Les formulaires

Les propriétés

Nous allons voir ici comment utiliser les propriétés spécifiques aux formulaires :
value, disabled, checked.

Une propriété classique : value

Commençons par la propriété la plus connue et la plus utilisée : value. Cette propriété permet de définir une valeur pour différents éléments d'un formulaire comme les <input>, les <button>, etc. On lui assigne une valeur (une chaîne de caractères ou un nombre qui sera alors converti implicitement) et elle est immédiatement affichée sur votre élément HTML. Exemple :

```
<input id="text" type="text" size="60" value="Vous n'avez pas le focus !" />

<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>
```

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 :

```
<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 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 :

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

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

    text.disabled = true;
```

```
</script>
```

Pour la propriété checked avec une checkbox, il suffit d'opérer de la même manière qu'avec la propriété disabled. Pour 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 boucle for pour vérifier quel bouton radio a été sélectionné :

```
<label><input type="radio" name="check" value="1" /> Case n°1</label><br />
<label><input type="radio" name="check" value="2" /> Case n°2</label><br />
<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>
```

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() :

```
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 le suivant :

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

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

```
    list.addEventListener('change', function() {

        // On affiche le contenu de l'élément <option> ciblé par la propriété selectedIndex
        alert(list.options[list.selectedIndex].innerHTML);

    });
</script>
```

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.

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.
Ces deux méthodes ont le même rôle que les éléments <input> de type submit ou reset.

Pour utiliser ces deux méthode il suffit juste de les appeler sans aucun paramètre (elles n'en ont pas) :

```
var element = document.getElementById('un_id_de_formulaire');

element.submit(); // Le formulaire est expédié
element.reset();  // Le formulaire est réinitialisé
```

On peut utiliser les évènements submit et reset afin de traiter un submit ou un reset. Il est important de préciser que la méthode submit() du JavaScript ne déclenchera jamais l'événement submi

voici un exemple complet :

```
<form id="myForm">
    <input type="text" value="Entrez un texte" />
    <br /><br />
    <input type="submit" value="Submit !" />
    <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>
```

La gestion du focus et de la sélection

Comme pour les événements permettant de détecter l'activation ou la désactivation du focus sur un élément, il existe aussi deux méthodes, focus() et blur(), permettant respectivement de donner et retirer le focus à un élément. Leur utilisation est très simple :

```
<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();" />
```

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 :

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

Cette méthode ne fonctionne que sur des champs de texte comme un <input> de type text ou bien un <textarea>.