Répétez des instructions

Introduction

Lorsque nous devons **réutiliser une tâche** dans notre code, nous regroupons souvent cette action dans une **fonction**. De même, lorsque nous voyons qu'**un processus doit se répéter plusieurs fois de suite**, nous écrivons une **boucle**. Les boucles nous permettent de créer un code efficace qui automatise les processus pour créer des programmes évolutifs et gérables.

Imaginez, par exemple, que dans un cours, pour un acte obscur que vous avez fait..., vous avez obtenu une conséquence négative qui vous demande d'écrire **1000 fois,** la phrase suivante: *Je ferai toujours les exercices avant d'attendre ou d'aller voir les solutions...*

WOW!!!! Vous allez sûrement espérer qu'un outil quelconque vous sauvera la vie et vous permettra d'écrire cela ***en un seul clic...*** *Heureusement, pour cela... les* ***boucles de programmation*** *seront d'une très grande aide...*

Les boucles de programmation

Une boucle est un **outil de programmation qui répète** un ensemble d'instructions **jusqu'à ce qu'une condition spécifiée**, appelée **condition d'arrêt**, soit **atteinte.** En tant que futur(e) programmeur(e) d'application multimédia interactive, vous constaterez assez vite que vous utiliserez des boucles très souvent! Vous entendrez aussi le terme générique **itérer** lorsque vous allez vous référer à des boucles; **itérer** signifie simplement «**répéter**».

Au lieu d'écrire le même code (ou les mêmes phrases…) encore et encore, il est possible d'**indiquer au programme de répéter un bloc de code** donné par ses propres moyens. Une façon de donner ces instructions à l'ordinateur consiste à utiliser différents types de boucle de programmation. Celles qui seront étudiées dans le cadre de ce cours seront: les **boucle for** et les **boucles for...of**.

La boucle for

La **boucle for** permet de **répéter** des instructions un **certain nombre de fois** simplement et rapidement et ce, jusqu'à ce qu'une condition donnée ne soit plus vérifiée.

La syntaxe de la **boucle for** est la suivante. À l'intérieur de parenthèses **( )**, elle contient **trois expressions** séparées par des **;** (point-virgule) , puis entre des accolades **{ }** , le bloc de code à exécuter est défini :

```
for (let variableInitiale = 0 ;  conditionExecution ;  iteration) {
    // instructions exécutées tant que la condition est vérifiée (true)
}
```

La boucle **for** initialise généralement une **variable d'itération** qui apparaît généralement dans les trois expressions. La **variable est initialisée,** vérifiée par rapport à la **condition d'exécution** et une **nouvelle valeur est affectée AUTOMATIQUEMENT à chaque itération de la boucle**.

Les variables itérateur peuvent avoir n'importe quel nom, mais il est recommandé d'utiliser un nom de variable *significatif*, malgré qu'il est fréquent d'utiliser la variable d'itération: ***i***.

Examinons de façon plus spécifique, le fonctionnement d'une boucle for avec l'exemple suivant:

```
for (let compteur = 0; compteur < 5 ;  compteur++) {
    console.log (compteur );
}
```

1. La **première étape** est l'**initialisation** qui se produit **une seule fois,** au début de l'exécution. Cette étape permet de définir la valeur initiale d'une variable qui sera impliquée dans la condition de la boucle. Dans l'exemple ci-haut, la variable locale "**compteur**" est initialisée avec la valeur **0**. Cette étape lance l'exécution de la boucle;
2. À la **deuxième étape**, la condition d'exécution est évaluée avant chaque tour de boucle. Si elle est vraie (true), un nouveau tour de boucle est effectué i.-e. que les instructions contenues dans la boucle sont exécutées. **Sinon, la boucle for se termine.** Dans notre exemple, la condition sera vérifiée tant que la valeur de la variable locale sera inférieure à 5. Dès qu'elle atteindra la valeur de 5, l'exécution de la boucle sera terminée;
3. À la **troisième étape**, la ou les instruction(s) du corps de la boucle sont exécutée(s). Dans notre exemple, la valeur de la variable de la boucle "**compteur**" sera affichée dans la console à chaque itération;
4. Lors de la **quatrième étape**, l'incrémentation de la variable locale est réalisée AUTOMATIQUEMENT par le programme après chaque tour de boucle. Elle sert à modifier la valeur de la variable initialisée, selon l'incrément indiqué. Dans notre exemple, la valeur de la variable "**compteur**" est augmentée de 1 à chaque itération. Notons que notre exemple, utilise l'opérateur **++**. Il est aussi possible d'utiliser l'opérateur d'affectation **+=**, comme par exemple: compteur **+= 1**;
5. **On retourne ensuite à l'étape 2**.

Pour revenir à notre exemple cité au début de cette note de cours, pour écrire automatiquement les 1000 phrases, le code serait simplement le suivant:

```
for (let iPhrase = 0 ; iPhrase  < 1000 ;  iPhrase ++ ) {
    console.log ("Je ferai toujours les exercices du cours avant d'attendre ou d'aller voir les solutions ...");
}
```

Et voilà… mine de rien… 1000 phrases sont écrites…

IMPORTANT!!! La **condition d'exécution** d'une boucle for doit **TOUJOURS être fausse** à **un moment donné SINON** la boucle n'arrêtera jamais et le code tournera à l'infini ce qui aura pour effet de faire planter votre code…**;-(**

Exemple d'affichage d'images avec une boucle

```
// Afficher l'image 3 fois à une distance de 50 l'un de l'autre sur le X
for (let compteur = 0 ; compteur  < 3 ;  compteur ++ ) {
            ctx.drawImage(vieImg, posX, 0);
           posX = posX + 50;
}
```

Le mot-clé break

Le mot-clé break permet aux programmes de "***sortir***" de la boucle depuis le bloc de code de la boucle. Dans l'exemple qui suit, on souhaite arrêter l'exécution de la boucle ŝi la valeur de i est > 3:

```
//Instruction pour la boucle
for (let i = 0 ; i < 10 ;  i ++ ) {
    // On sort de la boucle si i > 3
    if (i > 3 ) {
        break;
    }
     console.log(i);
}
//Retournera: 0 1 2 et 3
```

La boucle for...of

La boucle **for…of** est **spécialement créée** pour parcourir des éléments énumérables, comme le sont les tableaux (voir note de cours : *LesTableauxAvecJavaScript.pdf*) (Array). Regardons immédiatement comment utiliser ce type de boucle pratique avec le tableau suivant:

let lesVilles = ["New-York", "Beijing", "Montréal", "Paris", "Nairobi"];

Pour afficher la liste des villes présentes dans le tableau **lesVilles** avec la boucle **for...of,** le code sera le suivant:

```
for (let uneVille of lesVilles) {
  console.log(uneVille);
}

//Retournera:
"New-York"
"Beijing"
"Montréal",
"Paris"
"Nairobi"
```

Ici, on définit une variable **let uneVille** (on peut lui donner le nom qu'on souhaite) qui va stocker les différentes valeurs de notre tableau **lesVilles** une à une. La boucle **for… of** va en effet exécuter son code en boucle jusqu'à ce qu'elle arrive à la fin du tableau. **Ainsi, la boucle for...of permet d'exécuter simplement une ou plusieurs instructions pour la valeur de chaque index d'un tableau.**