Dynamisez vos sites web avec JavaScript !

40 heures Moyenne

Mis à jour le 16/04/2019

Les formulaires

Après l'étude des événements, il est temps de passer aux formulaires ! Ici commence l'interaction avec l'utilisateur grâce aux nombreuses propriétés et méthodes dont sont dotés les éléments HTML utilisés dans les formulaires.

Il s'agit cette fois d'un très court chapitre, cela vous changera un peu du bourrage de crâne habituel !

Les propriétés

Les formulaires sont simples à utiliser, cependant il faut d'abord mémoriser quelques propriétés de base.
Comme vous le savez déjà, il est possible d'accéder à n'importe quelle propriété d'un élément HTML juste en tapant son nom, il en va donc de même pour des propriétés spécifiques aux éléments d'un formulaire comme `value`, `disabled`, `checked`, etc. Nous allons voir ici comment utiliser ces propriétés spécifiques aux formulaires.

Une propriété classique : `value`

Commençons par la propriété la plus connue et la plus utilisée : `value` ! Pour ceux qui ne se souviennent pas, cette propriété permet de définir une valeur pour différents éléments d'un formulaire comme les `<input>`, les `<button>`, etc. Son fonctionnement est simple comme bonjour, on lui assigne une valeur (une chaîne de caractères ou un nombre qui sera alors converti

```
<script>
    var text = document.getElementById('text');

    text.addEventListener('focus', function(e) {
        e.target.value = "Vous avez le focus !";
    });

    text.addEventListener('blur', function(e) {
        e.target.value = "Vous n'avez pas le focus !";
    });
</script>
```

Essayer le code

Alors par contre, une petite précision ! Cette propriété s'utilise aussi avec un élément `<textarea>` ! En effet, en HTML, on prend souvent l'habitude de mettre du texte dans un `<textarea>` en écrivant :

html

```
<textarea>Et voilà du texte !</textarea>
```

Du coup, en JavaScript, on est souvent tenté d'utiliser `innerHTML` pour récupérer le contenu de notre `<textarea>`, cependant cela ne fonctionne pas : il faut bien utiliser `value` à la place !

Les booléens avec `disabled`, `checked` et `readonly`

Contrairement à la propriété `value`, les trois propriétés `disabled`, `checked` et `readonly` ne s'utilisent pas de la même manière qu'en HTML où il vous suffit d'écrire, par exemple, `<input type="text" disabled="disabled" />` pour désactiver un champ de texte. En JavaScript, ces trois propriétés deviennent booléennes. Ainsi, il vous suffit de faire comme ceci pour désactiver un champ de texte :

html

```
<input id="text" type="text" />

<script>
    var text = document.getElementById('text');

    text.disabled = true;
</script>
```

Il n'est probablement pas nécessaire de vous expliquer comment fonctionne la propriété `checked` avec une checkbox, il suffit d'opérer de la même manière qu'avec la propriété `disabled`. En revanche, mieux vaut détailler son utilisation avec les boutons de type radio. Chaque bouton radio coché se verra attribuer la valeur `true` à sa propriété `checked`, il va donc nous falloir utiliser une

```
<label><input type="radio" name="check" value="3" /> Case n°3</label><br />
<label><input type="radio" name="check" value="4" /> Case n°4</label>
<br /><br />
<input type="button" value="Afficher la case cochée" onclick="check();" />

<script>
    function check() {
        var inputs = document.getElementsByTagName('input'),
            inputsLength = inputs.length;

        for (var i = 0; i < inputsLength; i++) {
            if (inputs[i].type === 'radio' && inputs[i].checked) {
                alert('La case cochée est la n°' + inputs[i].value);
            }
        }
    }
</script>
```

Essayer le code

L'intérêt de cet exemple était de vous présenter l'utilisation de la propriété `checked`, sachez cependant qu'il est possible de simplifier ce code grâce à la méthode `querySelectorAll()` :

javascript

```
function check() {
    var inputs = document.querySelectorAll('input[type=radio]:checked'),
        inputsLength = inputs.length;

    for (var i = 0; i < inputsLength; i++) {
        alert('La case cochée est la n°' + inputs[i].value);
    }
}
```

Toutes les vérifications concernant le type du champ et le fait qu'il soit coché ou non sont faites au niveau de `querySelectorAll()`, on peut ainsi supprimer l'ancienne condition.

Les listes déroulantes avec `selectedIndex` et `options`

Les listes déroulantes possèdent elles aussi leurs propres propriétés. Nous allons en retenir seulement deux parmi toutes celles qui existent : `selectedIndex`, qui nous donne l'index (l'identifiant) de la valeur sélectionnée, et `options` qui liste dans un tableau les éléments `<option>` de notre liste déroulante. Leur principe de fonctionnement est on ne peut plus classique :

html

```
<select id="list">
    <option>Sélectionnez votre sexe</option>
```

```
9
10      list.addEventListener('change', function() {
11
12          // On affiche le contenu de l'élément <option> ciblé par la propriété selectedIndex
13          alert(list.options[list.selectedIndex].innerHTML);
14
15      });
16  </script>
```

Essayer le code

> Dans le cadre d'un `<select>` multiple, la propriété `selectedIndex` retourne l'index du premier élément sélectionné.

Les méthodes et un retour sur quelques événements

Les formulaires ne possèdent pas uniquement des propriétés, ils possèdent également des méthodes dont certaines sont bien pratiques ! Tout en abordant leur utilisation, nous en profiterons pour revenir sur certains événements étudiés au chapitre précédent.

Les méthodes spécifiques à l'élément `<form>`

Un formulaire, ou plus exactement l'élément `<form>`, possède deux méthodes intéressantes. La première, `submit()`, permet d'effectuer l'envoi d'un formulaire sans l'intervention de l'utilisateur. La deuxième, `reset()`, permet de réinitialiser tous les champs d'un formulaire.
Si vous êtes un habitué des formulaires HTML, vous aurez deviné que ces deux méthodes ont le même rôle que les éléments `<input>` de type `submit` ou `reset`.

L'utilisation de ces deux méthodes est simple comme bonjour, il vous suffit juste de les appeler sans aucun paramètre (elles n'en ont pas) et c'est fini :

javascript

```
1  var element = document.getElementById('un_id_de_formulaire');
2
3  element.submit(); // Le formulaire est expédié
4  element.reset();  // Le formulaire est réinitialisé
```

Maintenant revenons sur deux événements : `submit` et `reset`, encore les mêmes noms ! Il n'y a sûrement pas besoin de vous expliquer quand l'un et l'autre se déclenchent, cela paraît évident. Cependant, il est important de préciser une chose : envoyer un formulaire avec la méthode `submit()` du JavaScript ne déclenchera jamais l'événement `submit` ! Mais dans le doute, voici un exemple complet dans le cas où vous n'auriez pas tout compris :

```
    <input type="reset" value="Reset !" />
</form>

<script>
    var myForm = document.getElementById('myForm');

    myForm.addEventListener('submit', function(e) {
        alert('Vous avez envoyé le formulaire !\n\nMais celui-ci a été bloqué pour que vous ne changiez pas de page.');
        e.preventDefault();
    });

    myForm.addEventListener('reset', function(e) {
        alert('Vous avez réinitialisé le formulaire !');
    });
</script>
```

Essayer le code

La gestion du focus et de la sélection

Vous vous souvenez des événements pour détecter l'activation ou la désactivation du focus sur un élément ? Eh bien il existe aussi deux méthodes, `focus()` et `blur()`, permettant respectivement de donner et retirer le focus à un élément. Leur utilisation est très simple :

html

```
<input id="text" type="text" value="Entrez un texte" />
<br /><br />
<input type="button" value="Donner le focus" onclick="document.getElementById('text').focus();" /><br />
<input type="button" value="Retirer le focus" onclick="document.getElementById('text').blur();" />
```

Essayer le code

Dans le même genre, il existe la méthode `select()` qui, en plus de donner le focus à l'élément, sélectionne le texte de celui-ci si cela est possible :

html

```
<input id="text" type="text" value="Entrez un texte" />
<br /><br />
<input type="button" value="Sélectionner le texte" onclick="document.getElementById('text').select();" />
```

Essayer le code

Bien sûr, cette méthode ne fonctionne que sur des champs de texte comme un `<input>` de type `text` ou bien un `<textarea>`.

Explications sur l'événement `change`

modifications sans attendre la perte de focus, il vous faudra plutôt utiliser d'autres événements du style `keyup` (et ses variantes) ou `click`, cela dépend du type d'élément vérifié.
Et, deuxième et dernier point, cet événement est bien entendu utilisable sur n'importe quel `input` dont l'état peut changer, par exemple une `checkbox` ou un `<input type="file" />`, n'allez surtout pas croire que cet événement est réservé seulement aux champs de texte !

En résumé

- La propriété `value` s'emploie sur la plupart des éléments de formulaire pour en récupérer la valeur.
- Les listes déroulantes fonctionnent différemment, puisqu'il faut d'abord récupérer l'index de l'élément sélectionné avec `selectedIndex`.
- Les méthodes `focus()` et `blur()` permettent de donner ou de retirer le focus à un élément de formulaire.
- Attention à l'événement `onchange`, car il ne fonctionne pas toujours comme son nom le suggère, en particulier pour les champs de texte.

- Q.C.M.
- Questions ouvertes
- Tout cocher et tout décocher
- Vérifier deux adresses email
- Éditer les cellules d'un tableau

J'AI TERMINÉ CE CHAPITRE ET JE PASSE AU SUIVANT

LES ÉVÉNEMENTS

MANIPULER LE CSS

Johann Pardanaud

Découvrez aussi ce cours en...

Livre PDF

OpenClassrooms

L'entreprise

Alternance

Forum

Blog

Nous rejoindre

Entreprises

Employeurs

En plus

Devenez mentor

Aide et FAQ

Conditions Générales d'Utilisation

Politique de Protection des Données Personnelles

Nous contacter

Français