

# **Les variables et constantes en Dart**

# Table des matières

<b>I. Variables et leurs types</b>	<b>3</b>
A. Comprendre la notion de variable.....	3
B. Types de variables.....	4
C. Variables d'instance et programmation orientée objet.....	4
<b>II. Exercice : Quiz</b>	<b>4</b>
<b>III. Variables finales et constantes</b>	<b>5</b>
A. Variables finales .....	5
B. Constantes.....	6
<b>IV. Exercice : Quiz</b>	<b>7</b>
<b>V. Essentiel</b>	<b>8</b>
<b>VI. Auto-évaluation</b>	<b>8</b>
A. Exercice .....	8
B. Test.....	8
<b>Solutions des exercices</b>	<b>9</b>

## I. Variables et leurs types

### Contexte

Nous avons tous un prénom, et il y a bien une raison à cela ! Le prénom nous permet de faire référence à des personnes, de pouvoir les différencier, et même de pouvoir les appeler.

En programmation, puisque nous devons utiliser et faire référence à bon nombre d'objets (des nombres, des chaînes de caractères, etc.), nous avons dû utiliser un peu la même logique.

Dans ce cours, nous allons vous apprendre l'indispensable science des noms en Dart ! Nous verrons dans un premier temps les variables et leurs différents types, avant d'étudier les constantes et variables finales.

### A. Comprendre la notion de variable

Dans un programme, nous avons souvent besoin de manipuler bon nombre de données. Pour s'y retrouver, il est souvent nécessaire de nommer ces données, autrement dit de les stocker dans ce que nous appelons une « *variable* ».

Une variable est ainsi composée de 3 éléments :

- Un type : une variable peut avoir tous types de valeurs. Par exemple, nous pouvons stocker aussi bien du texte dans une variable qu'un nombre.
- Un nom : une variable a obligatoirement un nom, choisi à la discrétion du programmeur.
- Une valeur : ce que contient la variable.

### Exemple

```
1 Int nomdemavariabale = 4.
```

Ici, le type de `nomdemavariabale` est `Int`, et sa valeur est 4.

### Remarque

Il existe une méthode pour assigner la valeur à une variable plus tard dans une fonction. En écrivant `late String nomdemastring = « mavalueur »`, vous n'assignez aucune valeur à votre variable (même pas `null`). Néanmoins, si vous oubliez de lui donner une valeur plus tard, vous aurez des erreurs au moment de l'exécution de votre programme et de l'utilisation de cette variable. Il ne faut donc pas oublier !

### Exemple

```
1 late String nom;
2
3 void main() {
4   nom = 'Theodore';
5   print(nom);
6 }
```

### Méthode

#### Déclarer une variable en DART

## B. Types de variables

En Dart, il existe une dizaine de types de variables différents. Les principaux sont :

- `num` : ce sont des nombres. Ces nombres peuvent être de type `int` ou de type `double`.
- `String` : ce sont les chaînes de caractères. Exemple : `String texte = 'montexte'`.
- `bool` : peut être de valeur « *vrai* » ou « *faux* ». Exemple : `bool abc = true`.
- `List` : dans certains langages, `List` est appelé `Array`. C'est une liste de valeurs contenue dans un tableau. Exemple : `List animaux = ['oiseau', 'cochon', 'chèvre']`.
- `Set` : très proche de `List`, le `Set` permet cependant de garder de façon désordonnée des données, qui peuvent être d'un seul type. De plus, une même valeur ne peut pas exister plusieurs fois dans un `Set`.
- `Map` : `Map` permet de créer des paires clé-valeur. Ainsi, chaque valeur sera liée à une clé. Par exemple : `{ 'monage' = 42 }`.

Dans le cas de `List`, `Map` ou `Set`, il est possible d'indiquer le type de données qu'ils contiennent. Cela se fait grâce aux `<>`. Ainsi, si nous souhaitons créer une liste de textes, nous pouvons initialiser notre variable avec le type `List<String>`. Pour une `Map` de nombres, ce sera `Map<num>`.

Si vous souhaitez déclarer une variable dont vous n'avez pas encore d'informations sur le type, vous pouvez également utiliser `var`. Exemple : `var resultatdemafunction = mafunction()` ; Plus vous êtes précis en indiquant les types de variables, mieux c'est !

## C. Variables d'instance et programmation orientée objet

Il vous est possible de créer vos propres variables, à savoir des variables d'instance. Une variable d'instance est elle-même constituée de variables et/ou de fonctions.

```
1 class Point {
2   double x = 0;
3   double y = 0;
4 }
```

### Complément

Une variable d'instance peut aussi être appelée « *objet* » et est utilisée dans la programmation orientée objet. La programmation orientée objet est un paradigme informatique, consistant à créer des interactions entre des « *objets* ». Par exemple, un bouton dans un site Internet peut être un objet qui interagit avec un autre objet, comme un autre bouton.

L'utilisation de ces objets pourrait nécessiter tout un cours, c'est pourquoi nous nous contenterons ici de savoir que, grâce aux variables d'instance, il nous est possible de créer des variables contenant d'autres variables et fonctions !

## Exercice : Quiz

[solution n°1 p.11]

### Question 1

Parmi ces affirmations, laquelle est vraie ?

- ☐ Une valeur est constituée de variables, d'un type et d'un nom
- ☐ Des variables sont constituées d'un type, d'autres variables et d'un nom
- ☐ Une variable est constituée d'un type, d'un nom et d'une valeur

### Question 2

Cochez la/les bonne(s) syntaxe(s).

- ☐ `nomdemavariab String = 'mavaleur';`
- ☐ `String nomdemavariab = 'mavaleur';`
- ☐ `String nomdemavariab = mavaleur;`
- ☐ `var nomdemavariab = 'mavaleur';`

#### Question 3

`late` est utilisée pour :

- ☐ Donner le nom d'une variable plus tard
- ☐ Donner la valeur d'une variable plus tard
- ☐ Donner la variable à une valeur plus tard

#### Question 4

Si je souhaite créer une liste de chaînes de caractères, quel est le type de variable que je peux déclarer ? Plusieurs réponses possibles.

- ☐ `Array<String>`
- ☐ `Map<String>`
- ☐ `List`
- ☐ `List<String>`

#### Question 5

Une variable peut contenir d'autres variables.

- ☐ Vrai
- ☐ Faux

## III. Variables finales et constantes

### A. Variables finales

Une variable finale est une variable qui ne change pas de valeur.

**Une variable qui n'est pas variable**, eh oui !

La variable finale se déclare comme suit :

```
1 final String prenom = 'Gertrude';
```

Après avoir initialisé votre variable comme suit, vous aurez une erreur à chaque fois que vous essaieriez de changer la valeur de la variable `prenom`.

```
1 prenom = 'bernard'; // Error: a final variable can only be set once.
```

**Remarque**

Peut-être vous demandez-vous l'intérêt de déclarer une variable finale, alors que vous pourriez simplement vous contenter de ne jamais en changer la valeur. Il y a deux intérêts principaux : la lisibilité et la sécurité. Plus vous donnez d'informations sur la variable au moment de sa déclaration, plus vous comprendrez quel était votre objectif au moment de sa déclaration ; et si jamais vous en changez la valeur sans faire exprès, Dart pourra vite vous rappeler à l'ordre !

## B. Constantes

Mais alors, s'il est possible de créer des variables finales, que peuvent bien être les constantes ?

Tout comme les variables finales, les constantes ne peuvent jamais changer de valeur, mais elles diffèrent dans le moment où la variable est initialisée.

Une variable `const` sera prise en compte au moment de la compilation du programme, et rangée de sorte à ce qu'elle soit disponible à tout instant, au cas où votre programme en aurait besoin.

La variable finale, elle, doit être initialisée pendant que le programme tourne. Si celle-ci doit être initialisée lors de l'utilisation d'une fonction, mais que cette dernière n'est finalement pas utilisée pendant tout le temps où le programme aura fonctionné, cela signifiera que la variable finale n'aura jamais été initialisée. Elle n'aura donc jamais existé !

Cette distinction, ainsi que l'utilisation des deux types de variables au sein d'un même programme, vous permettent d'avoir une meilleure gestion de la mémoire vive ou du poids de votre programme.

Une variable initialisée au moment de la compilation sera disponible tout de suite, mais rendra votre fichier compilé plus lourd. Une variable initialisée pendant l'utilisation de votre programme rendra le fichier compilé plus léger, mais peut prendre plus de temps à être disponible et ralentir votre programme.

**Méthode** Déclarer une constante en DART

**Remarque**

Vous vous souvenez quand nous parlions de `late` plus tôt dans ce cours ? `late` peut aussi être utilisé dans le cas où vous voudriez éviter l'initialisation de la valeur d'une variable qui pourrait être trop lourde, au cas où celle-ci n'est jamais appelée.

Par exemple, en écrivant : `late String dayoftheweek = dayOfTheWeek();`, la fonction `dayOfTheWeek` ne sera appelée que si la variable `dayoftheweek` est appelée.

Ceci est bien évidemment exagéré, car, pour vos premiers programmes, vous ne verrez pas la différence. C'est primordial à partir d'un certain niveau de complexité et d'un certain volume de données dans votre programme, mais autant commencer à prendre les bonnes habitudes dès maintenant !

Autre spécificité de la variable `const` : celle-ci n'est valable que si la valeur qui la constitue est disponible au moment de la compilation. Par exemple, vous pouvez tout à fait écrire `const num nomdemavariabale = 4 + 5`. Par contre, si vous essayez de mettre la valeur de la date d'aujourd'hui avec `DateTime.now()`, vous aurez une erreur.

Les variables constantes sont dites « *profondément* » immuables. Si vous déclarez par exemple une liste constante, tous les éléments de la liste seront constants.

En résumé :

- Les variables `const` sont initialisées au moment de la compilation,
- Elles doivent être créées à partir de données disponibles au moment de la compilation,
- Elles ne seront jamais dupliquées.

**Attention**

Les variables d'instance peuvent être finales, mais pas constantes !

**Exercice : Quiz**

[solution n°2 p.12]

## Question 1

Si je souhaite déclarer une variable dont la valeur ne changera pas tout au long de l'exécution de mon programme, que puis-je utiliser ?

- ☐ Une variable constante
- ☐ Une variable finale
- ☐ Une variable constante ou une variable finale

## Question 2

Si je déclare une instance variable constante, les variables contenues dans ma variable doivent elles aussi être constantes.

- ☐ Vrai
- ☐ Faux
- ☐ Cela n'est pas possible

## Question 3

Si je souhaite créer une variable dont je suis sûr d'utiliser la valeur, qu'elle ne prend pas beaucoup de place et que celle-ci ne changera pas, que devrais-je utiliser ?

- ☐ `const`, sous certaines conditions
- ☐ `final`

## Question 4

Je dois créer une variable dont la valeur dépend d'une fonction coûteuse en mémoire vive, que je ne suis pas sûr d'utiliser et dont le résultat ne peut être disponible au moment de la compilation. Quelle est la meilleure approche ?

- ☐ Créer une variable `late final`
- ☐ Créer une variable `const`
- ☐ Créer une variable `final`
- ☐ Créer une variable `late const`

## Question 5

Ma variable finale n'a jamais été appelée, qu'est-ce que cela signifie ?

- ☐ Elle n'a pas de valeur
- ☐ Elle a été stockée au moment de la compilation et pourra servir plus tard
- ☐ Elle a tout de même été créée, au cas où

## V. Essentiel

### 1. Les variables et leurs types :

- Une variable est ainsi composée de 3 éléments : un type, un nom, une valeur.
- Il existe une méthode pour assigner la valeur à une variable plus tard dans une fonction : `late`.
- En Dart, il existe une dizaine de types de variables différents. Les principaux sont : `num`, `String`, `bool`, `List`, `Set` et `Map`.
- Dans le cas de `List`, `Map` ou `Set`, il est possible d'indiquer le type de données qu'ils contiennent. Cela se fait grâce aux `<>`.
- Il est possible de créer vos propres types de variables, à savoir des variables d'instance. Une variable d'instance est elle-même constituée de variables et/ou de fonctions.

### 2. Les variables finales et les constantes :

- Une variable finale est une variable qui ne change pas de valeur.
- Tout comme les variables finales, les constantes ne peuvent jamais changer de valeur, mais elles diffèrent dans le moment où la variable est initialisée. Une variable constante est initialisée au moment de la compilation du programme.
- Cette distinction, ainsi que l'utilisation des deux types de variables au sein d'un même programme, vous permettent d'avoir une meilleure gestion de la mémoire vive ou du poids de votre programme.
- Les variables constantes ont notamment 3 spécificités :
  - Les variables `const` sont initialisées au moment de la compilation,
  - Elles doivent être créées à partir de données disponibles au moment de la compilation,
  - Elles ne seront jamais dupliquées.

## VI. Auto-évaluation

### A. Exercice

Si vous avez un peu joué aux jeux vidéo, vous avez alors sûrement eu l'occasion de pouvoir imaginer un avatar. De la couleur de ses yeux à son âge, en passant par sa tenue vestimentaire, ce sont autant de variables qui vous étaient proposées et qui vous permettaient de créer le personnage de vos rêves.

Il est temps de s'y replonger !

#### Question 1

[solution n°3 p.13]

Créez une instance `Personnage`, contenant elle-même les variables `prénom`, `couleur` de cheveux et `âge` dont les valeurs par défaut seront nulles. À vous d'indiquer ces variables comme finales si vous jugez qu'elles ne changeront jamais !

#### Question 2

[solution n°4 p.13]

Créez la même instance, mais rajoutez les valeurs par défaut « *Charlie* », « *bleue* », « *24* ».

#### Question 3

[solution n°5 p.13]

Imaginez qu'il existe une fonction `creerPersonnage()` qui permette de créer un personnage avec des caractéristiques aléatoires, mais dont l'exécution prend du temps. Dans ma fonction, je ne suis pas sûr d'avoir besoin de cette variable `personnage`. Comment puis-je déclarer ma variable ?

### B. Test

#### Exercice 1 : Quiz

[solution n°6 p.13]

Question 1



`late` peut être utilisée pour éviter l'initialisation de la valeur d'une variable qui pourrait être trop lourde.

- ☐ Vrai
- ☐ Faux

#### Question 2

Si je souhaite créer une `Map` de numéros, quel est le type de variable que je peux déclarer ?

- ☐ `Array<String>`
- ☐ `Map<num>`
- ☐ `Map`
- ☐ `Map<Map>`

#### Question 3

Il est possible de créer nos propres variables.

- ☐ Vrai
- ☐ Faux

#### Question 4

Je souhaite créer une variable dont la valeur ne changera pas. Néanmoins, la valeur n'est pas disponible au moment de la compilation. Je peux utiliser `const`.

- ☐ Vrai
- ☐ Faux

#### Question 5

Je peux avoir plusieurs données identiques dans `Set`.


- ☐ Vrai
- ☐ Faux

## Solutions des exercices



**Exercice p. 4 Solution n°1****Question 1**


Parmi ces affirmations, laquelle est vraie ?

- ☐ Une valeur est constituée de variables, d'un type et d'un nom
- ☐ Des variables sont constituées d'un type, d'autres variables et d'un nom
- ☒ Une variable est constituée d'un type, d'un nom et d'une valeur
-  Une valeur n'a pas de nom si celle-ci n'est pas stockée dans une variable. Et une variable n'est pas nécessairement constituée d'autres variables. La bonne réponse est donc la dernière !

**Question 2**


Cochez la/les bonne(s) syntaxe(s).

- ☐ `nomdemavariabale String = 'mavaleur';`
- ☒ `String nomdemavariabale = 'mavaleur';`
- ☐ `String nomdemavariabale = mavaleur;`
- ☒ `var nomdemavariabale = 'mavaleur';`

 Le type de la variable doit être annoncé avant le nom. Concernant la 3<sup>e</sup> option, si le type est `String`, `mavaleur` doit être une `String`, ce qui n'est pas le cas ici ! Les bonnes réponses sont : `String nomdemavariabale = 'mavaleur';` et `var nomdemavariabale = 'mavaleur';`. `var` peut être de tout type.

**Question 3**


`Late` est utilisée pour :

- ☐ Donner le nom d'une variable plus tard
- ☒ Donner la valeur d'une variable plus tard
- ☐ Donner la variable à une valeur plus tard
-  `Late` permet de déclarer une variable sans avoir à lui assigner de valeur, pas même la valeur `null`. Cependant, il ne faut pas oublier de lui assigner une valeur avant qu'elle ne soit appelée !

**Question 4**

Si je souhaite créer une liste de chaînes de caractères, quel est le type de variable que je peux déclarer ? Plusieurs réponses possibles.

- ☐ `Array<String>`
- ☐ `Map<String>`
- ☒ `List`
- ☒ `List<String>`


 `Array` n'existe pas en Dart. Il faut utiliser `List` à la place. `Map` n'est pas une liste. Les bonnes réponses sont les deux dernières options. L'indication du type de données contenues dans la liste (`<String>`) n'est pas obligatoire.

### Question 5

Une variable peut contenir d'autres variables.

☒ Vrai

☐ Faux

 Une variable peut tout à faire contenir d'autres variables ! Il s'agit alors d'une variable d'instance, qui peut être créée grâce à `class`.

### Exercice p. 7 Solution n°2


#### Question 1

Si je souhaite déclarer une variable dont la valeur ne changera pas tout au long de l'exécution de mon programme, que puis-je utiliser ?

☐ Une variable constante

☐ Une variable finale

☒ Une variable constante ou une variable finale

 Il vous est tout à fait possible de déclarer une variable `const` ou une variable finale. Le choix ne dépend que de vous et de votre gestion de la mémoire du programme !


#### Question 2

Si je déclare une instance variable constante, les variables contenues dans ma variable doivent elles aussi être constantes.

☐ Vrai

☐ Faux

☒ Cela n'est pas possible


 Il n'est pas possible de créer une instance variable constante. Par contre, il est possible de créer une variable constante contenant d'autres valeurs (dans une liste, par exemple). Dans ce cas de figure, ces valeurs sont toutes constantes !

#### Question 3

Si je souhaite créer une variable dont je suis sûr d'utiliser la valeur, qu'elle ne prend pas beaucoup de place et que celle-ci ne changera pas, que devrais-je utiliser ?

☒ `const`, sous certaines conditions

☐ `final`

 `const` est la bonne réponse, à condition que la valeur de la variable déclarée soit disponible au moment de la compilation du code.

#### Question 4

Je dois créer une variable dont la valeur dépend d'une fonction coûteuse en mémoire vive, que je ne suis pas sûr d'utiliser et dont le résultat ne peut être disponible au moment de la compilation. Quelle est la meilleure approche ?

- ☒ Créer une variable `late final`
- ☐ Créer une variable `const`
- ☐ Créer une variable `final`
- ☐ Créer une variable `late const`

🔍 Si le résultat ne peut être obtenu au moment de la compilation, la variable ne peut être constante. Ensuite, si nous ne sommes pas sûrs d'utiliser cette variable, la fonction nous donnant la valeur ne devrait être lancée que si nous utilisons cette variable. Cela est possible grâce à `late`.

### Question 5

Ma variable finale n'a jamais été appelée, qu'est-ce que cela signifie ?

- ☒ Elle n'a pas de valeur
  - ☐ Elle a été stockée au moment de la compilation et pourra servir plus tard
  - ☐ Elle a tout de même été créée, au cas où
- 🔍 Si votre variable `late` n'a jamais été appelée, cela signifie qu'elle n'a absolument aucune valeur qui lui a été assignée.

### p. 8 Solution n°3

```
1 class Personnage {  
2     final String prenom;  
3     String cheveux;  
4     num age;  
5 }
```

### p. 8 Solution n°4

```
1 class Personnage {  
2     final String prenom = 'Charlie';  
3     String cheveux = 'bleue';  
4     num age = 24;  
5 }
```

### p. 8 Solution n°5

```
1 Late Personnage personnage = creerPersonnage();
```

En effet, si l'exécution prend du temps et que nous ne sommes pas sûrs d'avoir à utiliser la variable, `late` permet de ne lancer la fonction uniquement si la variable est utilisée.


### Exercice p. 8 Solution n°6

### Question 1

`late` peut être utilisée pour éviter l'initialisation de la valeur d'une variable qui pourrait être trop lourde.

☒ Vrai

☐ Faux

 La variable `late` peut être utilisée pour éviter l'initialisation de la valeur d'une variable qui pourrait être trop lourde, au cas où celle-ci n'est jamais appelée.

### Question 2

Si je souhaite créer une `Map` de numéros, quel est le type de variable que je peux déclarer ?

☐ `Array<String>`

☒ `Map<num>`

☒ `Map`

☐ `Map<Map>`


 Préciser le type des variables contenues dans la `Map` n'est pas obligatoire, mais c'est mieux !

### Question 3

Il est possible de créer nos propres variables.

☒ Vrai

☐ Faux


 Il vous est possible de créer vos propres variables, plus précisément des variables d'instance. Une variable d'instance est elle-même constituée de variables et/ou de fonctions.

### Question 4

Je souhaite créer une variable dont la valeur ne changera pas. Néanmoins, la valeur n'est pas disponible au moment de la compilation. Je peux utiliser `const`.

☐ Vrai

☒ Faux


 Il n'est malheureusement pas possible d'utiliser `const`, car pour cela il faudrait que la valeur soit disponible au moment de la compilation. Il est cependant possible d'utiliser `final`.

### Question 5

Je peux avoir plusieurs données identiques dans `Set`.

☐ Vrai

☒ Faux

 Impossible d'avoir plusieurs valeurs identiques dans `Set` ! Pour cela, il faudrait utiliser `Map`.