Quelques autres objets JavaScript

Nous n'avons repris jusqu'à présent que les éléments essentiels du JavaScript. Il existe néanmoins d'autres objets, qu'il est difficile de ne pas présenter mais dont l'étude sera moins détaillée.

1. L'objet Date

L'objet *Date* permet de gérer les informations relatives au temps, soit la date et l'heure.

a. new Date();

L'objet *Date* en JavaScript présente une particularité dans la mesure où il faut d'abord le créer, avant de pouvoir l'utiliser.

En effet, avant de pouvoir appliquer les multiples fonctions dédiées à la gestion du temps, il faut créer une instance de la date et l'heure de l'ordinateur de l'utilisateur.

Cette instance est créée par `new Date()`.

```
variable=new Date();
```

Ainsi `variable` sera une chaîne de caractères au format :

```
Fri Nov 7 05 :12 :30 UTC+0100 2014
```

```
Soit vendredi (Fri pour Friday) novembre (Nov pour november) 7 pour
le 7ème jour du mois 05:12:30 pour 5 heures 12 minutes et 30 secondes.
Cette heure a été notée à l'heure UTC (temps universel coordonné) +
1 heure. 2014 définit l'année.
```

Commentaires :

- La date et l'heure en JavaScript commencent au 1er janvier 1970. Toute référence à une date antérieure donne un résultat aléatoire.
- L'unité pour le calcul interne du temps est le millième de seconde.

b. Méthodes

`getYear()`

Retourne en principe les deux derniers chiffres de l'année dans l'objet Date. Microsoft Internet Explorer retourne cependant les quatre chiffres soit ici 2007. Des problèmes de compatibilité sont donc possibles.

```
objetdate=new Date();
annee=objetdate.getYear();
```

`getFullYear()`

Retourne les quatre chiffres de l'année. La compatibilité est totale.

```
objetdate=new Date();
annee=objetdate.getFullYear();
```

getMonth()

Retourne le mois sous forme d'un entier compris entre 0 et 11 (0 pour janvier, 1 pour février, 2 pour mars, etc.).

```
objetdate=new Date();
mois=objetdate.getMonth();
```

getDate();

Retourne le jour du mois sous forme d'un entier compris entre 1 et 31.

```
objetdate=new Date();
jour=objetdate.getDate();
```

getDay();

Retourne le jour de la semaine sous forme d'un entier compris entre 0 et 6 (0 pour dimanche, 1 pour lundi, 2 pour mardi, etc.).

```
objetdate=new Date();
semaine= objetdate.getDay();
```

getHours();

Retourne l'heure sous forme d'un entier compris entre 0 et 23.

```
objetdate=new Date();
heure=objetdate.getHours();
```

getMinutes();

Retourne les minutes sous forme d'un entier compris entre 0 et 59.

```
objetdate=new Date();
minute=objetdate.getMinutes();
```

getSeconds();

Retourne les secondes sous forme d'un entier compris entre 0 et 59.

```
objetdate=new Date();
seconde=objetdate.getSeconds();
```

Il existe de nombreuses autres méthodes mais elles n'entrent pas dans le cadre de cet ouvrage et de notre étude de JavaScript.

c. Exemple

Afficher la date et l'heure de façon usuelle en français.

Remplacer le numéro du mois retourné avec la méthode `getMonth()` par son nom (soit 4 par avril).

Pour ce faire, nous allons créer un tableau indicé (*Array*) en prenant soin de commencer à l'indice 0 pour le mois de janvier.

```
var tableau_mois = Array();
tableau_mois[0] = "janvier";
tableau_mois[1] = "février";
tableau_mois[2] = "mars";
tableau_mois[3] = "avril";
tableau_mois[4] = "mai";
tableau_mois[5] = "juin";
tableau_mois[6] = "juillet";
tableau_mois[7] = "août";
tableau_mois[8] = "septembre";
tableau_mois[9] = "octobre";
tableau_mois[10] = "novembre";
tableau_mois[11] = "décembre";
```

Pour afficher le jour de la semaine en toutes lettres, nous procédons de la même façon par un tableau.

```
var tableau_jours = Array();
tableau_jours[0] = "dimanche";
tableau_jours[1] = "lundi";
tableau_jours[2] = "mardi";
tableau_jours[3] = "mercredi";
tableau_jours[4] = "jeudi";
tableau_jours[5] = "vendredi";
tableau_jours[6] = "samedi";
```

On souhaite harmoniser l'affichage des heures, minutes et secondes avec deux chiffres. Ainsi au lieu d'afficher 7 h, ce sera 07 h.

Cette manipulation peut se réaliser avec un test conditionnel. Si le chiffre est inférieur à 10, on ajoutera un 0. Sinon, on garde le chiffre tel quel.

Ce test peut s'écrire :

```
if (min<10) {
min="0" + min;
}
else {
min="" + min;
}
```

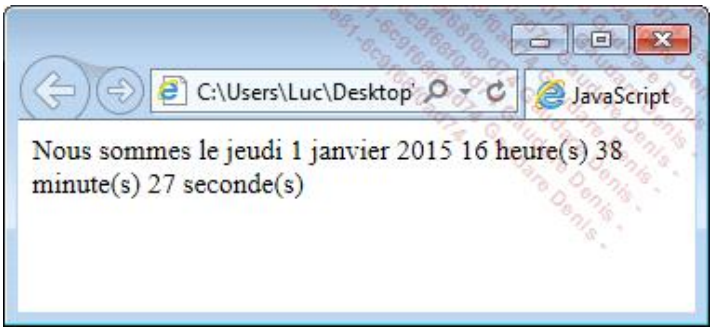
Ce qui peut se noter de façon abrégée :

```
min = ((min<10) ? "0" : "") + min;
```

Le script devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
var objetdate= new Date();
var mois=objetdate.getMonth();
var tableau_mois = Array();
tableau_mois[0] = "janvier";
tableau_mois[1] = "février";
tableau_mois[2] = "mars";
tableau_mois[3] = "avril";
tableau_mois[4] = "mai";
tableau_mois[5] = "juin";
tableau_mois[6] = "juillet";
tableau_mois[7] = "août";
tableau_mois[8] = "septembre";
tableau_mois[9] = "octobre";
tableau_mois[10] = "novembre";
tableau_mois[11] = "décembre";
var nom=tableau_mois[mois];
var jour=objetdate.getDay();
var tableau_jours = Array();
tableau_jours[0] = "dimanche";
tableau_jours[1] = "lundi";
tableau_jours[2] = "mardi";
tableau_jours[3] = "mercredi";
tableau_jours[4] = "jeudi";
tableau_jours[5] = "vendredi";
tableau_jours[6] = "samedi";
var nomjour=tableau_jours[jour];
document.write("Nous sommes le ");
document.write(nomjour + " ");
document.write(objetdate.getDate());
document.write(" ");
document.write(nom);
document.write(" ");
document.write(objetdate.getFullYear());
document.write(" ");
heure= objetdate.getHours();
heure = ((heure<10) ? "0" : "") + heure;
document.write(heure + " heure(s) ");
min=objetdate.getMinutes();
min = ((min<10) ? "0" : "") + min;
document.write(min + " minute(s) ");
sec=objetdate.getSeconds();
sec = ((sec<10) ? "0" : "") + sec;
document.write(sec + " seconde(s) ");
</script>
</head>
<body>
```

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



2. L'objet Math

AJAX se focalise surtout sur la manipulation des chaînes de caractères mais il n'est pas exclu qu'il faille effectuer des opérations sur des chiffres. L'objet `Math` met à la disposition du programmeur JavaScript une panoplie de méthodes permettant d'effectuer des calculs complexes.

La syntaxe est :

```
x = Math.méthode(paramètre);
```

Les méthodes les plus courantes de l'objet `Math` sont :

abs(y)

La méthode `abs(y)` renvoie la valeur absolue (valeur positive) de y. Elle supprime, en quelque sorte, le signe négatif d'un nombre.

```
Y = -4;
```

`x = Math.abs(y);` a comme résultat 4.

ceil(y)

La méthode `ceil(y)` renvoie l'entier supérieur ou égal à y.

Attention ! Cette fonction n'arrondit pas le nombre. Comme l'illustre l'exemple suivant, si y = 1.01, la valeur de x vaut 2.

```
y=1.01;
```

`x=Math.ceil(y);` a comme résultat 2.

floor(y)

La méthode `floor(y)` renvoie l'entier inférieur ou égal à y.

Attention ! Cette fonction n'arrondit pas le nombre. Si y = 1.99, la valeur de x vaut 1.

```
y=1.99;
```

`x=Math.floor(y);` a comme résultat 1.

round(y)

La méthode `round(y)` arrondit le nombre à l'entier le plus proche.

```
Y=20.355;
```

`x=Math.round(y);` a comme résultat 20.

Attention ! Certains calculs réclament une plus grande précision. Ainsi, pour avoir deux décimales après la virgule, la formule suivante est utilisée :`x=Math.round(y*100))/100;` et dans ce cas, x vaut 20.36.

max(y,z)

La méthode `max(y,z)` renvoie le plus grand des deux nombres y et z.

```
y=20; z=10;
```

`x=Math.max(y,z);` a comme résultat 20.

min(y,z)

La méthode `min(y,z)` renvoie le plus petit des 2 nombres y et z.

```
y=20; z=10;
```

`x=Math.min(y,z); a comme résultat x=10.`

pow(y,z)

La méthode `pow(y,z)` calcule la valeur d'un nombre y à la puissance z.

```
y=2; z=8
```

`x=Math.pow(y,z);` a comme résultat 2^8 soit 256.

random()

La méthode `random()` renvoie la valeur d'un nombre aléatoire choisi entre 0 et 1.

Ce nombre aléatoire pourrait être 0.042105102268759464.

sqrt(y)

La méthode `sqrt(y)` renvoie la racine carrée de y.

```
y=25;
```

`x=Math.sqrt(y);` a comme résultat 5.

parseFloat(var)

Cette fonction convertit une chaîne en un nombre à virgule flottante. Particulièrement utile pour transformer le contenu d'une zone de texte (qui est considéré comme une chaîne de caractères) en un

nombre en vue d'un traitement de calcul.

```
str='-.12345';
```

`x=parseFloat(str);` aura comme résultat le nombre -.12345.

Attention ! Le résultat risque d'être surprenant si JavaScript rencontre autre chose dans la chaîne que des chiffres, les signes + et -, le point décimal ou un exposant. S'il trouve un caractère "étranger", la fonction ne prend en compte que les caractères avant le caractère "étranger".

Si le premier caractère n'est pas un caractère admis, x est égal à 0 ou à NaN (*Not a Number*).

```
str='€ 5.50';
```

`x=parseFloat(str);` a comme résultat NaN.

parseInt(var)

Retourne la partie entière d'un nombre avec une virgule.

```
str='1.2345';
```

`x=parseInt(str);` a comme résultat 1.

eval(var)

Cette fonction évalue une chaîne de caractères sous forme de valeur numérique. Dans la chaîne peuvent être stockées des opérations numériques, des opérations de comparaison, des instructions et même des fonctions.

```
str='5 + 10';
```

`x=eval(str);` a comme résultat 15.

Exemple

Élaborons un convertisseur FF vers euro.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
var montant_francs;
var montant_euro;
function convertir() {
montant_francs = document.getElementById("entree").value;
montant_francs=montant_francs.replace(/,/,".");
montant=parseFloat(montant_francs);
montant_euro=montant/6.55957;
montant_euro=(Math.round(montant_euro*100))/100;
document. getElementById("sortie").value=montant_euro;
if (isNaN(montant_francs))
{
```

```
alert("Entrez un nombre");
document.form.entree.value = "";
document.form.sortie.value = "";
}
}
</script>
</head>
<body>
<form>
<input type="text" size="10" id="entree" name="entree"> FF
<input type="button" value="Convertir" onclick="convertir()">
<input type="text" size="10" id="sortie" name="sortie"> €
</form>
</body>
</html>
```

Pour commencer, les variables `montant_francs` et `montant_euro` sont définies.

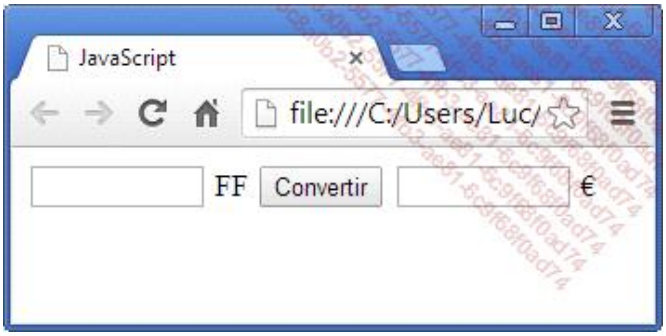
Dans `montant_francs`, la valeur de la zone de texte d'entrée est récupérée, soit la valeur de `document.form.entree.value`.

Le visiteur risque de saisir le montant avec une virgule. La fonction `replace()` est utilisée pour convertir l'éventuelle virgule en point (voir le chapitre La manipulation des chaînes de caractères - replace()).

Pour transformer en nombre la chaîne de caractères contenue dans la variable `formule`, la méthode `parseFloat()`, présentée ci-avant, est utilisée.

Ce nombre est divisé par 6.55957, puis est arrondi (`Math.round()`).

Enfin, si l'utilisateur n'a pas encodé un nombre (NaN), une boîte d'alerte apparaît et les zones de texte sont réinitialisées.



3. L'objet navigator

L'objet `navigator` est particulièrement utile pour adapter les modalités d'exécution des scripts aux différences d'implémentation de certaines instructions JavaScript liées au navigateur et à la version utilisés.

a. Propriétés

L'objet `navigator` de JavaScript fournit un ensemble d'informations sur l'environnement de travail du visiteur et tout particulièrement sur le navigateur qu'il utilise.

Passons en revue quelques-unes de ses propriétés et méthodes.

Propriété	Description
appCodeName	Retourne le "surnom" (*CodeName*) du navigateur. Historiquement, cette propriété a été inventée par Netscape : en effet le navigateur Netscape portait (déjà) le surnom de "Mozilla".
appName	Retourne le nom du navigateur.
appVersion	Retourne la version du navigateur.
language	Spécifie la langue utilisée par le navigateur. Cette propriété n'existe pas sous Internet Explorer.
userLanguage	Spécifie la langue utilisée par le navigateur. Cette propriété n'est valable que sous Internet Explorer.
platform	Retourne le système d'exploitation du navigateur de l'internaute.
userAgent	Spécifie des informations relatives au navigateur (*user agent* signifie navigateur).

Appliquées à la machine de l'auteur, ces propriétés fournissent les informations suivantes :

navigator.appCodeName

Internet Explorer 11	Mozilla
Firefox 29	Mozilla
Google Chrome 35	Mozilla

Comme Internet Explorer, Netscape, Mozilla, Opera ou Firefox renvoient la même valeur, cette propriété n'est pas très utile dans la pratique.

navigator.appName

Internet Explorer 11	Netscape
Firefox 29	Netscape
Google Chrome 35	Netscape

navigator.appVersion

Internet Explorer 11	5.0 (Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; MAAU; .NET4.0C; .NET4.0E; rv:11.0) like Gecko
Firefox 29	5.0 (Windows; fr-FR)
Google Chrome 35	5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36

navigator.language

Internet Explorer 11	fr-FR
Firefox 29	fr
Google Chrome 35	fr

navigator.userLanguage

Internet Explorer 11	fr-FR
Firefox 29	undefined
Google Chrome 35	undefined

navigator.platform

Internet Explorer 11	Win32
Firefox 29	Win32
Google Chrome 35	Win32

navigator.userAgent

Internet Explorer 11	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; MAAU; .NET4.0C; .NET4.0E; rv:11.0) like Gecko
Firefox 29	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:29.0) Gecko/20100101 Firefox/29.0
Google Chrome 35	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36

Ces informations peuvent être utilisées pour identifier le navigateur. En effet, Firefox contient le mot Firefox et Google Chrome le mot Chrome. Pour Internet Explorer 11, on se basera sur le mot Trident. Ces indications seront utilisées dans l'exemple suivant.

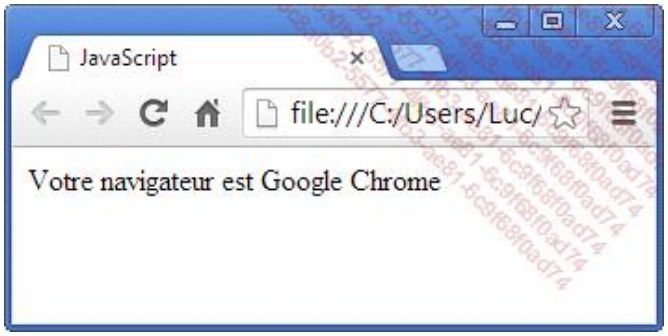
b. Distinguer Firefox, Chrome et Internet Explorer

Utilisons la propriété `userAgent` pour identifier le navigateur.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
if (navigator.userAgent.indexOf("Firefox") != -1) {
navigateur = "Firefox";
}
else {
if (navigator.userAgent.indexOf("Chrome") != -1) {
navigateur = "Google Chrome";
}
else {
if (navigator.userAgent.indexOf("Trident") != -1) {
navigateur = "Internet Explorer";
}
else {
navigateur ="autre";
```

```
}
}
}
document.write("Votre navigateur est " + navigateur);
</script>
</head>
<body>
</body>
</html>
```

Voici le résultat sous Google Chrome :



c. Identifier les versions d'Internet Explorer

Soit les informations transmises par `navigator.userAgent` pour Internet Explorer.

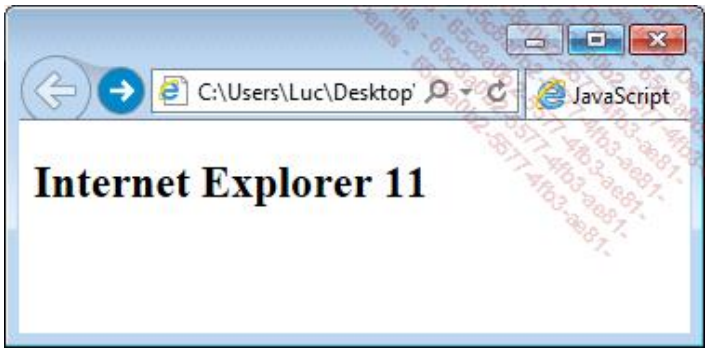
IE 6	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)
IE 7	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
IE 8	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
IE 9	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0)
IE 10	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
IE 11	Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko

Recherchons la présence des chaînes partielles MSIE X pour les versions antérieures à Internet Explorer 11. Pour cette dernière, nous l'identifierons par `rv:11`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
if (navigator.userAgent.indexOf("MSIE 6") != -1) {
document.write("<h2> Internet Explorer 6</h2>");
}
if (navigator.userAgent.indexOf("MSIE 7") != -1) {
document.write("<h2> Internet Explorer 7</h2>");
}
```

```
if (navigator.userAgent.indexOf("MSIE 8") != -1) {
document.write("<h2> Internet Explorer 8</h2>");
}
if (navigator.userAgent.indexOf("MSIE 9") != -1) {
document.write("<h2> Internet Explorer 9</h2>");
}
if (navigator.userAgent.indexOf("MSIE 10") != -1) {
document.write("<h2> Internet Explorer 10</h2>");
}
if (navigator.userAgent.indexOf("rv:11") != -1) {
document.write("<h2> Internet Explorer 11</h2>");
}
</script>
</head>
<body>
</body>
</html>
```

Pour Internet Explorer 11 :



Pour Internet Explorer 6 :

4. L'objet window

Certaines méthodes de l'objet `window` ne vous sont pas inconnues. Ainsi les méthodes `alert()`, `confirm()`, `prompt()` et `setTimeout` ont déjà été abordées au chapitre consacré aux fonctions et méthodes JavaScript.

a. La fenêtre pop-up

Les méthodes `open()` et `close()` de l'objet `window` permettent d'ouvrir (*open*) ou de fermer (*close*) une nouvelle fenêtre du navigateur. Il sera en outre possible de définir de nombreuses caractéristiques de cette nouvelle fenêtre.

L'usage des nouvelles fenêtres, appelées aussi fenêtres pop-up, est souvent utile pour afficher des informations complémentaires sans surcharger la page (ou la fenêtre) de départ. Elles comportent cependant deux désagréments notoires. En effet, si le concepteur n'a pas prévu un moyen de fermer celles-ci, un nombre important de fenêtres pop-up peut apparaître. En outre, il faut admettre que la toile est "polluée" par ces fenêtres pop-up, le plus souvent utilisées à des fins publicitaires. Nous vous conseillons de bien fermer les fenêtres pop-up et de les utiliser avec modération et discernement.

La syntaxe de `open()` est :

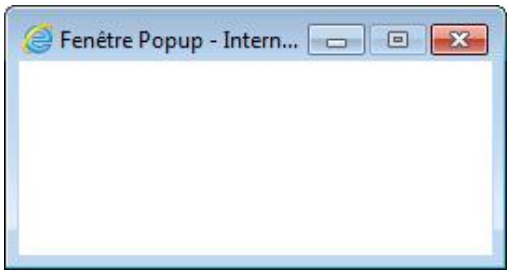
```
window.open("URL","nom_fenêtre","caractéristiques_fenêtre");
```

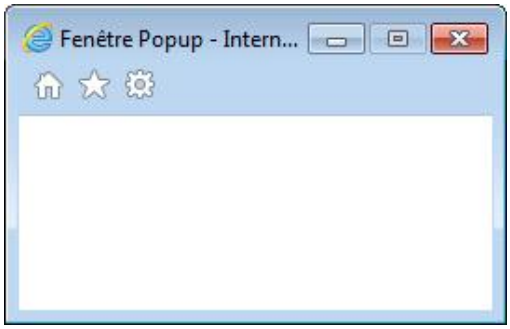
Commentaires :

- La mention de `window` est facultative car `window` étant l'objet avec la plus haute priorité, celui-ci est repris par défaut par le JavaScript.
- `URL` est l'adresse de la page que l'on désire afficher dans la nouvelle fenêtre. Si on ne désire pas afficher un fichier htm existant, on indiquera simplement " ".
- `nom_fenêtre` désigne le nom de la nouvelle fenêtre : cette variable peut être reprise en référence dans le code JavaScript.
- `caractéristiques_fenêtre` est une liste (voir ci-après) de certaines ou de toutes les caractéristiques de la fenêtre. Celles-ci se notent à la suite les unes des autres, séparées par des virgules, sans retour à la ligne. Ces caractéristiques doivent être impérativement écrites sur une seule et même ligne.

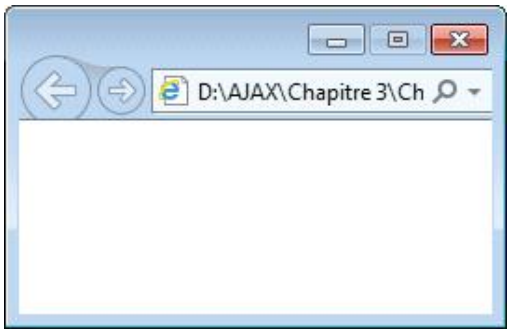
Pour le moteur JavaScript, les caractéristiques de la fenêtre doivent être impérativement écrites sur une seule et même ligne.

Ces caractéristiques sont :

Caractéristiques	Description
`width=x` `height=x`	Largeur de la nouvelle fenêtre en pixels. Hauteur de la nouvelle fenêtre en pixels. La valeur minimale est de 100 pixels.
	
`toolbar=yes ou no`	Affichage de la barre d'outils. La valeur par défaut est yes.

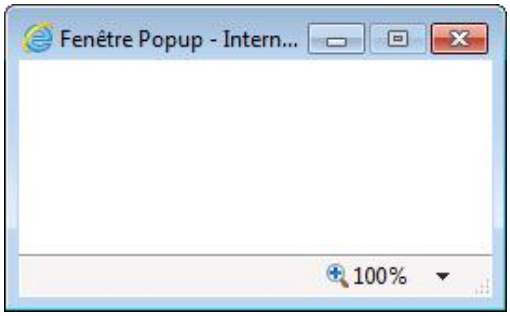


`location=yes ou no` Affichage de la barre d'adresse.



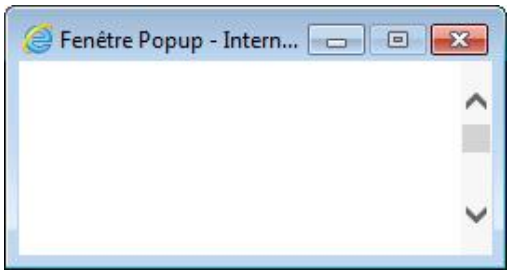
`directories=yes ou no` Affichage des boutons d'accès rapide (barre de liens). Pas d'affichage depuis IE 7.

`status=yes ou no` Affichage de la barre d'état.



`menubar=yes ou no` Affichage de la barre de menus.
Pas d'affichage.

`scrollbars=yes ou no` Affichage des barres de défilement.



Remarque : de nombreux paramètres possibles pour l'ouverture des pop-ups ont été désactivés dans les versions récentes de Firefox et de Google Chrome.

`resizable=yes ou no` Redimensionnement de la fenêtre.

`fullscreen=yes ou no` Affichage de la nouvelle fenêtre en plein écran. Internet Explorer uniquement.

`top=x`	Position en pixels par rapport au bord supérieur de l'écran.
`left=x`	Position en pixels par rapport au bord gauche de l'écran.

La notation 1 ou 0 à la place de yes ou no est également possible.

La syntaxe de `close()` est la suivante :

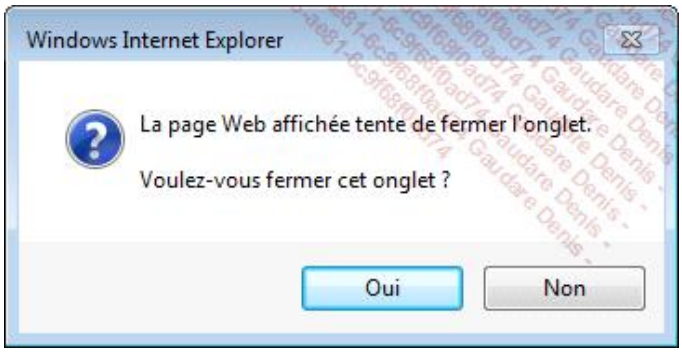
`window.close("nom_de_la_fenêtre")` où `window` est facultatif.

Dans la pratique, pour éviter toute confusion, le raccourci `self` est utilisé pour indiquer la fenêtre en cours.

```
window.self.close();
```

La méthode `close()` entre dans le cadre du concept de sécurité de JavaScript. Cela signifie qu'une fenêtre, à partir du moment où elle possède un historique (parce que l'utilisateur a déjà appelé plusieurs pages), ne se laisse plus fermer sans demande de confirmation de la part du navigateur. Il n'est pas possible d'empêcher cette demande.

La méthode `close()` entraîne sous Internet Explorer l'affichage d'une fenêtre de sécurité pour confirmer la fermeture de la fenêtre. Les versions récentes de Firefox et de Chrome ferment la fenêtre pop-up sans cet écran de confirmation.



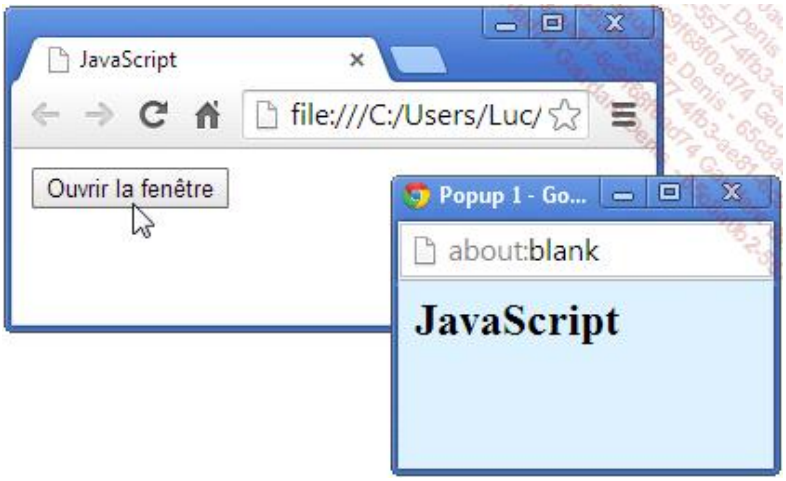
Exemple 1

Plutôt que de charger un fichier htm, il est possible d'écrire en JavaScript dans la nouvelle fenêtre.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function popup() {
nouvelle=open(’’, ’’ , ’width=200, height=100, toolbar=no,
location=no, directories=no, status=no, menubar=no, scrollbars=no,
resizable=no’);
nouvelle.document.write(’<html><head><title>Popup
1</title></head><body
style=\"background:rgb(215,235,255);\"><h2>JavaScript</h2></body><
/html>’);
}
```

```
</script>
</head>
<body>
<form>
<input type ="button" value="Ouvrir la fenêtre" onclick="popup()">
</form>
</body>
</html>
```

Pour rappel, les caractéristiques de la nouvelle fenêtre et le code de l'instruction `write` doivent être impérativement écrits sur une seule et même ligne dans le code.



Exemple 2

Fermer une nouvelle fenêtre avec un bouton.

La méthode `close()` requiert le nom de la fenêtre comme argument. Pour éviter toute confusion, on utilise `self` pour indiquer la fenêtre en cours.

Voici le fichier qui lance la fenêtre pop-up :

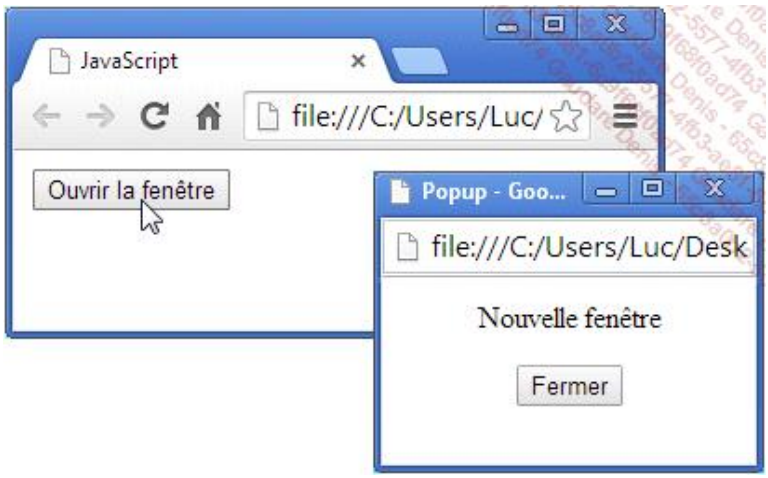
```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function popup() {
open('popup.htm','nouvelle', 'width=200, height=100, toolbar=no,
location=no, directories=no, status=no, menubar=no, scrollbars=no,
resizable=no');
}
</script>
</head>
<body>
<form>
<input type ="button" value="Ouvrir la fenêtre" onclick="popup()">
</form>
```

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

Pour rappel, les caractéristiques de la nouvelle fenêtre doivent être impérativement écrites sur une seule et même ligne dans le code.

Le fichier popup.htm de la nouvelle fenêtre :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Popup</title>
<meta charset="UTF-8">
<style>
input { margin-top: 15px;}
body { text-align: center;
       margin: 12px;}
</style>
</head>
<body>
Nouvelle fenêtre
<form>
<input type="button" value= "Fermer" onclick="self.close()">
</form>
</body>
</html>
```



b. La zone utile du navigateur

La zone disponible pour la page dans la fenêtre du navigateur peut être déterminée par les propriétés `innerWidth` et `innerHeight` de l'objet `window`.

`innerHeight`	Hauteur utile de la fenêtre (hors cadre et barre de défilement), en pixels.
`innerWidth`	Largeur utile de la fenêtre (hors cadre et barre de défilement), en pixels.

```
<!DOCTYPE html>
```

```
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
</head>
<body>
<script>
var largeur;
var hauteur;
if (window.innerWidth) {
largeur = window.innerWidth;
}
if (window.innerHeight) {
hauteur = window.innerHeight;
}
document.write("La largeur disponible est de " + largeur +"
pixels.<br>");
document.write("La hauteur disponible est de " + hauteur +"
pixels.");
</script>
</body>
</html>
```

Capture d'écran pour Google Chrome :