Personnalisation d'un formulaire XHTML en Javascript et CSS

par Josselin Willette (Page d'accueil)

Date de publication : 19/11/2007

Dernière mise à jour :

Ce document a pour but de vous expliquer comment personnaliser le moindre élément des formulaires (X)HTML.

I - Introduction

N'aviez-vous jamais rêvé de personnaliser intégralement vos formulaires (X)HTML, du simple champ texte en passant par les boutons radio, les listes déroulantes et même les champs *file* ?
Cet article va vous montrer une des nombreuses (pas si nombreuses) méthodes de réaliser ce tour de force.
Pour une meilleure compatibilité avec les différents navigateurs et différentes configurations, nous allons essayer de limiter le Javascript au maximum, bien qu'il soit nécessaire pour certains éléments. Aussi, ces derniers ne fonctionneront pas correctement si l'internaute désactive Javascript.
Rassurons-nous, avec l'avènement du Web 2 et ses sites qui tendent vers une utilisation parfois abusive de l'AJAX, peu d'internautes désactivent Javascript sous peine de ne plus pouvoir naviguer sur de nombreux sites.
De plus, je vous conseille fortement d'utiliser un **DOCTYPE** complet pour vos développements, ce qui gommera les différences d'interprétation des propriétés CSS utilisées dans cet article entre navigateurs.

II - Les inputs

II-A - text, password

Nous allons commencer simple avec la personnalisation des champs de type *text* et *password*.
Tout d'abord nous nous contenterons d'habiller les champs :

(X)HTML

```
<fieldset id="conteneurInput">
    <label for="login">Login :</label>
    <input type="text" name="login" id="login" />
    <label for="password">Mot de passe :</label>
    <input type="password" name="password" id="password" />
</fieldset>
```

CSS

```
label
{
    background      : #fc6;
    padding-right   : 5px;
    margin-right    : 3px;
}

#conteneurInput input
{
    border          : 1px solid #999;
    background      : #def;
}
```

Voir le résultat
Ce qui est bien, mais pas top.
Pour donner plus d'allure à nos champs, arrondir les angles serait parfait. Pour cela, il nous faut créer quatre images qui nous serviront de coins.
En outre, une modification conséquente du code s'avère obligatoire :

(X)HTML

```
<fieldset id="conteneurInput">
    <label for="login">Login :</label>
    <div>
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
```

(X)HTML

```
        <input type="text" name="login" id="login" />
    </div>
    <label for="password">Mot de passe :</label>
    <div>
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <input type="password" name="password" id="password" />
    </div>
</fieldset>
```

CSS

```
label
{
    display         : block;
    padding-right   : 5px;
    float           : left;
    background      : #fc6;
    margin-right    : 3px;
}

#conteneurInput div
{
    background      : #def;
    position        : relative;
    border          : 1px solid #999;
    text-align      : center;
    float           : left;
}

#conteneurInput input
{
    background      : none;
    border          : 0;
    padding         : 0 6px;
    width           : 130px;
}

span.top-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    left            : -1px;
    background      : url("top-left.gif");
}

span.bottom-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    left            : -1px;
    background      : url("bottom-left.gif");
}

span.bottom-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
```

CSS
```
    right           : -1px;
    background      : url("bottom-right.gif");
}

span.top-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    right           : -1px;
    background      : url("top-right.gif");
}
```

Voir le résultat
C'est bien mieux !

II-B - radio

La personnalisation des boutons radio (ou radiobuttons), quant à elle, requiert une petite partie de Javascript, celle qui permettra de permuter les images qu'on va utiliser pour présenter nos boutons.
Tout d'abord, le code HTML :

(X)HTML
```
<fieldset id="radio">
    <p>
        <label for="mlle">Mlle :</label>
        <input type="radio" id="mlle" name="civilite2" onclick="turnImgRadio(this)" />
        <img src="radio1.gif" id="img_radio_mlle" alt="Bouton radio" />
    </p>
    <p>
        <label for="mme">Mme :</label>
        <input type="radio" id="mme" name="civilite2" onclick="turnImgRadio(this)" />
        <img src="radio1.gif" id="img_radio_mme" alt="Bouton radio" />
    </p>
    <p>
        <label for="mr">Mr :</label>
        <input type="radio" id="mr" name="civilite2" onclick="turnImgRadio(this)" />
        <img src="radio1.gif" id="img_radio_mr" alt="Bouton radio" />
    </p>
</fieldset>
```

Comme on s'en doute à la lecture de ce code, l'image présente sous le bouton radio sera celle qui le remplacera et l'attribut *onclick* du bouton radio servira à permuter cette image.
Comment faire apparaître l'image à la place du bouton radio ? En le masquant bien sûr.

CSS
```
label
{
    padding-right   : 3px;
    margin-right    : 3px;
    background      : #fc6;
    vertical-align  : top;
}

#conteneurRadio p
{
    position        : relative;
    float           : left;
    margin          : 0;
```

CSS
```
}

#conteneurRadio input
{
    opacity         : 0; /* pour !IE */
    filter          : alpha(opacity=0); /* pour IE */
    width           : 20px;
    height          : 20px;
    position        : absolute;
    right           : 0;
    top             : 0;
}
```

Voilà nos boutons radio invisibles, mais actifs !
Autant le dire tout de suite, ce code CSS n'est pas valide W3C. Vous pouvez à la rigueur mettre le filtre Microsoft dans un commentaire conditionnel, mais il faudra attendre le CSS3 pour voir apparaître la propriété **opacity**.
Il ne reste plus qu'à créer notre fonction Javascript qui va simplement modifier l'image sélectionnée :

Javascript
```
function turnImgRadio(objRadio)
{
    var t_img = document.getElementById('conteneurRadio').getElementsByTagName('img');

    for (var i = 0; i < t_img.length; i++)
    {
        t_img[i].src = 'radio1.gif';
    }

    var img = document.getElementById('img_radio_' + objRadio.id);
    img.src = 'radio2.gif';
}
```

Voir le résultat

II-C - checkbox

Pas de grands changements avec le paragraphe précédent pour personnaliser les cases à cocher (ou checkboxes), si ce n'est la fonction Javascript qui perd quelques lignes.

(X)HTML
```
<fieldset id="conteneurCheckbox">
    <p>
        <label for="beau">Beau :</label>
        <input type="checkbox" id="beau" name="qualite" onclick="turnImgCheck(this)" />
        <img src="check1.gif" id="img_check_beau" alt="Checkbox" />
    </p>
    <p>
        <label for="fort">Fort :</label>
        <input type="checkbox" id="fort" name="qualite" onclick="turnImgCheck(this)" />
        <img src="check1.gif" id="img_check_fort" alt="Checkbox" />
    </p>
    <p>
        <label for="intelligent">Intelligent :</label>
        <input type="checkbox" id="intelligent" name="qualite" onclick="turnImgCheck(this)" />
        <img src="check1.gif" id="img_check_intelligent" alt="Checkbox" />
    </p>
</fieldset>
```

CSS

```
label
{
    padding-right   : 3px;
    margin-right    : 3px;
    background      : #fc6;
    vertical-align  : top;
}

#conteneurCheckbox p
{
    position        : relative;
    float           : left;
    margin          : 0;
}

#conteneurCheckbox input
{
    opacity         : 0; /* pour !IE */
    filter          : alpha(opacity=0); /* pour IE */
    width           : 20px;
    height          : 20px;
    position        : absolute;
    right           : 0;
    top             : 0;
}
```

Javascript

```
function turnImgCheck(objCheck)
{
    var img = document.getElementById('img_check_' + objCheck.id);
    var t = img.src.split('/');
    img.src = (t[t.length-1] == 'check2.gif') ? 'check1.gif' : 'check2.gif';
}
```

Voir le résultat

II-D - button, submit, reset

Rien de bien compliqué dans la personnalisation des boutons, comme vous pourrez en juger :

(X)HTML

```
<fieldset id="conteneurButton">
    <p>
        <input type="button" name="envoyer" value="Envoyer" class="ok" />
        <input type="reset" name="effacer" value="Effacer" class="nok" />
    </p>
</fieldset>
```

CSS

```
#conteneurButton input.ok
{
    border          : 1px solid #def;
    background      : #6c3;
    cursor          : pointer;
    padding         : 3px 30px;
    margin          : 0 10px;
}

#conteneurButton input.nok
```

CSS
```
{
    border          : 1px solid #def;
    background      : #e6484d;
    cursor          : pointer;
    padding         : 3px 30px;
    margin          : 0 10px;
}
```

Voir le résultat

II-E - file

Est-ce vraiment possible de personnaliser un input de type *file* ?
Et même modifier ce maudit texte "Parcourir..." ?
Cette opération est certes plus complexe à mettre en oeuvre que celle du paragraphe précédent, mais elle repose sur la même technique que celle des checkboxes et radiobuttons : masquer l'input et non l'émuler.

(X)HTML
```
<fieldset id="conteneurFile">
    <div id="divFile">
        <input type="text" id="input_text_file" class="inputText" readonly="readonly" />

  <input type="file" onmousedown="return false" onkeydown="return false" class="inputFile" onchange="document.get
        <span>Ajouter...</span>
    </div>
</fieldset>
```

CSS
```
#file #divFile
{
    position        : relative;
    width           : 250px;
    text-align      : right;
}

#conteneurFile .inputFile
{
    opacity         : 0; /* pour !IE */
    filter          : alpha(opacity=0); /* pour IE */
    position        : absolute;
    right           : 0;
    top             : 0;
}

#conteneurFile .inputText
{
    border          : 1px solid #999;
    padding         : 0px 6px;
    background      : #def;
    width           : 130px;
}

#conteneurFile span
{
    border          : 1px solid #def;
    background      : #ffc;
    width           : 80px;
    padding         : 1px 10px;
}
```

Voir le résultat

Les évènements *onmousedown* et *onkeydown* sur le champ *file* interdisent la saisie de texte dans ce champ invisible, ça pourrait perturber l'internaute de voir qu'il saisit du texte qui ne s'affiche pas.
L'évènement *onchange*, quant à lui, fait juste apparaître le nom du fichier dans notre champ texte, pour que l'internaute ne soit pas déboussolé.
Pour positionner correctement le bouton "Parcourir..." invisible par rapport au texte de substitution, vous pouvez modifier l'opacité pour voir où vous en êtes.
À partir de ce code, on peut évidemment imaginer plein de façons différentes de personnaliser notre input *file*, comme mettre une image à la place du texte "Ajouter...", modifier l'apparence du champ texte avec des angles arrondis selon la méthode du premier paragraphe de cet article, etc.
À vous d'imaginer le meilleur.

III - Le textarea

La méthode de personnalisation d'un textarea est strictement similaire à celle utilisée pour les champs texte.
Voici donc le code pour avoir des angles normaux :

(X)HTML

```
<fieldset id="conteneurTextarea">
    <label for="message">Message :</label>
    <textarea name="message" id="message" cols="30" rows="5"></textarea>
</fieldset>
```

CSS

```
label
{
    background      : #fc6;
    padding-right   : 5px;
    margin-right    : 3px;
}

#conteneurTextarea textarea
{
    border          : 1px solid #999;
    background      : #def;
    vertical-align  : top;
    overflow        : auto;
}
```

Voir le résultat

Et le code pour avoir des angles arrondis, toujours avec nos quatre images :

(X)HTML

```
<fieldset id="conteneurTextarea">
    <label for="message">Message :</label>
    <div>
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <textarea name="message" id="message" cols="30" rows="5"></textarea>
    </div>
</fieldset>
```

CSS

```
label
```

CSS

```
{
    display         : block;
    padding-right   : 5px;
    float           : left;
    background      : #fc6;
    margin-right    : 3px;
}

#conteneurTextarea div
{
    background      : #def;
    position        : relative;
    border          : 1px solid #999;
    text-align      : center;
    float           : left;
}

#conteneurTextarea textarea
{
    background      : none;
    border          : 0;
    padding         : 0 6px;
    width           : 300px;
    overflow        : auto;
}

span.top-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    left            : -1px;
    background      : url("top-left.gif");
}

span.bottom-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    left            : -1px;
    background      : url("bottom-left.gif");
}

span.bottom-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    right           : -1px;
    background      : url("bottom-right.gif");
}

span.top-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    right           : -1px;
    background      : url("top-right.gif");
}
```

Voir le résultat

IV - Le select

Contrairement aux autres éléments qui n'avaient quasiment pas besoin de Javascript, pour pouvoir personnaliser un select, au contraire, il est indispensable.
En effet, pour afficher une liste sur un clic, on est forcé d'utiliser les évènements Javascript.
Dans l'exemple ci-dessous, les angles seront arrondis à la manière décrite dans le premier paragraphe, vous pouvez évidemment vous en dispenser ou même l'améliorer.

(X)HTML

```
<fieldset id="conteneurSelect">
    <label>Pays :</label>
    <div class="inputsSelect">
        <span class="top-left"> </span>
        <span class="bottom-left"> </span>
        <span class="bottom-right"> </span>
        <span class="top-right"> </span>
        <p class="selects" onclick="showHideSelect('listeSelect1')">-- Choisissez --</p>
        <ul id="listeSelect1">

  <li><a href="javascript:void(0)" onclick="validAndHide('', this, 'countryCode', 'select1')">--
 Choisissez --</a></li>

  <li><a href="javascript:void(0)" onclick="validAndHide('FR', this, 'countryCode', 'select1')">France</a></li>

  <li><a href="javascript:void(0)" onclick="validAndHide('DE', this, 'countryCode', 'select1')">Allemagne</a></li

  <li><a href="javascript:void(0)" onclick="validAndHide('IT', this, 'countryCode', 'select1')">Italie</a></li>

  <li><a href="javascript:void(0)" onclick="validAndHide('VG', this, 'countryCode', 'select1')">Saint-
Vincent-et-les Grenadines</a></li>
        </ul>
    </div>
    <input type="hidden" name="countryCode" id="countryCode" />
</fieldset>
```

Le input de type *hidden* sert à récupérer la valeur sélectionnée pour pouvoir l'envoyer à la soumission du formulaire.

CSS

```
label
    {
    display         : block;
    padding-right   : 5px;
    float           : left;
    background      : #fc6;
    margin-right    : 3px;
}

p
{
    margin          : 0;
}

#conteneurSelect .inputsSelect
{
    background      : #def url("fleche.gif") right center no-repeat;
    position        : relative;
    border          : 1px solid #999;
    text-align      : center;
    float           : left;
}

.inputsSelect .selects
```

CSS

```
{
    padding         : 3px 14px 3px 3px;
    font            : normal 12px verdana;
    cursor          : default;
    width           : 95px;
    white-space     : nowrap;
    overflow        : hidden;
}

.inputsSelect ul
{
    position        : absolute;
    text-align      : left;
    border          : 1px solid #999;
    white-space     : nowrap;
    font            : normal 12px verdana;
    padding         : 5px;
    display         : none;
    background      : #eff7ff;
    z-index         : 100;
    list-style      : none;
    margin          : 0;
}

.inputsSelect ul li a
{
    display         : block;
    cursor          : default;
    color           : #000;
    text-decoration : none;
    background      : #eff7ff;
    width           : 100%;
}

.inputsSelect ul li a:hover
{
    color           : #fff;
    background      : #093e6d;
}

span.top-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    left            : -1px;
    background      : url("top-left.gif");
}

span.bottom-left
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    left            : -1px;
    background      : url("bottom-left.gif");
}

span.bottom-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    bottom          : -1px;
    right           : -1px;
    background      : url("bottom-right.gif");
```

CSS
```
}

span.top-right
{
    position        : absolute;
    width           : 4px;
    height          : 4px;
    overflow        : hidden;
    top             : -1px;
    right           : -1px;
    background      : url("top-right.gif");
}
```

Javascript
```
function showHideSelect(select)
{
    var objSelect = document.getElementById(select);
    objSelect.style.display = (objSelect.style.display == 'block') ? 'none' : 'block';
}

function validAndHide(txt, obj, input, select)
{
    document.getElementById(input).value = txt;
    obj.parentNode.parentNode.style.display = 'none';
    document.getElementById(select).innerHTML = obj.innerHTML;
}
```

Voir le résultat

V - Liens complémentaires

Personnalisez vos formulaires en CSS, par A. Pellegrini
Contrôlez vos formulaires avec PHP, par G. Rossolini
Toutes les questions que vous vous posez sur les formulaires en Javascript

VI - Remerciements

Un grand merci à **buchs** pour sa relecture et à l'équipe de Developpez.com qui m'a beaucoup aidé dans la conception de cet article.