JS et les formulaires

Dr N. BAME

Introduction

Outre les événements, les formulaires permettent l'interaction avec l'utilisateur grâce aux nombreuses propriétés et méthodes dont sont dotés les éléments HTML utilisés dans les formulaires.

Les propriétés

- 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.

La propriété classique **value**

- Cette propriété permet de définir une valeur pour différents éléments d'un formulaire comme les <input>, les <button>, etc.
- Son fonctionnement est simple,

  on lui assigne une valeur et elle est immédiatement affichée sur l'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){
    var target=e.target;
    target.value="vous avez le focus !";
    target.style.backgroundColor='rgb(200,150,190)';
    },true);
    text.addEventListener("blur", function(e){
    var target=e.target;
    target.value="vous n'avez plus le focus !";
    target.style.backgroundColor='rgb(100,200,150)';
    },true);
  </script>
```

La propriété classique value

- 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>

Exemple

```
<textarea id='txta'>
    ceci est une zone de textarea
</textarea>
<script>
    var txta=document.getElementById('txta');
    txta.addEventListener('click', function(e){
    var target=e.target;
    alert(target.value);
    target.value="Le  contenu a changé après le clic!";
},true);
</script>
```

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 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>
```

Les booléens avec disabled, checked et readonly

- Pour ce qui concerne 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 boucle for pour vérifier quel bouton radio a été sélectionné :

Les booléens avec disabled, checked et readonly

```
<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>
```

Les listes déroulantes avec selectedIndex et options

- Les listes déroulantes possèdent elles aussi leurs propres propriétés.
- Parmi celles qui existent :
  - **selectedIndex**, donne l'index (l'identifiant) de la valeur sélectionnée,
  - **options** qui liste dans un tableau les éléments <option> de la liste déroulante.
- Leur principe de fonctionnement est un plus classique :

Les listes déroulantes

```
<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);

  }, true);
</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 pratiques.

- Tout en abordant leur utilisation, nous en profiterons pour revenir sur certains événements

Les méthodes spécifiques à l'élément <form>

- Un formulaire, ou plus exactement l'élément <form>, possède deux méthodes intéressantes.
- submit(), permet d'effectuer l'envoi d'un formulaire sans l'intervention de l'utilisateur.
- 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
- L'utilisation de ces deux méthodes est simple, il suffit juste de les appeler sans aucun paramètre :

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

Les méthodes spécifiques à l'élément <form>

- Il est important de préciser une chose : envoyer un formulaire avec la méthode submit() du Javascript ne déclenchera jamais l'événement submit!

Exemple

```
<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! \nMais celui
    e.preventDefault();}, true);
    myForm.addEventListener("reset", function(e){
    alert("Vous avez réinitialisé le formulaire !");
    },true);
</script>
```

La gestion du focus et de la sélection

- Pour **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.

Exemple

```
<input id="text3" type="text" value="Entrez un texte"/>
<br/><br/>
<input type="button" value="Donner le focus"
       onclick="document.getElementById('text3').focus();" />
<input type="button" value="Retirer le focus"
       onclick="document.getElementById('text3').blur();" />
```

La gestion du focus et de la sélection

- 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="text4" type="text" value="Entrez un texte"/>
<br/><br/>
<input type="button" value="Sélectionner le texte"
       onclick="document.getElementById('text4').select();" />
```

- 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

- Il est important de revenir sur cet événement afin de clarifier quelques petits problèmes que vous pourrez rencontrer en l'utilisant.
- Tout d'abord, il est bon de savoir que cet événement attend que l'élément auquel il est attaché **perde** le focus avant de se déclencher s'il y a eu modification du contenu de l'élément.

- Si vous souhaitez vérifier l'état d'un input à chacune de ses 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.

- Cet événement est utilisable sur n'importe quel input dont l'état peut changer,
  - par exemple une checkbox ou un <input type="file" />

TP

- Reprendre le formulaire d'inscription d'un étudiant (voir TP PHP)
- Ecrire un code javascript qui impose à l'utilisateur à renseigner le numéro d'étudiant avant de saisir son nom : gérer les événements focus sur les deux éléments concernés
- Avec l'évt focus sur le numéro d'étudiant pré-remplir ce champ avec l'année en cours (créer un nouveau objet detype Date et utiliser la méthode getFullYear
- Ecrire un code javascript qui efface et/ou désactive le champ nom mari lors le sexe de l'étudiant est M.