JS 17 : Les expressions régulières

Définition

Une expression régulière (du mot « règle ») ou encore expression rationnelle est une suite de caractère qui va permettre, de travailler sur des chaînes de caractères pour :

- vérifier la validité d'une chaîne (nombre et/ou présence de caractères, longueur)
- effectuer des recherches dans une chaîne (par exemple trouver une mot ou une portion de mot)
- effectuer des remplacements de caractères ou de mots dans une chaîne

Une **expression régulière** agit comme un **filtre de vérification** ou une **contrainte** que l'on applique à une chaîne de caractères.

Les expressions régulières existent dans différents langages (Javascript, PHP, C#...) :

les filtres (écriture des expressions) **restent les mêmes d'un langage à l'autre**, par contre les fonctions d'exécution de ces filtres seront propres à chaque langage.

Les expressions régulières sont plutôt efficaces en termes de contrôle des données, la difficulté réside dans l'écriture des règles qui peut paraître complexes pour les débutants.

Exemple

L'expression régulière suivante, très simple, va contrôler que le mot « javascript », une chaîne de caractères donc, comporte bien uniquement des lettres et uniquement en minuscules :

```
var filtre = new RegExp("^[a-z]+$");
var resultat = filtre.test("javascript");
console.log(resultat);
```

- Ligne 1 : on crée un filtre, qui est en fait un objet expression régulière RegExp de Javascript.

Cet objet prend en argument la règle d'écriture `^[a-z]+$` .

L'ensemble est affecté à une variable, ici la variable nommée **filtre**.

- Ligne 2 : on va évaluer la chaîne passée en argument **"javascript"**, avec le **filtre** en utilisant la `fonction` nommée `test()` , une méthode de l'objet RegExp de Javascript.

Cette **méthode renvoie** un **booléen**, ici resultat vaut TRUE puisque la condition évaluée par le filtre est correcte.

Ressources Complémentaires

Le cours d'Openclassrooms sur les expressions régulières version PHP mais adaptable au javascript :

- OpenclassRooms

Un excellent outil en ligne pour construire et tester vos regex (Pensez à cliquer sur ECMAScript(JavaScript)) :

- REGEX 101

Un exercice complet sur la validation de formulaires HTML en JavaScript par le créateur de contenu Pierre Giraud :

- Le TD de Pierre Giraud sur la validation de formulaire

Ecriture des règles

Deux types de caractères peuvent être utilisés dans les expressions régulières :

les méta caractères **. \ ? * + { } ( ) [ ]** et les caractères normaux tout autre caractère, y compris les symboles précédents qui devront être alors représentés par des séquences d'échappement.

Les chaînes alphabétiques

[A-Za-z] permet de vérifier qu'un caractère fait partie des lettres **"A" à "Z"** et **"a" à "z"** (en dehors de tout caractère accentué).

Les slashes / délimitent l'expression régulière.

Mais on peut aussi bien mettre des guillemets :

/[A-Za-z]/ ou **"[A-Za-z]"**

Pour indiquer que nous ne voulons QUE des caractères alphabétiques, il faut écrire :**/^[A-Za-z]+$/**

Les 2 caractères « ^ » et « **$** » indiquent qu'il faut établir le contrôle du **début à la fin de la chaîne**.

Le caractère « **+** » **indique** que le caractère alphabétique doit être **présent au moins une fois**.

Les chaînes numériques

`/^[0-9]*$/`

L'expression [0-9] indique qu'on n'attend que des chiffres.

On peut aussi utiliser « \d » à la place de [0-9].

L'astérisque «*****» signifie que le caractère peut être absent ou présent plusieurs fois (alors que « **+** » implique que le caractère soit présent au moins une fois).

Donc si la chaîne est vide, elle sera considérée comme correcte, selon l'expression régulière.

Les dates

- **/^[0-9]+\/[0-9]+\/[0-9]+$/**

  Il s'agit seulement d'une date au format numérique, dans le style "14/7/2003". Dans l'expression régulière, on a 3 parties :

  le jour, le mois et l'année, qui sont des chiffres répétés plusieurs fois.

  Ils sont séparés par le signe « \/ » : on ne peut pas mettre simplement « / », parce que ce signe indique l'encadrement de l'expression régulière, donc il faut « échapper » le slash par un antislash (« \ »).

```
/^[0-9][0-9]?\/[0-9][0-9]?\/[0-9][0-9]([0-9][0-9])?$/
```

Avec cette expression régulière on est obligé d'écrire le jour et le mois avec un ou 2 chiffres, et on doit écrire l'année sur 2 ou 4 chiffres.

C'est-à-dire : « 14/7/03 » ou « 14/7/2003 » ou « 14/07/2003 »

On a introduit quelques nouveautés :

- le ? indique que le caractère précédent peut être présent 0 fois ou 1 fois, donc [0-9][0-9]? signifie qu'on peut écrire un ou 2 chiffres;
- les parenthèses permettent d'affecter le quantificateur (**+**, *****, ?) à la série de caractères entre ces parenthèses : ([0-9][0-9])? signifie que ces 2 caractères peuvent être présents 0 fois ou 1 fois.

Si l'on souhaite d'autre séparateur que le / pour la date, il faudra utiliser « **(\/|-|.)** » qui autorise soit le slash, soit le tiret, soit le point par exemple

Les adresses mail

Une adresse e-mail, par exemple `nom.prenom@site.fr` est constituée de 3 parties :

L'utilisateur (ici : `nom.prenom`) :

est donc contrôlé par la séquence [_a-z0-9-]+(.[_a-z0-9-]+) *où [_a-z0-9-] représente les caractères alphanumériques, plus le caractère de soulignement, plus le tiret. La 2ème partie (.[_a-z0-9-]+)* permet d'ajouter des mots séparés par un point.

Le nom de domaine (ici : `site`) :

Il suit l'arobase et ne peut contenir que des caractères alphanumériques, le caractère de souligne-ment, et le tiret : **[a-z0-9-]+**

L'extension TLD (top level domain) (ici : `fr`) :

Il est constitué seulement d'un point suivi de caractères alphanumériques, éventuellement répété plusieurs fois : « (.[a-z0-9]+) »

Voici un exemple :

- **^[a-z0-9_-]+@[a-z0-9-.]{2,}.[a-z]{2,4}$**

Les numéros de téléphones

Numero de telephone de type 0111111111 :

- **^0[1-9]{9}$**

Numero de telephone de type 0111111111 ou 01.11.11.11.11 ou 01 00 00 00 00 ou 01-11-11-11-11 :

- **^0[1-9]([-.]?[0-9]{2}){4}$**

la même chose que

- **^0[1-9]([-.]?\d{2}){4}$**

Autres cas

Type	Règle à vérifier	Regex à appliquer
Chaîne alphanumérique	On peut aussi utiliser le caractère « \w », qui autorise les caractères alphanumériques ou le caractère de soulignement (_)	/^[0-9A-Za-z]+$/
Code couleur	Un code couleur est constitué d'un « # », suivi d'un nombre hexadécimal, qui ne peut contenir que des chiffres ou des lettres de A à F.	/^#[0-9A-F]+$/

Caractères spéciaux

Voici les principaux caractères spéciaux que vous pourrez utiliser dans vos expressions régulières.

Caractères spéciaux	Signification
\	Si le caractère suivant est n'est pas un caractère spécial, cela signifie que ce caractère doit être considéré comme un caractère spécial. Exemple : \d signifie un chiffre Si le caractère suivant est un caractère spécial, cela signifie qu'il faut prendre ce caractère de façon littérale. Exemple * cherche la présence d'un astérisque *
^	indique l'emplacement où doit commencer la chaîne de caractère à contrôler
$	indique l'emplacement où doit finir la chaîne de caractère à contrôler
*	indique que le caractère précédent doit être présent 0 fois ou plusieurs fois (soit absent soit présent ou répété)
+	indique que le caractère précédent doit être présent 1 fois ou plusieurs fois (donc au moins une fois)
?	indique que le caractère précédent doit être présent 0 ou 1 fois (soit absent soit présent mais pas répété)
(x)	le caractère doit correspondre à x et permet de mettre ce caractère en mémoire (sert pour la fonction "exec" qui découpe la chaîne testée dans un tableau)
|	ex : x|y, le caractère doit correspondre à x OU à y ("|" est le caractère CTL-ALT 6 ou AltGr 6 sur PC)
{n}	si n est un nombre, le caractère précédent doit être présent n fois
{n, p}	si n et p sont des nombres, le caractère précédent doit être présent au minimum n fois et au maximum p fois
[abc]	le caractère doit correspondre aux caractères entre crochets ("a", "b" ou "c")
[^abc]	le caractère ne doit pas correspondre aux caractères entre crochets ("a", "b" ou "c")
\s	le caractère doit correspondre à un espace, un retour chariot, ou un caractère de tabulation
\S	correspond à tous les caractères sauf l'espace
\d	correspond à [0-9], c'est-à-dire à un chiffre
\D	tout caractère sauf un chiffre
\w	correspond aux caractères alphanumériques + le "_" (équivalent à [A-Za-z0-9_])
\W	correspond à tous les caractères sauf les caractères alphanumériques et le "_" (équivalent à [^A-Za-z0-9_])

Fonctions

Fonction	Objet et exemples
test	Retourne vrai si la chaîne respecte l'expression régulière, faux sinon.
exec	applique l'expression régulière à la chaîne et renvoie le résultat
match	applique l'expression régulière à la chaîne et renvoie le(s) résultat(s)
replace	remplace le(s) sous-chaîne(s) vérifiant l'expression régulière par la 2ème chaîne passée en argument et renvoie le résultat
search	renvoie la position de la première sous-chaîne vérifiant l'expression régulière
split	découpe une chaîne selon l'expression régulière