

Les boucles en Dart

Table des matières

I. Introduction aux boucles en Dart	3
A. Définition	3
B. Boucles définies et indéfinies	3
II. Exercice : Quiz	4
III. Boucles for et for ... in	5
A. For	5
B. For ... in	5
IV. Exercice : Quiz	6
V. Boucles while et do ... while	7
A. While	7
B. Do ... while	7
VI. Exercice : Quiz	7
VII. Instructions break et continue	9
A. break	9
B. Continue	9
C. Labels.....	10
VIII. Exercice : Quiz	10
IX. Essentiel	11
X. Auto-évaluation	11
A. Exercice	11
B. Test.....	12
Solutions des exercices	13

I. Introduction aux boucles en Dart

Contexte

Vous souvenez-vous lorsque, petit(e), vous avez dû écrire 50 fois « *Je ne discuterai pas avec mon voisin en classe* », en repréailles du fou rire que vous aviez eu avec votre voisin de table ?

Je ne discuterai pas avec mon voisin en classe

Je ne discuterai pas avec mon voisin en classe

...

Eh bien, si nous avions eu accès à un ordinateur, nous aurions pu créer une « *boucle* » permettant à deux lignes de codes d'écrire presque instantanément 50 fois la phrase souhaitée.

Quand il s'agit d'algorithmie, les boucles sont des éléments essentiels pour faciliter l'exécution de multiples instructions.

Dans ce cours, nous étudierons les différents types de boucles en Dart, ainsi que les instructions `break` et `continue`, afin que vous n'ayez plus à écrire 3 fois la même chose !

A. Définition

Définition

Une boucle est une instruction qui permet de répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste vraie.

Fondamental Les composants d'une boucle

Une boucle nécessite 4 éléments :

- une variable,
- une condition liée à cette variable,
- une instruction menant à la modification de la valeur de cette variable,
- un bloc d'instructions à exécuter pendant que la condition reste vraie.

B. Boucles définies et indéfinies

Fondamental

On distingue les boucles définies et indéfinies. Une boucle `for` prend en charge l'incrémentement de la variable liée à la condition.

```
1 for (int i = 0; i < 5; i++) {  
2   print('Bonjour');  
3 }
```

Le changement de valeur de la variable dans une boucle `while` doit se faire dans le corps de celle-ci.

```
1 while (i < 5) {  
2   print('Bonjour');  
3   i++;  
4 }
```

Remarque

L'erreur la plus courante, lorsqu'on utilise des boucles, est de créer des boucles infinies. Une boucle infinie est une boucle qui ne se termine jamais, car la condition indiquée en paramètre ne devient jamais fausse.

```
1 For (int i = 0; i < 5; i--) {
2   print('J'adore le Dart');
3 }
```

Ici, la valeur de `i` est décrémentée de 1 à chaque tour. Pourtant, nous indiquons bien que, tant que la valeur de `i` est inférieure à 5, il faut continuer la boucle. Or, la valeur de `i` ne pourra jamais atteindre 5, ce qui crée une boucle infinie.

Ce genre d'erreur peut paraître facilement évitable, mais arrive très souvent !

Exercice : Quiz

[solution n°1 p.15]

Question 1

Une boucle permet :

- ☐ De répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste fausse
- ☐ De répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste vraie
- ☐ De répéter l'exécution d'une partie d'un programme pendant que `i` reste inférieur à une certaine valeur

Question 2

Une boucle nécessite :

- ☐ Une variable à modifier
- ☐ Un bloc d'instruction
- ☐ Un paramètre de type `int`
- ☐ Une condition
- ☐ Une instruction menant à la modification de la variable

Question 3

Une boucle `while` est :

- ☐ De type défini
- ☐ De type indéfini

Question 4

Si ma boucle ne se termine jamais, qu'est-ce que cela peut signifier ?

- ☐ Je ne modifie jamais la variable liée à ma condition.
- ☐ La condition ne peut jamais devenir vraie.
- ☐ La condition ne peut jamais devenir fausse.

Question 5

Dans une boucle `for`, où dois-je déclarer la variable liée à la condition ?

- ☐ Dans la déclaration de la boucle `for`
- ☐ Dans le bloc d'instructions

III. Boucles for et for ... in

A. For

Définition

La boucle `for` est une boucle à pré-condition. Cela signifie que la condition est évaluée AVANT l'exécution du bloc d'instructions.

```
1 For (var i = 0; i < 2; i++) {  
2   print(i);  
3 }
```

Ici, nous commençons par initialiser une variable à 0. Cette variable est créée pour la boucle, et n'existe pas en dehors de celle-ci. Elle peut cependant être utilisée dans le bloc d'instructions, comme ici, lorsque nous l'affichons avec `print()`.

La condition est en deuxième position et utilise la variable précédemment déclarée. Ici, nous voulons que notre boucle se répète tant que la variable `i` est inférieure à 2.

En troisième position se situe la formule que nous souhaitons appliquer à notre variable à chaque tour. Ici, nous demandons une incrémentation de 1 à chaque fois.

Regardons ce qu'il se passe dans les détails :

- Première évaluation : `i` est égale à 0, elle est donc bien inférieure à 2. `i` est incrémentée de 1 à la fin de l'exécution du bloc d'instructions.
- Seconde évaluation : `i` est égale à 1, elle est toujours inférieure à 2. `i` est incrémentée de 1 à la fin de l'exécution du bloc.
- Troisième évaluation : `i` est égale à 2, elle n'est plus inférieure à 2, la boucle s'arrête et le bloc d'instructions n'est pas exécuté.

B. For ... in

Définition

La boucle `for ... in` est une boucle de parcours, c'est-à-dire qu'elle permet de faire une itération dans une collection.

```
1 Var animaux = ['Chien', 'Chat', 'Hamster'];  
2 For (var a in animaux) {  
3   print(a);  
4 }
```

Cette boucle s'exécutera tant que la fin de la liste n'a pas été atteinte. Dans le premier tour de la boucle, la variable `a` aura la valeur `Chien`, dans le second, elle aura la valeur `Chat` et ainsi de suite. En tout, puisque la liste contient 3 éléments, le bloc d'instructions sera exécuté 3 fois.

Remarque

Les classes itérables ont une méthode appelée `forEach` permettant de faire la même chose qu'une boucle `for ... in`.

```
1 var collection = ['Abeille', 'Chocolat', 'Soleil'];
2 collection.forEach((a) => print(a));
```

Comme avec `for ... in`, la variable `a` prendra la valeur de chacun des éléments du tableau jusqu'à ce que celui-ci soit terminé.

Exercice : Quiz

[solution n°2 p.16]

Question 1

La boucle `for` est :

- ☐ Une boucle à pré-condition
- ☐ Une boucle à post-condition

Question 2

`forEach` :

- ☐ S'utilise exactement de la même façon que `for`
- ☐ Permet de faire la même chose que `for ... in`
- ☐ Est une méthode des classes itérables

Question 3

La boucle `for ... in` est :

- ☐ Une boucle de parcours
- ☐ Une boucle ne se terminant jamais
- ☐ Une boucle à pré-condition

Question 4

```
For (var i = 0; i < 2; i++) {
  print(i);
}
```

Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais
- ☐ À l'infini

Question 5

```
For (var i = 1; i < 0; i++) {
  print(i);
}
```

Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais
- ☐ À l'infini

V. Boucles while et do ... while

A. While

Définition

La boucle `while` est une boucle à pré-condition. Cela signifie que la condition est évaluée AVANT l'exécution du bloc d'instructions.

```
1 var i = 0;
2 while (i < 3) {
3   print(i);
4   i++;
5 }
```

La boucle `while` se différencie de la boucle `for`, car il est ici nécessaire de déclarer la variable en dehors de la boucle. L'incrément (ou le changement de valeur de la variable, quel qu'il soit), se fait, elle, à l'intérieur de la boucle.

Remarque

Le plus souvent, il est plus intéressant d'utiliser une boucle `for`, car cela nous évite de déclarer inutilement des variables en dehors de la boucle et qui ne sont utilisées que pour une itération. De plus, le fait qu'il y ait toujours 3 paramètres à renseigner nous évite d'oublier l'opération d'incrément ou le changement de la variable, et donc de créer une boucle infinie !

B. Do ... while

Définition

La condition dans une boucle `do ... while` est évaluée après le bloc d'instructions. Il peut en effet arriver qu'il soit nécessaire d'effectuer une première fois le bloc d'instructions, avant d'évaluer s'il est nécessaire de l'exécuter de nouvelle fois.

```
1 var i = 0;
2 do {
3   i++;
4 } while (i < 3);
```

Lorsque la condition sera évaluée une première fois, `i` aura déjà été incrémentée et aura la valeur 1.

Exercice : Quiz

[solution n°3 p.17]

Question 1

La boucle `while` est :

- ☐ Une boucle à pré-condition comme `for`
- ☐ Une boucle à post-condition
- ☐ Une boucle de parcours

Question 2

Dans une boucle `while`, la déclaration de variable se fait :

- ☐ Dans les paramètres de la boucle
- ☐ Dans le corps de la boucle
- ☐ En dehors de la boucle

Question 3

Dans une boucle `while`, la modification de la variable liée à la condition se fait :

- ☐ Dans le corps d'instructions de la boucle
- ☐ Dans les paramètres de la boucle
- ☐ En dehors de la boucle

Question 4

La boucle `do ... while` est :

- ☐ Une boucle à pré-condition comme `for`
- ☐ Une boucle dont la condition est vérifiée après l'exécution du corps de la boucle
- ☐ Une boucle de parcours

Question 5

Pour ce morceau de code :

```
Var i = 5;
While (i > 5) {
    i++;
}
```

Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais
- ☐ À l'infini

VII. Instructions break et continue

A. break

Définition

L'instruction `break` termine l'exécution de la boucle qui l'englobe, même si la condition de départ de l'exécution de cette boucle est toujours vraie.

```
1 For (var i = 0; i < 5; i++) {  
2   if (i == 3) {  
3     break;  
4   }  
5   print(i);  
6 }
```

Ainsi, même si la boucle devrait normalement se terminer uniquement si la valeur de `i` atteignait 5, elle va se terminer, car la valeur de `i` a atteint 3.

`break` termine toujours la boucle qui l'englobe directement, mais ne terminera pas une boucle qui englobait une autre boucle.

```
1 For (var i = 0; i < 5; i++) {  
2   for (var j = 0; i < 10; j++) {  
3     if (j == 3) {  
4       break;  
5     }  
6     j++;  
7   }  
8   print(i);  
9 }
```

Ici, la boucle ayant pour variable de condition `i` continuera même si la boucle ayant pour variable `j` se termine avec `break`. Nous aurons donc bien 5 fois la valeur de `i` affichée.

B. Continue

Définition

Alors que `break` permet de terminer une boucle, `continue` permet d'ignorer un tour de boucle et de passer directement au suivant.

```
1 For (var i = 0; i < 10; i++ {  
2   if (i == 3) {  
3     continue;  
4     print('Cette instruction n'est pas prise en compte');  
5   }  
6   print(i);  
7 }
```

Grâce à `continue`, lorsque `i` aura pour valeur 3, toutes les instructions suivantes seront ignorées. Ainsi, nous aurons bien 2 et 4 qui se seront affichés, mais pas 3.

C. Labels

Les labels permettent de faire référence à certaines boucles. Par exemple, nous souhaitons parfois, depuis une certaine boucle, en arrêter une autre qui l'englobe. Pour créer un label, il suffit de créer un identifiant suivi de deux points.

```
1 premiereboucle: for (var i = 0; i < 5; i++) {
2   secondeboucle: for (var j = 0; j = 0; j++) {
3     if (i == 5) {
4       break premiereboucle;
5     }
6   }
7 }
```

Alors que `break` permettait alors de n'arrêter uniquement la boucle qui l'englobait directement, il a ici permis d'arrêter la boucle un niveau au-dessus.

Exercice : Quiz

[solution n°4 p.18]

Question 1

L'instruction `break` termine l'exécution de la boucle qui l'englobe, même si :

- ☐ La condition de la boucle qui l'englobe est toujours fausse
- ☐ La condition de la boucle qui l'englobe est toujours vraie

Question 2

`break` permet d'arrêter toutes les boucles dans lesquelles il était imbriqué.

- ☐ Vrai
- ☐ Faux

Question 3

`continue` permet :

- ☐ De ne pas réaliser la suite des instructions d'une boucle pendant un tour
- ☐ De faire tourner la boucle à l'infini
- ☐ De passer directement au tour de boucle suivant

Question 4

```
For (i = 0; i < 2; i++) {
  if (i == 1) {
    continue;
  }
  print('je suis là');
}
```

Combien de fois verrons-nous « *je suis là* » apparaître ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais

Question 5

Un label me permet de :

- ☐ Faire référence à une boucle différente de celle dans laquelle je me trouve
- ☐ Faire référence à la boucle dans laquelle je me trouve

IX. Essentiel

Rappel

Une boucle est une instruction qui permet de répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste vraie :

- Il existe les boucles indéfinies et les boucles définies.
- Les boucles `for` et `for ... in` sont des boucles définies.
- Les boucles `while` et `do ... while` sont des boucles indéfinies.

Fondamental Les boucles `for` et `for ... in`

- La boucle `for` est une boucle à pré-condition : la condition est vérifiée avant l'exécution des instructions du corps de la boucle.
- La boucle `for ... in` est une boucle de parcours, c'est-à-dire qu'elle s'exécutera jusqu'à ce qu'elle aura atteint la fin des éléments d'un itérable. La méthode `forEach` est une alternative à `for ... in`.

Fondamental Les boucles `while` et `do ... while`

- La boucle `while` est une boucle à pré-condition, contrairement à la boucle `do ... while`.
- Pour l'utilisation d'une boucle `while`, il est nécessaire d'avoir déclaré la variable d'itération en dehors de la boucle.
- Il ne faut pas oublier de changer la valeur de la variable d'itération dans le corps de la boucle.

Fondamental Les instructions `break` et `continue`

- L'instruction `break` termine l'exécution de la boucle qui l'englobe, même si la condition de départ de l'exécution de cette boucle est toujours vraie.
- L'instruction `continue` permet de passer directement au prochain tour de boucle, en ignorant les instructions qui suivaient.
- Les labels permettent de faire référence à une boucle précise.

X. Auto-évaluation

A. Exercice

Maintenant que les boucles n'ont plus de secret pour vous, il est temps de mettre vos connaissances en pratique !

Vous venez d'apprendre les boucles en Dart, pour effectuer un test, nous vous demandons de mettre en application votre apprentissage avec un exercice pratique.

Question

[solution n°5 p.19]

Vous devrez créer un triangle dans le terminal, composé de * grâce à l'exécution successive d'instructions, comme ci-dessous :

```
*****
*****
*****
*****
*****
*****
****
***
**
*
```

Indice :

Note : pour aller à la ligne, il ne faut pas oublier le « \n » !

B. Test

Exercice 1 : Quiz

[solution n°6 p.20]

Question 1

L'instruction `break` termine l'exécution de la boucle qui l'englobe, même si :

- ☐ La condition de la boucle qui l'englobe est toujours fausse.
- ☐ La condition de la boucle qui l'englobe est toujours vraie.

Question 2

La boucle `while` est :

- ☐ Une boucle de pré-condition comme `for`
- ☐ Une boucle de post-condition
- ☐ Une boucle de parcours

Question 3

Une boucle nécessite :

- ☐ Une variable à modifier
- ☐ Un bloc d'instruction
- ☐ Un paramètre de type `int`
- ☐ Une condition
- ☐ Une instruction menant à la modification de la variable

Question 4

`forEach:`

- ☐ S'utilise exactement de la même façon que `for`
- ☐ Permet de faire la même chose que `for... in`
- ☐ Est une méthode des classes itérables

Question 5

```
Var i = 5;  
While (i > 5) {  
  i++;  
}
```


Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais
- ☐ À l'infini

Solutions des exercices


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

Une boucle permet :

- ☐ De répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste fausse
- ☒ De répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste vraie
- ☐ De répéter l'exécution d'une partie d'un programme pendant que i reste inférieur à une certaine valeur
-  Une boucle permet de répéter l'exécution d'une partie d'un programme, pendant qu'une condition reste vraie. Lorsqu'elle est fausse, la boucle s'arrête.


Question 2

Une boucle nécessite :

- ☒ Une variable à modifier
- ☒ Un bloc d'instruction
- ☐ Un paramètre de type `int`
- ☒ Une condition
- ☒ Une instruction menant à la modification de la variable
-  Une boucle nécessite une variable à modifier, un bloc d'instructions dans le corps de la boucle, une condition pour indiquer lorsque la boucle doit s'arrêter, et une instruction menant à la modification de la valeur de la variable.


Question 3

Une boucle `while` est :

- ☐ De type défini
- ☒ De type indéfini
-  Une boucle `while` est une boucle indéfinie, car la variable servant d'itérateur est déclarée en dehors de la boucle, et l'incrément ou la décrémentation de cette variable se fait dans le corps de la boucle.

Question 4

Si ma boucle ne se termine jamais, qu'est-ce que cela peut signifier ?

- ☒ Je ne modifie jamais la variable liée à ma condition.
- ☐ La condition ne peut jamais devenir vraie.
- ☒ La condition ne peut jamais devenir fausse.
-  Lorsqu'une boucle ne se termine jamais, c'est qu'une condition ne devient jamais fausse. Cela peut par exemple signifier qu'il y a une erreur dans la déclaration de cette condition, ou un oubli dans la modification de la variable servant la condition.

Question 5

Dans une boucle `for`, où dois-je déclarer la variable liée à la condition ?

- ☒ Dans la déclaration de la boucle `for`
- ☐ Dans le bloc d'instructions
- ☒ La variable est déclarée dans les paramètres de la boucle `for`.

Exercice p. 6 Solution n°2

Question 1

La boucle `for` est :

- ☒ Une boucle à pré-condition
- ☐ Une boucle à post-condition
- ☒ La boucle `for` est une boucle à pré-condition. Cela signifie que la condition est vérifiée avant l'exécution des instructions du corps de la boucle.

Question 2

`forEach` :

- ☐ S'utilise exactement de la même façon que `for`
- ☒ Permet de faire la même chose que `for... in`
- ☒ Est une méthode des classes itérables
- ☒ `forEach` est une méthode des classes itérables, permettant de faire la même chose que `for... in`. La syntaxe n'est cependant pas la même.

Question 3


La boucle `for ... in` est :

- ☒ Une boucle de parcours
- ☐ Une boucle ne se terminant jamais
- ☒ Une boucle à pré-condition
- ☒ La boucle `for ... in` est une boucle de parcours, c'est-à-dire qu'elle va s'exécuter pour chaque élément d'un itérable. Il s'agit également d'une boucle à pré-condition, puisque la condition sera vérifiée avant l'exécution du corps de la boucle.

Question 4

```
For (var i = 0; i < 2; i++) {
  print(i);
}
```


Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
 - ☒ 2 fois
 - ☐ Jamais
 - ☐ À l'infini
-  La variable est initialisée à 0, et est incrémentée de 1 à chaque tour. La boucle s'exécutera donc 2 fois.

Question 5

```
For (var i = 1; i < 0; i++) {  
  print(i);  
}
```


Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
 - ☐ 2 fois
 - ☒ Jamais
 - ☐ À l'infini
-  *i* est initialisée à 1. Or, la condition est vraie uniquement si *i* est inférieure à 0. La condition ne sera donc jamais vraie, et les instructions de la boucle ne seront jamais exécutées.

Exercice p. 7 Solution n°3


Question 1

La boucle `while` est :

- ☒ Une boucle à pré-condition comme `for`
 - ☐ Une boucle à post-condition
 - ☐ Une boucle de parcours
-  La boucle `while` est une boucle à pré-condition : la condition est vérifiée avant l'exécution des instructions de la boucle.


Question 2

Dans une boucle `while`, la déclaration de variable se fait :

- ☐ Dans les paramètres de la boucle
 - ☐ Dans le corps de la boucle
 - ☒ En dehors de la boucle
-  Dans une boucle `while`, la déclaration de la variable doit avoir été faite en dehors de la boucle.


Question 3

Dans une boucle `while`, la modification de la variable liée à la condition se fait :

- ☒ Dans le corps d'instructions de la boucle
- ☐ Dans les paramètres de la boucle
- ☐ En dehors de la boucle
-  Dans une boucle `while`, la modification de la variable se fait dans la boucle.

Question 4

La boucle `do ... while` est :


- ☐ Une boucle à pré-condition comme `for`
- ☒ Une boucle dont la condition est vérifiée après l'exécution du corps de la boucle
- ☐ Une boucle de parcours
-  La boucle `do ... while` vérifie la condition seulement après l'exécution du corps de la boucle.

Question 5

Pour ce morceau de code :

```
Var i = 5;
While (i > 5) {
    i++;
}
```


Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais
- ☒ À l'infini
-  La boucle ainsi construite n'a pas de fin.

Exercice p. 10 Solution n°4

Question 1

L'instruction `break` termine l'exécution de la boucle qui l'englobe, même si :


- ☐ La condition de la boucle qui l'englobe est toujours fausse
- ☒ La condition de la boucle qui l'englobe est toujours vraie
-  Si la condition de la boucle qui l'englobe est fausse, la condition `break` n'est jamais exécutée.

Question 2

`break` permet d'arrêter toutes les boucles dans lesquelles il était imbriqué.

☐ Vrai

☒ Faux

 `break` permet seulement d'arrêter la boucle dans laquelle l'instruction se trouve.


Question 3

`continue` permet :

☒ De ne pas réaliser la suite des instructions d'une boucle pendant un tour

☐ De faire tourner la boucle à l'infini

☒ De passer directement au tour de boucle suivant

 `continue` permet de ne pas réaliser la suite des instructions se trouvant dans la boucle, et de passer directement au tour de boucle suivant.

Question 4

```
For (i = 0; i < 2; i++) {  
  if (i == 1) {  
    continue;  
    print('je suis là');  
  }  
}
```

Combien de fois verrons-nous « *je suis là* » apparaître ?

☐ 1 fois

☐ 2 fois

☒ Jamais


 Les instructions après `continue` sont toujours ignorées.

Question 5

Un label me permet de :

☒ Faire référence à une boucle différente de celle dans laquelle je me trouve

☒ Faire référence à la boucle dans laquelle je me trouve

 Un label fait référence à la boucle que vous voulez, que ce soit à la boucle dans laquelle vous vous trouvez ou à une boucle à des niveaux supérieurs.


Il s'agit d'une solution possible, mais on peut également utiliser des boucles `while` !

```
1 import 'dart:io';
2
3 main() {
4   var z = 10; //le nombre de lignes à imprimer
5   for (var i = 0; i < z; i++) {
6     for (var j = 10; j > i; j--) {
7       stdout.write("*");
8     }
9     stdout.write("\n");
10  }
11  return 0;
12 }
```

Exercice p. 12 Solution n°6


Question 1

L'instruction `break` termine l'exécution de la boucle qui l'englobe, même si :

- ☐ La condition de la boucle qui l'englobe est toujours fausse.
- ☒ La condition de la boucle qui l'englobe est toujours vraie.
-  Si la condition de la boucle qui l'englobe est fausse, la condition `break` n'est jamais exécutée.


Question 2

La boucle `while` est :

- ☒ Une boucle de pré-condition comme `for`
- ☐ Une boucle de post-condition
- ☐ Une boucle de parcours
-  La boucle `while` est une boucle de pré-condition : la condition est vérifiée avant l'exécution des instructions de la boucle.

Question 3


Une boucle nécessite :

- ☒ Une variable à modifier
- ☒ Un bloc d'instruction
- ☐ Un paramètre de type `int`
- ☒ Une condition
- ☒ Une instruction menant à la modification de la variable
-  Une boucle nécessite une variable à modifier, un bloc d'instructions dans le corps de la boucle, une condition pour indiquer lorsque la boucle doit s'arrêter, et une instruction menant à la modification de la valeur de la variable.

Question 4

`forEach` :

- ☐ S'utilise exactement de la même façon que `for`
- ☒ Permet de faire la même chose que `for... in`
- ☒ Est une méthode des classes itérables


 `forEach` est une méthode des classes itérables, permettant de faire la même chose que `for... in`. La syntaxe n'est cependant pas la même.

Question 5

```
Var i = 5;
While (i > 5) {
  i++;
}
```

Combien de fois la boucle s'exécutera ?

- ☐ 1 fois
- ☐ 2 fois
- ☐ Jamais
- ☒ À l'infini

 `i` est initialisée à 1. Or, la condition est vraie uniquement si `i` est inférieure à 0. La condition ne sera donc jamais vraie, et les instructions de la boucle ne seront jamais exécutées.