Guide Debugging Regex

Erreurs courantes et solutions

Formation DWWM - La Plateforme_

Erreur 1 : Oublier ^ et $

Symptômes

La regex valide des chaînes partielles au lieu de valider la chaîne complète.

❌ ERREUR :

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

emailRegex.test("Mon email: user@site.com ici");  // true !
// Valide alors qu'il y a du texte avant et après
```

✅ SOLUTION : Ajouter ^ (début) et $ (fin)

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

emailRegex.test("user@site.com");         // true
emailRegex.test("texte user@site.com");   // false
emailRegex.test("user@site.com texte");   // false
```

💡 ^ et $ forcent la regex à matcher TOUTE la chaîne, pas juste une partie

Erreur 2 : Échapper dans strings

Symptômes

```
SyntaxError: Invalid regular expression
```

Cause

Dans une string, \ doit être doublé : \\

❌ ERREUR :

```
// Avec new RegExp
const regex = new RegExp("\d+");  // Erreur !
// \d est interprété comme un caractère d'échappement
```

✅ SOLUTION : Doubler les backslashes

```
// Doubler \
const regex = new RegExp("\\d+");  // OK

// Ou mieux : utiliser littéral
const regex = /\d+/;  // Recommandé
```

💡 Préfère toujours /pattern/ plutôt que new RegExp() pour éviter ce problème

Erreur 3 : Valider email trop strictement

Problème

Une regex email trop stricte rejette des emails valides.

❌ TROP STRICT :

```
// Rejette les + et autres caractères valides
const emailRegex = /^[a-z]+@[a-z]+\.[a-z]+$/;

emailRegex.test("user+tag@site.com");  // false (devrait être true)
emailRegex.test("user.name@site.fr");  // false (devrait être true)
```

✅ ÉQUILIBRÉ :

```
// Accepte la plupart des formats valides
const emailRegex = /^[\w.-]+@[\w.-]+\.[a-z]{2,}$/i;

emailRegex.test("user+tag@site.com");   // true
emailRegex.test("user.name@site.fr");   // true
```

💡 Ne cherche pas la perfection, 95% des emails suffit

Erreur 4 : Oublier le flag i

Symptômes

La regex rejette les majuscules/minuscules.

❌ SENSIBLE À LA CASSE :

```
const regex = /^[a-z]+$/;  // Pas de flag i

regex.test("hello");   // true
regex.test("Hello");   // false
regex.test("HELLO");   // false
```

✅ INSENSIBLE : Ajouter flag i

```
const regex = /^[a-z]+$/i;  // Flag i
```

```
regex.test("hello");   // true
regex.test("Hello");   // true
regex.test("HELLO");   // true
```

Erreur 5 : Point sans échappement

Cause

. = n'importe quel caractère (pas le point littéral)

❌ PROBLÈME :

```
// Voulait matcher "site.com"
const regex = /site.com/;

regex.test("siteXcom");  // true (match !)
// . matche n'importe quel caractère
```

✅ SOLUTION : Échapper le point

```
const regex = /site\.com/;

regex.test("site.com");   // true
regex.test("siteXcom");   // false
```

💡 Caractères à échapper : . * + ? ^ $ { } ( ) | [ ] \

Erreur 6 : Téléphone avec espaces

Problème

La regex ne valide pas si l'utilisateur met des espaces.

❌ ERREUR :

```
const telRegex = /^0[1-9]\d{8}$/;

telRegex.test("0612345678");     // true
telRegex.test("06 12 34 56 78"); // false (espaces rejetés)
```

✅ SOLUTION 1 : Nettoyer avant

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

```
validerTel("06 12 34 56 78");  // true
```

✅ SOLUTION 2 : Accepter espaces

```
// \s? = espace optionnel
const telRegex = /^0[1-9](\s?\d{2}){4}$/;

telRegex.test("0612345678");     // true
telRegex.test("06 12 34 56 78"); // true
```

Erreur 7 : Test incomplet mot de passe

Problème

Tester uniquement la longueur ne suffit pas.

❌ INCOMPLET :

```
const pwdRegex = /^.{8,}$/;  // Juste 8+ caractères

pwdRegex.test("12345678");  // true (trop faible !)
```

✅ COMPLET : Vérifier chaque critère

```
function validerPassword(pwd) {
  return (
    pwd.length >= 8 &&
    /[a-z]/.test(pwd) &&      // Minuscule
    /[A-Z]/.test(pwd) &&      // Majuscule
    /\d/.test(pwd) &&          // Chiffre
    /[@$!%*?&]/.test(pwd)     // Spécial
  );
}
```

Erreur 8 : Valider date avec regex seule

Limite

Regex ne peut pas vérifier les dates réelles (ex: 31/02).

❌ INSUFFISANT :

```
const dateRegex = /^\d{2}\/\d{2}\/\d{4}$/;

dateRegex.test("31/02/2024");  // true (mais invalide !)
dateRegex.test("99/99/9999");  // true (n'importe quoi)
```

✅ SOLUTION : Regex + Date

```
function validerDate(dateStr) {
  // 1. Format
  if (!/^\d{2}\/\d{2}\/\d{4}$/.test(dateStr)) {
    return false;
  }

  // 2. Date réelle
  const [jour, mois, annee] = dateStr.split("/");
  const date = new Date(annee, mois - 1, jour);

  return (
    date.getDate() == jour &&
    date.getMonth() == mois - 1 &&
    date.getFullYear() == annee
  );
}
```

💡 Regex pour format, Date pour validité

Checklist debugging

☐ Tu as ajouté ^ et $?

☐ Tu as échappé les caractères spéciaux ?

☐ Tu as ajouté le flag i si besoin ?

☐ Tu nettoies l'input avant de valider ?

☐ Tu testes avec des cas limites ?

☐ Tu valides aussi côté serveur ?

Outils de test

- regex101.com - Tester en ligne
- regexr.com - Visualiser et expliquer
- Console Chrome - Tester rapidement

Résumé

- Toujours ^ et $ pour match complet
- Échapper \ dans new RegExp()
- Nettoyer input avant validation

- Flag i pour insensible casse
- Échapper . * + ? $ etc.
- Valider côté client ET serveur