

Secteur Tertiaire Informatique  
Filière étude - développement.

Développer des composants d'interface  
**Créer des formulaires de saisie**

**JAVASCRIPT – JS04-Controles de saisie-Consignes**

**Accueil**

**Apprentissage**

**Période en  
entreprise**

**Evaluation**



**CODE BARRE**

# SOMMAIRE

I	CREATION D'UNE EXPRESSION REGULIERE.....	3
II	ECRITURE D'UNE EXPRESSION REGULIERE .....	3
III	TEST D'UNE CHAINE EN UTILISANT UNE EXPRESSION REGULIERE.....	5
IV	EXERCICE D'APPLICATION .....	5
IV.1	JS04.....	5

## I CREATION D'UNE EXPRESSION REGULIERE

Les expressions régulières permettent de définir en quelque sorte, un modèle, auquel on pourra comparer les champs que l'on veut contrôler. Elles sont utilisées principalement pour effectuer des contrôles de validation (dans notre cas), et pour effectuer des recherches et remplacement des textes ou partie de texte. Associées à JavaScript, elles simplifient considérablement la programmation des contrôles coté client, et peuvent être utilisées notamment pour valider les formats des données saisies (N° téléphone, N° carte bleue, e-mails, etc....).

Les expressions régulières sont créées de la manière suivante (manière la plus simple, sachant qu'il existe d'autres méthodes et d'autres paramètres) :

```
var nomExpression = /chaîne/;
```

Le nom de l'expression régulière créée sera le nom utilisé lors des contrôles de validité. La chaîne va représenter le format de l'expression que l'on veut obtenir. Cette chaîne est composée d'un ensemble de caractères (représentant les valeurs voulues, ou plages de valeurs), et d'autres caractères représentant des opérateurs entre valeur, des positionnements dans la chaîne, des nombres d'occurrence, des caractères spéciaux, etc....

## II ECRITURE D'UNE EXPRESSION REGULIERE

Les caractères et autres symboles spéciaux représentent des caractères non-imprimables comme des sauts de lignes (\n), des tabulations (\t) ou des options particulières ou encore des plages de lettres ou de chiffres, etc.

		Expression	Résultat
<b>^</b>	Début de chaîne	/^ab/	Recherche si la chaîne commence par ab
<b>\$</b>	Fin de chaîne	/ab\$/	Recherche si la chaîne finit par ab
<b>.</b>	Un caractère quelconque	/.../	Recherche si la chaîne a au moins 3 caractères
<b>\r</b>	désigne une nouvelle ligne (caractère «CR»)		
<b>\n</b>	désigne un retour en début de ligne (caractère «LF»)		
<b>\t</b>	désigne une tabulation horizontale		
<b>\v</b>	désigne une tabulation verticale		
<b>\d</b>	caractère numérique		Teste si le caractère est numérique
<b>\D</b>	caractère non numérique		Teste si le caractère est non numérique

<code>\s</code>	caractère blanc (espace, tabulation,...)		Teste si le caractère est blanc
<code>\S</code>	caractère non blanc		Teste si le caractère n'est pas numérique
<code>\w</code>	caractère alphanumérique (lettre ou chiffre)		Teste si le caractère est alphanumérique
<code>\W</code>	caractère non alphanumérique		Teste si le caractère n'est pas alphanumérique
<code>[abc]</code>	caractère a, b ou c		Teste si la chaîne ne contient que les caractères a, b ou c
<code>[^abc]</code>	caractère autre que a, b ou c		Teste si la chaîne ne contient que des caractères autres que a, b ou c
<code>[a-h]</code>	tout caractère entre a et h inclus		Teste si la chaîne est composée de caractères entre a et h
<code>[^a-h]</code>	tout caractère non entre a et h inclus		Teste si la chaîne ne contient pas de caractères entre a et h
<code>*</code>	Le caractère précédent peut exister de zéro à plusieurs fois	<code>/is*/</code>	mie, <b>mis</b> , <b>mission</b>
<code>+</code>	Le caractère précédent doit être trouvé de une à plusieurs fois	<code>/is+/</code>	<b>Maison</b> , <b>mission</b>
<code>?</code>	Le caractère précédent est optionnel	<code>/is?/</code>	mie, <b>mis</b> , <b>mission</b>
<code>{n}</code>	Le caractère précédent doit être trouvé un nombre <i>n</i> fois	<code>/o{2}/</code>	<b>zoo</b>
<code>{n,m}</code>	Le caractère précédent doit être trouvé au moins <i>n</i> fois et au plus <i>m</i> fois	<code>/1{2,4}/</code>	<b>211</b> , <b>131111</b>
<code>{n, }</code>	Le caractère précédent doit être trouvé au moins <i>n</i> fois	<code>/1{2,}/g</code>	<b>211</b> , <b>131111</b> , <b>111111</b>
<code>ch1 ch2</code>	présence soit de ch1 soit de ch2	<code>/mois jour semaine/</code>	..la durée de 7 <b>jours</b> soit une <b>semaine</b>

**Attention :** pour tester un caractère qui est un méta caractère du langage (par exemple `^` ou `.`), il faut le faire précéder d'un `\` (caractère d'échappement).

### Exemples d'expression régulière

Pour tester si un champ est non vide :

```
var nonvideExp = /. /;
```

Pour tester si un champ est composé de 3 chiffres :

```
var codeExp = /^[d]{3}/;
```

Pour tester si un code de plusieurs lettres+2chiffres :

```
var code2Exp = /^[a-z]+[0-9]{2}/;
```

Pour tester un montant (avec ou sans virgule, en dollars) :

```
var montantExp = /^\\d*$|^\\d*\\.\\d{2}$/;
```

Etc....

### III TEST D'UNE CHAÎNE EN UTILISANT UNE EXPRESSION RÉGULIÈRE

Pour tester si une chaîne, après avoir défini l'expression régulière (nommée expReg), il suffit d'appeler la méthode test comme suit :

```
expReg.test(chaîne).
```

Ceci renvoie un Booléen à true si une correspondance est trouvée dans chaîne avec l'expression expReg, et renvoie false sinon.

Il sera judicieux, dans les pages HTML, d'inclure ces contrôles dans des fonctions JavaScript.

D'autres méthodes sont aussi disponibles, qui permettent de remplacer des chaînes, de récupérer des chaînes, etc....

### IV EXERCICE D'APPLICATION

#### IV.1 JS04

Cet exercice va nous permettre d'utiliser une des grandes fonctionnalités de JavaScript, qui est extrêmement importantes et largement utilisée, à savoir l'utilisation de JavaScript pour effectuer des contrôles locaux (sur le client) : contrôles de saisie dans un formulaire. Pour cela, plusieurs manières de faire sont envisageables : on va utiliser ici les expressions régulières qui sont très intéressantes. Suite à ces contrôles locaux, si tout est OK, le formulaire sera envoyé, et une page html suivante sera affichée.

Dans le corrigé qui vous est proposé, les contrôles sont effectués au niveau du formulaire dans son ensemble. Il est bien entendu possible d'avoir des contrôles au niveau d'un champ (événement onblur). De même, on vous propose plusieurs manières d'imposer une saisie numérique (événement onkeypress, événement onblur..).

Ecrire la page HTML suivante :

contrôle de saisie - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils ?

Précédente Recherche

Adresse D:\profs\fayolle\DevSi\E07-Internet\E OK Liens

nom :

prénom :

numero membre :

adresse :

ville :

code postal :

email :

telephone :

validation

Terminé Poste de travail

Effectuer en utilisant les expressions régulières, les contrôles suivants :

- Le nom est obligatoire
- Le prénom est obligatoire
- Le numéro de membre est composé de 8 chiffres
- L'adresse est obligatoire
- Le code postal est composé de 5 chiffres
- L'e-mail est sous la forme [nom1@nom2.nom3](#) dans lesquels nom1 commence par une lettre minuscule, et est suivi d'une série de lettres minuscules, chiffres, underscore, tiret ou point. nom2 est une série de lettres minuscules, chiffres, underscore, tiret ou point, et nom3 est composé de 2 ou 3 lettres minuscules.
- Le téléphone est composé de 10 chiffres.

Si les contrôles sont bons, vous soumettrez le formulaire en affichant une page suivante.

## **Etablissement référent**

*AFPA Champs Sur Marne*

## **Equipe de conception**

*J.C RIGAL*

### **Reproduction interdite**

Article L 122-4 du code de la propriété intellectuelle.  
«toute représentation ou reproduction intégrale ou partielle faite sans le  
consentement de l'auteur ou de ses ayants droits ou ayants cause est illicite. Il  
en est de même pour la traduction, l'adaptation ou la reproduction par un art ou  
un procédé quelconques.»

Date de mise à jour 2008  
afpa © Date de dépôt légal mois année

