Chapitre 3
Le langage Swift

1. Introduction

Depuis iPhone OS 2.0 apparu en 2008, le langage de choix pour le développement sur iPhone était Objective-C, déjà utilisé par Apple depuis longtemps. En 2014, Apple a présenté Swift, un nouveau langage qui devint rapidement la voie recommandée pour le développement sur ses plates-formes. Ce chapitre présente les bases nécessaires pour comprendre le code rédigé par la suite. Les notions plus avancées seront présentées au cours des chapitres suivants.

Swift propose un environnement de développement clair et sécurisé sans pour autant faire de compromis sur la performance. Sa syntaxe s'accorde avec celle des autres langages de la famille C (C++, C#, Java, JavaScript) tout en conservant le côté expressif que pouvait avoir Objective-C. On y retrouve la plupart des notions de développement orienté objet ainsi que des fonctionnalités plus sophistiquées.

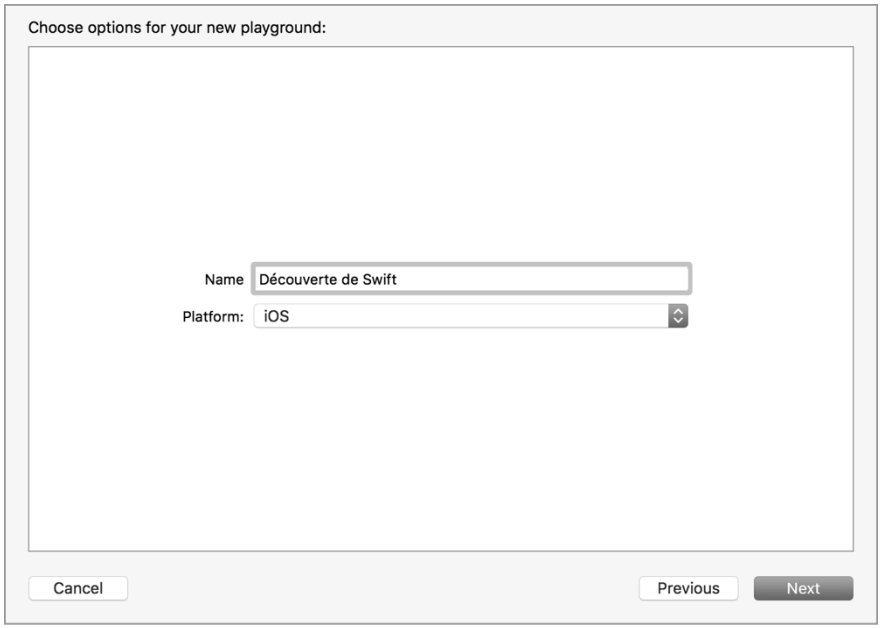
Il est important de noter que lors du développement sur iOS, la plupart des API (*Application Programming Interface*) que l'on utilise sont encore en Objective-C, mais un pont permet de les utiliser dans son code Swift sans avoir à s'en soucier. Swift et Objective-C savent s'exécuter dans le même environnement (runtime) et leurs syntaxes d'appel peuvent être traduites dans les deux sens : les deux langages cohabitent sans problème au sein de la même application.

2. Création d'un Playground

Afin de suivre au mieux les exemples de code qui suivent, il est conseillé de créer un *Playground* (terrain de jeu). Il s'agit d'une fonction d'Xcode qui permet d'écrire du code Swift et de visualiser en temps réel le résultat de chaque ligne. C'est de loin la meilleure façon d'expérimenter le langage car elle ne nécessite pas la création d'un projet complet ni la répétition du cycle de compilation.

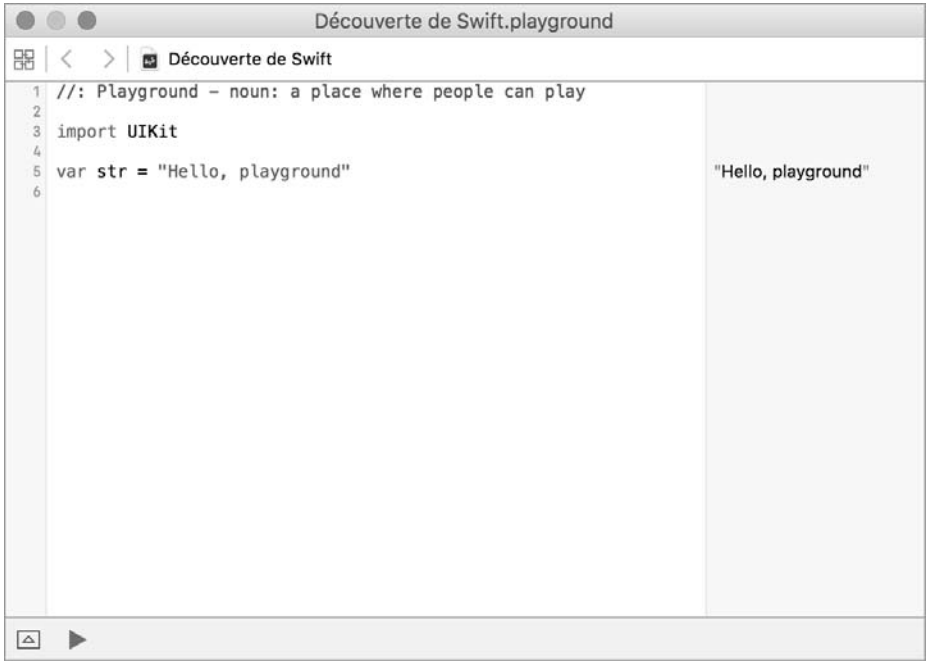
Créez maintenant un nouveau **Playground** :

- Dans le menu **File - New**, choisissez l'option **Playground**.
- Dans la fenêtre qui s'ouvre, choisissez un nom pour votre Playground (exemple : "Découverte de Swift") et assurez-vous que la plateforme choisie est bien **iOS**.



- Dans la fenêtre suivante, choisissez l'endroit où vous souhaitez enregistrer votre Playground.

- Le nouveau Playground s'affiche dans une nouvelle fenêtre d'Xcode :



La colonne de gauche permet d'écrire le code Swift. Trois lignes sont écrites par défaut. Dans la colonne de droite s'affichent des informations relatives à certaines lignes de code, on peut voir ici **Hello Playground** en regard de la ligne 5.

- Modifiez la valeur entre guillemets dans le code et observez les changements dans la colonne de droite.
- Avant de suivre les exemples de code suivants, effacez tout le contenu du Playground.

3. Variables

Le code sert d'ordinaire à manipuler des valeurs par le biais de variables. La déclaration d'une nouvelle variable s'effectue avec le mot-clé `var`.

```
var nomDeVariable = valeur
```

Le code suivant permet de déclarer une variable nommée `hello` contenant la chaîne de caractères **Hello World!**.

```
var hello = "Hello World!"
```

Il est possible d'attribuer par la suite une nouvelle valeur à la variable `hello` de cette manière :

```
var hello = "Hello World!"
hello = "Hello Swift!"
```

La colonne de droite confirme la valeur assignée à la variable.

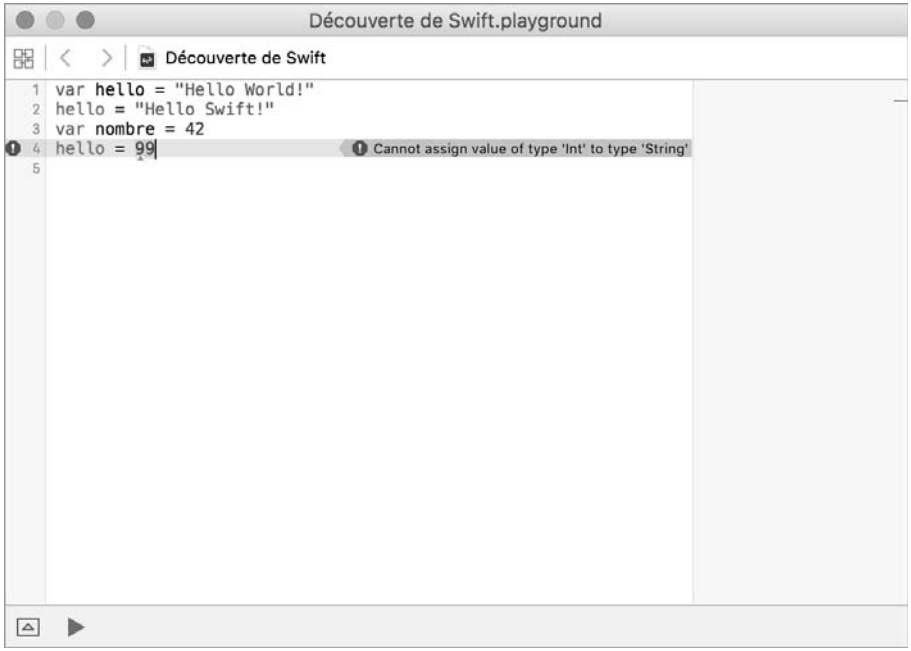
Il est possible d'assigner une valeur numérique à une nouvelle variable :

```
var hello = "Hello World!"
hello = "Hello Swift!"
var nombre = 42
```

Néanmoins, si l'on essaye d'assigner un nombre à la variable `hello` il survient une erreur.

```
var hello = "Hello World!"
hello = "Hello Swift!"
var nombre = 42
hello = 99
```

▶Cliquez sur la pastille rouge dans la marge pour afficher le message d'erreur.



L'erreur affichée est "*Cannot assign value of type 'Int' to type 'String'*" (Impossible d'assigner une valeur de type entier vers le type chaîne de caractères).

Swift est un langage dit fortement typé : une fois le type d'une variable défini, il est impossible de lui assigner une valeur d'un autre type.

Remarque

Le compilateur déduit le type d'une variable en fonction de la valeur qu'on lui attribue. Ce système d'inférence de type est couvert plus tard dans ce chapitre.

▶Effacez la ligne provoquant l'erreur.

Remarque

Lorsqu'une erreur survient dans un Playground, elle empêche son fonctionnement et l'affichage des valeurs dans la colonne de droite n'est plus mis à jour ; mieux vaut les traiter dès qu'elles surviennent.

Le mot-clé `let` permet de définir une nouvelle constante. Il est conseillé de l'utiliser en priorité à moins que la valeur soit censée changer.

```
let nomDeConstante = valeur
```

Il est ainsi possible d'ajouter une constante au Playground :

```
var hello = "Hello World!"
hello = "Hello Swift!"
var nombre = 42
let constNombre = nombre
```

Comme `constNombre` est une constante, il est impossible de lui assigner une nouvelle valeur. L'assignation d'une nouvelle valeur provoque une erreur :

```
var hello = "Hello World!"
hello = "Hello Swift!"
var nombre = 42
let constNombre = nombre
constNombre = 21
```

Le message d'erreur affiché indique "*Cannot assign to value: 'constNombre' is a 'let' constant*" (Impossible d'assigner à la valeur: 'constNombre' est une constante 'let'). Cette fois, une pop-up apparaît sous la ligne incriminée et propose un *Fix-it*, une correction immédiate qui permet de corriger l'erreur. Dans ce cas, la correction proposée est de changer `let` en `var` à la déclaration de notre constante.

▶Effacez plutôt la ligne provoquant l'erreur.

4. Types

On retrouve les types standards habituels :

– Nombres : `Int`, `Float`, `Double`
– Textes : `String`, `Character`
– Booléen : `Bool`
– Collections : `Array<Element>`, `Dictionary<Key: Hashable, Value>`, `Set<Element: Hashable>`