La manipulation des chaînes de caractères

L'objet `String` permet de manipuler des chaînes de caractères. JavaScript met à la disposition du programmeur une série de propriétés et de méthodes qui permettent de traiter les caractères de la chaîne.

La manipulation des caractères est très utilisée dans les applications AJAX.

Instruction	Description
`length`	Propriété qui renvoie un entier indiquant la longueur de la chaîne de caractères.
`charAt()`	Méthode qui permet d'accéder à un caractère isolé d'une chaîne.
`indexOf()`	Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée, en commençant au début de la chaîne, soit en position 0.
`lastIndexOf()`	Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée, en commençant à la fin de la chaîne, soit en position length - 1.
`subString(x,y)`	Méthode qui renvoie une chaîne partielle située entre la position x et la position y-1.
`replace(x,y)`	Remplace x par y.
`toLowerCase()`	Transforme toutes les lettres en minuscules.
`toUpperCase()`	Transforme toutes les lettres en majuscules.

1. La propriété length()

La propriété `length` retourne un entier qui indique le nombre d'éléments dans une chaîne de caractères. Si la chaîne est vide (`""`), le nombre vaut zéro.

La syntaxe est simple :

```
X = variable.length;
X = ("chaîne de caractères").length;
```

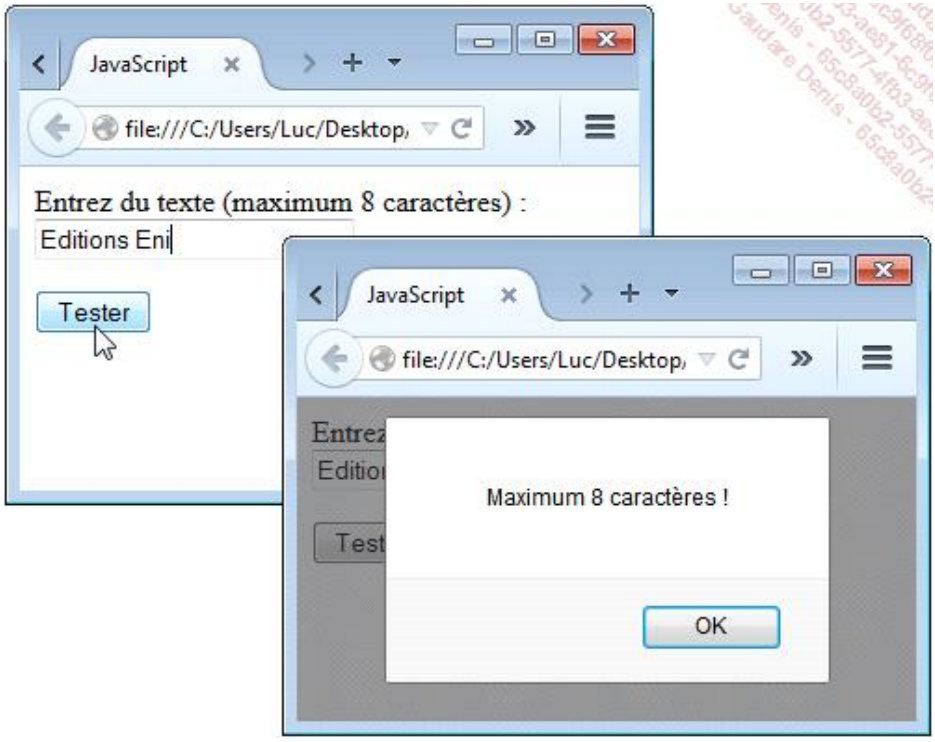
Exemple

Ce script vérifie que les caractères saisis dans une zone de texte ne dépassent pas la limite fixée à huit caractères.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function valider(){
input = document.getElementById("entree").value;
if (input.length>8) {
alert("Maximum 8 caractères !");
}
}
</script>
</head>
<body>
```

```
<form>
Entrez du texte (maximum 8 caractères) :<br>
<input type="text" id="entree" name="entree" size="25" value="">
<p><input type="button" value="Tester" onclick="valider()"></p>
</form>
</body>
</html>
```

La fonction `valider()` reprend d'abord dans la variable `input` la valeur de la zone de texte (`input = document.getElementById("entree").value`). La longueur de la chaîne de caractères est fournie par la propriété `length` associée à la variable `input` (`input.length`). Grâce à un test, on vérifie que cette longueur dépasse les 8 caractères ; si c'est le cas une boîte d'alerte est déclenchée.



Il faut noter que la propriété `length` est valable pour les chaînes de caractères, mais aussi pour connaître la longueur ou le nombre d'éléments :

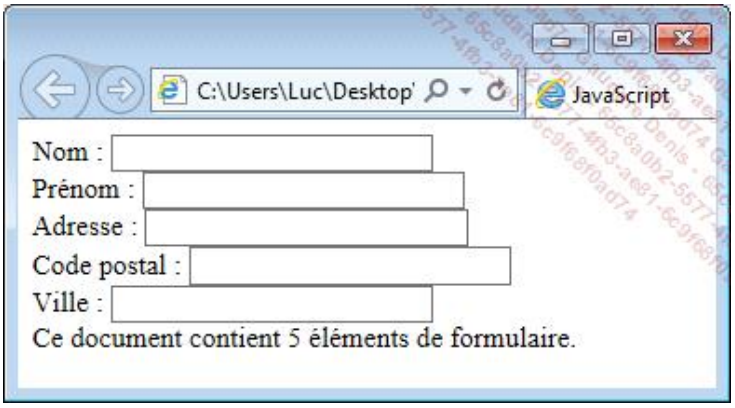
- Formulaires : combien y a-t-il de formulaires différents dans le document ?
- Boutons d'option : combien y a-t-il de boutons radio dans le formulaire ?
- Cases à cocher : combien y a-t-il de cases à cocher dans le formulaire ?
- Options dans un menu déroulant : combien y a-t-il d'options dans une balise `<select>` ?
- Cadres, ancres, liens.
- etc.

Exemple

Ce script indique le nombre d'éléments de formulaire présents dans le document XHTML.

```
<!DOCTYPE html>
```

```
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
</head>
<body>
<form id="form">
Nom : <input type="text" size="25"><br>
Prénom : <input type="text" size="25"><br>
Adresse : <input type="text" size="25"><br>
Code postal : <input type="text" size="25"><br>
Ville : <input type="text" size="25">
</form>
<script>
a="Ce document contient " + document.getElementById("form").length
+ " éléments de formulaire.";
document.write(a);
</script>
</body>
</html>
```



2. La méthode charAt()

La méthode `chartAt()` retourne la lettre ou le signe qui occupe une position déterminée dans une chaîne de caractères. La position souhaitée est fournie en paramètre.

Il faut d'abord noter que les caractères sont comptés de gauche à droite et que la position du premier caractère est 0. La position du dernier caractère est donc la longueur totale (*length*) de la chaîne de caractères moins 1.

```
chaîne :      JavaScript
              ||||||||||
position :    0123456789  (dernier caractère = 9 soit longueur totale -1)
```

Si la position indiquée en paramètre est inférieure à zéro ou plus grande que la longueur moins 1, JavaScript retourne une chaîne vide.

La syntaxe de `charAt()` est :

```
resultat = chaine_départ.charAt(x);
```

où x est un entier compris entre 0 et la longueur de la chaîne à analyser moins 1.

Notez les exemples suivants :

```
var chaine="Javascript";                    La réponse est "J"
resultat=chaine.charAt(0);
resultat="Javascript".charAt(0);
```

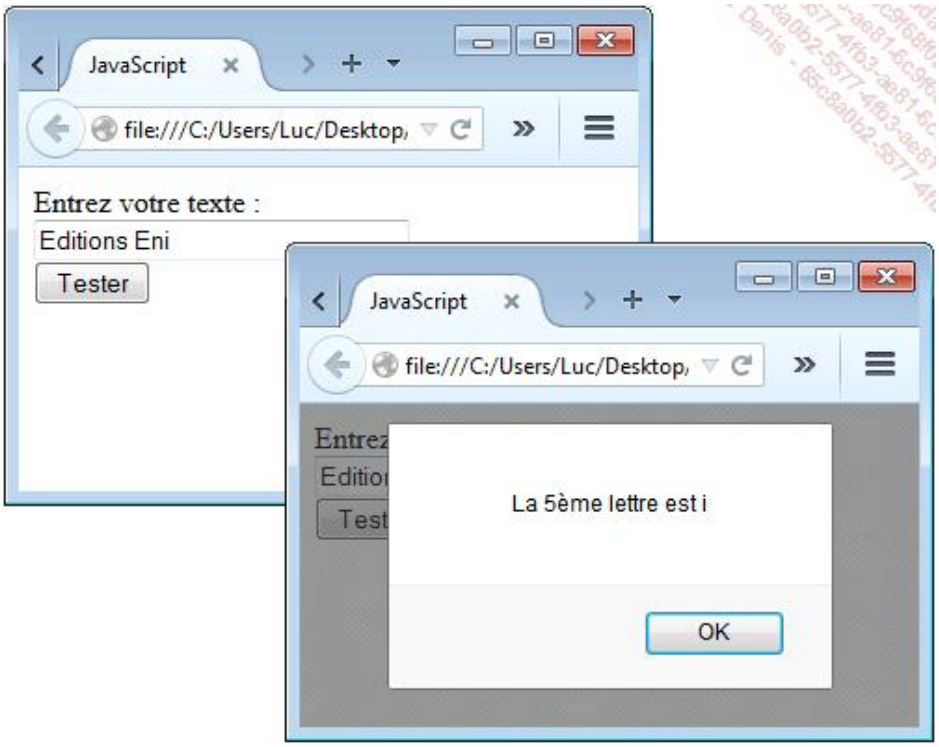
```
var str="Javascript";                       La réponse est "t"
var chr=str.charAt(9);
var chr=charAt(str,9);
```

Exemple

Ce script retourne la lettre en cinquième position.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function valider() {
var x=document.getElementById("texte").value;
var a=x.charAt(4);
alert("La 5ème lettre est " + a);
}
</script>
</head>
<body>
<form>
Entrez votre texte :<br>
<input type="text" id="texte" name="texte" size="30"><br>
<input type="button" value="Tester" onclick="valider()">
</form>
</body>
</html>
```

Le texte saisi dans la zone de texte est sauvegardé dans la variable `x`. La méthode `charAt` est appliquée à cette dernière (`x.charAt(4)`). Notez le paramètre 4 qui correspond en JavaScript à la cinquième position.



3. La méthode indexOf()

Cette méthode `indexOf()` retourne la position d'une chaîne partielle (lettre unique, groupe de lettres ou mot) dans une chaîne de caractères. Cette chaîne partielle est transmise en paramètre.

Il est possible, mais facultatif et peu retenu dans la pratique, de transmettre comme second paramètre la position à partir de laquelle la recherche doit commencer. S'il n'est pas spécifié, la recherche commence à la position 0.

Pour chercher l'arobase dans une chaîne de caractères, la syntaxe est :

```
variable="chaîne_de_caractères";
x=variable.indexOf("@");
```

Commentaires :

- La position retournée est celle de la première occurrence de la chaîne partielle dans la chaîne de caractères.
- Si la chaîne partielle n'est pas trouvée dans la chaîne de caractères à analyser, la valeur retournée est égale à -1.

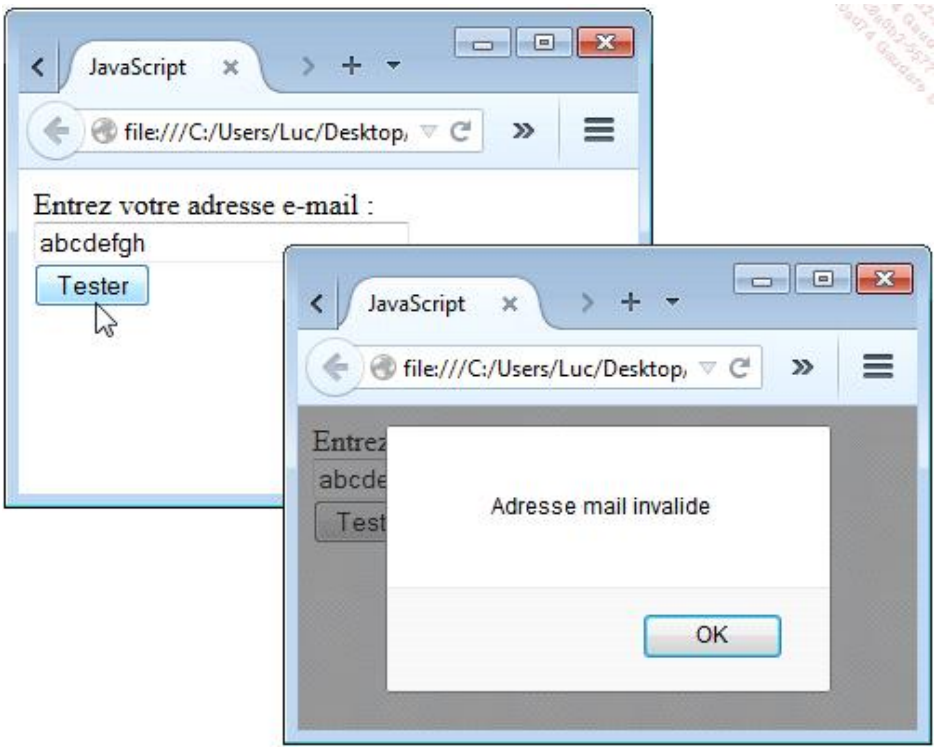
Exemple

Ce script teste si une adresse e-mail contient le signe @. Il notifiera, par un message d'alerte, que le signe @ n'est pas trouvé (soit valeur -1).

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
```

```
function valider() {
var a=document.getElementById("email").value.indexOf("@");
if (a == -1) {
alert("Adresse mail invalide");
}
else {
alert("OK");
}
}
</script>
</head>
<body>
<form>
Entrez votre adresse e-mail :<br>
<input type="text" id="email" name="email" size="30"><br>
<input type="button" value="Tester" onclick="valider()">
</form>
</body>
</html>
```

L'adresse e-mail est fournie par `document.getElementById("email")`. La méthode `indexOf` est appliquée avec le signe @ à rechercher, transmis en paramètre. Si l'arobase a été trouvée, la méthode retourne sa position qui ne peut être qu'un chiffre positif. Si la position retournée vaut -1, cela indique que l'arobase n'a pas été trouvée et que l'adresse e-mail n'est donc pas valide.



Ce test est assez sommaire et peut être affiné. Outre l'arobase, une adresse e-mail contient également au moins un point. Le script suivant va tester la présence du point et ne déclarer l'e-mail valide que si les deux conditions sont remplies.

```
<!DOCTYPE html>
```

```
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function valider() {
var a=document.getElementById("email").value.indexOf("@");
var b= document.getElementById("email").value.indexOf(".");
if (a == -1 || b == -1) {
alert("Adresse mail invalide");
}
else {
alert("OK");
}
}
</script>
</head>
<body>
<form>
Entrez votre adresse e-mail :<br>
<input type="text" id="email" name="email" size="30"><br>
<input type="button" value="Tester" onclick="valider()">
</form>
</body>
</html>
```

Dans la variable a, on obtient la position du signe @ ou son absence (-1). Idem pour la variable b qui fournit la position du point ou son absence (-1). L'absence de l'arobase ou du point détermine si l'adresse e-mail est non valide.

Pour des procédures de validation de formulaires plus sophistiquées, reportez-vous à la section Les expressions régulières de cette section.

4. La méthode lastIndexOf()

Cette méthode ressemble fortement à `indexOf()` sauf que la recherche s'effectue cette fois de droite à gauche (en commençant donc par la fin).

Notons que même lorsque la recherche s'effectue à partir de la fin de la chaîne, la position retournée est numérotée depuis le début de la chaîne avec la numérotation commençant à zéro.

```
x=variable.lastIndexOf(chaîne_partielle);
```

Les exemples suivants montrent la différence entre `indexOf()` et `lastIndexOf()` :

Soit `variable="Javascript"`. On recherche la position de la lettre a.

`x = variable.indexOf(a)` retourne la position 1.

`x=variable.lastIndexOf(a)` retourne la position 3 (pour le second a).

Cette méthode est peu utilisée dans la pratique.

5. La méthode substring()

La méthode `substring()` sera particulièrement utile pour extraire des données d'une chaîne de caractères.

La syntaxe est `substring(x,y)` où x désigne la position du premier signe à extraire dans la chaîne de caractères et y la position du premier signe ne devant plus être extrait de la chaîne de caractères. Pour rappel, la numérotation commence à 0.

Si x est égal à y, `substring()` retourne (logiquement) une chaîne vide.

Voici quelques exemples :

```
Javascript
||||||||||
0123456789

str="Javascript";
str1=str.substring(0,4);
str2=str.substring(6,9);
str3=str.substring(7,10);
```

Les résultats sont :

```
str1="Java"; soit les positions 0,1, 2 et 3.
str2="rip"; soit les positions 6, 7 et 8.
str3="ipt" soit les positions 7, 8 et 9.
```

Exemple

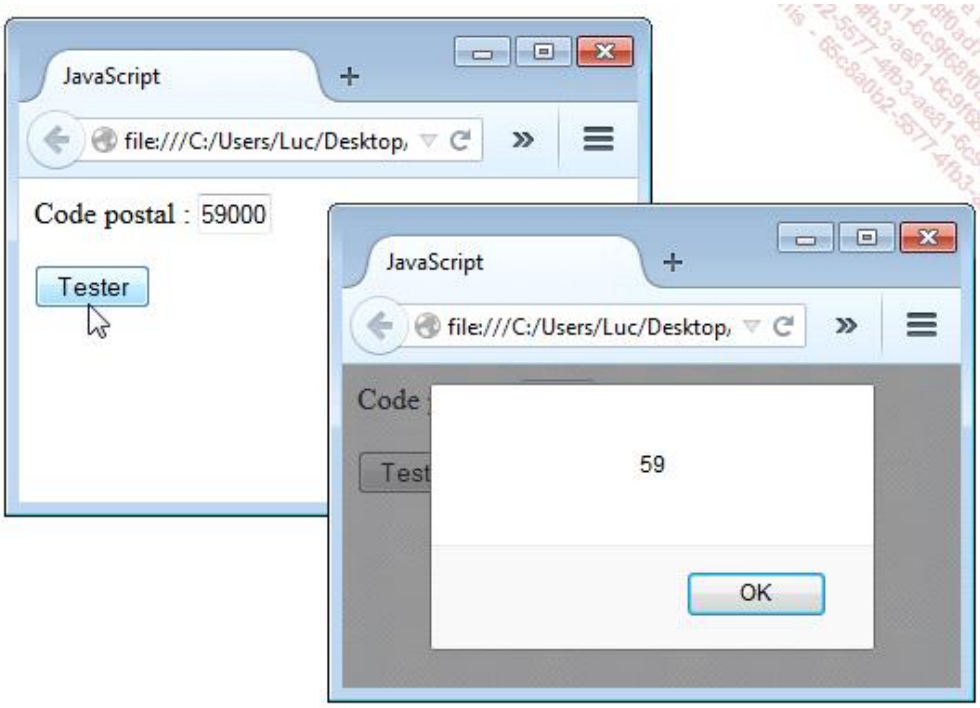
Retrouver les deux premiers chiffres d'un code postal.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function extraire() {
dep = document.getElementById("cp").value;
dep = dep.substring(0,2);
alert(dep);
document.getElementById("cp").value = "";
}
</script>
</head>
<body>
<form>
Code postal :
<input type="text" id="cp" name="cp" size="3" maxlength="5"><br>
<p><input type="button" name="bouton" value="Tester"
onclick="extraire()"></p>
</form>
```

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

La fonction `extraire()` récupère les 5 chiffres du code postal saisi dans la zone de texte (`document.getElementById("cp").value`). Les deux premiers chiffres sont extraits (`dep.substring(0,2)`) et sont affichés dans une boîte d'alerte.

La zone de texte est réinitialisée à 0 par `document.getElementById("cp")=""`.



6. La méthode toLowerCase()

Cette méthode transforme tous les caractères (majuscules ou minuscules) d'une chaîne de caractères en minuscules.

```
variable="chaîne de caractères";
x=variable.toLowerCase();
```

Exemple

```
str="JavaScript";
str1=str.toLowerCase();
```

Le résultat sera :

```
str1="javascript";
```

7. La méthode toUpperCase()

Cette méthode transforme les caractères (majuscules ou minuscules) d'une chaîne de caractères en majuscules.

```
variable="chaîne de caractères";
x=variable.toUpperCase();
```

Exemple

```
str="JavaScript";
str2=str.toUpperCase();
```

Le résultat sera :

```
str2="JAVASCRIPT";
```

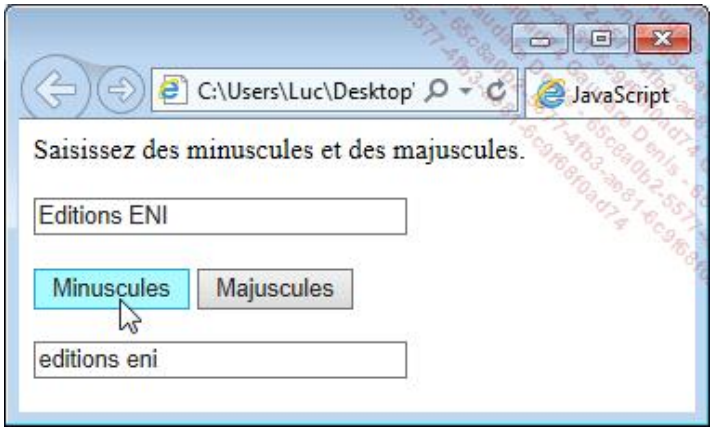
Exemple

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<style>
input { margin-top: 18px;}
</style>
<script>
function minuscules() {
a = document.getElementById("entree").value.toLowerCase();
document.getElementById("sortie").value = a;
}
function majuscules() {
a = document.getElementById("entree").value.toUpperCase();
document.getElementById("sortie").value = a;
}
</script>
</head>
<body>
<form>
Saisissez des minuscules et des majuscules.<br>
<input id="entree" name="entree" size="30"><br>
<input type="button" value="Minuscules" onclick="minuscules()">
<input type="button" value="Majuscules" onclick="majuscules()">
<br>
<input id="sortie" name="sortie" size="30">
</form>
</body>
</html>
```

Pour la fonction `minuscules()`, la variable a repris le contenu de la zone de texte d'entrée (`document.getElementById("entree").value`) auquel la méthode `toLowerCase()` est appliquée. Le résultat est renvoyé dans la zone de texte de sortie (`document.getElementById("sortie").value`).

Pour la fonction `majuscules()`, la variable a repris le contenu de la zone de texte d'entrée (`document.getElementById("entree").value`) auquel la méthode `toUpperCase()` est appliquée. Le

résultat est renvoyé dans la zone de texte de sortie (`document.getElementById("sortie").value`).



8. La méthode replace()

La méthode `replace(x,y)` remplace les caractères x dans la chaîne de caractères par y.

Il est important de noter que le premier paramètre est défini entre deux barres obliques /.

```
str="Je programme en PHP";
str=str.replace(/PHP/,"JavaScript")
```

Le résultat est : "Je programme en JavaScript".

Il faut noter que la méthode `replace(x,y)` ne va remplacer que la première occurrence des caractères correspondant à x dans la chaîne de caractères.

Pour remplacer toutes les occurrences dans la chaîne de caractères, il faut adopter la notation suivante :

```
str=str.replace(/PHP/g,"JavaScript");
```

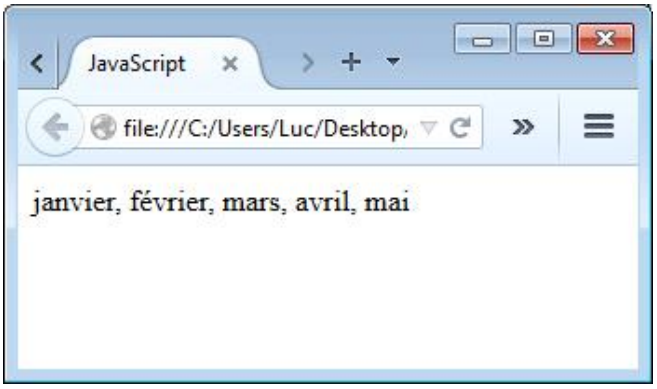
Exemple

JavaScript retourne une chaîne de caractères initiale `"janvier,février, mars,avril,mai"`.

Pour afficher une présentation plus conviviale avec les mêmes termes séparés par des espaces `"janvier, février, mars, avril, mai"`, il suffit de remplacer toutes les chaînes virgules par les chaînes virgule espace.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
chaine="janvier,février,mars,avril,mai"
chaine=chaine.replace(/,/g,", ");
document.write(chaine);
</script>
```

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



9. Les expressions régulières

Les méthodes de manipulation de caractères de l'objet `String` (JavaScript 1.1) décrites plus haut se sont rapidement révélées trop limitées.

On a alors introduit en JavaScript les expressions régulières et l'objet RegExp (JavaScript 1.2), qui comportent des critères de recherche plus sophistiqués et qui, en outre, permettent de modifier de façon dynamique des chaînes de caractères pendant l'exécution du script.

Pour cela, il vous faut définir une instance de l'objet **RegExp**. C'est ce qu'on appellera une **exp**ression **rég**ulière. Les éléments de cette expression régulière sont précisés entre deux barres obliques. Les différents éléments sont séparés par une virgule.

a. Déclaration

La création d'un objet RegExp se crée à l'aide de la syntaxe suivante :

```
expression = /critère/;
```

Il est également possible de créer un tel objet de manière plus classique à l'aide de son constructeur :

```
expression = new RegExp("critère");
```

On peut, de plus, préciser le comportement de l'expression régulière de façon optionnelle par :

- `g` qui indique une recherche globale de toutes les occurrences.
- `i` pour une recherche non sensible à la casse, soit indépendamment de la présence de majuscule ou minuscule dans la chaîne.
- `gi` qui combine les deux.

```
expression = /critère/g;
expression = new RegExp("critère","i");
```

b. Syntaxe

Les expressions régulières, issues du langage de programmation Perl, en reprennent la concision mais aussi la complexité...

Le tableau suivant présente une liste de composants pour former une expression régulière.

Caractère	Description
xyz	Trouve xyz n'importe où dans la chaîne de caractères.
^xyz	Trouve xyz au début de la chaîne de caractères.
xyz$	Trouve xyz à la fin de la chaîne de caractères.
xyz*	Trouve xy suivi de zéro ou plusieurs z.
xyz+	Trouve xy suivi de un ou plusieurs z.
xyz?	Trouve xy suivi de zéro ou un seul z.
.xyz	Trouve xyz précédé d'un caractère joker.
.+xyz	Trouve xyz précédé de plusieurs caractères joker.
\bxyz\b	Trouve xyz en tant que mot distinct.
\Bxyz\B	Trouve uniquement xyz à l'intérieur de mots.
x|y	Trouve x ou y.
[xyz]	Trouve x ou y ou z.
[a-z]	Trouve n'importe quel caractère de a à z (alphabet ASCII, soit sans caractères accentués).
[0-9]	Trouve n'importe quel chiffre de 0 à 9.
x{n}	Trouve x exactement n fois.
x{n,}	Trouve x au moins n fois.
x{n,m}	Trouve x entre n et m fois

Pour rechercher un caractère faisant partie des caractères spéciaux, il suffit de le faire précéder d'une barre oblique inverse.

Caractère spécial	Notation
\	\\
.	\.
$	\$
[	\[
]	\]
(	\(
)	\)
{	\{
}	\}
^	\^
?	\?

*	*
+	\+
-	\-

On note également :

Caractère	Description
\f	Trouve un signe de saut de page.
\n	Trouve un saut de ligne.
\r	Trouve un retour chariot.
\t	Trouve une tabulation horizontale.
\v	Trouve une tabulation verticale.
\s	Trouve toute sorte d'espace vide donc espace, retour chariot, tabulation, saut de ligne, saut de page.
\S	Trouve un caractère quelconque qui ne soit pas un espace vide.
\w	Trouve tous les caractères alphanumériques (minuscules ou majuscules) ainsi que le tiret de soulignement.
\W	Trouve tous les caractères non alphanumériques.
\d	Trouve tous les caractères numériques soit chiffre de 0 à 9.
\D	Trouve tous les caractères non numériques.

Exemple 1

Un nombre (entier ou décimal).

```
var reg = /^\d+[.]?\d*$/;
```

- le signe ^ signale que l'on commence au début de la chaîne de caractères.
- \d+ pour plusieurs caractères numériques.
- [.]? pour zéro ou une fois un point décimal.
- \d* pour zéro ou plusieurs caractères numériques.
- $ pour fin du nombre.

Exemple 2

Login valide de 3 à 8 caractères (numériques ou alphanumériques, majuscules ou minuscules) sans caractères spéciaux.

```
var exp=new RegExp("^[a-zA-Z0-9]{3,8}$");
```

- le signe ^ signale que l'on commence au début de la chaîne de caractères.
- [a-zA-Z0-9] note que les caractères alphanumériques peuvent être en minuscule de a à z, en majuscules de A à Z et des chiffres de 0 à 9.
- {3,8} vérifie si la chaîne comporte entre 3 et 8 caractères.
- $ signale la fin de la chaîne de caractères et interdit tout caractère supplémentaire.

c. Méthodes

Certaines méthodes de l'objet `String` peuvent utiliser les expressions régulières.

Méthode	Description
`match()`	Effectue une recherche de correspondance dans une chaîne. Elle renvoie un tableau de valeurs ou *null* en cas d'échec.
`replace()`	Effectue une recherche de correspondance dans une chaîne et remplace la sous-chaîne capturée par la chaîne de remplacement.
`split()`	Utilise une expression rationnelle ou une chaîne pour diviser une chaîne en un tableau de sous-chaînes.

Par ailleurs l'objet RegExp possède certaines méthodes qui lui sont propres.

Méthode	Description
`exec()`	Effectue une recherche de correspondance dans une chaîne. Elle renvoie un tableau de valeurs.
`test()`	Teste la correspondance d'une chaîne. Elle renvoie la valeur `true` ou `false`.
`search()`	Teste la correspondance d'une chaîne. Elle renvoie l'index de la capture, ou `-1` en cas d'échec.

Exemple avec split()

Soit une liste de noms dont les séparateurs ne sont pas uniformes : "abc ,def ; hij klm - nop". Isolons tous les noms de cette liste.

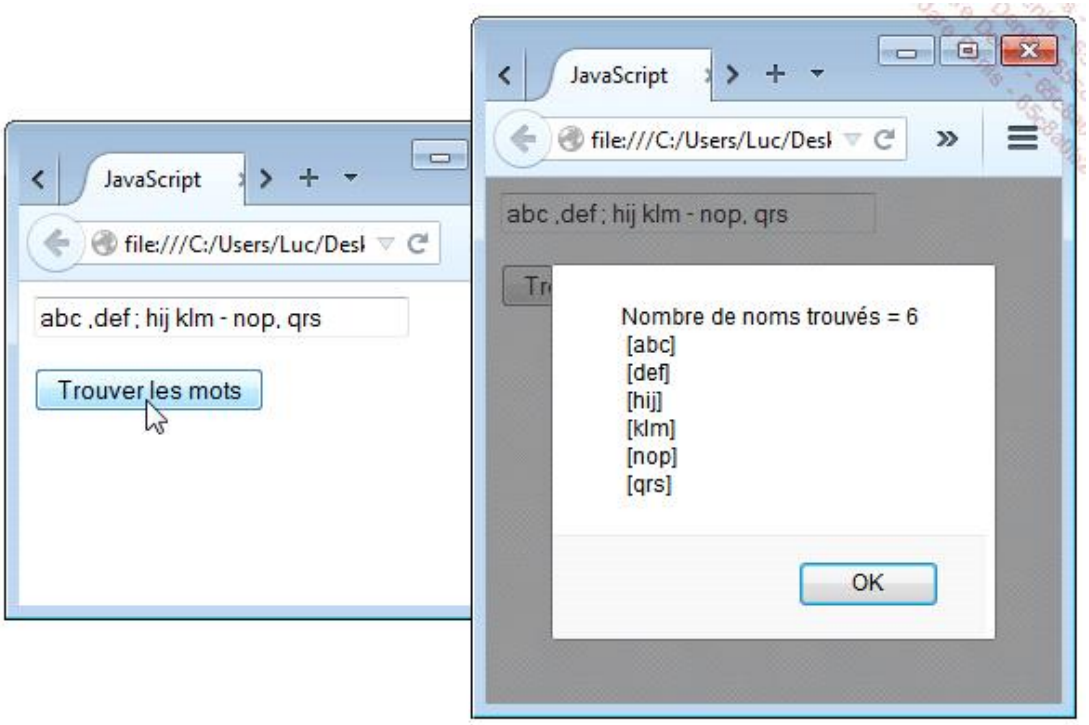
L'expression régulière va prévoir des séparateurs sous forme de virgule, point-virgule, double point, tiret, un ou des espaces. Celle-ci devient :

```
var expression=new RegExp("[,;:- ]+","g");
```

Le code complet devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function extraire(form) {
var test=document.getElementById("exemple").value;
var expression = new RegExp("[, ;-]+","g");
var tableau = test.split(expression);
var affichage = "Nombre de noms trouvés = " + tableau.length +
"\n";
for (var i=0;i<tableau.length;i++){
affichage=affichage + " ["+ tableau[i] + "]\n";
}
alert(affichage);
```

```
}
</script>
</head>
<body>
<form name="form">
<input type="text" id="exemple" name="exemple" size="30"
value="abc ,def ; hij klm - nop">
<p><input type="button" value="Trouver les mots"
onclick="extraire(form)"></p>
</form>
</body>
</html>
```



Exemple avec match()

Continuons sur le même thème en utilisant la méthode `match()` pour isoler les mots d'une phrase.

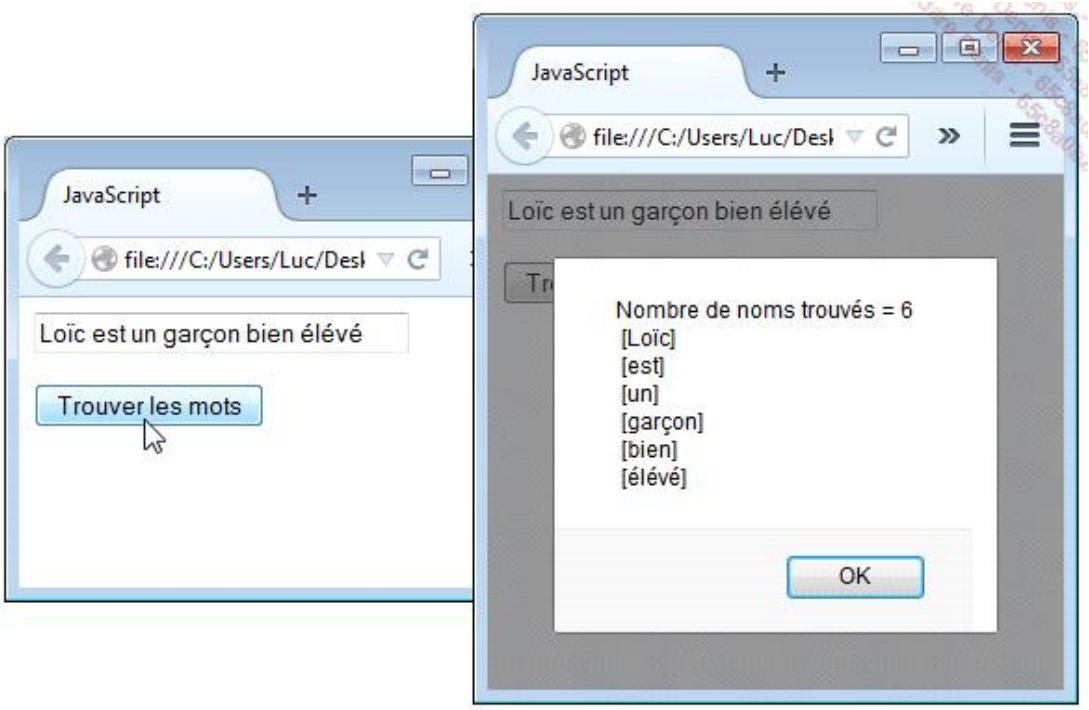
L'expression régulière doit reconnaître tous les caractères alphanumériques, en majuscules ou minuscules, de la chaîne de caractères. Il est prudent de prévoir quelques caractères accentués.

```
var expression=new RegExp("[a-zA-Zéèàçûï\-]+","gi");
```

Le code complet devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
```

```
function extraire(form) {
var test=document.getElementById("exemple").value;
var expression=new RegExp("[a-zA-Zéèàçûï\-]+","gi");
var tableau=test.match(expression);
var affichage="Nombre de noms trouvés = " + tableau.length + "\n";
for (var i=0;i<tableau.length;i++){
affichage=affichage + " ["+ tableau[i] + "]\n";
}
alert(affichage);
}
</script>
</head>
<body>
<form name="form" >
<input type="text" id="exemple" name="exemple" size="30"
value="Loïc est un garçon bien élevé">
<p><input type="button" value="Trouver les mots"
onclick="extraire(form)"></p>
</form>
</body>
</html>
```



Exemple avec test()

Parmi les méthodes de l'objet RegExp, la méthode `expression.test("chaîne")` est assurément la plus utilisée. Elle teste une chaîne de caractères selon les critères de l'expression régulière. Cette méthode retourne `true` si la recherche est fructueuse et `false` dans le cas contraire.

Appliquons-la à la vérification de la validité d'une adresse e-mail.

Une adresse e-mail comporte toujours le signe @ et un point pour le nom du serveur.

```
exp = new RegExp("@.");
```

C’est un peu simpliste. Entre l’arobase et le point, il doit y avoir quelques caractères (au moins 2). De plus, après le point, il doit avoir deux, trois ou quatre caractères pour le nom de domaine.

```
exp = new RegExp("@[a-z]{2,}[.][a-z]{2,4}$");
```

Le signe $ signifie la fin de l’e-mail.

Il y a bien entendu des caractères (ici en minuscules) avant l’arobase. La présence du point, du tiret ou du soulignement est également prévue.

```
exp = new RegExp("^[a-z0-9.-_]+@[a-z]{2,}[.][a-z]{2,4}$");
```

Le signe ^ fait démarrer le test depuis le début de la chaîne.

Certains ont encore des majuscules dans leur adresse.

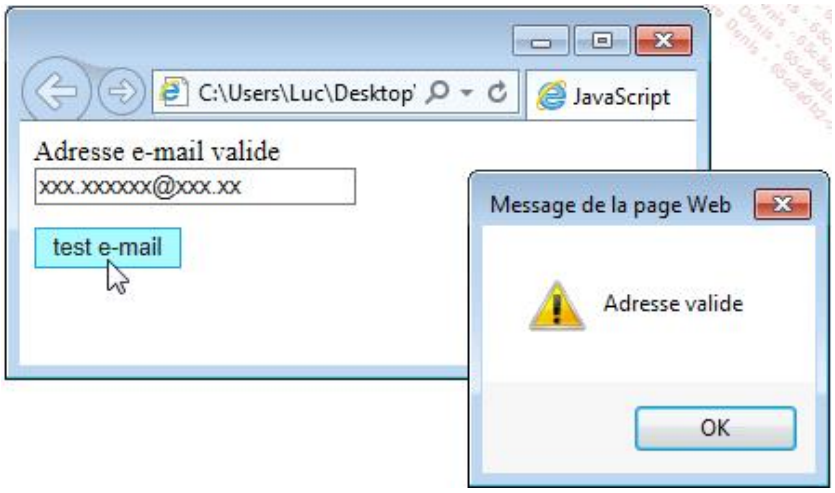
```
exp = new RegExp("^[a-z0-9.-_]+@[a-z]{2,}[.][a-z]{2,4}$","i");
```

L’équivalent de cette expression régulière en JavaScript "classique" aurait réclamé de nombreuses lignes de code.

Le code devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
function trouver() {
var exp = new RegExp("^[a-z0-9.-_]+@[a-z]{2,}[.][a-z]{2,4}$","i");
var email = document.getElementById("mail").value;
if (exp.test(email)) {
alert("Adresse valide");
}
else {
alert("Non valide");
document.getElementById("mail").value="";
}
}
</script>
<style>
#bouton { margin-top: 12px;}
</style>
</head>
<body>
<form name="form">
Adresse e-mail valide <br>
<input type="text" id="mail" name="mail" size="25"><br>
<input type="button" id="bouton" value="test e-mail"
onclick="trouver()">
```

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



La vérification de la validité des adresses de courrier électronique peut également se réaliser avec l'expression régulière suivante, aussi efficace mais a priori assez hermétique.

```
RegExp("^\\w[\\w\-\_\.]*\\w@\\w[\\w\-\_\.]*\\w\\.\\w{2,4}$");
```

^\\w	un caractère au début de l'e-mail.
*[\\w\-_\.]**	ensuite une série de caractères avec éventuellement un tiret, un soulignement et un point.
\\w	un caractère seul devant l'arobase.
@	le signe arobase.
\\w	un caractère seul.
*[\\w\-_\.]**	divers caractères avec éventuellement un tiret, un soulignement et un point.
\\w	un caractère pour finir cette série.
\\.	un point.
\\w{2,4}	de deux à quatre caractères pour le nom de domaine.
$	fin de la chaîne.

Une expression régulière peut même être prévue pour s'assurer qu'il n'y a pas de signes interdits dans l'adresse.

```
var illegal = new RegExp("[\(\),;: àéè\']+","g");
```

On interdit ici les parenthèses ouvrante et fermante, la virgule, le point-virgule, le double point, l'espace, les caractères accentués et l'apostrophe.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>JavaScript</title>
<meta charset="UTF-8">
<script>
```

```
function trouver() {
var illegal = new RegExp("[\(\),;: àéè\’]+","g");
var exp = new RegExp("^\\w[\\w\-\_\.]*\\w@\\w[\\w\-
\_\.]*\\w\\.\\w{2,4}$");
var email = document.getElementById("mail").value;
if (illegal.test(email)==false) {
if (exp.test(email)==true) {
alert("Adresse valide");
}
}
else {
alert("Recommencez");
document.getElementById("mail")="";
}
}
</script>
<style>
#bouton { margin-top: 12px;}
</style>
</head>
<body>
<form name="form">
Adresse email valide <br>
<input type="text" id="mail" name="mail" size="25"><br>
<input type="button" id="bouton" value="test email"
onclick="trouver()">
</form>
</body>
</html>
```