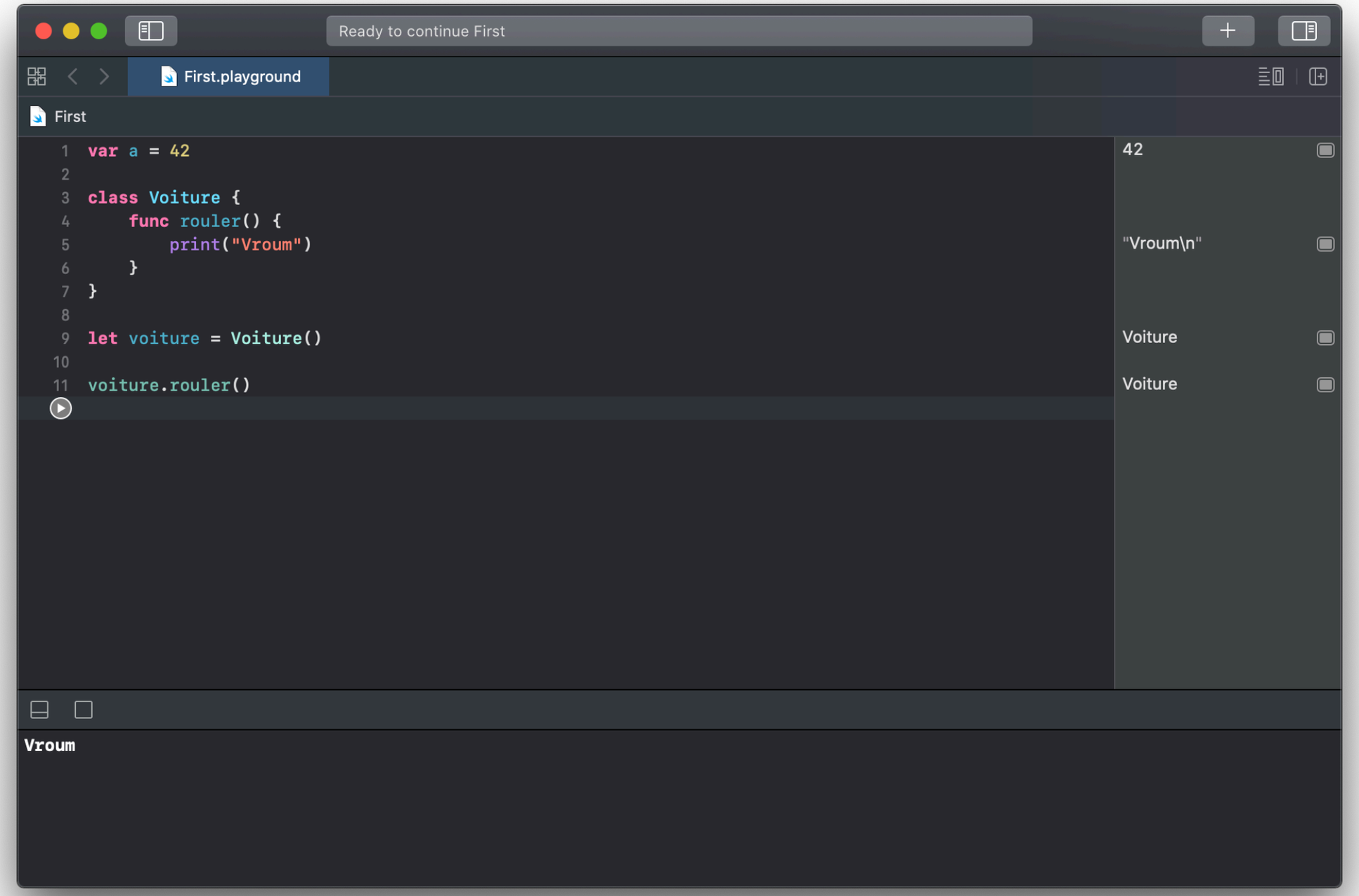
Xcode


Ready | Today at 12:01
No Selection
Choose a template for your new playground:
iOS
macOS
tvOS
Filter
Playground
Blank
Game
Map
Single View
Cancel
Previous
Next

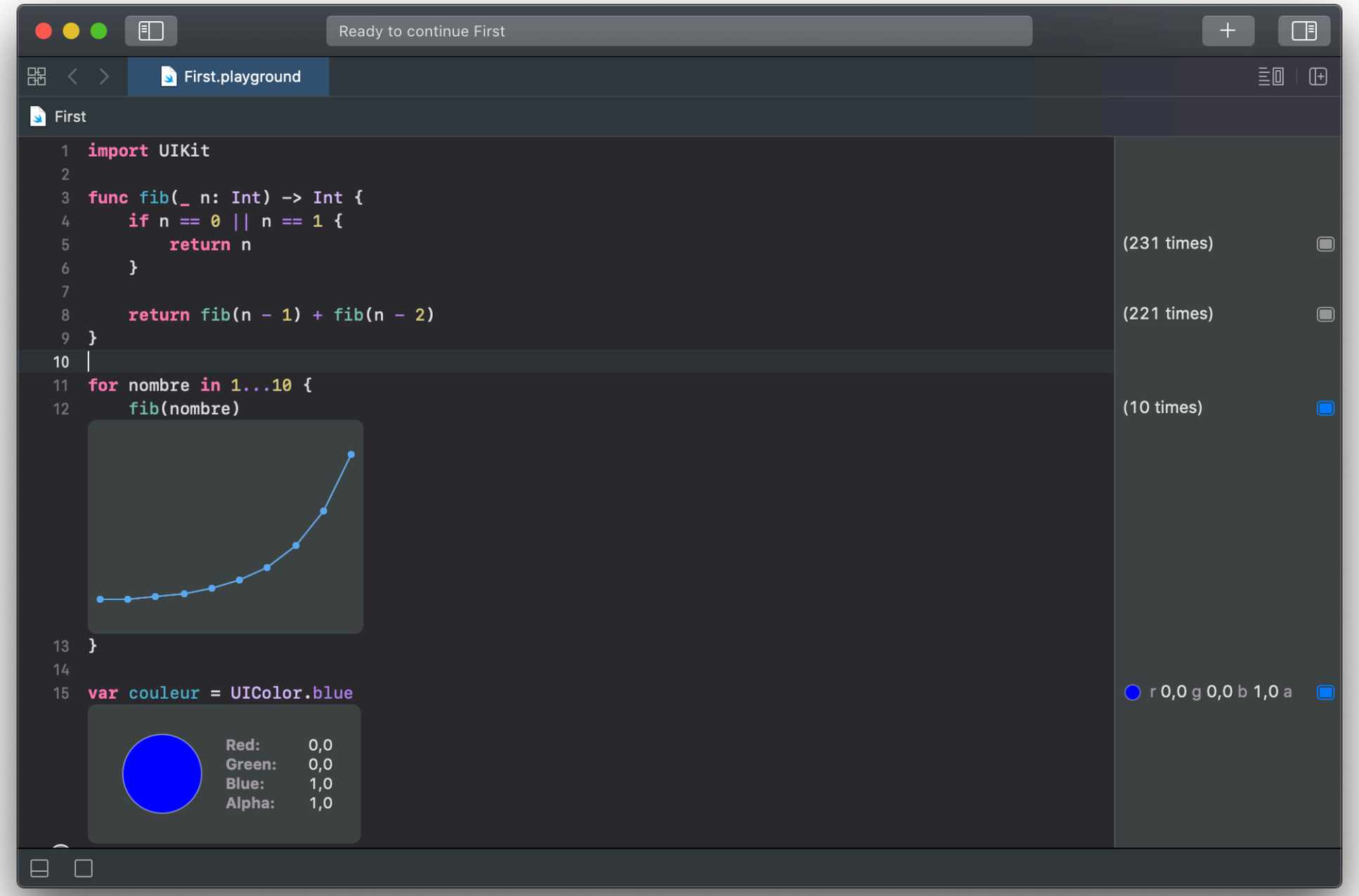
Xcode


First | Build First (Playground) 2: Succeeded | Today at 12:01
First.playground
First
Sources
Resources
First
var a = 42
class Voiture {
func rouler() {
print("Vroum")
}
}
let voiture = Voiture()
voiture.rouler()
Filter

Xcode


Ready to continue First
First.playground
First
1 var a = 42
2
3 class Voiture {
4 func rouler() {
5 print("Vroum")
6 }
7 }
8
9 let voiture = Voiture()
10
11 voiture.rouler()
42
"Vroum\n"
Voiture
Voiture
Vroum

Xcode


Ready to continue First
First.playground
First
import UIKit
func fib(_ n: Int) -> Int {
if n == 0 || n == 1 {
return n
}
return fib(n - 1) + fib(n - 2)
}
for nombre in 1...10 {
fib(nombre)
}
var couleur = UIColor.blue
Red: 0,0
Green: 0,0
Blue: 1,0
Alpha: 1,0
(231 times)
(221 times)
(10 times)
r 0,0 g 0,0 b 1,0 a

Quelques liens

- Livre Swift officiel (anglais) : https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html

- Exercices Swift (anglais) : https://exercism.io/tracks/swift