Validez les données saisies par vos utilisateurs

Avant d'envoyer nos données à un service web, il est nécessaire de les **valider**. C'est-à-dire que nous allons vérifier leur **cohérence** par rapport à ce que le service web attend.

Les données que l'on souhaite envoyer à un service web viennent généralement de ce que l'utilisateur saisit dans un formulaire ; or, ce n'est pas parce que vous lui avez explicitement demandé d'entrer un numéro de téléphone qu'il va le faire...

Ne jamais faire confiance à un utilisateur !

Ce qui se traduit littéralement par : **Ne faites jamais confiance aux données saisies par vos utilisateurs !**

Il est bon de le rappeler :

Ne faites jamais confiance à vos utilisateurs !
Bon, maintenant je pense qu'il est temps d'expliquer pourquoi il faut être si méfiant envers vos utilisateurs, et pourquoi c'est si important.

Certains de vos utilisateurs peuvent être **malveillants** ou ils peuvent **ne pas bien comprendre** ce que vous souhaitez qu'ils fassent. Bref, ils ne vont pas toujours entrer les données que vous attendez d'eux. Or, cela peut se révéler désastreux pour votre site web, et ce, de plusieurs façons :

- Si l'utilisateur ne comprend pas ce que vous attendez de lui, il peut entrer quelque chose qui ne vous convient pas. Votre service web ne fera pas ce qu'il faut et l'utilisateur ne comprendra pas pourquoi ça ne marche pas. Vous risquez de **perdre** cet utilisateur parce que vous n'avez pas validé les données saisies et affiché un message d'erreur indiquant que les données ne sont pas celles attendues ;

- Pire encore, l'utilisateur pourrait **attaquer** votre service web en entrant des **données malveillantes** dans un champ alors que vous attendiez simplement un nom, par exemple. Dans ce cas, il pourrait prendre le contrôle de votre service web, collecter des données utilisateurs, se faire passer pour un administrateur, et j'en passe... (il existe des tonnes de

ressources sur Internet donnant des exemples de ces types d'attaques. Si cela vous intéresse, je vous encourage à faire quelques recherches) ;

- De la même manière, l'utilisateur pourrait **faire planter** votre application s'il entrait du texte dans un champ où vous attendiez un nombre.

Nous allons apprendre ici à valider les données utilisateur sur votre site web avant de les envoyer à votre service web. Cela constitue un premier rempart face aux problèmes cités plus haut, même si ce n'est pas suffisant, **il faudra toujours avoir une validation poussée des données utilisateurs sur le service web**.

C'est important, car rien n'empêchera un utilisateur malveillant d'utiliser un logiciel pour envoyer directement les requêtes HTTP malveillantes à votre service web. De cette manière, elles ne passeront pas par la case validation côté site web...

Validez les données suite à des événements

Afin de valider les données utilisateurs, vous pouvez vous aider des événements du DOM. Ainsi, vous pouvez écouter l'événement `onChange` pour vérifier la donnée, dès que l'utilisateur a fini de l'éditer. Ou bien vous pouvez écouter l'événement `onInput` pour vérifier la donnée à chaque nouveau caractère.

Par exemple, vous pouvez vérifier que ce qui est saisi commence par `Hello` avec le code suivant :

```
myInput.addEventListener('input', function(e) {
    var value = e.target.value;
    if (value.startsWith('Hello ')) {
        isValid = true;
    } else {
        isValid = false;
    }
});
```

Faites une validation plus complexe avec les Regex

Si vous n'avez jamais entendu parlé des **Regex**, sachez qu'il s'agit d'un format spécial qui permet de matcher du texte, c'est-à-dire de vérifier qu'un texte corresponde à une description que l'on a définie. Ainsi, si l'on veut savoir si notre texte commence par la lettre `e` et est suivi d'au moins 3 chiffres, on écrira la regex suivante :

```
function isValid(value) {
    return /^e[0-9]{3,}$/.test(value);
}
```

Cela peut paraître barbare comme notation, mais c'est très puissant et très pratique !

Pour en savoir plus, https://regexr.com/.

Découvrez les contraintes HTML5

Depuis HTML version 5, il est possible d'ajouter de la validation **directement** dans le code HTML, sans avoir besoin d'écrire la moindre ligne de JavaScript.

Pour cela, différents **attributs** sont ajoutés et permettent d'**empêcher la soumission d'un formulaire** si toutes les validations ne sont pas respectées.

L'attribut type pour les inputs

Pour valider les informations saisies dans une balise `input`, il est possible d'utiliser l'attribut `type`.

L'attribut `type` de la balise `input` ne prend pas seulement comme valeurs `text` et `password`. Cela peut aussi être `email`, `tel`, `URL`, `date` et bien d'autres.

Lorsque vous ajoutez un élément `input` avec un attribut `type="email"`, le navigateur empêchera la soumission du formulaire si ce n'est pas une adresse email correcte.

Les attributs de validation simples

En fonction du `type` de l'`input`, vous pouvez utiliser différents attributs pour **perfectionner** votre validation :

- `min` / `max` : fonctionne avec des champs de type **nombre** ou **date**. Cela permet de définir une valeur minimum et une valeur maximum autorisées ;

- `required` : fonctionne avec à peu près **tous les types de champs**. Cela rend obligatoire le remplissage de ce champ ;

- `step` : fonctionne avec les **dates** ou les **nombres**. Cela permet de définir une valeur d'incrément lorsque vous changez la valeur du champ via les flèches ;

- `minlength` / `maxlength` : fonctionne avec les **champs textuels** (`text`, `url`, `tel`, `email`...). Cela permet de définir un nombre de caractères minimum et maximum autorisé.

Les patterns

Nous avons vu qu'il était possible d'avoir une validation complexe grâce aux Regex en JavaScript. Eh bien c'est aussi possible directement en HTML5 avec l'attribut `pattern`. Il suffit de définir une Regex dans cet attribut, et vous obligez la valeur du champ correspondant à la respecter.

Par exemple, si on prend le code suivant :

```
<input type="text" pattern="[0-9]{,3}" />
```

il empêchera un utilisateur d'entrer autre chose que des chiffres, et limitera leur nombre à 3 chiffres.

Pour en savoir plus, https://developer.mozilla.org/fr/docs/Web/Guide/HTML/HTML5/Constraint_validation