Expressions régulières

Programmation Web et multimédia I

JS - Cours 11

RegEx

Table des matières

Introduction

Les expression régulières, ça sert à quoi?

Source de l'image: https://www.vectorstock.com/royalty-free-vector/regular-expressions-emblem-vector-28362362

Introduction

Définition

Une expression régulière (« regular expression ») est une suite de caractères qui représente un **motif (pattern)** qu'on **souhaite rechercher dans une chaîne de caractères** afin de valider ou de modifier celle-ci.

Exemple

- On souhaite vérifier si la sous-chaîne «vert» existe dans une chaîne de caractères contenue dans une variable mémorisant une couleur.
- Le motif recherché est donc constitué des 4 caractères `/vert/`.

```
let strChaine = 'bleu-vert';
let motif = /vert/;
console.log(motif.test(strChaine));
// Résultat affiché: true, parce que le motif se retrouve
// dans la chaîne (même si autre chose s'y trouve aussi).
// Retournerait false si le motif ne se trouve pas dans la chaîne.
```

Introduction

Utilité des RegEx

Une expression régulière pourra être utilisée pour :

- Rechercher la présence d'un motif dans une chaîne (motif ou masque, les deux termes sont utilisés).
- Extraire les occurrences du motif qui sont présentes dans la chaîne et les mémoriser dans un tableau.
- Remplacer les occurrences du motif qui sont présentes dans la chaîne.

Dans cette introduction nous ne couvrons que le cas de la recherche de la présence d'un motif dans une chaîne, ceci afin de valider si la chaîne en question répond au format attendu.

Introduction

Implémentation de motifs de RegEx

Il y a deux façons de définir une expression régulière.

1. La première consiste à la délimiter avec des `/`

```
let strChaine = 'bleu-vert';
let motif = /vert/;
console.log(motif.test(strChaine)); // Imprime true
```

2. La seconde consiste à la construire avec `new RegExp()`

```
let strChaine = 'bleu-vert';
let motif = new RegExp('vert');
console.log(motif.test(strChaine)); // Imprime true
```

Une fois votre motif défini, vous pourrez le tester avec des chaînes à l'aide de la méthode `test()`.

Introduction

Validateur de RegEx en ligne

https://regexr.com/

Syntaxes

1. Vérifier que le motif contient un caractère déterminé ou une suite de caractères déterminés
2. Vérifier que le motif contient un caractère qui doit appartenir à un ensemble de caractères (classe)
3. Indiquer le nombre de répétitions d'un élément du motif
4. Préciser le début et la fin du motif dans la chaîne
5. Créer des alternatives d'éléments dans un motif
6. Regrouper des éléments dans un motif

Syntaxes

1. Vérifier que le motif contient un caractère déterminé ou une suite de caractères déterminés

Il suffit de préciser le caractère ou la suite de caractères:

```
/vert/
```

```
let strChaine = 'bleu';
let motif = /vert/;
console.log(motif.test(strChaine));
// Imprime false
```

ou

```
new RegExp('vert');
```

```
let strChaine = 'bleu';
let motif = new RegExp('vert');
console.log(motif.test(strChaine));
// Imprime false
```

Syntaxes

1. Vérifier que le motif contient un caractère déterminé ou une suite de caractères déterminés - suite

Attention, plusieurs caractères jouent un rôle bien précis dans la grammaire des expressions régulières. Si ces caractères font partie du motif recherché il faudra les faire précéder d'un `\`, sinon ils joueront le rôle qui leur est associé.
Il s'agit des caractères :

```
^ $ . * + ? = ! : | \ / ( ) [ ] { }
```

Exemple

```
let motif = /3\.45\$/;
```

Le rôle de la plupart de ces caractères sera expliqué plus loin, certains dépassent toutefois le cadre de cette introduction.

Syntaxes

2. Vérifier que le motif contient un caractère qui doit appartenir à un ensemble de caractères (classe)

On mentionne entre crochets l'ensemble des caractères de la classe.

Lorsqu'un `^` se trouve dans les crochets et précède les caractères de la classe cela signifie qu'on recherche un caractère différent de ceux de la classe.

Intervalle	Equivalent	Description
`[a-z]`	`[abcdefghijklmonpqrstuvwxyz]`	Lettres minuscules de `a` à `z`
`[A-Z]`	`[ABCDEFGHIJKLMNOPQRSTUVWXYZ]`	Lettres majuscules de `A` à `Z`
`[0-9]`	`[0123456789]`	Chiffres entre `0` à `9`
`[a-z0-9]`	`[abcdefghijklmonpqrstuvwxyz0123456789]`	Lettres minuscules de `a` à `z` ou chiffres entre `0` à `9`

Syntaxes

2. Vérifier que le motif contient un caractère qui doit appartenir à un ensemble de caractères (classe) – suite 1

Intervalle	Equivalent	Description
`[a-d]`	`[abcd]`	Lettres minuscules de **a** à **d**
`[à-ÿ]`	–	Tous les caractères accentués minuscules
`[^a-z]`	`[^abcdefghijklmonpqrstuvwxyz]`	Ne sont pas des lettres minuscules de **a** à **z**
`[^0-9]`	`[^0123456789]`	Caractère qui n'est pas un chiffre de l'alphabet latin classique
`[aB\#]`	–	Vérifie si la lettre minuscule **a**, la lettre majuscule **B** ou le symbole **#** est présent.
`Prix [0-9]$`	`Prix [0123456789]$`	Vérifie si le chiffre utilisé dans la chaine de caractère `'Prix <chiffre>$'` se situe entre **0** et **9**.

Syntaxes

2. Vérifier que le motif contient un caractère qui doit appartenir à un ensemble de caractères (classe) – suite 2

Notez que certaines classes courantes ont des **raccourcis** comme:

Intervalle	Equivalent	Description
`\d`	`[0-9]`	Chiffres entre **0** à **9**
`\D`	`[^0-9]`	Caractère qui n'est pas un chiffre de l'alphabet latin classique
`\w`	`[A-Za-z0-9_]`	N'importe quel caractère alpha-numérique de l'alphabet latin classique (*sans les accents) et aussi au tiret bas (underscore).

Syntaxes

3. Indiquer le nombre de répétitions d'un élément du motif

On utilise, à la suite de l'élément à répéter, une des syntaxes présentées ci-dessous

Quantificateur	Equivalent	Description
`{n,m}`	–	Indique que l'élément doit être répété au minimum `n` fois et au maximum `m` fois
`{n,}`	–	Indique que l'élément doit être répété `n` fois ou plus
`{n}`	–	Indique que l'élément doit être répété exactement `n` fois
`?`	`{,1}`	Indique que l'élément peut être présent `0` ou `1` fois
`+`	`{1,}`	Indique que l'élément doit être présent au moins `1` fois
`*`	`{0,}`	Indique que l'élément peut être présent `0` ou plusieurs fois

Syntaxes

3. Indiquer le nombre de répétitions d'un élément du motif - suite

Exemples de quantificateur

Motif	Description
`a{2}`	La lettre minuscule **a** doit être répété **2** fois de suite
`C{2,}`	La lettre majuscule **C** doit être répété **2** fois de suite ou plus
`[a-d]{2,5}`	Des lettres minuscules entre **a** et **d** doivent être répétés de **2** à **5** fois de suite
`[1-3]+`	Un chiffre entre 1 et 3 doit être présent au moins 1 fois

Syntaxes

4. Préciser le début et la fin du motif dans la chaîne

Délimiteur	Description
^	Indique que l'élément du motif qui suit le ^ doit se trouver au début de la chaîne
$	Indique que l'élément du motif qui précède le $ doit se trouver à la fin de la chaîne

Exemple

```
const strChaine = 'Bonjour Simon';

const motifA = /^Bonjour/; // La chaîne doit commencer par bonjour
console.log(motifA.test(strChaine)); // Imprime true

const motifB = /Bonjour$/; // La chaîne doit finir par bonjour
console.log(motifB.test(strChaine)); // Imprime false

const motifC = /^Bonjour$/; // La chaîne doit commencer et finir par bonjour
console.log(motifC.test(strChaine)); // Imprime false
```

Syntaxes

4. Préciser le début et la fin du motif dans la chaîne - suite

L'utilisation des deux délimiteurs en même temps nous permet de nous assurer que non seulement la chaine contient le motif mais que tout son contenu correspond au motif.

Attribut HTML5 `pattern`

En HTML5, les pattern réagissent par défaut comme si ces délimiteurs étaient présents. En JavaScript, il est important de ne pas les oublier.
Pensez à les mettre dans votre HTML si vous prévoyez récupérer vos patterns pour les utiliser aussi en JavaScript.

Syntaxes

5. Créer des alternatives d'éléments dans un motif

Comparaison	Description
|	Il sépare différentes alternatives comme l'opérateur || en JavaScript

Exemple

```
const strChaine = 'a1b';

const motif = /[a-c]{2,}|[0-5]{2,}/; // La chaîne doit contenir au moins deux caractères
                                     // de suite entre a et c ou contenir au moins
                                     // deux caractères de suite entre 0 et 5
console.log(motif.test(strChaine)); // Imprime false
```

Syntaxes

6. Regrouper des éléments dans un motif

Regroupement	Description
( )	Les parenthèses permettent de regrouper des éléments afin de leur appliquer une quantité ou de les impliquer dans une alternative

Exemple

```
const strChaine = 'J\'aime les bonbon';

const motif = /(bon){2}/; // La chaîne doit contenir au moins deux fois
                          // de suite le texte bon
console.log(motif.test(strChaine)); // Imprime true
```

Syntaxes

RegEx cheatsheet

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Cheatsheet

Syntaxes

Exemple de RegEx d’adresse courriel

https://regexr.com/3e48o