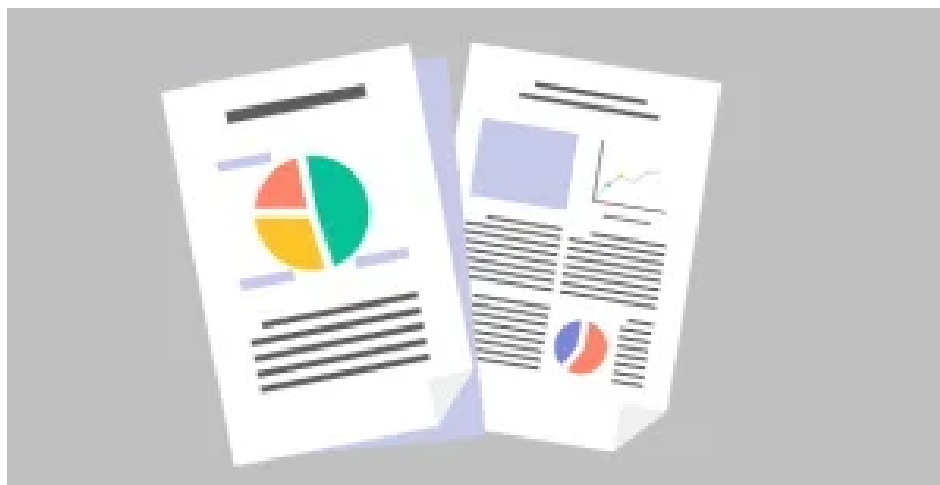



Guide complet sur le JavaScript



3.5 étoiles sur 5 a partir de 31 votes. 

Votez ce document: 

1/Introduction

2/La structure

2.1/La casse

2.2/Les espaces

2.3/Le point-virgule

2.4/Le point

2.5/Les sauts de ligne

2.6/Les commentaires

2.7/Les types de valeurs

3/Les variables

3.1/Déclaration et affectation

3.2/La portée

3.3/Affectation du type

3.4/Les littéraux

3.5/Les tableaux de littéraux

3.6/Les littéraux booléens

3.7/Les littéraux entiers

3.8/Les littéraux à virgule flottante

3.9/Les littéraux objets

3.10/Les littéraux chaînes de caractères

3.11/Les littéraux

3.11.1/Affichage des caractères

3.11.2/Les séquences d'échappement

3.11.3/compatibilité avec le code ASCII et ISO

3.12/Les identificateurs

3.13/Les mots réservés

4/Les instructions

4.1/Les groupes

4.2/Les labels

4.3/Les instructions conditionnelles

4.3.1/If else

4.3.2/With

4.3.3/Switch case

4.4/Les boucles 4.4.1/For

4.4.2/Do while

Articles similaires

[8 erreurs que les étudiants en anglais commettent et leurs solutions](#)

[Exercice comptabilité : méthodes coût complet, sections homogènes](#)


[Exercice HTML: Réalisation d'un Formulaire de QCM](#)

[WhatLaptops : un guide exceptionnel pour choisir son ordinateur](#)


[Exercice HTML: Formulaire Html avec Fonction Javascript](#)

[Offre d'emploi: Web designer en Freelance](#)


Documents similaires

 [Cours Complet de JavaScript](#)


Cours Complet de JavaScript

 [Guide complet sur le JavaScript](#)


Guide complet sur le JavaScript

 [Cours détaillé en JavaScript](#)


Cours détaillé en JavaScript

 [Management des ressources humaines guide complet](#)

Management des ressources humaines guide complet

 [Guide complet technique secretariat bureautique](#)

Guide complet technique secretariat bureautique

 [Guide complet de l'economie de l'education](#)

Guide complet de l'economie de l'education

5.12/In

5.13/Contactez-nous

5.14/New
A propos de nous

5.15/This

On recrute

5.16/Typeof

5.17/Recherche
Rechercher dans le site

5.18/Void
Politique de confidentialité

6/Les expressions régulières
Droit d'auteur/Copyright

6.1/Créations d'expressions régulières constantes

6.2/Plan du site
Plan du site

6.3/Les attributs d'extensions de recherche

6.4/Les caractères spéciaux

6.5/Les méthodes et les objets

6.6/Les propriétés et les méthodes de RegExp

7/Les fonctions



7.1/Function

7.2/new Function

7.3/Appel d'une fonction

7.4/Les arguments

7.5/Les tableaux d'arguments

7.6/Eval

7.7/isFinite

7.8/isNaN

7.9/parseFloat

7.10/parseInt

7.11/Number

7.12/String

7.13/Escape

7.14/Unescape

7.15/Return

8/**Les objets**

8.1/ActiveXObject

8.2/Anchor

8.3/applet

8.4/Arguments

8.5/Array

8.6/Boolean

8.7/Button

8.8/Checkbox

8.9/Date

8.10/Dictionary

8.11/Document

8.12/Drive

8.13/Drives

8.14/Enumerator

8.15/Error

8.16/Event

8.17/File

8.18/Files

8.19/FileSystemObject

8.19/FileSystemObject

8.20/FileUpload

8.21/Folder

8.22/Folders

8.23/Form

8.24/Frame

8.25/Function

8.26/Global

8.27/hidden

8.28/History

8.29/HTMLElement

8.30/Image

8.31/Input

8.32/JavaArray

8.33/JavaClass

8.34/JavaObject

8.35/JavaPackage

8.36/JSObject

8.37/Layer

8.38/Link

8.39/Location

8.40/Math

8.41/MimeType

8.42/Navigator

8.43/Number

8.44/Object

8.45/Option

8.46/Packages

8.47/Password

8.48/Plugin

8.49/PrivilegeManager

8.50/Radio

8.51/RegExp

8.52/Reset

8.53/Screen

8.54/Session

8.54/Select

8.55/String

8.56/Style

8.57/Submit

8.58/Text

8.59/Textarea

8.60/TextRange

8.61/TextStream

8.62/VBArray

8.63/Window

9/Les objets DHTML pour Internet Explorer

10/Les objets Javascript pour Netscape

11/Les événements

12/Les chaînes de requêtes

1 / Introduction

Les programmes Javascript permettent de rendre dynamique un site internet développé par le langage HTML. Comme nous avons pu le constater auparavant, le langage HTML crée des pages statiques dont les carences dans le domaine de l'interactivité peuvent désormais provoquer une désaffection du public pour votre site.

Créer un site interactif et dynamique peut constituer un atout indéniable pour **améliorer ou conforter votre audience** mais aussi afin de permettre à l'utilisateur **une navigation plus aisée** et de fournir **un aspect attractif** à votre site.

Le langage Javascript permet de développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet. Néanmoins, plus le degré de complexité de ces applications est important, plus la puissance de calcul du processeur sera sollicité jusque dans ses confins. Ainsi, **un programme Javascript requiert une machine puissante et rapide chez l'utilisateur.**

Contrairement à un applet Java qui est un programme compilé, les scripts écrits en Javascript sont interprétés, cela explique en partie la caractéristique précitée.

De la même manière de sorte à ne pas confondre les différents programmes : **le Java, représenté par un ou plusieurs fichiers autonomes** dont l'extension sera **.class* ou **.jar*, est invoqué par une balise HTML spécifique, alors que **le JavaScript est écrit directement au sein du document HTML** sous forme d'un script encadré par des balises HTML spéciales.

Cependant, **le Javascript est un langage de script simplifié orienté objet dont la syntaxe est basée sur celle du Java** qui lui-même prend sa source dans les langages de développement C et C++.

Le langage Javascript a été initialement élaboré par Netscape en association avec Sun Microsystems. Plus tard, Microsoft développera son propre langage Javascript officiellement connu sous le nom de *JScript*.

Tout comme, le langage HTML ou les feuilles de style, **le Javascript est standardisé par un comité**

spécialisé, en l'occurrence l'ECMA (European Computer Manufacturers Association).

Afin de ne favoriser aucun des concepteurs précités, **cette association a décrété que ce langage de script porterait le nom commun de *ECMAScript*** dont les références se trouvent dans le standard ECMA-262. Ce dernier regroupe la totalité des fonctionnalités du langage Javascript aussi bien présentes dans le langage de Netscape ou de celui de Microsoft mais aussi d'autres qui ne seraient implémentées ni dans l'un, ni dans l'autre.

Enfin, **les programmeurs avertis ne ressentiront guère de difficultés à se familiariser à ce langage et les débutants devraient par une bonne connaissance du langage HTML et certainement par un bon investissement personnel réussir à rapidement maîtriser cet outil**. Dans tous les cas, le jeu en vaut la chandelle !

Voici quelques uns des éditeurs Javascript ou HTML offrant un module Javascript:

WebExpert intègre un module Javascript en plus de ces fonctionnalités HTML,

Frontpage accueille naturellement son propre langage JScript,

HotMetal Pro offre également la possibilité d'écrire des scripts, DreamWeaver se distingue dans la création de page dynamique, Golive propose des fonctionnalités équivalentes.

2 / La structure

La syntaxe définit des règles d'écritures sans lesquelles un programme Javascript ne pourrait fonctionner correctement.

La syntaxe du langage Javascript s'appuie sur le modèle du Java, ressemblant lui-même à celui du C ou du C++.

Ainsi, les programmeurs ne rencontreront aucunes difficultés à se familiariser avec cette structure. Les dilettantes n'y verront pas plus d'obstacles.

2.1 / La casse

En ce qui concerne la casse, le langage Javascript y est extrêmement sensible. Ainsi, le strict respect des majuscules et minuscules est une condition sine qua non au bon fonctionnement de vos scripts.

En conséquence soyez vigilant sur la casse des variables et instructions.

Par exemple, écrivez systématiquement les instructions et les variables en lettres minuscules et les constantes en lettres majuscules.

```
var START = "début"; var Start = "un"; var start = 0; var log = LOG10E; var racine = SQRT2;
```

2.2 / Les espaces

L'insertion des espaces peut s'effectuer n'importe où dans le script comme lors d'une rédaction habituelle.

C'est-à-dire, des espaces peuvent séparer les valeurs des opérateurs et eux-mêmes des identificateurs, etc..

```
//Dans ce cas les deux méthodes sont bonnes var identificateur="Valeur"; var identificateur = "Valeur";
```

```
//Par contre ici le mot-clé var. var initialisation = i + 1;
```

```
//ne peut être accolé à l'identificateur. var initialisation=i+1;
```

```
//Cette méthode est correct. document.write("Chapitre");  
//Celle-ci l'est également.  
document.write ( "Chapitre" );  
//Ici ce n'est pas bon !  
document . write ( "Chapitre" );
```

2.3 / Le point-virgule

Chaque commande doit être terminée par un point-virgule (;).

Bien que cela ne soit pas obligatoire, il est préférable pour de bonnes règles de programmation de respecter systématiquement cette règle.

```
var increment = i+1; var jour = "Lundi";  
if (choix == "non")  
{  
document.write('Votre choix est l\'abandon');  
}
```

2.4 / Le point

Dans les opérations mathématiques, un nombre à virgule flottante est séparé en France par une virgule, mais celle-ci a une signification particulière dans le langage Javascript.

C'est pourquoi, ces nombres doivent être séparés par un point (.) comme le montre ces exemples :

```
var PI = 3.141592654; var prix = 2.50; nb = 5.3 * 14.025;
```

2.5 / Les sauts de ligne

Les sauts de ligne peuvent être placés partout où se trouve un espace, y compris au sein d'une commande.

Néanmoins, afin d'éviter toutes ambiguïtés, il vaudra mieux éviter de découper sur plusieurs lignes une instruction car si les sauts de ligne ne sont pas correctement interprétés, un point-virgule pourrait être inséré automatiquement en fin de ligne rendant la commande inopérante.

```
var texte = "ce texte est bien trop long pour tenir sur une seule ligne, c'est pourquoi un saut de ligne  
sera nécessaire"; var init = 0;
```

```
document.write("document.write(""  
+ "Rémunération sur Internet  
+ ""  
+ "  
+ "Erreur ! Nom du fichier non spécifié.");
```

2.6 / Les commentaires

Les commentaires permettent de rendre votre code lisible et surtout d'en faciliter ultérieurement la maintenance.

En général, l'insertion de commentaire se fait soit en fin de ligne, soit sur une nouvelle ligne mais en aucun cas au sein d'une ligne de commande.

Il existe deux méthodes permettant d'intégrer des commentaires à vos scripts.

La première consiste à placer un double slash (//) devant le texte comme dans l'exemple ci-dessous :

```
var image = ""; //image de fond  
if (x=2) //redirection vers la seconde page  
{ url = (""); //action  
}  
else  
{ //sinon retourne à la page d'accueil url = (""); }
```

La seconde solution est d'encadrer le texte par un slash suivi d'une étoile (/*) et la même séquence inversée (*/) comme le montre l'exemple suivant :

```
/*Voici un commentaire*/  
/*  
Encore un autre commentaire  
*/  
/*Ce commentaire est écrit sur deux lignes*/
```

2.7 / Les types de valeurs

Le langage Javascript reconnaît plusieurs types de valeurs :

Les nombres entiers ou à virgule flottante comme "42" ou "3.14159".

Les valeurs logiques (Booléennes), *true* (vrai) et *false* (faux).

Les caractères comme 'a', '5' '.', etc..

Les chaînes de caractères comme "Bonjour !".

Null, un mot-clé spécial symbolisant une valeur nulle; le *null* est aussi une valeur primitive. Parce que JavaScript est sensible à la casse, *null* n'est pas le même comme le *Null*, le *NULL*, ou une autre variante.

Undefined, est **une propriété de niveau supérieur dont la valeur est non définie;** *undefined* est aussi une valeur primitive.

Ce jeu relativement restreint de types de valeurs ou de types de données, permet néanmoins d'exécuter des fonctions utiles avec vos applications.

Il n'y a aucune distinction explicite entre l'entier et les nombres réels.

Il n'y a également pas de type de données sous forme de dates explicites dans Javascript. Cependant, vous pouvez utiliser l'objet de Date et ses méthodes afin de manipuler des dates.

3 / Les variables

Vous utilisez des variables comme identificateurs pour affecter des valeurs qui peuvent être sollicitées n'importe où dans le script.

Ces variables peuvent contenir des littéraux composés soit de valeurs numériques, soit de chaînes de caractères.

Par ailleurs, elles doivent se conformer à certaines règles, comme d'éviter de donner à la variable un nom identique à celui d'un mot clé.

3.1 / Déclaration et affectation

Le mot-clé *var* permet de déclarer une ou plusieurs variables. Après la déclaration de la variable, il est possible de lui affecter une valeur par l'intermédiaire du signe d'égalité (=).

Si une valeur est affectée à une variable sans que cette dernière ne soit déclarée, alors Javascript la déclare automatiquement. Cependant, la lecture d'une variable non déclarée provoque une erreur risquant de perturber votre programme.

D'autre-part, une variable correctement déclarée mais dont aucune valeur n'est affectée, est indéfinie (undefined). Ainsi, il est préférable de systématiquement après la déclaration d'une variable de lui associer une valeur.

```
//Déclaration de i, de j et de k. var i, j, k;
```

```
//Affectation de i. i = 1;
```

```
//Déclaration et affectation de prix. var prix = 0;
```

```
//Déclaration et affectation de caractere var caractere = ["a", "b", "c"];
```

3.2 / La portée

Dans le langage Javascript, les variables peuvent être globales ou locales.

Une variable globale est déclarée en début de script et est accessible à n'importe quel endroit du programme.

Une variable locale est déclarée à l'intérieur d'une fonction et n'est utilisable que dans la fonction elle-même. Dans certain cas, une variable n'a de portée qu'au sein des accolades au sein de laquelle elle a été déclarée.

Une variable globale peut être appelée au sein d'une fonction par l'intermédiaire du mot-clé *this*.

```
//Variables globales.
```

```
var i = 0; j = 64;
```

```
function() {
```

```
    //Variables locales.
```

```
    var i = 1; j = 128;
```

```
    document.write(Valeurs de i et j : " + i + " " + j);
```

```
}
```

```
//La variable k est locale et //n'est utilisable que dans la boucle for. for(k = 0; k < 10; k++){
```

```
    document.write("valeur de k : " + k);
```

```
}
```

```
//Variable globale. var x = 72;
```

```
//paramètre de la fonction.
```

```
function affiche(x){
```

```
    //Appel de la variable globale à l'aide du mot-clé this.
```

```
document.write("Valeurs : " + x + " " + this.x);  
}  
affiche(12);
```

3.3 / Affectation du type

Une particularité du langage Javascript est d'avoir des variable sans type (untyped), c'est-à-dire que le type comme les nombres entiers ou à virgule flottante, est automatiquement affecté à la variable.

```
//i est du type entier. var i = i + 1;  
//taux est du type à virgule flottante var taux = 0.66;  
//texte est du type string  
var texte = "Une chaîne de caractère quelconque";
```

3.4 / Les littéraux

Les littéraux sont employés pour représenter des valeurs dans JavaScript.

Ceux-ci permettent de fixer des valeurs et non-pas des variables, que vous fournissez littéralement dans vos scripts.

"une chaîne de caractères" true

2.789

'une autre chaîne de caractères'

0x26

3.5 / Les tableaux de littéraux

Un littéral de tableau est une liste de zéro ou plusieurs expressions, dont chacune représente un élément du tableau, entouré par des parenthèses carrées ([]).

Lors de la création d'un tableau employant un littéral de tableau, il est initialisé avec les valeurs indiquées comme ses éléments et sa longueur est calculé selon le nombre d'arguments indiqués.

```
semaine      ["lundi","mardi","mercredi","jeudi",      "vendredi","samedi","dimanche"]      nom  
["Marc","Ludovic","Angélique"]
```

Dans le premier exemple, le littéral de tableau comporte sept éléments et une longueur de sept.

Dans le second, le littéral comporte trois éléments ainsi que trois vides. La longueur est ici de six avec :

```
nom[0] = 'Marc'; nom[2] = 'Ludovic'; nom[5] = 'Angélique'
```

3.6 / Les littéraux booléens

Le type booléen a deux valeurs littérales : true (vrai) et false (faux).

Ne confondez pas les valeurs primitives booléennes *true* et *false* avec les valeurs de l'objet *Boolean()*.

L'objet *Boolean()* est un emballage autour du type de données primitif Booléen.

```
var ouverte = true; var fermee = false; porte [ouverte, fermee]
```

```
var on = true; var off = false; lumiere [on, off]
```

3.7 / Les littéraux entiers

Les entiers peuvent être exprimés en décimal (base 10), hexadécimal (base 16) et octal (base 8). Un littéral entier décimal consiste en une séquence de chiffres situé entre 0 et 9.

Des entiers hexadécimaux peuvent inclure des chiffres de 0 à 9 et les lettres a-f et A-F. Des entiers octaux peuvent inclure seulement les chiffres 0-7.

... ..

3.11 / Les littéraux

Unicode est une norme universelle codant les caractères pour l'échange et l'affichage des principales langues écrites.

Il couvre les langues d'Amérique, d'Europe, du Moyen-Orient, d'Afrique, de l'Inde, de l'Asie et du Pacifique, aussi bien que des scripts historiques et des symboles techniques. Unicode tient compte de la conversion, le traitement et l'affichage de textes multilingues, aussi bien que l'utilisation de symboles communs techniques et mathématiques. Il espère résoudre les problèmes d'internationalisation de calcul multilingue, comme des différents standards de caractères nationaux.

Cependant, tous les scripts modernes ou archaïques ne sont pas actuellement soutenus.

Le jeu de caractère Unicode peut être employé pour tout les codages connus. Unicode a été créé après le jeu de caractère ASCII (American Standard Code for Information Interchange). Il utilise une valeur numérique et un nom pour chaque caractère.

Le codage de caractères spécifie l'identité du caractère et sa valeur numérique (la position du code), aussi bien que la représentation de cette valeur en bit. La valeur numérique sur 16 bits (la valeur du code) est définie par un nombre hexadécimal et un préfixe U, par exemple, U+0041 représente A. Le nom unique pour cette valeur est la MAJUSCULE LATINE A.

Les fonctionnalités Unicode ne sont reconnus dans Javascript qu'à partir de la version 1.3.

Unicode est compatible avec les caractères ASCII et est supporté par beaucoup de programmes. Les 128 premiers caractères Unicode correspondent aux caractères ASCII et ont la même valeur d'octet. Les caractères Unicode de U+0020 jusqu'à U+007E sont équivalents des caractères ASCII de 0x20 jusqu'à 0x7E. À la différence de l'ASCII, qui supporte l'alphabet latin et utilise le jeu de caractère à 7 bits, Unicode emploie une valeur de 16 bits pour chaque caractère. Il prend en compte des dizaines de milliers de caractères. La version 2.0 Unicode contient 38 885 caractères. Il supporte aussi un mécanisme d'extension, l'UTF (Unicode Transformation Format), nommé UTF-16, qui permet l'encodage de plus d'un million de caractères en employant des couples de caractère 16 bits. UTF tourne le codage aux bits réelles.

3.11.1 / Affichage des caractères

Le langage Javascript utilise des séquences d'échappement d'Unicode différente du Java. Dans JavaScript, la séquence d'échappement n'est jamais interprétée comme un caractère spécial.

Par exemple, une séquence d'échappement fin de ligne à l'intérieur d'une chaîne de caractère ne la termine pas avant qu'elle ne soit interprétée par la fonction.

JavaScript ignore n'importe quelles séquences d'échappement si elles sont employées dans des commentaires. Dans le Java, si une séquence d'échappement est employée dans une seule ligne de commentaire, elle est interprétée comme un caractère Unicode.

Pour un littéral chaîne de caractères, le compilateur de Java interprète les séquences d'échappement

Pour un littéral chaîne de caractères, le compilateur de Java interprète les séquences d'échappement en premier. Par exemple, si un le caractère d'échappement fin de ligne (`\u000A`) est employé dans Java, il termine le littéral chaîne de caractère.

Dans le Java, cela mène à une erreur, parce qu'on ne permet pas de fins de ligne dans des littéraux chaînes de caractères.

Vous devez employer `\n` pour un retour à la ligne dans un littéral chaîne de caractères. En Javascript, la séquence d'échappement fonctionne de la même manière qu'un saut de ligne `\n`.

Vous pouvez utiliser Unicode pour afficher des caractères de langues différentes ou des symboles techniques. Pour que des caractères soient montrés correctement, une plateforme cliente comme le navigateur de Netscape ou de Microsoft doit supporter Unicode. De plus, une police Unicode appropriée doit être disponible chez le client et la plateforme cliente doit supporter la technologie Unicode.

Souvent, les polices Unicode ne montrent pas tous les caractères Unicode. Quelques plateformes, comme celle de Windows 95, fournissent un appui partiel pour Unicode.

Pour recevoir l'entrée du caractère non-ASCII, le client doit l'envoyer comme Unicode. L'utilisation d'une norme a étendu le clavier, le client ne peut pas facilement saisir des caractères supplémentaires supportés par Unicode. Souvent, la seule manière d'entrée des caractères Unicode est en utilisant des séquences d'échappement Unicode. La spécification Unicode, toutefois, n'exige pas l'utilisation de séquences d'échappement. Unicode trace une méthode pour une interprétation des caractères spéciaux Unicode employant un caractère composé. Il indique l'ordre des caractères qui peuvent être employés pour créer un caractère composé, où le caractère de base vient d'abord, suivi par un ou plusieurs marques de non-espacement.

Cependant, les mises en oeuvre communes d'Unicode incluant la mise en oeuvre JavaScript ne soutiennent pas cette option. JavaScript n'essaye pas la représentation de l'Unicode des ordres se combinant. Autrement dit, une entrée d'un `a` et `'` ne produit pas `à`. JavaScript interprète un `'` comme deux caractères distincts 16 bits Unicode. Vous devez employer une séquence d'échappement Unicode ou un littéral Unicode pour le caractère `à`.

Voir les caractères Unicode disponibles.

3.11.2 / Les séquences d'échappement

Vous pouvez employer la séquence d'échappement d'Unicode dans des littéraux chaînes de caractères. L'ordre d'échappement comporte six caractères ASCII : `u` et un nombre hexadécimal à quatre chiffres. Par exemple, `u00A9` représente le symbole de droit d'auteur.

