Validation des formulaires

La validation des formulaires permet de contrôler que les données saisies sont bien celles attendues (champs requis, formats, plages de données possibles…). Elle peut se faire :

- Côté client : à partir des contrôles de validité HTML5 et Javascript
- Côté serveur (variable en fonction du langage et de la technologie utilisée)

L'avantage d'une validation côté client est qu'elle évite d'envoyer vers le serveur des données invalides (ce qui sollicite inutilement le réseau et le serveur), et est assez déplaisant pour l'utilisateur.
La présence d'une validation côté client ne doit pas conduire le développeur à s'abstenir d'une validation côté serveur (pour des raisons de sécurité).

Validation intégrée HTML5

Champs requis

Contraindre la saisie se fait via l'attribut **required** sur les champs **input** :

```
<form>
  <label for="choose">Quel est votre animal de compagnie préféré ?</label>
  <input id="choose" name="i_like" required>
  <button>Valider</button>
</form>
```

L'attribut **required** existe sur tous les champs input, excepté ceux qui ont une valeur par défaut (**color** et **range**), et sur les **select** et **textarea**.

Pattern de validation

La vérification de contenu peut se faire avec l'attribut **pattern** qui prend en charge les expressions régulières.

Exemple : valeurs attendues : **Chien ou chat**

```
<form>
  <label for="choose">Quel est votre animal de compagnie préféré ?</label>
  <input id="choose" name="i_like" required pattern="[Cc]h(ien|at)">
  <button>Valider</button>
</form>
```

#validation-form input { width: 200px; display: inline-block; } #validation-form input + span { position: relative; } #validation-form input + span::before { position: absolute; right: -20px; top: 0px; margin-right: 30px; } #validation-form input:invalid { border: 2px solid #db4764; } #validation-form input:invalid + span::before { content: '✖'; color: #db4764; } #validation-form input:valid + span::before { content: '✓'; color: green; } #validation-form input:required + span::after { font-size: 1rem; position: absolute; content: "obligatoire"; color: white; background-color: #db4764; padding: 5px 10px; top: -36px; left: -70px; border-radius: 4px; }

Quel est votre animal de compagnie préféré ? Valider

Expressions régulières

Classes

Pattern	Description
[abc]	1 caractère parmi a, b ou c
[^abc]	1 caractère sauf a, b ou c
[a-z]	1 caractère dans la plage a..z
[a-z|A-Z]	1 caractère dans la plage a..z ou A..Z
.	1 caractère quelconque
a|b	a ou b
\s	Tout caractère blanc (espace, tabulation, nouvelle ligne, retour charriot)
\S	Tout caractère non blanc
\d	Tout chiffre
\D	Tout caractère excepté un chiffre
\w	Tout “word” caractère, équivalent à **[a-zA-Z_0-9]**
\W	Tout “non word” caractère, équivalent à **[^a-zA-Z_0-9]**
(?:...)	Match avec l'expression entre parenthèses
(...)	Capture l'expression entre parenthèses

Quantifieurs

Pattern	Description
a?	0 ou 1 **a**
a*	0 ou plusieurs **a**
a+	1 ou plusieurs **a**
a{3}	Exactement 3 **a**
a{3,}	3 **a** ou plus
a{3,6}	Entre 3 et 6 **a**

Ancres

Pattern	Description
^	Début de la chaîne
$	Fin de la chaîne

Testeur en ligne regex101

Personnalisation des messages

Il n'est pas possible de personnaliser l'apparence des messages par défaut, mais uniquement leur contenu, via javascript :

```
<form id="form-messages">
  <label for="mail">Pourriez-vous nous fournir une adresse mail ?</label>
  <input type="email" id="mail" name="mail">
  <button>Envoyer</button>
</form>
```

```
let email=document.getElementById("mail");
email.addEventListener("change", function (event) {
  if(email.validity.typeMismatch) {
    email.setCustomValidity("Cet email n'en est pas un !");
  } else {
    email.setCustomValidity("");
  }
});
```

Pourriez-vous nous fournir une adresse mail ? Envoyer

Validation Javascript

Il est maintenant possible d'utiliser l'api de validation de contraintes HTML5, supportée par la plupart des navigateurs :

Propriétés des éléments

Propriété	Rôle
validationMessage	Retourne le message de validation retourné par l'élément, prenant en compte les contraintes non satisfaites (en lecture seule)
validity.customError	renvoie **true** si l'élément a un message d'erreur personnalisé
validity.patternMismatch	renvoie **true** si la valeur de l'élément ne respecte pas le pattern fourni
validity.rangeOverflow	renvoie **true** si la valeur de l'élément est supérieure au max défini
validity.rangeUnderflow	renvoie **true** si la valeur de l'élément est inférieure au min défini
validity.tooLong	renvoie **true** si la taille de l'élément est supérieure à celle définie
validity.typeMismatch	renvoie **true** si la valeur saisie a une incohérence de type avec celui attendu
validity.valid	renvoie **true** si l'élément passe tous les contrôles de validité
validity.valueMissing	renvoie **true** si un élément requis n'a pas de valeur
willValidate	Renvoie **true** si l'élément est validé lors du submit

Méthodes

Exemple

L'attribut **novalidate** du form désactive la validation HTML5 par défaut.

```
<form novalidate>
  <label for="mail-v">Pourriez-vous nous fournir une adresse mail ?</label>
  <input type="email" id="mail-v" name="mailv">
  <button>Envoyer</button>
  <div class="my-error"></div>
</form>
```

.my-error { color: white; background-color: #db4764; border-radius: 4px; padding: 8px; display: none; margin-top: 10px; } .my-error.active { display: block; } .my-error.valid { background-color: green; }
Pourriez-vous nous fournir une adresse mail ? Envoyer