JavaScript
Regex - partie 2*

IFNTI Sokodé – L3

05 Mai 2021

Nous avons vu comment créer des expressions régulières simples et comment les utiliser avec les méthodes des objets String et Regex. Il est temps de descendre un peu plus dans les profondeurs des expressions régulières.

1 Classes de Caractères

Les classes de caractères vont nous permettre de fournir différents choix de correspondance pour un caractère en spécifiant un ensemble de caractères qui vont pouvoir être trouvés. Elles vont nous permettre de rechercher n'importe quel caractère d'une chaîne qui fait partie de la classe de caractères fournie dans le masque, ce qui va rendre les expressions régulières déjà beaucoup plus puissantes et flexibles qu'une recherche classique (rappelez-vous les majuscules).

Pour déclarer une classe de caractères dans notre masque, nous allons utiliser une paire de crochets `[ ]` qui vont nous permettre de délimiter la classe en question.

```
let masque1 = /[aeiouy]/g;
let masque2 = /j[aeiouy]/g;
let masque3 = /[aeiouy][aeiouy]/g;
```

Dans l'exemple si dessus, nous recherchons :
— masque1 : une voyelle
— masque2 : 'j' suivi d'une voyelle
— masque3 : deux voyelles consécutives

Rappelons nous si nous cherchons un motif : il suffit de concaténer les caractères (e.g. /je/ recherchera les motifs constitués d'un 'j' et d'un 'e' consécutifs.
Si nous souhaitons un classe de caractère (dans une liste), nous spécifions la liste entre crochet. (e.g. `/[aeiouy]/` pour rechercher une voyelle).
Nous pouvons combiner les deux techniques en accolant des classes de caractère (e.g. `/j[aeiouy]/` ou `/[aeiouy][aeiouy]/`)

On essaie ?

2 Caractères et méta-caractères

L'objectif d'une expression régulière est d'offrir un motif de caractérisation d'une séquence de caractère. Pour ce faire, il est définie une syntaxe ce qui signifie que certains symboles ne sont pas des caractères mais possèdent une signification spéciale : répétitions du symboles précédents ...

Ce sont les **méta-caractères**. Vous en connaissez déjà deux : `[` et `]`.

ATTENTION : Il existe de nombreux métacaractères qu'on va pouvoir inclure dans nos masques. Cependant, au sein des classes de caractères (dans les crochets), la plupart de ces métacaractères perdent leur signification spéciale. Il faudra donc toujours faire bien attention à bien distinguer les sens de ces caractères selon qu'ils sont dans une classe de caractères ou pas.

*Repris des cours de Pierre Giraud

Commençons par ceux qui ont une signification spéciale au sein d'une classe de caractère (c'est à dire : à l'intérieur des crochets).

— `\` : caractère d'échappement qui permet :
  — de neutraliser un sens spécial d'un métacaractère
  — de donner un sens spécial à des caractères.
— `^` : si placé en début : spécifie que le masque est inversé. C'est à dire que le masque spécifie les caractères qui ne sont pas dans la classe.
— `-` : si placé entre deux caractères, il spécifie un intervalle entre ces deux caractères inclus.

Si on souhaite rechercher le caractère représenté par l'un des métacaractères ci-dessus plutôt que de l'utiliser pour son sens spécial (par exemple si on souhaite rechercher le signe moins), il faudra alors le protéger ou « l'échapper » avec un antislash.

Notez qu'il faudra également protéger les caractères de classe (les crochets) ainsi que le délimiteur de masque (le slash) si on souhaite les inclure pour les rechercher dans une classe de caractères. Dans le cas contraire, cela peut poser des problèmes car le navigateur pourrait penser par exemple que ce n'est pas «] » qui est cherché mais la classe qui est refermée.

2.1 Amusons nous un peu

Créons 8 masques :

1. un masque qui recherche autre choses qu'une voyelle
2. un masque qui recherche une voyelle ou `^`
3. un masque qui recherche une lettre minuscule suivie d'un « o »
4. un masque qui recherche une lettre minuscule ou majuscule suivie d'un « o »
5. un masque qui recherche soit la lettre « a » soit la lettre « z » soit le caractère « - »
6. un masque qui recherche soit une lettre minuscule, un chiffre ou « - »
7. un masque qui cherche soit un chiffre soit `[` soit `]`

Appliquons ces masque au texte suivant :

```
Bonjour, je suis Willy. Mon /numéro/ est le [70-02-26-00]
```

3 Classes de caractères abrégées ou prédéfinies

Rappelez vous que \ (antislash) va permettre non seulement d'échapper un caractère mais aussi de donner un autre sens à certains caractères « normaux ». En définissant des classes abrégées.

Les classes abrégées les plus intéressantes sont les suivantes (faites bien attention aux emplois de majuscules et de minuscules ici !) :

— `\w` : Représente tout caractère de « mot » (caractère alphanumérique + tiret bas).
— `\W` : Représente tout caractère qui n'est pas un caractère de « mot »
— `\d` : Représente un chiffre
— `\D` :Représente tout caractère qui n'est pas un chiffre.
— `\s` : Représente un caractère blanc (espace, retour chariot ou retour à la ligne)
— `\S` : Représente tout caractère qui n'est pas un caractère blanc
— `\t` : Représente une espace (tabulation) horizontale
— `\v` :Représente une espace (tabulation) verticale
— `\n` : Représente un saut de ligne

Exercice donnez les équivalents des quatres premières classes abrégées.

3.1 Entrainement

Nous allons créer 4 masques :

1. notre premier masque correspond à n'importe quel caractère alphanumérique ainsi qu'au tiret bas et nous permet de rechercher ces caractères.
2. notre deuxième masque nous permet lui de trouver tous les caractères qui n'appartiennent pas à la classe `[a-ZA-Z-0-9_]`, c'est-à-dire tout caractère qui n'est ni une lettre de l'alphabet de base ni un chiffre ni un underscore.

3. notre troisième masque nous permet de trouver tous les caractères qui sont des chiffres dans une chaîne de caractères.
4. finalement, notre dernier masque nous permet de trouver n'importe quel chiffre dans la chaîne de caractères ainsi que toutes les lettres minuscules (hors lettres accentuées et à cédille).

Vous pouvez remarquerez que pour le dernier masque on doit inclure notre classe abrégée dans une classe « classique » définie avec des crochets. Cela est en effet tout à fait autorisé.

4 On accélère

Nous avons appris à créer des classes de caractères et des métacaractères. Pour le moment nous n'en avons vu que trois qui ont un rôle au sein des classes de caractères.

Attardons nous sur ceux qui ont un rôles en dehors des classes de caractères.

4.1 Le point

Le point (« . ») va nous permettre de rechercher n'importe quel caractère à l'exception du caractère représentant une nouvelle ligne.

Pour rechercher le caractère « . » dans une chaîne de caractère, il faudra l'échapper ou le protéger avec un antislash dans notre masque comme pour tout métacaractère.

Exercice que recherchent ces masques :

— `/o./`
— `/o\./`
— `/o[.]/`
— `/[o.]/`

4.2 Les alternatives

Le métacaractère `|` (barre verticale) sert à proposer des alternatives. Concrètement, ce métacaractère va nous permettre de créer des masques qui vont pouvoir chercher une séquence de caractères ou une autre.

```
let masque1 = /o|j/;
let masque2 = /Fadilou|Wiyao/;
```

4.3 Les ancres

Il existe deux méta caractères `^` et `$` qui vont permettre d'ancrer les masques au début (`^`) ou à la fin (`$`) d'une chaîne de caractère à analyser.

Que faut-il faire si nous souhaitons cherchez exactement les caractères `^` et `$` ?

ATTENTION `^` a une autre signification s'il se trouve dans une classe.

4.3.1 Exercice

Que font ces masques :

1. `let masque1 = /^./g;`
2. `let masque2 = /^[A-Z]/g;`
3. `let masque3 = /.$/g;`
4. `let masque4 = /a\^\$b/g;`
5. `let masque5 = /[e$]/g;`
6. `let masque6 = /^[^a-z]/g;`
7. `let masque7 = /^...$/;`

4.4 Les quantificateurs

Plutôt que de répéter un caractère ou classe de caractère dans un masque les expressions régulières mettent à notre disposition les quantificateurs à travers quelques métacaractères.

— a`{X}` Une séquence qui contient X « a » consécutifs
— a`{X,Y}` Une séquence qui contient entre X et Y « a » consécutifs
— a`{X,}` Une séquence qui contient au moins X « a » consécutifs (sans limite supérieur)
— a`?` Une séquence qui contient 0 ou 1 « a » : équivalent à a`{0,1}`
— a`+` Une séquence qui contient au moins un « a » : équivalent à a`{1,}`
— a`*` Une séquence qui contient 0, 1 ou plusieurs « a » : équivalent à a`{0,}`

Exercice Que font les masques suivants :

1. `let masque1 = /er?/g;`
2. `let masque2 = /er+/g;`
3. `let masque3 = /^[A-Z].{10,}/g;`
4. `let masque4 = /^\d{10,10}$/;`

5 Exercices

5.1 Compter le nombre de mot dans une chaîne

Écrivez un programme JavaScript pour compter le nombre de mots dans une chaîne (qui doit être extrait d'un texteArea)

1. Supprimez les espaces blancs des positions de début et de fin.
2. Convertissez 2 espaces ou plus en 1.
3. Excluez les nouvelles lignes avec un espacement de début

5.2 Validité adresse email

Écrivez une fonction JavaScript pour vérifier la validité d'une adresse mail.

1. une adresse e-mail est de la forme xxxx`@`xxxxxx.xxx
2. une adresse e-mail sera jugée correcte si elle contient un caractère « `@` » et un caractère « `.` »

5.3 Adresse IP

Écrivez une fonction javaScript pour vérifier si la chaîne de caractère donnée est une adresse IP ou non.
— la chaîne contient au moins trois points
— la chaîne a plus de 15 caractères ou non
— les sous-chaine avant, entre et après les points ne soient que des nombre entre 0 et 255