Les notions fondamentales

Avant de passer à l'étude proprement dite de tout langage de programmation, aussi modeste soit le JavaScript, il importe de parcourir quelques règles d'écriture et de syntaxe.

1. La casse

Le JavaScript est sensible à la casse (*case sensitive*). Il fait donc la différence entre les majuscules et les minuscules.

Ainsi, la variable `var = numero` n'est pas égale à la variable `var = Numero` ni à `var = NUMERO`.

L'alphabet ASCII composé de 128 caractères est utilisé. Ainsi le code doit être écrit sans caractères accentués.

Les espaces, tabulations, retours chariot, lignes vierges sont (sauf exceptions) ignorés dans le code.

2. Les commentaires

Il est toujours utile d'annoter le code avec des commentaires en vue de faciliter sa lisibilité, surtout à la relecture.

En JavaScript, ces commentaires peuvent se présenter sous deux formes différentes :

- par une double barre oblique (//) devant le commentaire. Cette notation est bien adaptée pour les commentaires qui se notent sur une seule ligne.
  ```
  // Ceci est un commentaire
  ```
- par les signes `/* ... */`, ce qui permet d'étendre le commentaire sur plusieurs lignes.
  ```
  /* Pour les bavards, JavaScript a prévu
  des commentaires sur plusieurs lignes */
  ```

3. Les points-virgules

Chaque commande ou instruction JavaScript se termine par un point-virgule.

Exemple : `document.write ("ligne : " + compt + "<br>");`

Bien que cela ne soit plus obligatoire en toutes circonstances, il reste prudent de maintenir cette règle de façon systématique lorsqu'on est débutant en JavaScript.

4. Les constantes

Les constantes sont fixées par la valeur indiquée dans le code.

Les valeurs reconnues par le JavaScript sont :

- **Les nombres entiers ou à virgule flottante**. Notons que pour ces derniers, le point remplace la virgule habituellement utilisée en bureautique.

  2005

  16.6666

- **Les chaînes de caractères**. Elles sont constituées d'une suite de caractères quelconques, encadrée par des

guillemets ou des apostrophes. Une chaîne commencée avec des guillemets doit se terminer avec des guillemets, même chose pour l'apostrophe.

"CSS + DHTML"

'3.1416'

Si la chaîne de caractères contient un guillemet, une apostrophe ou une barre oblique à gauche (*backslash*), ces derniers risquent d'être interprétés comme du code JavaScript. Les caractères cités s'encodent alors précédés d'une barre oblique à gauche.

Soit :

\' pour l'apostrophe.

\" pour le guillemet.

\\ pour la barre oblique à gauche.

```
document.write ("Je lui ai dit \; \"Va à l\'école \"").
```

Une chaîne de caractères peut être vide. C'est l'équivalent de la valeur zéro pour une constante numérique. Dans ce cas, elle s'écrit au moyen de deux guillemets ou de deux apostrophes rigoureusement consécutifs (sans espace entre eux).

- **Les valeurs logiques** ou booléennes, soit *true* pour vrai et *false* pour faux.
- **null** lorsqu'il n'y a pas de valeur attachée à la variable.
- **undefined**. C'est la valeur d'une variable qui n'a pas été initialisée.

5. Les variables

Contrairement à la plupart des autres langages de programmation, JavaScript n'est que faiblement typé. En effet, il n'est pas nécessaire d'en déclarer le type (comme par exemple avec int, float, double, char de PHP) et une variable peut à tout moment changer de type.

Les variables se déclarent de deux façons :

- Explicitement par le mot clé `var`.
  ```
  var indice
  ```
- Implicitement par son apparition à gauche du signe égal.
  ```
  b = 256
  ```

Le nom de la variable doit respecter la syntaxe suivante :

- La variable doit commencer par une lettre ou un souligné "_".
- La variable peut comporter un nombre indéterminé de lettres, de chiffres ainsi que des caractères _ et $.
- Les espaces ne sont pas autorisés.
- Le nom de variable ne peut utiliser des mots dits "réservés". En effet, ces mots sont utilisés dans le code JavaScript même. On ne peut nommer une variable, par exemple, var, true, false, else, etc.
- Pour rappel, le JavaScript est sensible aux majuscules et minuscules.

Exemples de déclaration de variables valides :

- `var pi;`
- `var code_postal;`

- `var formulaire1;`
- `var result$;`

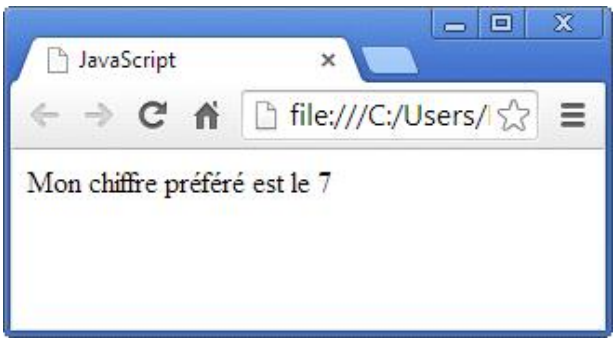
On peut initialiser une variable en même temps que sa déclaration au moyen du signe égal (=) suivi d'une valeur numérique, d'une chaîne de caractères ou d'une valeur booléenne. Ce qui évite que la valeur *undefined* soit retournée en cours d'exécution du script.

Exemples :

- `var pi = 3.1415926535;`
- `var code_postal = 59000;`
- `var formulaire1 = "Ville";`
- `var result$ = true;`

Exemple :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
</head>
<body>
<script>
var texte = "Mon chiffre préféré est le ";
var variable = 7;
document.write(texte + variable);
</script>
</body>
</html>
```



6. Les opérateurs

JavaScript comporte de nombreux opérateurs selon le type de valeurs.

a. Les opérateurs arithmétiques

Ce sont les opérateurs classiques de toutes les opérations de calcul.

Opérateur	Nom	Signification	Exemple
+	plus	addition	x + 100
-	moins	soustraction	x - 1
*	multiplié	multiplication	x * 2
/	divisé	division	x / 10
%	modulo	reste de la division par	356 % 5
=	égal	affectation de valeur	i = 4

Commentaires :

- Lorsque l'opérateur + porte sur deux opérateurs dont l'un au moins est une chaîne de caractères, il joue le rôle d'opérateur de concaténation (voir plus loin).
- L'opérateur = affecte simplement une valeur. On le confond souvent avec l'opérateur d'égalité ==.

b. Les opérateurs de comparaison

Ces opérateurs sont souvent utilisés lors de tests conditionnels.

Opérateur	Nom	Signification
<	inférieur	x < 10
<=	inférieur ou égal	x <= 10
==	égal	x == 10
>=	supérieur ou égal	x >= 10
>	supérieur	x > 10
!=	différent	x != 10

Pour rappel, le = est un opérateur d'attribution de valeur tandis que le == est un opérateur de comparaison. Cette confusion est une source fréquente d'erreurs de programmation.

Le résultat renvoyé par une opération de comparaison est une valeur booléenne `true` (vrai) ou `false` (faux).

c. Les opérateurs booléens (ou logiques)

Ces opérateurs sont également utilisés dans les tests conditionnels, mais logiques cette fois. Ils portent sur des opérandes booléens et le résultat renvoyé est également une valeur booléenne (`true` ou `false`).

Opérateur	Nom	Signification	Exemple
&&	et	les deux conditions sont vérifiées	condition1 && condition2
||	ou	une des deux conditions est vérifiée	condition1 || condition2
!	!	la condition n'est pas vérifiée	! condition1

d. Les opérateurs associatifs

Ces opérateurs associent deux opérateurs d'affectation. C'est une forme abrégée pour noter les opérateurs de calcul. Ceci est plutôt réservé à des programmeurs plus confirmés.

Opérateur	Nom	Exemple	Signification
+=	plus égal	x += y	x = x + y
-=	moins égal	x -= y	x = x - y
*=	multiplié égal	x *= y	x = x * y
/=	divisé égal	x /= y	x = x / y

e. Les opérateurs d'incrémentation

Un classique des langages de programmation mais souvent déroutant pour les novices. Cet opérateur d'incrémentation est très utilisé pour faire varier d'une unité les compteurs implémentés dans le code.

Ainsi pour une valeur de départ x = 0, au premier passage x++ vaut 1 (x=x + 1 ou 1=0 + 1). Au second passage, x++ vaut 2 (x=x +1 ou 2=1 + 1). Au troisième passage, x++ vaut 3 (x=x +1 ou 3=2 + 1). Et ainsi de suite.

Opérateur	Nom	Exemple	Signification
x++	incrémentation	x = x++	x++ est le même que x=x+1
x--	décrémentation	x = x--	x-- est le même que x=x-1

f. Les opérateurs de concaténation

Utilisé avec des chaînes de caractères, cet opérateur ajoute une chaîne de caractères à la suite d'une autre chaîne de caractères.

Opérateur	Nom	Exemple	Signification
+	concaténation	"chaine1" + "chaine2"	"chaine1 chaine2"

Lorsque vous concaténez des chaînes de caractères, il ne faut pas oublier les espaces au début ou à la fin de chacune d'elles, sous peine de coller les chaînes.

g. Autres opérateurs

Il existe encore d'autres opérateurs, mais ils dépassent largement le cadre de cet ouvrage.

Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title
<meta charset="UTF-8">
<script>
var variable = (100 - 50) * 2;
document.write("(100 - 50) * 2 = " + variable);
</script>
```

```
</head>
<body>
</body>
</html>
```

h. Priorité des opérateurs

Comme en algèbre, lorsqu'une expression comporte plusieurs opérateurs, celle-ci est évaluée selon la priorité respective des opérateurs.

Voici, pour les opérateurs repris dans notre étude, la liste des priorités, de la plus basse à la plus haute.

Opérateur	Symbole
virgule	,
association et affectation	= += -= *= /= < <= > >=
ou logique	||
et logique	&&
comparaison	= != < <= >= >
addition/soustraction	+ -
multiplication/division	* /
différent et incrémentation	! ++ --
parenthèses	( )

En cas de priorité égale d'opérateurs consécutifs, l'opération est effectuée de gauche à droite.

On peut toujours modifier une priorité d'opérateurs par l'emploi de parenthèses car celles-ci ont le plus haut niveau de priorité.

7. L'accès aux éléments du HTML

Une page HTML est un ensemble d'éléments ou d'objets. Ces éléments peuvent être par exemple une balise de titre, un paragraphe ou un champ de formulaires. Cet ensemble d'objets constitue ce que l'on appelle le DOM de la page, soit le document object model ou le modèle des objets du document. L'étude détaillée du DOM sera abordée plus loin dans cet ouvrage mais nous avons besoin d'une première approche pour mener à bonne fin les exemples des chapitres suivants.

Le JavaScript doit pouvoir sélectionner un ou des éléments pour y associer des fonctions ou des méthodes. Une

façon de sélectionner un objet est fournie par la méthode `getElementById()` qui permet d'accéder à un élément déterminé doté d'un identifiant. Ce dernier est défini par l'attribut HTML `id="identifiant"` de la balise de l'objet. Ce sélecteur fonctionne ainsi en deux temps :

- Définir dans le code HTML l'attribut id="identifiant" associé à une balise.
- Accéder en JavaScript à l'objet par la méthode `getElementById("identifiant")`.

Cette méthode `getElementById()` pourrait se traduire par "trouver un élément par son identifiant id". Il est important de respecter à la lettre les différentes majuscules et minuscules.

Cette sélection par l'identifiant est également reprise dans les feuilles de style CSS.

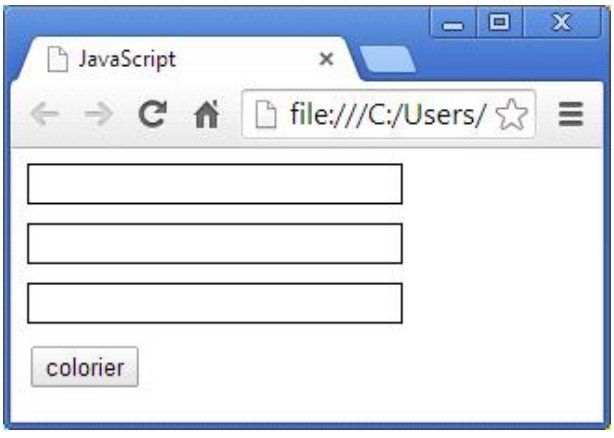
Exemple

Colorons une division au clic sur un bouton.

Le fichier HTML

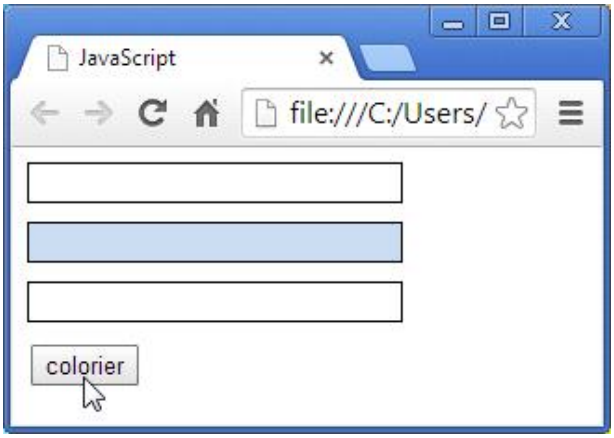
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<style>
div { width:200px; height: 20px;
      border: 1px solid black;
      margin-bottom: 10px;}
</style>
</head>
<body>
<div></div>
<div id="couleur"></div>
<div></div>
<button type="button" onclick="colorier()">colorier</button>
</body>
</html>
```

Soit 3 divisions dont une est identifiée par `id="couleur"`.



Ajoutons le script. Le document devient alors :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<style>
div { width:200px; height: 20px;
      border: 1px solid black;
      margin-bottom: 10px;}
</style>
</head>
<body>
<div></div>
<div id="couleur"></div>
<div></div>
<button type="button" onclick="colorier()">colorier</button>
<script>
function colorier() {
document.getElementById("couleur").style.backgroundColor="rgb(195,
215,235)";
}
</script>
</body>
</html>
```



Que les lecteurs familiarisés avec le JavaScript se rassurent, outre `getElementById` abordé ici, les autres méthodes comme `getElementsByName`, `getElementsByTagName`, et autres `document.getElementsByClassName` seront détaillées au chapitre Le DOM (Document Object Model) - Accéder aux objets.