Les opérateurs Dart

Table des matières

I. Définition d'un opérateur en programmation

Contexte

Dans la vie quotidienne, vous avez besoin de faire des calculs mathématiques grâce à des additions ou des soustractions. Il est donc logique de penser qu'il est aussi indispensable, dans un programme informatique, d'effectuer des opérations de ce genre.

Les opérateurs sont toutes les fonctions basiques pour effectuer des opérations spécifiques, manipuler des variables, les comparer, etc. Ils vont permettre d'indiquer de façon simple l'opération à faire ; par exemple, une addition ou une multiplication. Les opérateurs sont utilisés dans tous les langages de programmation ; cependant, les caractères réservés à ces derniers peuvent varier en fonction du langage utilisé.

Nous allons dans ce cours parcourir l'ensemble des opérateurs mis à disposition avec le langage Dart. Nous aborderons l'ensemble des types d'opérateurs : d'assignation, arithmétique, logique, de comparaison et de test de type.

À travers de multiples exemples, nous verrons les différents types d'opérateurs et leurs utilités. Cela vous permettra de comprendre et d'apprendre à utiliser ces outils nécessaires à tous les développeurs. Les différents exemples de code Dart mis à disposition illustreront leur utilisation pratique. Il est recommandé d'écrire le code de votre côté pour vous familiariser avec la notion d'opérateur.

Définition **Définition générale**

En programmation informatique, un opérateur est un élément qui se comporte comme une fonction. Il est défini par un identificateur spécifique qui lui est réservé au sein du langage informatique. On peut le considérer comme l'équivalent d'un opérateur mathématique pour la programmation.

Les opérateurs remplissent différentes fonctions : opérations arithmétiques, manipulation de chaînes de caractères, comparaison, opérations logiques, etc. Pour un langage de programmation donné, un ensemble d'opérateurs est disponible pour effectuer ces types d'action.

Liste des principaux opérateurs en Dart

On retrouve ainsi une multitude d'opérateurs pour le langage Dart, plus ou moins utilisés quotidiennement lorsque l'on code dans ce langage :

- Les opérateurs d'assignation : `= += -= *= /=`
- Les opérateurs arithmétiques : `+ - * / %`
- Les opérateurs logiques : `! || &&`
- Les opérateurs de comparaison : `== != > < >= <=`
- Les opérateurs de test de type : `as is is!`

Dans la suite de ce cours, nous allons lister un à un tous ces opérateurs et s'appuyer sur des exemples précis afin de comprendre en détail leur fonctionnement. Nous verrons également quelques opérateurs complémentaires plus rarement utilisés.

Exercice : Quiz

Question 1

À quoi peut être apparenté un opérateur ?

- Une variable
- Une fonction

Question 2

L'un des opérateurs permet de faire une division.

- Vrai
- Faux

Question 3

Les opérateurs ne permettent pas de manipuler des chaînes de caractères.

- Vrai
- Faux

Question 4

Les opérateurs en programmation sont :

- Facultatifs
- Utiles selon le contexte
- Indispensables quel que soit le programme informatique

Question 5

Combien y a-t-il d'opérateurs différents en Dart ?

- 4
- 10
- Plus de 20

II. Opérateurs d'assignation

Définition **Définition générale**

Les opérateurs d'assignation permettent d'assigner à l'opérande de gauche une nouvelle valeur. Selon l'opérateur, une action peut également être effectuée au moment de l'assignation (exemple : une opération arithmétique). Après l'assignation, la nouvelle valeur est donc stockée dans l'opérande de gauche.

Liste des opérateurs d'assignation

Voici les deux types d'opérateurs d'assignation :

- Opérateur d'assignation simple : `=`

  Il permet d'assigner à l'opérande de gauche la valeur de l'opérande de droite.

- Opérateurs d'assignation composée : `+=` `-=` `*=` `/=`

  Ils permettent d'assigner à l'opérande de gauche la valeur de l'opération suivante : opérande de gauche évaluée avec l'opérande de droite, suivant l'opérateur arithmétique spécifié avant le signe égal.

Exemple **Utilisation et mise en pratique de chaque opérateur**

Pour l'opérateur d'assignation simple :

La valeur 2 est assignée à la variable *a*.

```
var a = 2;
```

La variable *a* est donc égale à 2 après l'opération d'assignation.

Pour les opérateurs d'assignation composée :

La valeur 2 est assignée à la variable *a* (la variable est déclarée pour la suite des opérations).

```
var a = 2;
```

La valeur 3 est additionnée à la valeur de *a*, puis assignée à la variable *a*.

```
a += 3;
```

Cela est strictement équivalent à : a = a + 3

À ce stade, la variable *a* est donc égale à 5.

La valeur 1 est soustraite à la valeur de *a*, puis assignée à la variable *a*.

```
a -= 1;
```

Cela est strictement équivalent à : a = a - 1

À ce stade, la variable *a* est donc égale à 4.

La valeur de *a* est multipliée par la valeur 5, puis assignée à la variable *a*.

```
a *= 5;
```

Cela est strictement équivalent à : a = a * 5

À ce stade, la variable *a* est donc égale à 20.

La valeur de *a* est divisée par la valeur 10, puis assignée à la variable *a*.

```
a /= 10;
```

Cela est strictement équivalent à : a = a / 10

À ce stade, la variable *a* est donc égale à 2.

Exercice : Quiz

Question 1

L'assignation simple permet d'assigner une valeur à une variable.

- O Vrai
- O Faux

Question 2

Quels sont les deux types d'assignation ?

- Standard et complexe
- Simple et composée

Question 3

L'assignation composée permet de raccourcir l'expression.

- Vrai
- Faux

Question 4

Quel opérateur permet d'effectuer une assignation avec soustraction ?

- =-
- -
- -=

Question 5

Quel opérateur permet d'effectuer une assignation avec division ?

- *=
- /=

III. Opérateurs arithmétiques

Définition **Définition générale**

Les opérateurs arithmétiques permettent d'effectuer des opérations mathématiques entre plusieurs nombres. Ils s'utilisent de manière classique, comme on pourrait le faire en mathématiques.

L'opérateur arithmétique « **+** » permet également de concaténer des chaînes de caractères.

Liste des opérateurs arithmétiques

Voici les différents opérateurs arithmétiques que l'on peut retrouver :

- L'opérateur pour l'addition : `+`
- L'opérateur pour la soustraction : `-`

L'opérateur pour la multiplication : `*`

- L'opérateur pour la division : `/`
- L'opérateur pour le modulo : `%`

Pour rappel, le modulo est une opération qui, au couple *(a, b)* d'entiers, associe le reste *r* de la division euclidienne de *a* par *b*. Un exemple est vu ci-dessous.

Exemple **Utilisation et mise en pratique de chaque opérateur**

Pour les opérateurs arithmétiques classiques :

Une addition est assignée à la variable *a*.

```
var a = 2 + 3;
```

La variable *a* est donc égale à 5 après l'opération d'addition.

Une soustraction est assignée à la variable *a*.

```
a = 20 - 7;
```

La variable *a* est donc égale à 13 après l'opération de soustraction.

Une multiplication est assignée à la variable *a*.

```
a = 4 * 4;
```

La variable *a* est donc égale à 16 après l'opération de multiplication.

Une division est assignée à la variable *a*.

```
a = 6 / 2;
```

La variable *a* est donc égale à 3 après l'opération de division.

Pour l'opérateur arithmétique modulo :

Un modulo est assigné à la variable *a*.

```
var a = 10 % 3;
```

La variable *a* est donc égale à 1 après l'opération de modulo, car le reste de la division euclidienne de 10 par 3 est bien égale à 1.

Un autre modulo est assigné à la variable *a*.

```
a = 20 % 7;
```

La variable *a* est donc égale à 6 après l'opération de modulo, car le reste de la division euclidienne de 20 par 7 est bien égale à 6.

Pour l'opérateur arithmétique d'addition et la concaténation de chaînes de caractères :

Des chaînes de caractères sont assignées aux variables *a* et *b*.

```
var a = "Hello";
var b = "everyone";
```

L'addition (concaténation) de *a* et *b* est assignée à la variable *c*.

```
var c = a + b;
```

La variable *c* est donc égale à "Hello everyone" après l'opération de concaténation des variables *a* et *b*.

Exercice : Quiz

Question 1

À quoi peut être apparenté un opérateur arithmétique en Dart ?

- O Un opérateur mathématique classique
- O Un opérateur binaire classique

Question 2

Un opérateur arithmétique permet de faire une division et une multiplication sur la même ligne de code.

- O Vrai
- O Faux

Question 3

Les opérateurs arithmétiques sont rarement utilisés en Dart.

- O Vrai
- O Faux

Question 4

L'opérateur modulo retourne :

- O Le quotient de la division euclidienne
- O La dividende de la division euclidienne
- O Le reste de la division euclidienne

Question 5

Les opérateurs arithmétiques n'agissent pas sur les chaînes de caractères.

- O Vrai
- O Faux

IV. Opérateurs logiques

Définition **Définition générale**

Les opérateurs logiques permettent d'effectuer des opérations sur des expressions booléennes.

Pour rappel, une expression booléenne est une expression qui ne peut prendre que deux états distincts : VRAI (*true*) ou FAUX (*false*). Les opérateurs logiques permettent donc de les combiner ou de les inverser.

Liste des opérateurs logiques

On dénombre uniquement trois opérateurs logiques :

- L'opérateur logique qui **inverse la valeur** : `!`
- L'opérateur logique **OU** : `||`
- L'opérateur logique **ET** : `&&`

L'opérateur OU exige qu'au moins une des opérandes combinées soit égale à VRAI pour retourner VRAI. L'opérateur ET exige que toutes les opérandes combinées soient égales à VRAI pour retourner VRAI.

Exemple **Utilisation et mise en pratique de chaque opérateur**

Pour l'opérateur logique qui inverse la valeur :

Une valeur *true* est assignée à la variable *a*.

```
var a = true;
```

La variable *a* est assignée à la variable *b* en étant inversée.

```
var b = !a;
```

La variable *b* est donc égale à *false*.

Pour les opérateurs logiques OU et ET :

Des valeurs booléennes sont assignées aux variables *a*, *b* et *c*.

```
var a = true;
var b = false;
var c = true;
```

La combinaison de *a* OU *b* est assignée à la variable *d*.

```
var d = a || b;
```

La variable *d* est égale à *true* (VRAI), car au moins une des opérandes est égale à *true* (VRAI).

La combinaison de *a* ET *b* est assignée à la variable *d*.

```
d = a && b;
```

La variable *d* est égale à *false* (FAUX), car toutes les opérandes ne sont pas égales à *true* (VRAI).

La combinaison de *a* ET *c* est assignée à la variable *d*.

```
d = a && c;
```

La variable *d* est égale à *true* (VRAI), car toutes les opérandes sont égales à *true* (VRAI).

Exercice : Quiz

Question 1

Une variable booléenne ne peut prendre que deux valeurs.

- Vrai
- Faux

Question 2

En utilisant l'opérateur d'inversion sur une variable *false*, sa valeur reste à *false*.

- Vrai
- Faux

Question 3

Si au moins une opérande est égale à *true*, l'opérateur « && » (ET) retourne *true*.

- Vrai
- Faux

Question 4

Les opérateurs « && » et « || » permettent :

- De tester la combinaison d'expressions booléennes
- D'effectuer des opérations arithmétiques
- De réaliser des décalages de bits

Question 5

Quelle proposition est valide en Dart ?

- a &&& b
- (a && b) || (a || c)
- (a && b) || a c

V. Opérateurs de comparaison

Définition **Définition générale**

Les opérateurs de comparaison permettent de définir la relation entre deux opérandes : égalité, non-égalité, supériorité, infériorité, etc. Les valeurs des deux opérandes sont ainsi comparées, et, selon le résultat, l'opération retourne une expression booléenne VRAI (*true*) ou FAUX (*false*).

Par exemple, si l'opérateur d'égalité est utilisé et que les deux valeurs comparées ne sont pas égales, l'expression va retourner FAUX (*false*). Si au contraire les deux valeurs comparées sont égales, l'expression va retourner VRAI (*true*).

Liste des opérateurs de comparaison

On dénombre au total six opérateurs de comparaison :

- L'opérateur d'égalité : `==`
- L'opérateur d'inégalité : `!=`
- L'opérateur de supériorité stricte : `>`
- L'opérateur d'infériorité stricte : `<`
- L'opérateur de supériorité ou égalité : `>=`
- L'opérateur d'infériorité ou égalité : `<=`

Les opérateurs d'égalité ou d'inégalité peuvent être utilisés sur tout type d'objet : ils vont simplement comparer la valeur des opérandes et déduire le résultat.

Les opérateurs de supériorité ou d'infériorité sont généralement utilisés sur des nombres.

Exemple **Utilisation et mise en pratique de chaque opérateur**

Pour les opérateurs d'égalité ou d'inégalité :

Des valeurs sont assignées aux variables *a*, *b* et *c*.

```
var a = 1;
var b = 2;
var c = 1:
```

Une opération d'égalité est assignée à la variable *d*.

```
var d = a == b;
```

La variable *d* est égale à *false* (FAUX), car les valeurs de *a* et *b* ne sont pas égales.

Une autre opération d'égalité est assignée à la variable *d*.

```
d = a == c;
```

La variable *d* est égale à *true* (VRAI), car les valeurs de *a* et *b* sont égales.

Une opération d'inégalité est assignée à la variable *d*.

```
var d = a != b;
```

La variable *d* est égale à *true* (VRAI), car les valeurs de *a* et *b* sont inégales.

Nous avons exactement le même raisonnement avec des chaînes de caractères assignées aux variables a, b et c :

Des valeurs sont assignées aux variables *a*, *b* et *c*.

```
var a = "Hello"
var b = "Hi"
var c = "Hello":
```

Une opération d'égalité est assignée à la variable *d*.

```
var d = a == b;
```

La variable *d* est égale à *false* (FAUX), car les valeurs de *a* et *b* ne sont pas égales.

Une autre opération d'égalité est assignée à la variable *d*.

```
d = a == c;
```

La variable *d* est égale à *true* (VRAI), car les valeurs de *a* et *b* sont égales.

Une opération d'inégalité est assignée à la variable *d*.

```
var d = a != b;
```

La variable *d* est égale à *true* (VRAI), car les valeurs de *a* et *b* sont inégales.

Pour les opérateurs de supériorité et d'infériorité :

Des valeurs sont assignées aux variables *a*, *b* et *c*.

```
var a = 1;
var b = 2;
var c = 1:
```

Une opération de supériorité est assignée à la variable *d*.

```
var d = a > c;
```

La variable *d* est égale à *false* (FAUX), car la valeur de *a* n'est pas strictement supérieure à la valeur de *c*.

Une autre opération de supériorité est assignée à la variable *d*.

```
d = a >= c;
```

La variable *d* est égale à *true* (VRAI), car la valeur de *a* est supérieure ou égale à la valeur de *c*.

Une opération d'infériorité est assignée à la variable *d*.

```
d = a < b;
```

La variable *d* est égale à *true* (VRAI), car la valeur de *a* est strictement inférieure à la valeur de *b*.

Une autre opération d'infériorité est assignée à la variable *d*.

```
d = b < a;
```

La variable *d* est égale à *false* (FAUX), car la valeur de *a* n'est pas strictement inférieure à la valeur de *b*.

Exercice : Quiz

Question 1

L'opérateur d'égalité « == » ne peut être utilisé que sur des nombres.

- O Vrai
- O Faux

Question 2

Selon la situation, les opérateurs « > » et « >= » ne retourneront pas le même résultat.

- O Vrai
- O Faux

Question 3

Que retourne l'expression suivante : 8 <= 2 ?

- O True
- O False

Question 4

Les opérateurs d'infériorité sont généralement utilisés sur :

- O Des booléens
- O Des chaînes de caractères
- O Des nombres

Question 5

Les opérateurs de comparaison retournent toujours des valeurs booléennes.

- O Vrai
- O Faux
- O Cela dépend

VI. Opérateurs de test de type

Définition **Définition générale**

Les opérateurs de test de type, comme leur nom l'indique, permettent d'effectuer des tests concernant le type de l'opérande. Ils permettent également (si cela est possible) de *caster* un objet vers un type spécifique.

Pour rappel, « *caster une variable* » signifie changer le type de la variable en la convertissant implicitement.

Liste des opérateurs de test de type

Il existe uniquement trois opérateurs de test de type :

- L'opérateur pour caster : `as`
- L'opérateur pour confirmer un type : `is`
- L'opérateur pour infirmer un type : `is!`

Ces opérateurs sont moins courants que ceux vus précédemment, mais se révéleront utiles dans certains cas, pour s'assurer qu'on manipule bien les bons types d'objets.

Exemple **Utilisation et mise en pratique de chaque opérateur**

L'opérateur pour caster ne doit être utilisé que lorsque l'on est sûr que l'objet que l'on caste est bien du type avec lequel il va être casté.

```
(employee as Person).firstName = "Bob"
```

Dans l'exemple ci-dessus, il faut imaginer qu'une classe *Person* a été créée. La variable `employee` est alors castée vers le type *Person*, car cela permet dans ce cas d'utiliser l'attribut `firstName` (qui est propre à la classe *Person*). Dans le contexte donné, il faudra toutefois s'assurer expressément que la variable `employee` est bien de type *Person* avant d'effectuer l'opération de cast.

Pour les opérateurs de confirmation et d'infirmation de type :

Des valeurs sont assignées aux variables *a* et *b*.

```
int a = 1;
String b = "Hello";
```

Une opération de confirmation de type est assignée à la variable *c*.

```
var c = a is String;
```

La variable *c* est égale à *false* (FAUX), car la variable *a* n'est pas de type String, mais bien de type int.

Une opération d'infirmation de type est assignée à la variable *c*.

```
c = b !is int;
```

La variable *c* est égale à *true* (VRAI), car la variable *b* n'est effectivement pas de type int, mais bien de type String.

Exercice : Quiz

Question 1

Les opérateurs de test de type permettent de tester le type des opérateurs.

- O Vrai
- O Faux

Question 2

Les opérateurs de test de type sont moins fréquemment utilisés que les autres opérateurs vus précédemment.

- O Vrai
- O Faux

Question 3

Caster une variable revient à convertir son type de façon implicite.

- O Vrai
- O Faux

Question 4

On peut caster une variable :

- O Vers n'importe quel autre type d'objet
- O Vers n'importe quel autre type d'opérateur
- O Vers un type d'objet auquel elle correspond déjà

Question 5

Peut-on s'assurer qu'une variable est bien de type String avec l'opérateur de confirmation de type ?

- Non
- Oui, et cela fonctionne uniquement avec le type String
- Oui, et cela fonctionnerait pour tous les types d'objets

VIII. Autres opérateurs

Liste de quelques opérateurs complémentaires

Certains des opérateurs suivants sont plus rarement utilisés, mais doivent quand même être maîtrisés pour utiliser Dart à son plein potentiel :

- Les opérateurs au niveau du bit : `&` `|` `^` `<<` `>>`
- Les opérateurs d'expressions conditionnelles : `?` `??`
- La notation en cascade : `..`

Les opérateurs au niveau du bit nécessitent des connaissances préalables en langage binaire (il est probable que vous n'ayez jamais à les utiliser, des exemples seront toutefois vus ci-dessous).

Les deux autres types d'opérateurs listés sont simplement des notations raccourcies afin d'avoir un code plus optimisé et concis (voir exemples ci-dessous).

Exemple **Utilisation et mise en pratique de chaque opérateur**

Pour les opérateurs au niveau du bit :

Des valeurs (format hexadécimal) sont assignées aux variables *a* et *b*.

```
var a = 0x22;
var b = 0x0f;
```

Une opération au niveau du bit est assignée à la variable *c*.

```
var c = a & b;
```

La variable *c* est égale à 0x02 (masque AND).

Une opération au niveau du bit est assignée à la variable *c*.

```
c = a | b;
```

La variable *c* est égale à 0x2f (masque OR).

Une opération au niveau du bit est assignée à la variable *c*.

```
c = a ^ b;
```

La variable *c* est égale à 0x2d (masque XOR).

Une opération de décalage de bits est assignée à la variable *c*.

```
c = a << 4;
```

La variable *c* est égale à 0x220 (décalage de 4 bits à gauche).

Une opération de décalage de bits est assignée à la variable *c*.

```
c = a >> 4;
```

La variable *c* est égale à 0x02 (décalage de 4 bits à droite).

Pour les opérateurs d'expressions conditionnelles :

Cette expression

```
var a = b ? 1 : 2;
```

est équivalente à :

```
if (b) {
a = 1;
}
else {
a = 2;
}
```

L'opérateur « ? » s'appuie sur la condition de *b* pour déduire le résultat.

Cette expression

```
var a = b ?? 1;
```

est équivalente à :

```
if (b != null) {
a = b;
}
else {
a = 1;
}
```

L'opérateur « ?? » s'appuie sur la valeur de *b* pour déduire le résultat.

Pour la notation en cascade :

Cette expression

```
var paint = Paint();
..color = Colors.black;
..strokeCap = StrokeCap.round;
..strokeWidth = 5.0;
```

est équivalente à :

```
var paint = Paint();
paint.color = Colors.black;
paint.strokeCap = StrokeCap.round;
paint.strokeWidth = 5.0;
```

L'opérateur « .. » permet de ne pas avoir à répéter l'objet initial pour manipuler ses attributs.

Exercice : Quiz

Question 1

Les opérateurs au niveau du bit demandent des connaissances du langage binaire.

- Vrai
- Faux

Question 2

Avec les opérateurs au niveau du bit, on peut effectuer des décalages de bits mais pas de masques binaires.

- Vrai
- Faux

Question 3

Les opérateurs d'expressions conditionnelles permettent de raccourcir le code.

- Vrai
- Faux

Question 4

Si la variable *c* est non nulle, que retourne cette expression : *c ?? 5;* ?

- Nulle
- 5
- c

Question 5

L'opérateur « .. » permet :

- D'éviter les répétitions de l'objet initial pour accéder à son type
- D'éviter les répétitions de l'objet initial pour accéder à ses attributs
- D'éviter les répétitions de l'objet initial pour accéder à son parent

IX. Essentiel

Les opérateurs en Dart font partie intégrante du quotidien du développeur Dart. On les retrouve quasiment à chaque ligne de code, et ce sont eux qui permettent de jongler avec la logique, les variables, les conditions et les opérations diverses.

Nous avons vu dans ce cours l'ensemble des opérations réalisables via les opérateurs en Dart : assignations, arithmétique, logique, comparaisons, test, etc.

L'important est de bien comprendre l'opération engendrée par chacun de ses opérateurs. Ces opérations sont claires et précises. Leur but est d'accélérer la production du code et de rendre possible une infinité de fonctionnalités.

Avec le temps, ces opérateurs sont utilisés instinctivement par le développeur car ils sont omniprésents. Ils deviennent très simples à utiliser à mesure qu'on progresse.

Certains des opérateurs vus dans ce cours peuvent paraître compliqués au premier abord : dans la pratique, vous comprendrez vite leur intérêt et leur fonctionnement. Le tout est de pratiquer encore et encore pour les intégrer rapidement. Les opérateurs font partie des bases fondamentales d'un langage de programmation.

X. Auto-évaluation

A. Exercice :

Dans cet exercice, nous allons utiliser et mettre en application les différents opérateurs vus dans ce cours. Les réponses doivent être ajoutées successivement à un programme en Dart que vous devez compiler (pour simplifier l'environnement, le DartPad peut être utilisé : https://dartpad.dev/[1]).

Question 1

Déclarez une variable *a* et assignez lui la valeur 5.

Question 2

Utilisez une assignation composée pour monter la valeur de la variable *a* à 20.

Question 3

Déclarez une variable *b* et assignez lui la multiplication de la variable *a* par 10.

Question 4

Testez si la valeur de la variable *a* est supérieure strictement à 10 ET si la variable *b* est inférieure ou égale à 200. Affichez le résultat dans la console avec la fonction *print*.

Question 5

Testez si la variable *a* est de type String avec l'opérateur correspondant. Affichez le résultat dans la console avec la fonction *print*.

Question 6

Utilisez un opérateur d'expression conditionnelle pour assigner la variable *b* à la variable *a* si celle-ci est non nulle. Si celle-ci est nulle, assignez la valeur 2.

Question 7

Déclarez deux chaînes de caractères de votre choix puis concaténez-les avec un opérateur arithmétique. Affichez le résultat dans la console avec la fonction *print*.

B. Test

Exercice : Quiz

Question 1

Les opérateurs en Dart permettent de coder absolument toutes les fonctionnalités imaginables.

- ○ Vrai
- ○ Faux

Question 2

Un opérateur permet de récupérer le reste d'une division euclidienne.

- ○ Vrai
- ○ Faux

Question 3

[1] https://dartpad.dev/?null_safety=true

On peut tester si le type d'une variable est une String avec un opérateur.

- Vrai
- Faux

Question 4

Quel opérateur permet de réaliser un décalage de bits ?

- >=
- !=
- >>

Question 5

Quel opérateur permet de tester l'égalité entre deux objets ?

- =
- ==
- ===

Solutions des exercices

Exercice p. 3 Solution n°1

Question 1

À quoi peut être apparenté un opérateur ?

- ○ Une variable
- ◉ Une fonction

Un opérateur se comporte comme une fonction spécifique, et sa représentation est invariable.

Question 2

L'un des opérateurs permet de faire une division.

- ◉ Vrai
- ○ Faux

Certains opérateurs sont arithmétiques et permettent de faire des opérations mathématiques.

Question 3

Les opérateurs ne permettent pas de manipuler des chaînes de caractères.

- ○ Vrai
- ◉ Faux

Certains opérateurs permettent la manipulation de chaînes de caractères, comme la concaténation.

Question 4

Les opérateurs en programmation sont :

- ○ Facultatifs
- ○ Utiles selon le contexte
- ◉ Indispensables quel que soit le programme informatique

Les opérateurs sont indispensables pour n'importe quel type de fonctionnalité, on les retrouve quasiment à chaque ligne de code.

Question 5

Combien y a-t-il d'opérateurs différents en Dart ?

- ○ 4
- ○ 10
- ◉ Plus de 20

Il existe une multitude d'opérateurs en Dart, à utiliser selon le contexte.

Exercice p. 5 Solution n°2

Question 1

L'assignation simple permet d'assigner une valeur à une variable.

- [x] Vrai
- [ ] Faux

Il s'agit de l'assignation standard qui vise à affecter une valeur à une variable.

Question 2

Quels sont les deux types d'assignation ?

- [ ] Standard et complexe
- [x] Simple et composée

L'assignation simple vise à affecter une valeur à une variable, tandis que l'assignation composée ajoute une opération lors de l'assignation.

Question 3

L'assignation composée permet de raccourcir l'expression.

- [x] Vrai
- [ ] Faux

Elle permet effectivement de réécrire l'expression de façon plus compacte.

Question 4

Quel opérateur permet d'effectuer une assignation avec soustraction ?

- [ ] =-
- [ ] -
- [x] -=

Le symbole de l'opération mathématique se situe avant le signe égal.

Question 5

Quel opérateur permet d'effectuer une assignation avec division ?

- [ ] *=
- [x] /=

Le signe « / » correspond à la division.

Exercice p. 8 Solution n°3

Question 1

À quoi peut être apparenté un opérateur arithmétique en Dart ?

- [x] Un opérateur mathématique classique
- [ ] Un opérateur binaire classique

Un opérateur arithmétique en Dart s'utilise exactement comme en mathématiques.

Question 2

Un opérateur arithmétique permet de faire une division et une multiplication sur la même ligne de code.

- ◉ Vrai
- ○ Faux

Les opérateurs peuvent être utilisés en chaîne. Exemple : var a = 6 / 2 * 3

Question 3

Les opérateurs arithmétiques sont rarement utilisés en Dart.

- ○ Vrai
- ◉ Faux

Les opérateurs arithmétiques font partie des opérateurs les plus couramment utilisés dans le langage Dart.

Question 4

L'opérateur modulo retourne :

- ○ Le quotient de la division euclidienne
- ○ La dividende de la division euclidienne
- ◉ Le reste de la division euclidienne

L'opérateur modulo correspond au reste de la division euclidienne, division telle qu'on la faisait à l'école primaire.

Question 5

Les opérateurs arithmétiques n'agissent pas sur les chaînes de caractères.

- ○ Vrai
- ◉ Faux

L'addition est possible sur des chaînes de caractères et revient à les concaténer.

Exercice p. 9 Solution n°4

Question 1

Une variable booléenne ne peut prendre que deux valeurs.

- ◉ Vrai
- ○ Faux

Une variable booléenne ne dispose que de deux états : vrai ou faux.

Question 2

En utilisant l'opérateur d'inversion sur une variable *false*, sa valeur reste à *false*.

- ○ Vrai
- ◉ Faux

En inversant une telle variable, la nouvelle valeur sera donc *true*.

Question 3

Si au moins une opérande est égale à *true*, l'opérateur « && » (ET) retourne *true*.

- ○ Vrai
- ◉ Faux

Pour que l'opérateur « && » retourne *true*, il faut que l'intégralité des opérandes soient égales à *true*.

Question 4

Les opérateurs « && » et « || » permettent :

- ◉ De tester la combinaison d'expressions booléennes
- ○ D'effectuer des opérations arithmétiques
- ○ De réaliser des décalages de bits

Les opérateurs « && » et « || » servent effectivement à combiner des expressions booléennes pour retourner un résultat spécifique.

Question 5

Quelle proposition est valide en Dart ?

- ○ a &&& b
- ◉ (a && b) || (a || c)
- ○ (a && b) || a c

Les deux autres expressions ne sont absolument pas valides en Dart et bloqueraient la compilation.

Exercice p. 13 Solution n°5

Question 1

L'opérateur d'égalité « == » ne peut être utilisé que sur des nombres.

- ○ Vrai
- ◉ Faux

L'opérateur d'égalité fonctionne sur tout type d'objet, comme par exemple les chaînes de caractères.

Question 2

Selon la situation, les opérateurs « > » et « >= » ne retourneront pas le même résultat.

- ◉ Vrai
- ○ Faux

Le premier concerne une supériorité stricte, tandis que le second pourra retourner *true* (VRAI) en cas d'égalité des opérandes.

Question 3

Que retourne l'expression suivante : 8 <= 2 ?

- ○ True
- ◉ False

Elle retourne *false* (FAUX), car 8 n'est pas inférieur ou égal à 2.

Question 4

Les opérateurs d'infériorité sont généralement utilisés sur :

- ○ Des booléens
- ○ Des chaînes de caractères
- ◉ Des nombres

On voudra généralement utiliser les opérateurs d'infériorité sur des nombres, car ce type de comparaison s'applique difficilement à d'autres types d'objets.

Question 5

Les opérateurs de comparaison retournent toujours des valeurs booléennes.

- ◉ Vrai
- ○ Faux
- ○ Cela dépend

Les opérateurs de comparaison retournent systématiquement une valeur booléenne pour attester de la validation ou non de la comparaison effectuée.

Exercice p. 15 Solution n°6

Question 1

Les opérateurs de test de type permettent de tester le type des opérateurs.

- ○ Vrai
- ◉ Faux

Ils permettent de tester le type d'une variable ou d'un objet.

Question 2

Les opérateurs de test de type sont moins fréquemment utilisés que les autres opérateurs vus précédemment.

- ◉ Vrai
- ○ Faux

Ils s'appliquent à certaines situations spécifiques lorsque le type d'une variable doit être confirmé par exemple.

Question 3

Caster une variable revient à convertir son type de façon implicite.

- [x] Vrai
- [ ] Faux

Lorsque cela est possible, caster une variable revient à changer son type implicitement.

Question 4

On peut caster une variable :

- [ ] Vers n'importe quel autre type d'objet
- [ ] Vers n'importe quel autre type d'opérateur
- [x] Vers un type d'objet auquel elle correspond déjà

Si la variable castée n'a rien à voir avec le type visé, alors cela produira une erreur.

Question 5

Peut-on s'assurer qu'une variable est bien de type String avec l'opérateur de confirmation de type ?

- [ ] Non
- [ ] Oui, et cela fonctionne uniquement avec le type String
- [x] Oui, et cela fonctionnerait pour tous les types d'objets

L'opérateur « is » fonctionne avec n'importe quel type d'objet, il se contentera de retourner la correspondance ou non.

Exercice p. 19 Solution n°7

Question 1

Les opérateurs au niveau du bit demandent des connaissances du langage binaire.

- [x] Vrai
- [ ] Faux

De telles opérations se font au niveau du bit, il faut donc connaître le comportement de ce type de nombre pour utiliser ces opérateurs.

Question 2

Avec les opérateurs au niveau du bit, on peut effectuer des décalages de bits mais pas de masques binaires.

- [ ] Vrai
- [x] Faux

On peut également réaliser des masques AND, OR ou XOR, le plus souvent via un format hexadécimal.

Question 3

Les opérateurs d'expressions conditionnelles permettent de raccourcir le code.

- [x] Vrai
- [ ] Faux

Ce type d'opérateur permet effectivement de maintenir la logique sur une seule ligne de code et ainsi gagner en clarté.

Question 4

Si la variable *c* est non nulle, que retourne cette expression : *c ?? 5;* ?

- ○ Nulle
- ○ 5
- ◉ c

Si *c* est non nulle, alors l'expression retourne *c*, autrement l'expression aurait retourné 5.

Question 5

L'opérateur « .. » permet :

- ○ D'éviter les répétitions de l'objet initial pour accéder à son type
- ◉ D'éviter les répétitions de l'objet initial pour accéder à ses attributs
- ○ D'éviter les répétitions de l'objet initial pour accéder à son parent

L'opérateur « .. » se révèle effectivement utile lorsque l'on souhaite accéder aux attributs d'un objet et les manipuler successivement.

Exercice p. Solution n°8

On assigne la valeur 5 à la variable *a*.

```
var a = 5;
```

Exercice p. Solution n°9

On utilise une assignation composée pour monter la valeur à 20.

```
a += 15;
```

a *= 4 est aussi une solution valable.

Exercice p. Solution n°10

On assigne à la variable *b* la multiplication de la variable *a* par 10.

```
var b = a * 10;
```

Exercice p. Solution n°11

On compare la variable *a* strictement supérieure à 10 ET la variable *b* inférieure ou égale à 200.

```
print(a > 10 && b <= 200);
```

Exercice p. Solution n°12

On teste si la variable *a* est de type String.

```
print(a is String);
```

Exercice p. Solution n°13

On utilise une expression conditionnelle pour assigner *b* (si *b* est non nulle) ou 2 (si *b* est nulle), à la variable *a*.

```
a = b ?? 2;
```

Exercice p. Solution n°14

On déclare deux chaînes de caractères, et on utilise l'opérateur d'addition pour les concaténer.

```
var c = "Hello ";
var d = "World !";

print(c + d);
```

Exemple **Programme final**

```dart
void main() {
  var a = 5;

  a += 15;

  var b = a * 10;

  print(a > 10 && b <= 200);

  print(a is String);

  a = b ?? 2;

  var c = "Hello ";
  var d = "World !";

  print(c + d);
}
```

Résultat affiché dans la console :

```
Console

true
false
Hello World !
```

Exercice p. 20 Solution n°15

Question 1

Les opérateurs en Dart permettent de coder absolument toutes les fonctionnalités imaginables.

- ○ Vrai
- ◉ Faux

Les possibilités sont énormes, mais il y aura besoin d'autres outils disponibles en Dart dans certains cas.

Question 2

Un opérateur permet de récupérer le reste d'une division euclidienne.

- [x] Vrai
- [ ] Faux

Il s'agit de l'opérateur modulo « % ».

Question 3

On peut tester si le type d'une variable est une String avec un opérateur.

- [x] Vrai
- [ ] Faux

Oui, avec l'opérateur « is ».

Question 4

Quel opérateur permet de réaliser un décalage de bits ?

- [ ] >=
- [ ] !=
- [x] >>

L'opérateur« >> » permet d'effectuer un décalage de bits vers la droite sur un nombre.

Question 5

Quel opérateur permet de tester l'égalité entre deux objets ?

- [ ] =
- [x] ==
- [ ] ===

La première réponse concerne l'affectation de valeur, et la dernière n'existe pas en Dart.