

Support Regex pour Formulaires

Valider les champs comme un pro

Formation DWWM - La Plateforme_

1. C'est quoi une RegEx ?

Une RegEx (Regular Expression ou Expression Régulière) est un pattern qui permet de rechercher, valider ou manipuler du texte.

Cas d'usage

- Valider un email
- Valider un numéro de téléphone
- Valider un mot de passe (complexité)
- Valider un code postal
- Extraire des informations (prix, dates)
- Remplacer du texte

 RegEx = motif de recherche puissant pour valider des formats

2. Syntaxe de base

Créer une RegEx en JavaScript

```
// Avec littéral (recommandé)
const regex = /pattern/;

// Avec constructeur
const regex = new RegExp("pattern");

// Avec flags
const regex = /pattern/gi;
// g = global, i = insensible à la casse
```

Caractères spéciaux de base

Symbol	Signification	Exemple
.	N'importe quel caractère	/a.c/ → abc, a3c
^	Début de chaîne	/^Hello/ → Hello World
\$	Fin de chaîne	/end\$/ → The end

*	0 ou plusieurs fois	/ab*c/ → ac, abc, abbc
+	1 ou plusieurs fois	/ab+c/ → abc, abbc
?	0 ou 1 fois	/colou?r/ → color, colour
{n}	Exactement n fois	/a{3}/ → aaa
{n,}	Au moins n fois	/a{2,}/ → aa, aaa
{n,m}	Entre n et m fois	/a{2,4}/ → aa, aaa, aaaa

3. Classes de caractères

Classes prédéfinies

Classe	Équivalent	Signification
\d	[0-9]	Un chiffre
\D	[^0-9]	Pas un chiffre
\w	[A-Za-z0-9_]	Lettre, chiffre ou _
\W	[^A-Za-z0-9_]	Pas \w
\s	[\t\n]	Espace blanc
\S	[^ \t\n]	Pas un espace

Classes personnalisées

```
// Entre crochets []
[abc]      // a ou b ou c
[a-z]      // Lettre minuscule
[A-Z]      // Lettre majuscule
[0-9]      // Chiffre
[a-zA-Z]   // Lettre (min ou maj)
[^0-9]     // Tout sauf chiffre (^ = négation)
```

4. Méthodes JavaScript

.test() - Tester si correspond

```
const regex = /hello/i;

regex.test("Hello World"); // true
regex.test("Goodbye");    // false
```

💡 test() = méthode la plus utilisée pour valider des formulaires

.match() - Extraire les correspondances

```
const text = "Mon email: user@example.com";
const regex = /[\w.-]+@[ \w.-]+\.[a-z]{2,}/gi;
```

```
const emails = text.match(regex);
console.log(emails); // ["user@example.com"]
```

.replace() - Remplacer

```
const text = "Téléphone: 0123456789";
const regex = /(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})/;

const formatted = text.replace(regex, "$1 $2 $3 $4 $5");
console.log(formatted); // "Téléphone: 01 23 45 67 89"
```

.exec() - Extraire avec détails

```
const regex = /(\d{2})/(\d{2})/(\d{4})/;
const result = regex.exec("Date: 25/12/2024");

console.log(result[0]); // "25/12/2024"
console.log(result[1]); // "25" (jour)
console.log(result[2]); // "12" (mois)
console.log(result[3]); // "2024" (année)
```

5. Validation Email

Pattern simple

```
const emailRegex = /^[^\w.-]+@[^\w.-]+\.[a-z]{2,}$/i;

// Décomposition :
// ^          Début
// [\w.-]+    1+ lettres, chiffres, . ou -
// @          Arobase
// [\w.-]+    Domaine
// \.         Point
// [a-z]{2,}   Extension (2+ lettres)
// $          Fin
// i          Insensible à la casse
```

Fonction de validation

```
function validerEmail(email) {
  const regex = /^[^\w.-]+@[^\w.-]+\.[a-z]{2,}$/i;
  return regex.test(email);
}
```

```
// Tests
validerEmail("user@example.com");      // true
validerEmail("user.name@site.fr");      // true
validerEmail("invalid");               // false
validerEmail("no@domain");             // false
```

Cette regex couvre 95% des emails courants

6. Validation Téléphone France

Fixe ou mobile

```
// Format: 01 23 45 67 89 ou 0123456789
const telRegex = /^0[1-9](\d{2}){4}$/;

// Décomposition :
// ^          Début
// 0          Commence par 0
// [1-9]      Puis chiffre 1-9
// (\d{2}){4}  4 groupes de 2 chiffres
// $          Fin
```

Avec espaces optionnels

```
// Accepte avec ou sans espaces
const telRegex = /^0[1-9](\s?\d{2}){4}$/;

function validerTelephone(tel) {
  // Retirer espaces avant test
  const clean = tel.replace(/\s/g, "");
  const regex = /^0[1-9]\d{8}$/;
  return regex.test(clean);
}

// Tests
validerTelephone("0123456789");      // true
validerTelephone("01 23 45 67 89");    // true
validerTelephone("06 12 34 56 78");    // true
validerTelephone("1234567890");       // false (pas de 0)
```

7. Validation Mot de passe

Règles de complexité

Un bon mot de passe doit avoir :

- Au moins 8 caractères
- Au moins 1 majuscule
- Au moins 1 minuscule
- Au moins 1 chiffre
- Au moins 1 caractère spécial

Pattern avec lookahead

```
// Lookahead = regarder en avant sans consommer
const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;

// Décomposition :
// ^                      Début
// (?=.*[a-z])            Au moins 1 minuscule
// (?=.*[A-Z])            Au moins 1 majuscule
// (?=.*\d)                Au moins 1 chiffre
// (?=.*[$!%*?&])         Au moins 1 spécial
// [A-Za-z\d@$!%*?&]      Caractères autorisés
// {8,}                    Minimum 8 caractères
// $                      Fin
```

Fonction avec messages détaillés

```
function validerPassword(pwd) {
  const errors = [];

  if (pwd.length < 8) {
    errors.push("Minimum 8 caractères");
  }
  if (!/[a-z]/.test(pwd)) {
    errors.push("Au moins 1 minuscule");
  }
  if (!/[A-Z]/.test(pwd)) {
    errors.push("Au moins 1 majuscule");
  }
  if (!/\d/.test(pwd)) {
    errors.push("Au moins 1 chiffre");
  }
  if (!/[@$!%*?&]/.test(pwd)) {
    errors.push("Au moins 1 caractère spécial");
  }

  return {
    valid: errors.length === 0,
    errors: errors
  };
}
```

```
}
```

8. Validation Code postal

France

```
// Format: 5 chiffres
const cpRegex = /^[0-9]{5}$/;

cpRegex.test("75001");    // true (Paris)
cpRegex.test("13000");    // true (Marseille)
cpRegex.test("1234");     // false (trop court)
cpRegex.test("ABCDE");   // false (pas des chiffres)
```

9. Validation URL

```
// URL simple
const urlRegex = ^https?:\/\/[\.w\.-]+\.[a-z]{2,}\/.*?$/i;

// Tests
urlRegex.test("https://example.com");           // true
urlRegex.test("http://site.fr/page");            // true
urlRegex.test("ftp://server.com");              // false (ftp)
urlRegex.test("not-a-url");                     // false
```

10. Validation Date

Format JJ/MM/AAAA

```
// Pattern basique
const dateRegex = /^(0[1-9]|1[2]\d|3[01])\/(0[1-9]|1[012])\/(19|20)\d{2}$/;

// Décomposition :
// (0[1-9]|1[2]\d|3[01]) Jour: 01-31
// \
// (0[1-9]|1[012])        Mois: 01-12
// \
// (19|20)\d{2}            Année: 1900-2099

dateRegex.test("25/12/2024"); // true
dateRegex.test("32/01/2024"); // false (jour > 31)
dateRegex.test("15/13/2024"); // false (mois > 12)
```

⚠ Cette regex ne vérifie pas les jours réels (ex: 31/02 passe). Pour une validation complète, utilise Date en plus.

11. Exemple complet : Formulaire

```
// HTML
// <form id="inscription">
//   <input type="text" id="nom" placeholder="Nom">
//   <input type="email" id="email" placeholder="Email">
//   <input type="tel" id="tel" placeholder="Téléphone">
//   <input type="password" id="pwd" placeholder="Mot de passe">
//   <button type="submit">Valider</button>
// </form>

// JavaScript
const form = document.querySelector("#inscription");

form.addEventListener("submit", (e) => {
  e.preventDefault();

  const nom = document.querySelector("#nom").value;
  const email = document.querySelector("#email").value;
  const tel = document.querySelector("#tel").value;
  const pwd = document.querySelector("#pwd").value;

  // Validation
  const errors = [];

  // Nom (2-50 caractères, lettres)
  if (!/^[a-zA-Zà-ÿ\s-]{2,50}$/.test(nom)) {
    errors.push("Nom invalide");
  }

  // Email
  if (!/^\w+@\w+\.\w{2,}$/.test(email)) {
    errors.push("Email invalide");
  }

  // Téléphone
  const cleanTel = tel.replace(/\s/g, "");
  if (!/^0[1-9]\d{8}$/.test(cleanTel)) {
    errors.push("Téléphone invalide");
  }

  // Mot de passe
  if (pwd.length < 8 || !/[A-Z]/.test(pwd) || !/[a-z]/.test(pwd) || !/\d/.test(pwd)) {
    errors.push("Mot de passe trop faible");
  }
}
```

```

// Afficher erreurs
if (errors.length > 0) {
    alert(errors.join("\n"));
} else {
    alert("Formulaire valide !");
    // Envoyer au serveur
}
});

```

12. Tableau patterns courants

Champ	RegEx	Exemple
Email	/^[\w.-]+@[\w.-]+\.[a-z]{2,}\$/i	user@site.com
Téléphone FR	/^0[1-9]\d{8}\$/	0612345678
Code postal	/^0-9]{5}\$/	75001
Date JJ/MM/AAAA	/^\d{2}/\d{2}/\d{4}\$/	25/12/2024
URL	/^https?:\/\/.+\$/	https://site.com
Hexadécimal	/^#[0-9A-F]{6}\$/i	#FF5733
IPv4	/^\d{1,3}(\.\d{1,3}){3}\$/	192.168.1.1
Nom	/^[\a-zA-ZÀ-Ӄ\s-]{2,50}\$/	Jean-Pierre

13. Bonnes pratiques

- ✓ Teste toujours tes regex avec plusieurs exemples
- ✓ Utilise .test() pour valider (plus rapide)
- ✓ Échappe les caractères spéciaux avec \
- ✓ Utilise des regex simples et lisibles
- ✓ Ajoute des commentaires pour regex complexes
- ✓ Valide côté client ET serveur
- ✓ Donne des messages d'erreur clairs
- ✗ Ne valide pas TOUT avec regex (utilise Date pour dates)
- ✗ Ne fais pas confiance à 100% aux regex (sécurité)
- ✗ Ne réinvente pas la roue (utilise patterns éprouvés)

14. Résumé

- ❖ RegEx = pattern pour valider du texte
- ❖ .test() = tester si correspond (true/false)
- ❖ .match() = extraire les correspondances

- 📌 .replace() = remplacer avec pattern
- 📌 ^ = début, \$ = fin
- 📌 \d = chiffre, \w = lettre/chiffre, \s = espace
- 📌 + = 1 ou plus, * = 0 ou plus, ? = optionnel
- 📌 Toujours valider côté serveur aussi

Formation DWWM - La Plateforme_ - 2025