Les conditions et les boucles

1. Les conditions if ... else

Les langages de programmation n'échappent pas aux tests conditionnels. C'est ce qui fait d'ailleurs une de leurs richesses.

L'expression `if` (si) permet d'exécuter, ou non, une série d'instructions en fonction du résultat d'un test.

```
if (condition vraie) {
une ou plusieurs instruction(s);
}
```

Si la condition est vérifiée (`true`), les instructions s'exécutent. Si elle ne l'est pas (`false`), les instructions ne s'exécutent pas et le programme passe à la commande suivante.

Remarquons que les instructions sont comprises entre une accolade ouvrante et une fermante.

Voici une forme plus évoluée :

```
if (condition vraie) {
instructions 1;
}
else {
instructions 2;
}
```

Si la condition est vérifiée (`true`), le bloc d'instructions 1 s'exécute. Sinon (`else`), soit lorsque la condition renvoie la valeur false, le bloc d'instructions 2 s'exécute.

Exemple

Le visiteur entre un nombre compris entre 0 et 99. Le script annonce si le nombre saisi est inférieur ou égal à 50 ou supérieur à 50.

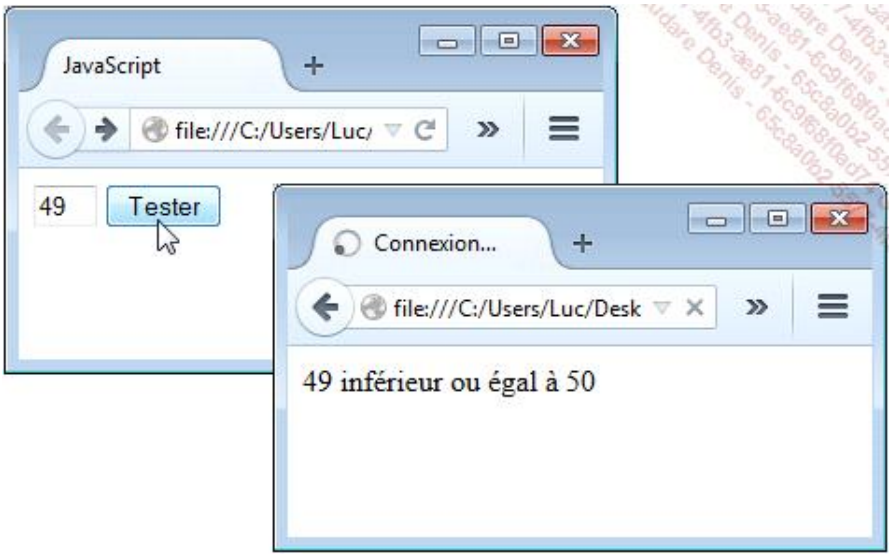
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function test(){
var a = document.getElementById("input").value;
if (a <= 50) {
document.write(a +"  inférieur ou égal à 50");
}
else {
document.write(a + " est supérieur à 50");
}
}
</script>
</head>
```

```
<body>
<form id="formulaire">
<input type="text" id="input" size="2" maxlength="2" value="">
<input type="button" value="Tester" onclick="test()">
</form>
</body>
</html>
```

- Le formulaire comporte une ligne de texte limitée à 2 positions (voir `maxlength="2"`) pour saisir un chiffre compris entre 0 et 99.
- En cliquant sur le bouton **Tester**, le visiteur appelle la fonction `test()` définie entre `<head> ... </head>`.
- Dans la variable `test()`, la variable a, chiffre saisi dans la zone de texte, est récupérée. Pour ce faire, le formulaire est identifié et la valeur de zone de texte, nommée input, est extraite.

Ce chemin s'écrit en JavaScript, `document.getElementById("input")`.

- La condition "a inférieur ou égal à 50" (a <= 50) est testée.
- Si cette condition est réalisée, l'information "chiffre saisi inférieur ou égal à 50" est écrite dans le document.
- Si cette condition n'est pas réalisée (`else`), l'information "chiffre saisi est supérieur à 50" est écrite dans le document.
- Remarquez les deux accolades de fin. La première termine le test if ... else. La seconde termine la fonction `test()`.



Il est également possible de concevoir des tests conditionnels multiples.

```
if (condition 1) {
instruction(s) à exécuter si la condition 1 est vraie
}
else if (condition 2) {
code à exécuter si la condition 2 est vraie
}
else
{
code à exécuter si tous les tests sont faux
```

```
}
```

Pour ceux qui aiment les notations concises, on peut aussi écrire :

```
(expression) ? instruction a : instruction b
```

Si l'expression entre parenthèses est vraie, l'instruction `a` est exécutée. Si l'expression entre parenthèses est fausse, c'est l'instruction `b` qui est exécutée. Remarquons le double point entre les deux instructions `a` et `b`.

Le script précédent deviendrait alors :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function test(){
var a = document.getElementById("input").value;
a <=50 ?  document.write(a + " est inférieur ou égal à 50") :
document.write(a +
" est supérieur à 50");
}
</script>
</head>
<body>
<form>
<input type="text" id="input" size="2" maxlength="2" value="">
<input type="button" value="Tester" onclick="test()">
</form>
</body>
</html>
```

Le test conditionnel `if ... else` est un pilier du JavaScript et de la programmation. Il sera fréquemment utilisé dans les scripts d'une certaine complexité.

2. La boucle for

L'expression `for` permet d'exécuter un bloc d'instructions un certain nombre de fois (boucle) en fonction de la réalisation d'un certain critère. L'instruction `for` prévoit d'emblée un compteur et une condition pour l'interruption.

La syntaxe générale est :

```
for (valeur initiale ; condition ; progression) {
instructions;
}
```

Prenons un exemple concret :

```
for (i=0; i<5; i++) {
```

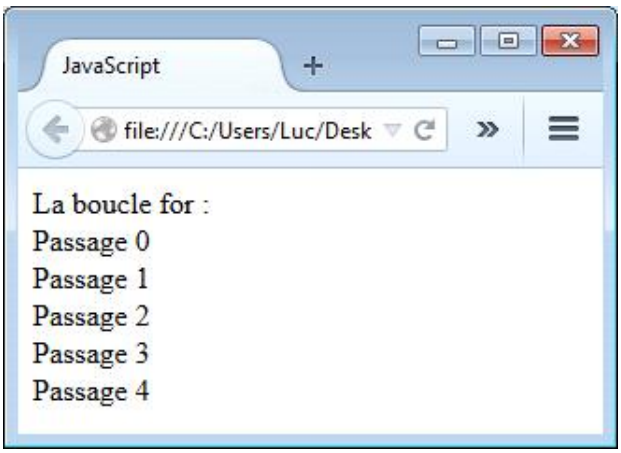
```
document.write(i + "<br>")
}
```

Au premier passage, la variable `i` vaut 0 (sa valeur initiale). Elle est bien inférieure à 5. Les instructions s'exécutent. La variable `i` est ensuite incrémentée d'une unité par l'opérateur d'incrémentation `i++` (`i` vaut alors 1) et la boucle for continue son exécution.

La variable `i` (qui vaut 1) est toujours inférieure à 5. L'instruction est à nouveau exécutée. La variable est augmentée de 1 par l'incrémentation.

Et ainsi de suite jusqu'à obtenir 5 pour `i`. La variable `i` ne remplit alors plus la condition `i` inférieur à 5. La boucle s'interrompt et le programme continue après l'accolade de fermeture.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
</head>
<body>
La boucle for :<br>
<script type="text/JavaScript">
for (i=0; i<5; i++) {
document.write("Passage " + i + "<br>")
}
</script>
</body>
</html>
```



Cette instruction `for` est très utilisée en JavaScript.

3. La boucle while

L'instruction `while` permet d'exécuter une instruction (ou un groupe d'instructions) un certain nombre de fois.

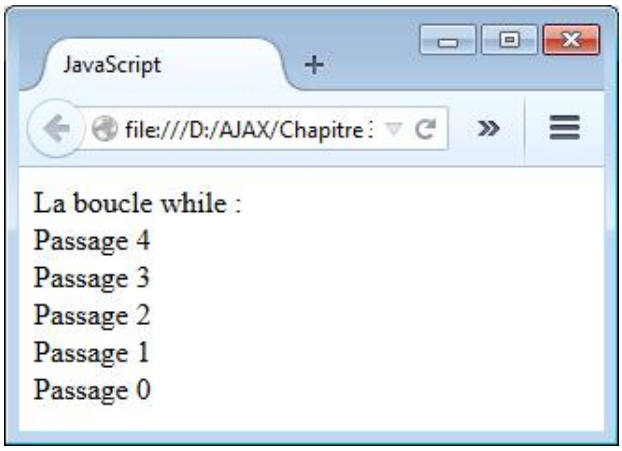
```
while (condition vraie) {
instruction(s)
```

```
}
```

Aussi longtemps que la condition est vérifiée, JavaScript continue à exécuter les instructions entre les accolades. Une fois que la condition n'est plus vérifiée, la boucle est interrompue et le script continue l'exécution.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
</head>
<body>
La boucle while :<br >
<script>
compt = 4;
while (compt>=0) {
document.write("Passage " + compt + "<br>");
compt--;
}
</script>
</body>
</html>
```



Le résultat est identique à l'instruction `for`. Dans la pratique, l'instruction `for` est souvent préférée à `while` car elle intègre un compteur.

Le risque de boucle infinie est d'autant plus présent avec `while` que le concepteur doit créer et incrémenter lui-même le compteur.

4. L'instruction break

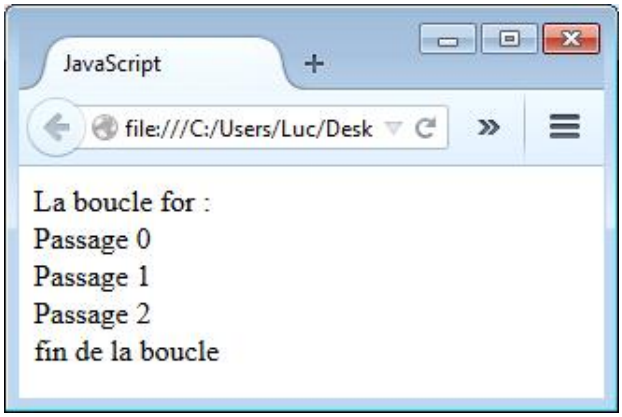
L'instruction `break` permet d'interrompre prématurément une boucle `for` ou `while`.

Pour illustrer ceci, reprenons notre exemple :

```
for (i=0; i<5; i++) {
if (i == 3)
break;
document.write("Passage " + i + "<br>")
}
document.write("fin de la boucle");
```

Le fonctionnement est semblable à l'exemple de la section La boucle for sauf lorsque le compteur vaut 3. À ce moment, on sort de la boucle par l'instruction `break` et le message "fin de la boucle" s'affiche.

Ce qui donne à l'écran :



5. L'instruction continue

L'instruction `continue` permet de sauter une instruction dans une boucle `for` ou `while` et de continuer ensuite le bouclage (sans sortir de celui-ci comme le fait `break`).

Soit l'exemple :

```
compt=0;
while (compt<5) {
if (compt == 3)
{
compt++
continue;
}
document.write ("ligne : " + compt + "<br>");
compt++;
}
document.write("fin de la boucle");
```

Ici, la boucle démarre. Lorsque le compteur vaut 3, l'instruction `document. write` ne s'exécute pas grâce à l'instruction `continue` (ligne : 3 n'est pas affiché) et la boucle poursuit son processus. Notons que compt a été incrémenté (compt++) avant l'instruction `continue` pour éviter un bouclage infini et que le navigateur ne rende la main.

Ce qui donne à l'écran :