INFORMATIQUE
TECHNIQUE

Expert

Swift 3 pour iPhone

Développez vos premières applications mobiles

Fichiers complémentaires à télécharger

Pascal BATTY

eni

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EI3SWI** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 1
Avant-propos

1. Prérequis 15
2. À qui s'adresse cet ouvrage ? 16
3. Méthode d'enseignement 16
4. Matériel et logiciel nécessaires 16
5. Utilisation du contenu téléchargeable 17
6. État de Swift 17

Chapitre 2
Une application iOS

1. Introduction 19
2. Modèle-Vue-Contrôleur 19
3. Application Cliqueur 21
   - 3.1 Présentation 21
   - 3.2 Contexte client 21
4. Création d'un projet d'application 22
5. Interface d'Xcode 25
   - 5.1 Barre d'outils 26
   - 5.2 Éditeur 27
     - 5.2.1 Barre de saut 29
     - 5.2.2 Raccourcis-clavier utiles 29
   - 5.3 Navigateur 30
   - 5.4 Utilitaires 31
   - 5.5 Zone de débogage 33
   - 5.6 Aide et documentation 33

6. Éléments du modèle d'application 34
6.1 Réglages du projet 36
6.2 Autres éléments du modèle 38
7. Premier lancement dans le simulateur 39
8. Création de l'interface 40
8.1 Mise en place 40
8.2 Personnalisation 44
9. Interaction entre la vue et le code 46
9.1 Ouverture de l'éditeur assistant 46
9.2 Création d'une Prise 48
9.3 Création d'une Action 51
9.4 Déconnexion et reconnexion 52
10. Implémentation du View Controller 56
10.1 Modélisation du compte actuel 56
10.2 Définition de l'état initial 57
10.3 Réaction aux actions 58
11. En résumé 59
12. Pour aller plus loin 59

Chapitre 3
Le langage Swift

1. Introduction 61
2. Création d'un Playground 62
3. Variables 64
4. Types 66
5. Inférence de type et déclaration explicite 67
6. Collections 69
6.1 Tableaux 69
6.2 Dictionnaires 70
6.3 Ensembles 72
7. Constructeurs 72

8. Propriétés et méthodes d'instance 74
8.1 Propriétés 74
8.2 Méthodes d'instance 74
9. Valeurs optionnelles 76
10. Boucles et conditions 79
11. Contrôle d'accès 80
11.1 Module et fichier 80
11.2 Modificateurs d'accès 81
11.3 La différence entre public et open 81
12. En résumé 82

Chapitre 4
Contrôles textuels et délégation

1. Introduction 83
2. L'application Décodeur 83
2.1 Présentation 83
2.2 Contexte client 84
3. Création de l'interface 85
4. Bases d'Auto Layout 87
4.1 Hiérarchie des vues 88
4.2 Attributs et contraintes 92
4.3 Positionnement du titre 94
4.4 Positionnement du champ texte et du libellé 98
4.5 Modifier une contrainte existante 103
5. Écriture de fonctions 104
5.1 Signature d'une fonction 104
5.2 Nom des paramètres et code expressif 104
5.3 Implémentation de la méthode décoder(message:) 106
6. Implémentation du View Controller 106
6.1 Branchement des Prises et des Actions 106
6.2 Implémentation des méthodes 107

7. Délégation 109
7.1 Présentation 109
7.2 Protocole 109
7.3 Utilisation de la délégation 110
7.4 Connexion du délégué 111
7.5 Empêcher certains changements 113
8. Affichage d'une alerte 114
8.1 Ajout de constantes 114
8.2 Implémentation de la méthode 114
9. En résumé 117
10. Pour aller plus loin 117

Chapitre 5
Combinaison de vues

1. Introduction 119
2. Contrôles de navigation 119
2.1 Tab Bar Controller 120
2.2 Navigation Controller 121
3. L'application Visiteur 122
3.1 Présentation 122
3.2 Contexte client 122
4. Création d'un Tab Bar Controller 123
4.1 View Controller Initial 125
4.2 Propriétés des onglets 127
5. Insertion d'une scène existante 130
5.1 Insertion du ViewController 131
5.2 Insertion de la Scène 132
5.3 Segues de relation 133
5.4 Masquage du clavier 136
6. Mise en forme avec Scroll View et Stack View 138
6.1 Scroll View 138
6.2 Stack View 139

6.3 Mise en forme 140
7. Ajout d'une vue modale de connexion 150
7.1 Segue de navigation 153
8. Segue de débobinage 155
8.1 Mise en place du Segue de retour vers l'écran Visiteur 156
9. Passage d'informations lors du débobinage 158
10. Modification de la vue Visiteur 162
10.1 Affichage de l'e-mail dans un libellé 162
10.2 Ajout du bouton Déconnexion 165
10.3 Factorisation à l'aide d'un Observateur de Propriété 166
11. En résumé 167
12. Pour aller plus loin 168

Chapitre 6
Animations

1. Introduction 169
2. Closures 170
2.1 Présentation 170
2.2 Simplification de la syntaxe 171
2.3 Capture 172
3. Généralités sur les animations 172
3.1 Animation de propriété 172
3.2 La fonction animate 173
3.3 Courbe d'animation 174
4. Disparition d'un élément 175
5. Contraintes, cadre et transformation 177
5.1 Changement de la position d'un élément 177
5.2 Contraintes 177
5.3 Cadre 177
5.4 Transformations 179
5.5 Choix de la méthode d'animation 180
6. Animation des coordonnées du cadre 182

7. Animation de contraintes 182
8. Animation de la transformation 184
9. En résumé 185

Chapitre 7
Liste d'éléments

1. Introduction 187
2. Contrôle de liste avec Table View 188
   2.1 Table View 188
      2.1.1 Relation avec le délégué et la source de données 189
      2.1.2 IndexPath 190
      2.1.3 Table View Controller 190
   2.2 Cellules 191
      2.2.1 Composition 191
      2.2.2 Accessoires 191
      2.2.3 Styles de cellules 192
      2.2.4 Performances et réutilisation des cellules 193
   2.3 Sections 194
3. L'application Agenda 195
   3.1 Présentation 195
   3.2 Contexte client 195
4. Réalisation de la liste d'éléments 196
   4.1 Création du projet 196
   4.2 Configuration de la cellule 199
   4.3 Création de la classe Table View Controller 199
   4.4 Premier test de la source de données 201
5. Types Valeur : structure et énumération 205
   5.1 Types valeur et types référence 205
      5.1.1 Copie à l'assignation 205
      5.1.2 Immutabilité 206
   5.2 Structure 207
   5.3 Énumération 208

6. Création d'objets modèle 209
6.1 La structure Événement 209
6.2 Propriétés et méthodes statiques 210
6.2.1 Explication 210
6.2.2 Tous les Evénements 210
6.3 Propriétés calculées et accesseurs 211
6.3.1 Explication 211
6.3.2 Nom des salles 212
6.4 Utilisation dans le Table View Controller 213
7. Navigation Master-Detail avec Navigation Controller 216
7.1 Navigation Controller 216
7.2 Principe de fonctionnement 216
7.3 Barre de navigation et Navigation Item 217
7.4 Barre d'outils 219
7.5 Combinaison avec le Tab Bar Controller 219
8. Ajout de l'écran de détails 220
8.1 Mise en place du Navigation Controller 220
8.2 Ajout du Segue de navigation 225
8.3 Conception de l'écran de détails 226
8.4 Injection de l'événement 229
9. Personnalisation 231
9.1 Création d'un modèle de cellule personnalisée 231
9.2 Personnalisation de la couleur globale 238
10. En résumé 240
11. Pour aller plus loin 240

Chapitre 8
Stockage d'informations

1. Introduction 241
2. Gestion des erreurs en Swift 242
2.1 Présentation 242
2.2 Cas d'exemple 242

2.3 Émission d'erreurs 243
2.4 Gestion de l'erreur avec catch 244
2.5 Essai conditionnel avec try? 245
2.6 Essai risqué avec try! 246
2.7 Remonter l'erreur avec throws 246
3. Le bac à sable de l'application 246
3.1 Concept 246
3.2 Bundle Container 247
3.3 Data Container 248
4. Accès aux fichiers 249
4.1 Ressources du Bundle 249
4.2 Images du bundle 250
4.3 Ressources dans le Data Container 250
4.4 Document 252
5. Manipulation de données 253
5.1 Data 253
5.1.1 Lecture et écriture 253
5.1.2 Changement en chaîne de caractères 253
5.1.3 Changement en image 254
5.2 Encodage et décodage 254
5.3 Archivage 257
6. Réglages utilisateur 258
6.1 Principe 258
6.2 UserDefaults 258
6.3 Trousseau d'accès 259
7. En résumé 260

Chapitre 9
Édition d'une liste d'éléments

1. Introduction 261
2. Animation de changements sur la liste 262
2.1 Rechargement simple 262

2.2 Modification des lignes 262
2.2.1 Principe 262
2.2.2 Rafraîchissement, suppression et insertion 263
2.2.3 Déplacement 265
2.3 Modification de sections 266
2.4 Changements en simultané 266
3. Réaction aux modifications de l'utilisateur 267
3.1 Interactions 267
3.2 Mode édition 268
3.3 Suppression d'une ligne 270
3.4 Déplacement d'une ligne 271
3.5 Suppression sans mode édition 272
4. L'application Notes 273
5. Préparation de l'application 274
6. Ajout du modèle de note 277
6.1 Code source 277
6.2 Énumération Couleur 279
6.2.1 Type Imbriqué 279
6.2.2 Énumération avec valeur brute 279
6.2.3 Propriété d'énumération 280
6.3 Valeurs par défaut 280
6.4 Propriétés 281
6.5 Déduction du titre 281
7. Liste de notes 282
7.1 Valeurs par défaut 285
7.2 Propriétés 285
7.3 Cycle de vie 285
7.4 Archivage, sauvegarde et chargement 286
7.4.1 Méthodes 286
7.4.2 Transformation de tableaux avec map et flatMap 286
7.4.3 Sauvegarde et chargement 287
7.5 Manipulation des notes, actions et méthodes de source de données 287

8. Implémentation de la délégation 288
8.1 Présentation 288
8.2 Principe 288
8.3 Références faibles et cycle de rétention 289
8.3.1 Gestion de la mémoire et ARC 289
8.3.2 Cycle de rétention et fuite mémoire 289
8.3.3 Références faibles 290
8.3.4 Convention pour la Délégation 291
9. Édition d'une note 291
9.1 Délégation 291
9.2 Le View Controller d'édition de note 293
9.2.1 Propriétés 295
9.2.2 Cycle de vie 295
9.2.3 Manipulation de la vue 295
9.3 Modification de la liste 296
10. En résumé 298
11. Pour aller plus loin 298

Chapitre 10
Géolocalisation et plans

1. Introduction 299
2. Extensions 300
2.1 Retour sur l'héritage 300
2.2 Extension de types 301
2.3 Extension de protocole 302
3. Géolocalisation avec Core Location 305
3.1 Présentation 305
3.2 Position géographique avec CLLocation 305
3.3 Géolocalisation avec CLLocationManager 306
3.3.1 Présentation 306
3.4 Disponibilité de la géolocalisation 307
3.5 Demande de géolocalisation 308

3.6 Impacts sur l'autonomie 310
4. Plans avec MapKit 311
4.1 Présentation 311
4.2 Présentation d'un plan 311
4.3 Affichage d'annotations 314
4.3.1 MKAnnotation 314
4.4 MKAnnotationView et délégation 316
5. L'application 7 Merveilles 318
6. Création du projet 319
6.1 Ajout des frameworks MapKit et CoreLocation 319
6.2 Ajout du texte de justification 322
6.3 Mise en place de l'interface 323
7. Demande de l'autorisation de géolocalisation 324
8. Ajout du modèle Merveille 326
9. Création du plan 327
10. Simulateur, plans et géolocalisation 330
10.1 Pincer et zoomer sur le simulateur 330
10.2 Géolocalisation sur le simulateur 331
11. Création de la liste 332
12. Affichage de la page associée 336
12.1 UIWebView et SFSafariViewController 336
12.2 Extension de UIViewController 337
13. En résumé 339
14. Pour aller plus loin 339

Chapitre 11
Caméra et photos

1. Introduction 341
2. Récupération de photo avec UIImagePickerController 342
2.1 Présentation 342
2.2 Autorisations 342
2.3 Présentation du récupérateur d'images 343

2.4 Disponibilité des sources de capture 344
2.5 Délégation 345
3. Exécution asynchrone et en parallèle 347
3.1 Principe 347
3.2 File d'acheminement 348
3.3 Exécution en file 349
4. Traitement des images 351
4.1 Présentation 351
4.2 Redimensionnement d'image 351
4.3 Exécution sur une file d'acheminement 353
5. En résumé 355

Chapitre 12
Accès aux services web

1. Introduction 357
2. Gestion de tâches réseau avec URLSession 358
2.1 Principe 358
2.2 Création de la session 358
2.3 Création de requête 358
2.4 Types de tâches 359
2.5 Création d'une tâche avec closure 359
2.6 Démarrer et interrompre une tâche 361
2.7 Utilisation de la délégation 362
2.8 Sécurité et App Transport Security 365
3. Exploitation de données JSON 366
3.1 Principe 366
3.2 Format JSON 366
3.3 Interprétation de JSON 367
3.4 Sérialisation en JSON 370
4. En résumé 371

Chapitre 13
Gestes et dessin

1. Introduction 373
2. Réaction au toucher avec UIResponder 374
   2.1 Présentation 374
   2.2 UITouch 375
   2.3 La chaîne de réponse 375
3. Dessin d'une vue personnalisée 377
   3.1 Présentation 377
   3.2 Méthode draw 378
   3.3 Dessin dans un CGContext 378
4. Réaction aux gestes avec UIGestureRecognizer 380
   4.1 Présentation 380
   4.2 UIGestureRecognizer 381
   4.3 Action 382
   4.4 Dépendances 384
   4.5 Délégation 385
5. Définition d'un View Controller en code 386
   5.1 Présentation 386
   5.2 Méthode loadView() 387
   5.3 Ajout d'une sous-vue 388
   5.4 Positionnement avec un masque de redimensionnement 390
   5.5 Positionnement en Auto Layout 391
6. En résumé 393

Chapitre 14
Débogage

1. Introduction 395
2. Points d'arrêt et pas-à-pas 396
   2.1 Présentation 396
   2.2 Mise en place d'un point d'arrêt 396
   2.3 Exécution avec un point d'arrêt 397

2.4 Mise en pause 399
3. Suivi des jauges 399
4. Débogage de la hiérarchie des vues 401
5. En résumé 402

Chapitre 15
Installation sur un appareil et déploiement

1. Introduction 403
2. Prérequis 404
3. Signature de code et profil d'approvisionnement 404
   3.1 Principe 404
   3.2 Environnements 405
      3.2.1 Développement 405
      3.2.2 Déploiement ad hoc 405
      3.2.3 Déploiement App Store 406
   3.3 Déploiement Entreprise 407
4. Xcode et signature du code 407
   4.1 Ajout d'un compte à Xcode 407
   4.2 Génération des certificats 410
   4.3 Définition de l'identité de signature pour une application . . . 411
   4.4 Lancement en débogage sur un appareil 412
   4.5 Création d'une archive pour déploiement 413
5. En résumé 414

Le mot de la fin 415

Index 417

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.

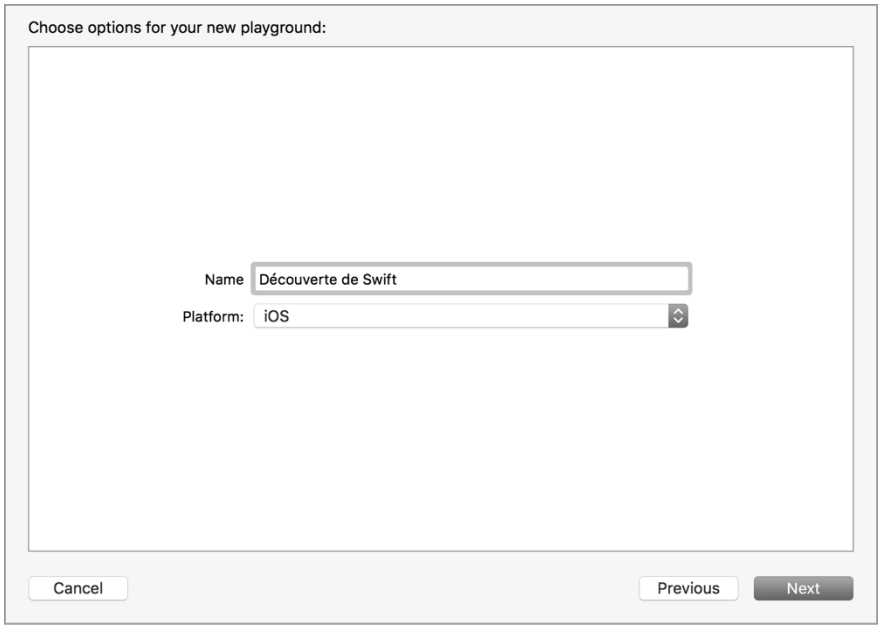
Il est important de noter que lors du développement sur iOS, la plupart des API (*Application Programming Interface*) que l'on utilise sont encore en Objective-C, mais un pont permet de les utiliser dans son code Swift sans avoir à s'en soucier. Swift et Objective-C savent s'exécuter dans le même environnement (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.

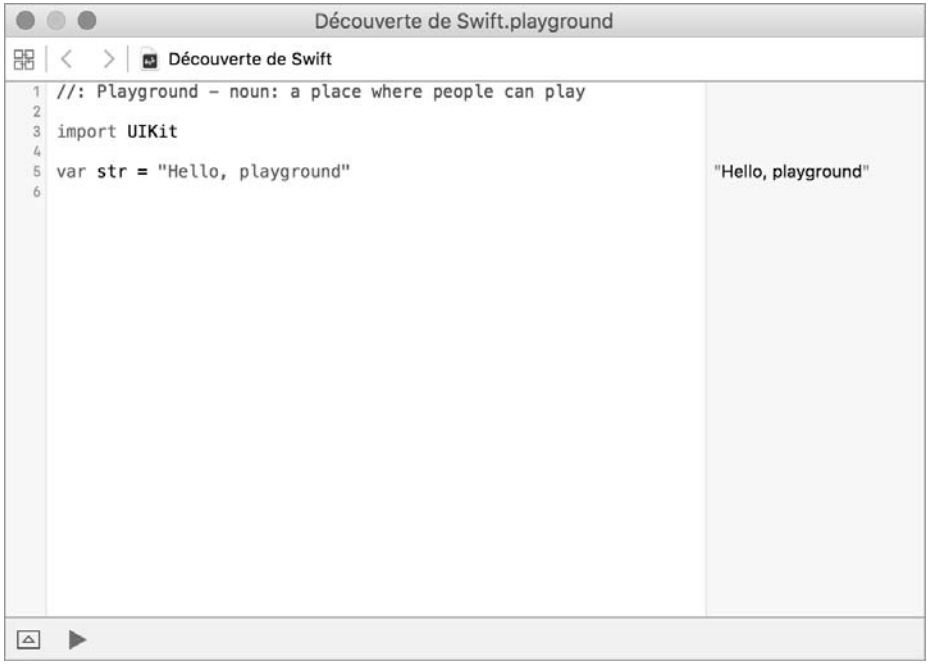
Créez maintenant un nouveau **Playground** :

- 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.

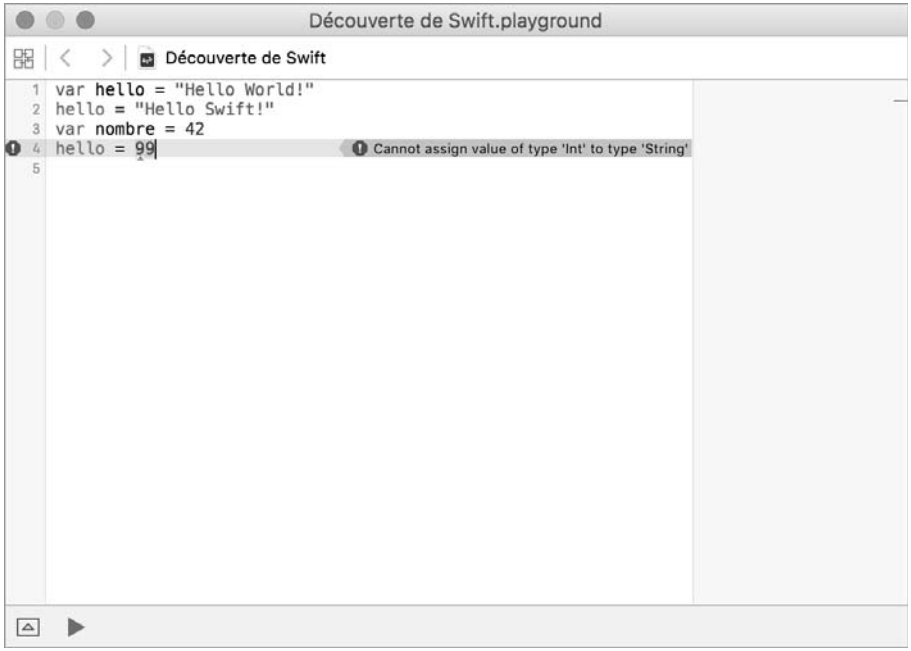
Il est possible d'assigner une valeur numérique à une nouvelle variable :

```
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>`