# Apprendre Javascript - Expressions rationnelles

Les expressions rationnelles (régulières) sont des objets qui représentent un modèle de recherche. Elles sont utilisées pour effectuer des recherches dans des chaînes de caractères.

## Structure d'une expression rationnelle

Une expression rationnelle est délimitée par des `/` et peut contenir des lettres, des chiffres, des caractères spéciaux et des métacaractères.

**À noter que les expressions rationnelles ne sont pas entourées de guillemets.**

```
const regex = /motif/;
```

Pour tester: https://regexr.com/

Pour aller plus loin: https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Regular_expressions

### Délimitateurs et options

- ^ : Début de la chaîne
- $ : Fin de la chaîne
- i : Insensible à la casse
- g : Recherche globale, c'est-à-dire recherche de toutes les occurrences

```
// Recherche le mot "motif" uniquement en début de chaîne
// et insensible à la casse et en global
const regex = /^motif$/gi;
```

### Caractères spéciaux

Les classes de caractères permettent de définir un ensemble de caractères possibles.

- [abc] : a, b ou c. Un seul caractère parmi la liste
- [a-z] : de a à z
- [A-Z] : de A à Z
- [a-zA-Z] : de a à z ou de A à Z
- [0-9] : de 0 à 9

- [^abc] : tout sauf a, b ou c
- [^a-z] : tout sauf de a à z
- . : n'importe quel caractère sauf un retour à la ligne
- \d : un chiffre
- \D : un caractère qui n'est pas un chiffre
- \w : un caractère alphanumérique
- \W : un caractère qui n'est pas alphanumérique
- \s : un espace
- \b : un mot entier
- {n} : exactement n occurrences
- {n,} : au moins n occurrences
- {n,m} : entre n et m occurrences
- ? : 0 ou 1 occurrence
- + : 1 ou plusieurs occurrences
- * : 0 ou plusieurs occurrences
- () : groupe de caractères

```
// Recherche d'un code postal canadien
// Contient un groupe contenant une lettre, un chiffre et une lettre
const regexCodePostal = /^([a-Z]\d[a-Z])\s?(\d[a-Z]\d)$/i;
const codePostal = "H1H 1H1";
console.log(regexCodePostal.test(codePostal)); // true
```

## Rechercher/remplacer un motif

### Rechercher un motif

Pour rechercher un motif dans une chaîne de caractères, on utilise la méthode `test()` de l'objet `RegExp`. Cela retourne `true` si le motif est trouvé et `false` sinon.

```
const regex = /motif/;
const chaine = "Ceci est un motif de recherche";
```

```
console.log(regex.test(chaine)); // true
```

### Rechercher toutes les occurrences

Pour rechercher toutes les occurrences d'un motif dans une chaîne de caractères, on utilise la méthode `match()` et `matchAll()` de l'objet `String`. Si on ajoute le modificateur `g`, toutes les occurrences du motif seront retournées sous forme de tableau.

```
const regex = /motif/g;
const chaine = "Ceci est un motif de recherche motif";

console.log(chaine.match(regex)); // ["motif", "motif"]
```

### Remplacer un motif

Pour remplacer un motif dans une chaîne de caractères, on utilise la méthode `replace()` et `replaceAll()` de l'objet `String`.

On peut également utiliser des groupes pour remplacer des motifs. Dans ce cas, on utilise `$1`, `$2`, etc. pour faire référence aux groupes. On peut donc recréer la chaîne de caractères en inversant les mots ou ajouter un préfixe ou un suffixe.

```
const regex = /motif/g;
const chaine = "Ceci est un motif de recherche motif";

console.log(chaine.replace(regex, "remplacement")); // "Ceci est un remplacement de
recherche remplacement"

// Remplacement avec un groupe
const regexGroupe = /(\w+)\s(\w+)/;
const chaineGroupe = "Ceci est un motif de recherche";

console.log(chaineGroupe.replace(regexGroupe, "$2 $1")); // "est Ceci un motif de
recherche"
```

## Pattern d'un champ de formulaire

Voici quelques exemples de pattern pour valider un champ de formulaire dans le code HTML. Si le champ ne respecte pas le pattern, un message d'erreur de type `PatternMismatch` sera affiché.

**À noter que les expressions rationnelles dans les attributs `pattern` ne sont pas entourées de `/`.**

```
```

```
<!-- Champ de texte -->
<input type="text" pattern="[\d]{3,}" title="3 caractères alphanumérique minimum" />

<!-- Champ de courriel avec un nom de domaine spécifique -->
<input type="email" pattern="^[\w.]{3,}@domaine.com$" title="Courriel avec le
domaine domaine.com" />
<!-- Champ de téléphone -->
<!-- Accepte les formats 123-456-7890, 123 456 7890, 1234567890 -->
<input type="tel" pattern="^(\d{3}[-\s]?\d{3}[-\s]?\d{4}$" title="Téléphone
canadien" />
```