Boucle while

1. Syntaxe

La boucle `while` permet de traduire une logique de Tant que avec un test en début de structure. La syntaxe est la suivante :

```
while(condition)
{
    séquence de code
}
```

Avec la boucle `while` une séquence de code est exécutée tant que la condition placée entre parenthèses est vraie.

Il faut bien évidemment avec cette itération que l'état de la condition puisse évoluer au cours de la séquence de code intégrée dans les accolades. Il faut aussi être extrêmement attentif au fait que la condition puisse être évaluable au premier passage sur celle-ci. Il s'agit d'une cause de dysfonctionnement fréquente. La condition porte aussi très souvent sur l'état d'une variable de type compteur. Bien sûr pour espérer que le compteur puisse atteindre une valeur limite, il faut le modifier au cours de la séquence de code (incrémentation ou décrémentation).

Enfin, vous noterez dans les scripts de ce livre et sur les nombreux exemples disponibles sur Internet, la syntaxe alternative suivante (que nous avions déjà évoquée pour la structure conditionnelle `if`) :

```
while(condition) {
    séquence de code
}
```

2. Exercice n°9 : Moyenne de 10 nombres saisis au clavier

Sujet

Calculer et afficher la moyenne de 10 nombres lus au clavier

En toute rigueur, cet exercice pourrait être traité par une boucle Pour dans la mesure où le nombre d'itérations (10) est connu en tout début de traitement.

Corrigé (partiel) en JavaScript

```
/* Déclaration de variables locales */
var nblu, cpt, somme, moyenne;

/* Initialisations */
cpt = 1;
somme = 0.0;

/* Boucle de traitement */
while (cpt<=10)
{
```

```
    nblu = parseFloat(prompt("Nombre n° " + cpt + " : "));
    somme = somme + parseFloat(nblu);
    cpt = cpt + 1;
}

/* Affichage du résultat */
moyenne = somme / 10;
document.write("Moyenne : "+ moyenne);
```

Commentaires du code JavaScript

Le script ne présente pas de difficultés particulières d'interprétation.

Il convient de faire attention tout de même à l'implémentation de la boucle `while` en JavaScript. La moyenne est divisée par 10 en fin de traitement. Une division par le compteur de tours de boucles `cpt` aurait donné un résultat non satisfaisait, la division se serait faite par 11.

N'oubliez pas non plus de réaliser les conversions dans la boucle en type réel pour la variable `nblu` (par l'intermédiaire de la méthode `parseFloat` pour les nombres réels) car la variable `nblu` bien qu'ayant été déclarée en début de script n'a pas de type explicite.

Vous avez peut-être noté la présence d'un espace entre le mot clé `while` et la condition ; cet espace est facultatif et permet d'aérer la syntaxe.

3. Exercice n°10 : Moyenne d'une série de n nombres saisis au clavier

Sujet

Calculer et afficher la moyenne de n nombres réels saisis au clavier (liste terminée par un zéro)

Nous nous trouvons ici face à une problématique qu'il n'est pas possible de traiter avec une structure itérative de type Pour (`for` en JavaScript).

Voyons comment réaliser ce calcul par l'intermédiaire d'un `while`.

Corrigé (partiel) en JavaScript

```
/* Initialisations */
cpt = 1;
somme = 0.0;

/* Saisie d'un premier nombre */
nblu = parseFloat(prompt("Nombre (0 pour finir) : "));

/* Boucle de traitement */
while (nblu != 0)
{
    somme = somme + parseFloat(nblu);
    nblu = parseFloat(prompt("Nombre (0 pour finir) : "));
    cpt = cpt + 1;
}
```

```
/* Affichage du résultat */
if (cpt == 1)
{
    document.write("Aucun nombre n'a été saisi");
}
else
{
    moyenne = somme / (cpt-1);
    document.write("Moyenne : "+ moyenne);
}
```

Commentaires du code JavaScript

Pour que la condition soit évaluable lors du premier passage sur `nblu != 0`, il a fallu faire une première saisie au clavier avant la boucle `while`. Il ne faut bien évidemment pas oublier dans la séquence de code intégrée dans la boucle de prévoir un second `prompt` pour que la lecture des nombres suivants soit possible.

Au niveau de l'affichage, il faut enfin prévoir un test pour isoler le cas particulier d'une suite de chiffres ne contenant que le "marqueur de fin de liste 0". Dans ce cas, il faut surtout éviter la division par zéro. En réalité en JavaScript, il n'y aurait pas de "plantage dur" du navigateur, la valeur affichée serait `NaN` (*Not a Number*).

Quelques rappels au niveau des conditions :

- Pour signifier différent, l'opérateur sera noté != (au lieu du <> vu au chapitre Développement à partir d'algorithmes).
- Pour la comparaison par rapport à 1 dans l'affichage du résultat, il faut bien prévoir un double égal (==). Le signe égal (=) en JavaScript est réservé à l'affectation.