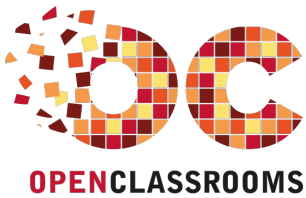
Une zForm avec Mootools

Par franckysolo



www.openclassrooms.com

Dernière mise à jour le 20/01/2012

Sommaire

Une zForm avec Mootools

Par franckysolo

Mise à jour : 20/01/2012
Difficulté : Intermédiaire

Bonjour à tous,
Nous allons nous entraîner à utiliser des objets du framework Mootools grâce à la création d'un ZForm avec une prévisualisation *via* Ajax. Ce mini-tuto vient en complément de « Découvrez mootools, ou bien commencer avec Javascript ».

Prérequis

- Avoir des notions de programmation objet
- Connaître le langage JavaScript
- Connaître le langage PHP
- Savoir utiliser les expressions régulières
- Avoir lu le tutoriel : « Découvrez mootools, ou bien commencer avec Javascript »
- Sujet traité en JavaScript natif : Insertion de balises dans une zone de texte

Le *débugger*

Pour pouvoir corriger vos scripts avec Mootools, je vous conseille d'utiliser un *débugger* du type Firebug.
Sommaire du tutoriel :

Le ZForm

Téléchargez le framework Mootools en version core et more. Une fois que c'est fait, rassemblez les deux fichiers en un seul, appelé *mootools.js*, et placez-le dans un dossier public et un sous-dossier `js`.

Les différents éléments

Le ZForm va nous permettre d'insérer des balises destinées à mettre en forme un texte, à ajouter des images, des titres et autres.
Notre ZForm va posséder plusieurs éléments HTML :

- le champ de texte : le textarea ;
- les boutons simples de mise en forme ;
- les boutons avec choix de mise en forme ;
- les listes de choix pour la mise en forme du texte.

Les boutons simples

Les boutons simples vont insérer, lors d'un clic, une paire de balises de mise en forme dans le champ de texte.

Par exemple, pour la propriété « texte en gras », lors du clic sur le bouton gras, notre ZForm va afficher à l'intérieur du champ de texte :

Code : Autre

```
<gras>texte</gras>
```

Voici les listes des boutons simples de mise en forme dont nous aurons besoin :

- gras ;
- italique ;
- barre ;
- souligne ;
- marge ;
- erreur ;
- infos ;
- question ;
- attention.

Voici les images que nous allons utiliser pour le TP :

Les boutons avec choix

La liste des boutons avec choix :

- lien ;
- lien vers une ancre ;
- ancre ;
- puce normale ;
- puce numérique ;
- tableau ;
- citation ;
- image ;
- clip.

Je vous donne aussi les images correspondantes, je suis sympa :

Les listes de choix

Ces boutons vont, lors du survol, faire apparaître un menu de choix à cliquer ; ils seront utiles pour les couleurs, polices d'écriture…
Les boutons à menu :

- police ;
- taille du texte ;

- position ;
- flottant ;
- couleur ;
- avertissement : contiendra les boutons simples (question, erreur, infos, attention) ;
- titre.

Les images pour les listes de choix :

Le formulaire html

Code : HTML - index.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
/>
<title>ZForm</title>

<script type="text/javascript" src="public/js/mootools.js"></script>
<script type="text/javascript" src="public/js/zform.js"></script>

<link href="public/css/zform.css"  rel="stylesheet" type="text/css"
media="all" />

<script type="text/javascript">

window.addEvent('domready', function () {
    var form = new ZForm();
     form.listeningEvent();
    });
</script>

</head>
<body>
    <div id="main">
        <form id="zform" action="" method="post">
            <fieldset><legend>ZForm</legend>

                <!-- Boutons simples -->
                <img data-bt="gras" class="splButtons bt-code"
src="public/images/boutons/bt-gras.gif"  alt="Gras" title="Gras"  />
                <img data-bt="italique" class="splButtons bt-code"
src="public/images/boutons/bt-italique-black.gif" alt="Itlalique"
title="Italique"/>
                <img data-bt="souligne" class="splButtons bt-code"
src="public/images/boutons/bt-souligne.gif" alt="Souligné"
title="Souligner"/>
                <img data-bt="barre" class="splButtons bt-code"
src="public/images/boutons/bt-surligne.gif" alt="Barré"
title="Barrer" />
                <img data-bt="marge" class="splButtons bt-code"
src="public/images/boutons/bt-marge.gif"  alt="Marge" title="Marge"
/>
                <img data-bt="gras" class="splButtons bt-code"
src="public/images/boutons/bt-gras.gif"  alt="Gras" title="Gras"  />

                <!-- Boutons choix -->
                <img data-bt="liste" class="dlgButtons bt-code"
src="public/images/boutons/bt-puce.gif" alt="Listes" />
                <img data-bt="listnum" class="dlgButtons bt-code"
src="public/images/boutons/bt-pucenum.gif"  alt="Listes Numériques"
/>
```

```
                <img data-bt="tableau" class="dlgButtons bt-code"
src="public/images/boutons/bt-tableau.gif"  alt="Tableau" />
                <img data-bt="url" class="dlgButtons bt-code"
src="public/images/boutons/bt-lien.gif" alt="lien" />
                <img data-bt="ancre" class="dlgButtons bt-code"
src="public/images/boutons/bt-ancre.gif" alt="Ancre" />
                <img data-bt="urlancre" class="dlgButtons bt-code"
src="public/images/boutons/bt-urlancre.gif" alt="Lien vers une
ancre" />
                <img data-bt="citation" class="dlgButtons bt-code"
src="public/images/boutons/bt-citation.gif" alt="Citation" />
                <img data-bt="image" class="dlgButtons bt-code"
src="public/images/boutons/bt-image.gif"  alt="Images" />
                <img data-bt="clip" class="dlgButtons bt-code"
src="public/images/boutons/bt-clip.gif"  alt="Clip Video"  />

               <!-- Boutons listes -->
                <img data-bt="police" class="bt-code listButtons"
src="public/images/boutons/bt-font.gif"  alt="Police" title="Police"
/>
                <div class="sliderBox">
                    <a class="slider-link" rel="Arial">Arial</a>
                    <a class="slider-link" rel="Comic Sans MS">Comic
Sans MS</a>
                    <a class="slider-link" rel="Verdana">Verdana</a>
                    <a class="slider-link" rel="Times New Roman"
>Times New Roman</a>
                    <a class="slider-link" rel="Rockwell Extra
Bold">Rockwell Extra Bold</a>
                </div>

                <img data-bt="taille" class="bt-code listButtons"
src="public/images/boutons/bt-size.gif"  alt="Taille de la police"
title="Taille de la police"  />
                <div class="sliderBox">
                    <a class="slider-link" rel="xx-small">Très très
petit</a>
                    <a class="slider-link" rel="x-small">Très
petit</a>
                    <a class="slider-link" rel="small">Petit</a>
                    <a class="slider-link" rel="medium">Normal</a>
                    <a class="slider-link" rel="large">Grand</a>
                    <a class="slider-link" rel="x-large">Très
grand</a>
                    <a class="slider-link" rel="xx-large">Très très
grand</a>

                </div>

                <img data-bt="couleur" class="bt-code listButtons"
src="public/images/boutons/bt-color.gif"  alt="Couleur de la police"
title="Couleur de la Police" />
                <div class="sliderBox">
                    <a class="slider-link bgWhite"
rel="#FFFFFF"> </a>
                    <a class="slider-link bgGray"
rel="#CCCCCC"> </a>
                    <a class="slider-link bgRed"
rel="#FF0000"> </a>
                    <a class="slider-link bgOrange"
rel="#FF6600"> </a>
                    <a class="slider-link bgYellow"
rel="#FFCC00"> </a>
                    <a class="slider-link bgGreen"
rel="#33CC00"> </a>
                    <a class="slider-link bgBlue"
rel="#0000CC"> </a>
                    <a class="slider-link bgMarine"
rel="#000033"> </a>
                    <a class="slider-link bgBlack"
```

```
        rel="#000000"> </a>
                    </div>

                    <img data-bt="titre" class="bt-code listButtons"
        src="public/images/boutons/bt-titre.gif"  alt="Titre" title="Titre"
        />
                    <div class="sliderBox">
                        <a class="slider-link" rel="h1">Titre 1</a>
                        <a class="slider-link" rel="h2">Titre 2</a>
                        <a class="slider-link" rel="h3">Titre 3</a>
                    </div>

                    <img data-bt="position" class="bt-code listButtons"
        src="public/images/boutons/bt-position.gif"  alt="Position du texte"
        />
                    <div class="sliderBox">
                        <a class="slider-link" rel="gauche">Gauche</a>
                        <a class="slider-link" rel="center">Centrer</a>
                        <a class="slider-link" rel="droite">Droite</a>
                        <a class="slider-link"
        rel="justifier">Justifier</a>
                    </div>

                    <img data-bt="flottant" class="bt-code listButtons"
        src="public/images/boutons/bt-float.gif"  alt="Couleur de la police"
        />
                    <div class="sliderBox">
                        <a class="slider-link" rel="gauche">Flottant à
        gauche</a>
                        <a class="slider-link" rel="droit">Flottant à
        droite</a>
                    </div>

                    <img data-bt="avert" class="bt-code listButtons"
        src="public/images/boutons/bt-avert.gif"  alt="Avertissements" />
                    <div class="sliderBox">
                        <img data-bt="question" class="splButtons bt-
        code" src="public/images/boutons/bt-question.gif" alt="Question" />
                        <img data-bt="attention" class="splButtons bt-
        code" src="public/images/boutons/bt-attention.gif"  alt="Attention"
        /><br />
                        <img data-bt="infos" class="splButtons bt-code"
        src="public/images/boutons/bt-infos.gif"  alt="Infos"  />
                        <img data-bt="erreur" class="splButtons bt-code"
        src="public/images/boutons/bt-erreur.gif"  alt="Erreur" />
                    </div>

                    <div class="block">
                        <textarea id="text" name="text" cols="100"
        rows="10"></textarea>
                    </div>
                    <div id="control">
                        <input type="reset" value="effacer" />
                        <input type="submit" value="Valider" />
                    </div>
                </fieldset>
            </form>

            <div id="prev">
                <h4>Prévisualisation : </h4>
            </div>

        </div>
    </body>
    </html>
```

Une petite précision ici, nous instancions et utilisons notre objet de cette manière :

Code : HTML

```
<script type="text/javascript">
// Lorsque le dom est lu
window.addEvent('domready', function () {
    var form = new ZForm(); // On crée notre objet ZForm
    form.listeningEvent();  // On écoute les événements du ZForm
});
</script>
```

La feuille de style

Code : CSS - zform.css

```
#parser {
 margin: auto;
 width: 80%;
}

 fieldset {
 margin: auto;
 width: 70%;
}

#control {
 float: none;
 display: block;
 clear: both;
 margin-left : 3%;
}
table {
 background-color: #ccc;
 padding : 1%;
}
td {
 border : 1px solid #eee;
 background-color: #fff;
}

.title-citation {
 background-color: #eee;
 font-weight: bold;
 border: #ccc 1px solid;
 border-bottom: none;
 text-indent : 1%;
 width: 20%;
 margin:0;
 margin-top : 1%;
}
.citation {
 margin:0;
 border: #ccc 1px solid;
 padding: 1%;
}

#prev {
 border: 1px solid #999;
 margin: auto;
 margin-top: 1%;
 padding: 1%;
 min-height: 200px;
 -moz-border-radius: 10px;
 overflow : auto;
 width: 78%;
```

```
}

.bt-code {
 cursor: pointer;
 width: 30px;
 height: 30px;
 padding-left: 0;
 /*float: left;*/
}

.sliderBox {
 border: 1px solid #ccc;
 padding: 0.3%;
 background-color: #eee;
 max-width : 200px;
}

.slider-link {
 background-color: #fff;
 border: 1px solid #ccc;
 cursor: pointer;
 padding: 1%;
 margin: 1%;
 min-width: 150px;
 display : block;
}
.slider-link:hover {
 border: 1px solid #999;
}

.parser-erreur{
 background: #eee url(public/images/zcode_erreur.png) no-repeat 10px
center;
 border : 1px solid red;
 color: red;
 width: 80%;
 padding: 2%;
 margin: auto;
 padding-left: 5%;

}
.parser-attention{
 background: #eee url(public/images/zcode_attention.png) no-repeat
10px center;
 border : 1px solid orange;
 color: orange;
 width: 80%;
 padding: 2%;
 margin: auto;
 padding-left: 5%;

}
.parser-question{
 background: #eee url(public/images/zcode_question.png) no-repeat
10px center;
 border : 1px solid blue;
 color: blue;
 width: 80%;
 padding: 2%;
 margin: auto;
 padding-left: 5%;

}
.parser-infos{
 background: #eee url(public/images/zcode_info.png) no-repeat 10px
center;
 border : 1px solid green;
 color: green;
 width: 80%;
```

```
 padding: 2%;
 margin: auto;
 padding-left: 5%;
}

.block {
 float: left;
}
.bgWhite {
 background-color: white;
}
.bgBlack {
 background-color: black;
}
.bgMarine {
 background-color: #003;
}
.bgGray {
 background-color: #ccc;
}
.bgOrange {
 background-color: orange;
}
.bgBlue {
 background-color: blue;
}
.bgRed {
 background-color: red;
}
.bgGreen {
 background-color: green;
}
.bgYellow {
 background-color: yellow;
}
```

Les boutons simples

L'objet ZForm

Nous allons coder notre objet Mootools. Pour ce faire, créez un fichier nommé `zform.js` et placez-le dans `public/js/`.

Le fichier est lié entre les balises `head` de votre `index.html`.

Construction de l'objet et initialisation

À la construction de l'objet, on initialise les paramètres qui vont servir au fonctionnement du ZForm. Voici quelques lignes de code pour commencer :

Code : JavaScript - zform.js

```
var ZForm = new Class({

 Implements : Options,

 options : {
  textId : 'text', // id du textarea
  preview : 'prev', // id de la div de prévisualisation
  buttons : '.splButtons', // les boutons simples
  dlgButtons : '.dlgButtons', // les boutons ouvrant une boîte de
dialogue
```

```
    listButtons : '.listButtons', // les boutons liste ouvrant un menu
de choix
    slider : '.sliderBox', // les boîtes contenant un menu de choix
    sliderLink : '.slider-link',  // les liens du menu de choix
    closeDelay : 500 // temps avant fermeture de la boîte de menu en
millisecondes
  },
        //Constructeur
  initialize : function(options) {
          this.setOptions(options);
   this.input = $(this.options.textId); // getElementById(textId)
   this.preview = $(this.options.preview); // getElementById(preview)
   this.timer = 0;
   //Buttons
   this.buttons = $$(this.options.buttons);  // tableau de tous les
boutons simples
   this.dlgButtons = $$(this.options.dlgButtons); // tableau de tous
les boutons ouvrant une boîte de dialogue
   this.sliders = $$(this.options.slider); // tableau de toutes les
boîtes contenant un menu de choix
   this.listButtons = $$(this.options.listButtons); // tableau de
tous les boutons liste ouvrant un menu de choix
   this.sliderLink = $$(this.options.sliderLink); // tableau de tous
les liens de menu de choix
   this.closeDelay = this.options.closeDelay;
  },

  // Écoute les événements
  listeningEvent : function() {
   this.input.focus(); // On donne le focus au champ de texte
  }
});
```

Les classes Mootools implémentent des options pour assigner des paramètres par défaut, ce qui vous permet, par exemple, de changer le nom de la classe CSS, ou l'id du textarea. Dans ce cas, il faudra assigner les nouveaux paramètres lors de l'instanciation :

Code : JavaScript

```
var zform = new ZForm({textId: 'message'});
```

Le principe des boutons simples

Étude de cas

Nous allons pour l'instant étudier le principe avec le bouton simple gras.

J'ai placé mes images dans un dossier `public/images/`.

Mettons-nous à la place de l'utilisateur ; deux cas de figure se présentent :

1. L'utilisateur tape un texte, sélectionne le texte et clique sur le bouton gras pour ajouter les balises ;
2. L'utilisateur clique sur le bouton gras et tape un texte entre les balises ajoutées.

Avec ces deux cas de figure, nous allons développer notre algorithme. Nous devons :

1. Écouter les événements et identifier le bouton cliqué ;
2. Récupérer le nom de la balise à insérer ;

3. Récupérer la sélection du texte s'il y en a une ;
4. Si aucun texte n'est sélectionné, placer le curseur entre les balises ajoutées ;
5. Si un texte est sélectionné, placer les balises de part et d'autre la sélection.

Regrouper les boutons

Pour éviter d'avoir à réécrire une fonction pour chaque bouton simple, Mootools nous permet de regrouper des éléments sous une même classe. Ici, nous allons nommer notre classe CSS **`splButtons`** pour tous les boutons simples.

Code : JavaScript

```
this.buttons = $$(this.options.buttons);
// correspond à
this.buttons = $$('.splButtons');
// et nous renvoie un tableau de tous les éléments ayant une classe
CSS splButtons
```

Le bouton gras se présente sous cette forme :

Code : HTML - index.html

```
<!-- Boutons simples -->
<img data-bt="gras" class="splButtons bt-code"
src="public/images/boutons/bt-gras.gif"  alt="Gras" title="Gras"  />
```

Pour récupérer le nom de la balise, l'astuce ici est d'utiliser une nouvelle fonctionnalité HTML : l'attribut data-bt de l'image.

Nous allons aussi utiliser la classe `Elements.Forms` et ses méthodes `getSelectedText`, `insertAtCursor` et `insertAroundCursor`, ici pour repérer où se trouve notre curseur dans le champ de texte, mais aussi quelle partie du texte l'utilisateur a sélectionnée. Codons les méthodes nécessaires au fonctionnement.
Nous allons expliquer un peu la différence de comportement des méthodes d'insertion de balises :

- `insertAtcursor` insère les balises à la position du curseur. Cette méthode prend en paramètre le texte à insérer et un booléen en deuxième paramètre :
  - s'il vaut `true` : une fois les balises insérées, la sélection englobe le contenu,
  - s'il vaut `false` : une fois les balises insérées, la sélection est annulée et le curseur se positionne à la fin de la balise de fermeture ;
- `insertAroundcursor` insère les balises autour du curseur. Une fois les balises insérées, la sélection s'applique au texte entre les balises. Cette méthode prend en paramètre un tableau :
  - clé `after` : définit la balise d'ouverture,
  - clé `before` : définit la balise de fermeture,
  - clé `defaultMiddle` : le texte entre les balises.

L'insertion des tags

Voici nos deux méthodes :

- les événements sur les boutons simples : `eventButton` ;
- l'insertion des tags : `insertTag`.

N'oubliez pas de lancer l'appel à `eventButton`, dans l'écouteur d'événement (méthode `listeningEvent`).

Code : JavaScript

```
[...]
 * Events sur les boutons simples
 */
eventButtons : function() {
 //Si c'est I.E. on utilise plutôt l'event mousedown
 var event = (Browser.ie) ? 'mousedown' : 'click';
 // On écoute tous les boutons simples
 this.buttons.each(function(button){
  // Si un event est détecté
  button.addEvent(event, function(){
   //On récupère l'attribut data-bt de l'image
   var name = button.getProperty('data-bt');
   //On appelle la méthode d'insertion de tags avec les bons
paramètres et la sélection du texte
   this.insertTag('<'+name+'>', '</'+name+'>',
this.input.getSelectedText());
  }.bind(this));
 }, this);
},

//Insertion des tags
insertTag : function(startTag, endTag, select) {
 //On donne le focus à notre champ de texte
 this.input.focus();
 //S'il n'y a pas de sélection
 if(select === null || undefined === select) {
  //On insère nos tags autour du curseur
  this.input.insertAroundCursor({'before' : startTag, 'after' :
endTag});
 } else {
  //Sinon on insère nos tags au curseur
  this.input.insertAtCursor(startTag + select + endTag, false);
 }
 this.input.focus();
},
//Écoute les événements
listeningEvent : function() {
 this.input.focus();
 this.eventButtons();
}
[...]
```

Analyse du code

Le code est commenté mais je vais quand même expliquer comment cela fonctionne.
Dans la méthode `eventButton`, j'exécute une boucle sur mon tableau de boutons ; si un événement est détecté, je récupère le nom du bouton que j'assigne en paramètre à la méthode d'insertion des balises.
La méthode `insertTag` m'indique si une partie de texte a été sélectionnée ou non dans le textarea ; en fonction des cas, les balises sont insérées au curseur ou de part et d'autre de la sélection.

Une condition est posée si on utilise Internet Explorer. En effet, un bug existe avec l'event `onclick`, car lors de la sélection du bouton, le champ de texte perd le focus et les méthodes de Mootools ne retrouvent pas la position du curseur. Pour éviter ce désagrément, on utilise l'event mousedown avec Internet Explorer.

Les boutons avec choix

Avant de commencer

Étude de cas

Ici, nous allons étudier pour l'exemple le cas d'un bouton lien.
Deux cas de figure se présentent :

- l'utilisateur tape le texte du lien, sélectionne le texte et clique sur le bouton lien pour ajouter les tags, une boîte de dialogue demande d'entrer l'adresse du lien, les tags sont ajoutés de part et d'autre de texte sélectionné ;
- l'utilisateur clique sur le bouton lien, une première boîte de dialogue demande d'entrer le texte du lien et une seconde boîte de dialogue demande d'entrer l'adresse du lien, les tags sont ajoutés au curseur.

La forme du tag pourra être :

Code : Autre

```
<url value="adresse du lien">texte du lien</url>
```

Pour pouvoir récupérer les paramètres utilisateur, nous allons utiliser prompt, un pour l'adresse et un pour le texte. Si l'adresse est vide, on mettra le paramètre à `null`, idem pour le texte.
Nous allons aussi essayer d'automatiser les tâches sauf que cette fois, chaque bouton ne va pas déclencher les mêmes actions.
En effet, nous allons avoir besoin de différents types de balises :

Code : Autre

```
// 1er type : citation, URL, urlancre, ancre
<balise value="value">text</balise>

// 2e type : image
<balise>text</balise>

// Les listes
<balise-liste>
    <balise-puce>text</balise-puce>
    <balise-puce>text</balise-puce>
</balise-liste>

// Tableau
<balise-tableau>
   <balise-tr>
    <balise-td>text</balise-td>
    <balise-td>text</balise-td>
   </balise-tr>
</balise-tableau>
```

Nous allons également avoir besoin de différents types de messages pour les commandes prompt. Il va falloir réfléchir un peu plus pour ces boutons.
Nous allons associer les groupes de tags en fonction du type de balise et des boîtes de dialogue à afficher.

- Les tags tableau et liste auront un comportement unique : nous devrons donc les dissocier.
- Le tag image : il sera de la forme `<tag>nom_de_l_image</tag>`.
  - Si aucune partie de texte n'est sélectionnée, on devra afficher une boîte de dialogue demandant d'entrer le chemin de l'image.
  - Sinon la partie de texte sélectionnée s'insérera automatiquement à l'intérieur des tags.
- Les tags url, urlancre, et ancre : le comportement et type de balise sera identique sauf au niveau du texte des boîtes de dialogue.
- La balise citation aura un attribut nommé `auteur` et non `value`.

Les événements sur les boutons de choix

Comment récupérer la liste de boutons de choix ?

Notre bouton :

Code : HTML

```
<!-- Boutons choix -->
<img data-bt="url" class="dlgButtons bt-code"
src="public/images/boutons/bt-lien.gif" alt="lien" />
```

Ici, la classe CSS qui identifie les boutons de choix se nomme **dlgButtons**. Nous allons procéder de la même manière que pour les boutons simples à quelque chose près. On lance une boucle sur les boutons de choix, on écoute les événements et selon le cas on récupère le texte sélectionné, les infos des commandes prompt, et enfin on assigne les bons paramètres pour construire le tag approprié pour l'insérer dans le champ de texte.

Étudions la méthode `eventMessageBox` :

Code : JavaScript

```
/**
* Lien ouvrant une fenêtre box pour insérer une ou des valeurs au
choix
*/
eventMessageBox : function() {
 var event = (Browser.ie) ? 'mousedown' : 'click';
 this.dlgButtons.each(function(bt){
  var link = $(bt);
  var text = value = '';
  if(undefined !== link) {
   link.addEvent(event, function(){
    var name = link.getProperty('data-bt');
    var startTag = endTag = '';
    var select = this.input.getSelectedText();
    switch(name) {
     case 'tableau':
      startTag = this.setTable();
      text = '';
     break;

     case 'liste':
     case 'listnum' :
      startTag = this.setList(name);
      text = '';
     break;

     case 'image' :
      text = (select !== '') ? select : prompt("Tapez le nom
d'emplacement de l'image : ");
      startTag = this.getSimpleTag(name, text);
     break;

     case 'clip':
      startTag = this.setObject(name);
     break;

     case 'citation' :
      value = prompt("Tapez le nom de l'auteur de la citation :");
      text = null;
      startTag = this.getTagValue(name, value, 'auteur');
     break;

     case 'url' :
      value = prompt("Tapez l'adresse du lien : ");
      text = (select === '') ? prompt("Tapez le texte du lien : "):
select;
      startTag = this.getTagValue(name, value);
```

```
            break;

            case 'urlancre' :
             value =  (select !== '') ? select : prompt("Tapez le nom de
l'ancre : ");
             text = prompt("Tapez le texte du lien : ");
             startTag = this.getTagValue(name, value);
             break;

            case 'ancre' :
             alert('Lier avec un lien de type:'+"\n"+'<urlancre
value="nomdeLancre">Texte du lien</url>');
             value = (select !== '') ? select : prompt("Tapez le nom de
l'ancre : ");
             text = prompt("Tapez le texte du lien : ");
             startTag = this.getTagValue(name, value);
            break;
           }
           endTag = '</'+name+'>';
           this.insertTag(startTag, endTag, text);
          }.bind(this));
        }
       }, this);
      },
      [...]
```

Analyse du code

On lance une boucle sur les boutons portant la classe CSS **dlgButtons**. Dès qu'un événement est détecté, on récupère l'attribut `data-bt` de l'image. Selon les cas, on récupère la valeur à insérer dans l'attribut de notre tag (ici adresse du lien), le texte sélectionné (ici le texte du lien). Ensuite, on construit notre tag d'ouverture (`startTag`) avec les bons paramètres, et on appelle la méthode `insertTag` avec nos paramètres assignés.
Voyons maintenant les méthodes spécifiques. Nous commencerons par les méthodes `getTagSimple` et `getTagValue` :

Construction du tag d'ouverture simple pour le tag image

On doit obtenir :

Code : Autre

```
//Si aucune sélection
<image>null
//Sinon
<image>
```

Deux paramètres seront indispensables : le nom du tag et le texte sélectionné s'il existe.

Code : JavaScript

```
/**
* Formate un tag de la forme <tag>text
*/
getSimpleTag : function(name, text) {
 var tag = '';
 //Si aucune sélection
 if (text === '') {
  tag = '<'+name+'>null'; // null entre les balises
 } else {
```

```
    tag = '<'+name+'>'; // Sinon on ajoute juste le tag (pour éviter
qu'il se répète avec insertTag)
   }
   return tag;
},
```

Le code est commenté et très facile à comprendre.

Construction du tag d'ouverture avec attribut

Le but du jeu est d'obtenir un tag d'ouverture avec attribut et sa valeur si elle existe ou une valeur nulle selon le cas.

Code : JavaScript

```
[...]
/**
* Formate un tag à la forme <name attribut="value">text
*/
getTagValue : function(name, value, attribut) {
 var tag = '';
 //Si aucun nom d'attribut n'est assigné
 if('' === attribut || undefined === attribut) {
  attribut = 'value';// l'attribut par défaut sera value
 }
 //Si aucune valeur d'attribut n'est assignée
 if (value === '') {
  tag += '<'+name+' '+attribut+'="null">';// attribut a pour valeur
null

 } else {
  //Sinon on construit notre tag d'ouverture
  tag = '<'+name+' '+attribut+'="'+ value +'">';
 }
 return tag;
},
[...]
```

Le code est assez commenté pour comprendre le principe. 

Les listes et tableaux

Les listes et les tableaux sont un peu plus complexes car ils vont imbriquer des tags.
Commençons par les listes. Nous aurons deux types de listes (normales et numériques). Les tags seront de la forme :

Code : Autre

```
<Typedeliste>
        <unepuce>Texte de la puce</unepuce>
</Typedeliste>
```

Les types de listes seront : liste et listnum.
Je vous donne l'exemple de la puce et vous aurez le tableau à faire en exercice  :

Code : JavaScript - zcode.js

```
[...]
   /**
```

```
        * Liste et liste numérique
        */
        setList : function(name) {
            var i = 1;
            var tag = '';
            var puce = '';
            var message = 'Tapez le texte de la puce : (Si vous voulez
        arrêter, cliquez sur Annuler)';
            var insertText = this.input.getSelectedText();
            if(insertText !== ''){
                puce = "\t<puce>" + insertText + "</puce>\n";
                tag = "" + tag + "" + puce + "";
            } else {
                insertText = prompt(message, "Puce " + i);
                while (insertText) {
                    i++;
                    if (insertText) {
                        puce = "\t<puce>" + insertText + "</puce>\n";
                        tag = "" + tag + "" + puce + "";
                        insertText = prompt (message,"Puce " + i);
                    } else {
                        break;
                    }
                }
            }
            tag = "\n<"+name+">\n" + tag;
            return tag;
        },

        [...]
```

Rien de très compliqué ici, si le client sélectionne un bout de texte, on insère une liste simple, sinon on lui demande le texte de la puce et tant que l'utilisateur ne clique pas sur annuler, on récupère le texte du prompt et on ajoute des puces.

À vous de coder la méthode `setTable` ! Les tags seront de la forme :

Code : Autre

```
<tableaux>
        <lignes>
                <colonnes>Texte de la colonne</colonne>
                <colonnes>Texte de la colonne</colonne>
        </ligne>
        <lignes>
                <colonnes>Texte de la colonne</colonne>
                <colonnes>Texte de la colonne</colonne>
        </lignes>
</tableaux>
```

J'ai mis le nom des tags au pluriel ici pour éviter le parsage du zCode du SdZ.

- Il va falloir demander le nombre de lignes et de colonnes que veut l'utilisateur pour son tableau…
- … et demander quel texte l'utilisateur veut insérer entre ses colonnes.

Correction de la méthode :

Secret (cliquez pour afficher)

Code : JavaScript - zform.js

```
    [...]
```

```
        /**
     * Tableau
     */
    setTable : function() {
        var tag = '';
        var lineNumber = 0;
        var rowNumber = 0;
        var select = this.input.getSelectedText();
        if(select !== ''){
            tag += "\t<ligne>\n";
            tag += "\t\t<colonne>" + select + "</colonne>\n";
            tag += "\t</ligne>\n";
        } else {
            lineNumber = prompt('Tapez le nombre de lignes que
    contiendra le tableau :','Nombre de lignes');
            rowNumber = prompt('Tapez le nombre de colonnes que
    contiendra le tableau :','Nombre de colonnes');
            for ( var i = 0; i < lineNumber; i++) {
                tag += "\t<ligne>\n";
                for (var j = 1; j <= rowNumber; j++) {
                    text = prompt("Ligne:" + (i + 1)+ "\nTapez le
    texte de cette colonne : ", "Texte colonne "+ j);
                    if (text) {
                        tag += "\t\t<colonne>" + text +
    "</colonne>\n";
                    } else {
                        break;
                    }
                }
                tag += "\t</ligne>\n";
            }
        }
        tag = "<tableau>\n" + tag;
        return tag;
    },
    [...]
```

Ici, si le client sélectionne du texte on insère un tableau à une ligne et une colonne, sinon je récupère le nombre de lignes et de colonnes avec un prompt. Je fais une boucle sur mes lignes et colonnes.
Sur chaque colonne, je demande le texte de la colonne avec un prompt et je construis mes tags (j'ai ajouté des tabulations et retours à la ligne pour le rendu).

Les clips

La méthode pour insérer des vidéos ou animations :

Code : JavaScript

```
[...]
setObject : function(name) {
 var tag = '';
 var app = prompt("Saisissez le type de l'application au choix :\n\r
flash|wmv|avi|mpeg");
 var video = prompt("Saisissez le chemin du clip");
 var w = prompt("Saisissez la largeur");
 var h = prompt("Saisissez la hauteur du clip");
 var autoStart = '0';
 var autoPlay = 'false';
 var type = '';
 switch(app) {
  case 'flash':
    type =  'x-shockwave-flash';
   break;
  case 'wmv':
```

```
          type =  'video/x-ms-wmv';
         autoPlay = '';
         break;
        case 'avi':
          type =  'video/x-msvideo';
         autoPlay = '';
         break;
        case 'mpeg':
          type =  'video/mpeg';
         break;
       }
       if(h ===''  || h > 600) h = 200;
       if(w === ''  || w > 600) w = 150;
       if(app == '') {
        alert("Vous devez saisir le nom de l'application");
        app = prompt("Saisissez le type de l'application au choix :\n\r
      flash|wmv|avi|mpeg");
       } else {
        if(video != '') {
         tag =
      '<'+name+'="'+type+':'+w+':'+h+':'+autoStart+':'+autoPlay+'">'+video;
        } else {
         tag =
      '<'+name+'="'+type+':'+w+':'+h+':'+autoStart+':'+autoPlay+'">';
        }
       }
       return tag;
      },
      [...]
```

Ajouter les events des boutons de choix à l'écouteur

Pour finir il ne faut pas oublier d'ajouter la méthode à l'écouteur :

Code : JavaScript

```
[...]
// Écoute les événements
listeningEvent : function() {
 this.input.focus();
 this.eventButtons();
 this.eventMessageBox();
},
[...]
```

Testez chaque bouton pour voir le résultat, le tag clip correspond à une balise object.
Voilà, notre objet commence à être bien complet. Il nous manque les listes de choix, allons-y de ce pas !

Les listes de choix

Avant de commencer

Le principe

Nous allons créer des listes de choix à l'aide de boutons, chaque bouton survolé faisant apparaître un menu de choix. Le choix sera soit un lien textuel, soit une image. Un exemple simple : le flottement. Un texte ou un élément HTML peut être disposé flottant à gauche ou à droite.
Les tags seront de la forme suivante pour un texte flottant à droite :

Code : Autre

```
<flottant value="droite">texte</flottant>
//Correspond à la balise
<div style="float: right;">Text</div>
```

Au clic du bouton, une boîte contenant les choix va apparaître. Au clic du choix, le tag choisi sera inséré dans le textarea.
Notre bouton :

Code : HTML - index.html

```
<!-- Boutons listes -->
<img data-bt="flottant" class="bt-code listButtons"
src="public/images/boutons/bt-float.gif"  alt="Couleur de la police"
/>
     <div class="sliderBox">
          <a class="slider-link" rel="gauche">Flottant à gauche</a>
          <a class="slider-link" rel="droit">Flottant à droite</a>
     </div>
```

Au survol de l'image portant un nom de classe listButtons, la div `class="slider-box"` va apparaître, pour afficher nos deux choix. Au clic sur l'un des liens `class="slider-link"`, on enverra le paramètre `rel` `gauche` ou `droite` à notre méthode.
Vous saisissez ? 

La sliderBox, ses événements et méthodes

Avant de coder, détaillons tous les événements et le nom des méthodes choisies.

- initListButtons : au chargement de la page les box doivent être cachées.
- eventList : event sur les listes de choix :
  - au survol du bouton, la box apparaît ;
  - lorsque la souris quitte le bouton, la box doit disparaître ;
  - mais si on survole la box et ses liens, elle ne doit pas disparaître.

- eventLinkList : event sur les liens de la liste :
  - au clic du lien, le tag doit s'insérer dans le textarea.

- eventBoxList : event sur les boîtes à liste :
  - au survol des liens, la box ne doit pas disparaître ;
  - lorsque la souris quitte la box et n'est pas sur le bouton liste, la box doit disparaître.

Voilà pour les events. Il y aura aussi, si vous avez suivi, les deux méthodes suivantes :

- hide : pour cacher la box ;
- show : pour l'afficher.

Cacher les boîtes de menus au démarrage

Il va falloir mettre notre sliderBox en position absolue pour qu'elle s'affiche juste en dessous du bouton survolé, et donc calculer la position left de la div. Elle sera appelée au chargement de la page, on lancera l'appel dans notre constructeur.
Vous devez coder la méthode initListButtons :

Code : JavaScript

```
/**
* Positionne et cache les boîtes des sélecteurs de liste
*/
initListButtons : function() {
 if(undefined !== this.sliders) {
  //On fait une boucle sur toutes les boîtes à menu
  this.sliders.each(function(slider){
   //Je récupère l'attribut name de l'image
   var img = slider.getPrevious('img');
   //Sa position
   var x = img.getPosition().x.toInt();
   var y = img.getPosition().y.toInt() + img.getSize().y.toInt();
   //Positionne les box par rapport au bouton cliqué
   slider.setStyles({
    'position' : 'absolute',
    'left' : x,
    'top' : y,
    'display' : 'none',
    'visibility' : 'hidden'
   });
  });
 }
},
```

Le code est commenté pour vous expliquer le cheminement.

La méthode eventList

Code : JavaScript

```
[...]
/**
* Events sur les boutons sélecteurs
*/
eventList : function() {
 //Événements sur les listes de choix
 this.listButtons.each(function(bt){
  if(undefined !== bt) {
   bt.addEvents({
    'mouseenter' : function(){ //On affiche la box et on écoute ses
events
     this.eventBoxList(bt);
    }.bind(this),
    'mouseleave' : function() { //On cache la box
     this.timer = this.hide.delay(this.closeDelay, this,
bt.getNext(this.options.slider));
    }.bind(this)
   });
  }
 }, this);
},
[...]
```

La méthode eventBoxList

Code : JavaScript

```
[...]
// Event sur les menus
```

```
eventBoxList : function(el) {
 var box = el.getNext('div');
 if(undefined !==  box) {
  this.show(el, box); //On affiche le menu
  box.addEvents({
   'mouseleave': function(){
    this.timer = this.hide.delay(this.closeDelay, this, box); //On
cache le menu au bout de 500 millisecondes
   }.bind(this),
   'mouseenter': function(){
    clearTimeout(this.timer);//On remet le timer à 0
   }.bind(this)
  });
 }
},
[...]
```

La méthode eventLinkList

Code : JavaScript

```
[...]
// Event sur les liens de la liste
eventLinkList : function() {
 if(undefined !== this.sliderLink) {
  //On boucle sur tous les lien slider-link
  this.sliderLink.each(function(link){
   //Si on clique
   link.addEvent('mousedown', function(){
    //On récupère l'attribut data-bt de l'image
    var name =
link.getParent('div').getPrevious('img').getProperty('data-bt');
    //On insère notre tag avec la valeur de l'attribut rel du lien
cliqué
    this.insertTag('<'+name+' value="'+link.get('rel')+'">',
'</'+name+'>');
   }.bind(this));
  }, this);
 }
},
[...]
```

Les méthodes show et hide

On utilisera Fx.Morph pour modifier les propriétés de style de la box :

Code : JavaScript

```
/**
* Cache les boîtes de menus
*/
hide :  function(box) {
 var fx = new Fx.Morph(box, {duration:800,transition:
Fx.Transitions.linear});
 fx.start({
  opacity : [1,0],
  display : 'block',
  visibility : 'hidden'
 });
},
```

```
/**
* Affiche les boîtes de menus
*/
show : function(el, box) {
 var fx = new Fx.Morph(box, {duration:800,transition:
Fx.Transitions.linear});
 fx.start({
  left : el.getPosition().x.toInt(),
  display : 'block',
  visibility : 'visible',
  opacity : [0,1]
 });
},
```

Analyse du code

- Pour hide : on assigne en paramètre la box, on fait disparaître notre box grâce aux propriétés css.
- Pour show : fonctionnement similaire sauf que nous avons besoin de la position du bouton (elle représente le bouton image et box la div) pour que la box s'affiche au bon endroit.

Dans listeningEvent, il faut ajouter :

Code : JavaScript

```
this.eventSelectors();
this.eventLinkList();
```

Et dans la méthode initialize, il faut ajouter :

Code : JavaScript

```
this.initListButtons();
```

Si le comportement de survol ne vous plaît pas et que vous préférez afficher les listes au clic du bouton, il faudra modifier la méthode eventList en conséquence, comme ajouter un event onclick et un booléen qui définit un statut ouvert ou fermé par exemple.

Testez donc chaque bouton pour voir si tout fonctionne.
Ça commence à en faire du code... Nous approchons de la fin avec le prochain chapitre : la visualisation automatique avec Ajax.

L'auto-visualisation avec Ajax

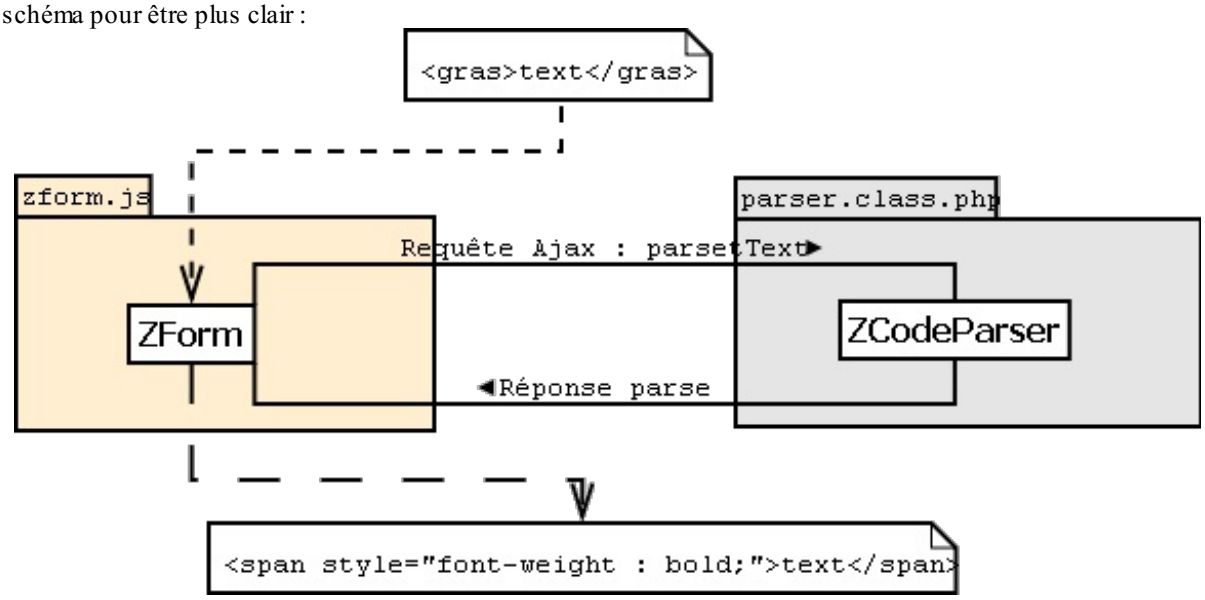
Choix du parsage

Le langage zCode fonctionne logiquement avec une visualisation PHP et XML/XSL. J'ai choisi pour le tuto un « parsage » à l'aide de PHP et des regex. Le « parsage » fait déjà l'objet de plusieurs tutoriels, ici il a seulement pour but d'avoir le rendu de notre ZForm et voir si tout fonctionne correctement.

Le principe

Nous allons envoyer une requête Ajax vers un fichier PHP, ce fichier va nous parser le texte en HTML et nous le retourner afin de l'afficher dans la div id="prev". Nous aurons donc deux méthodes JavaScript à coder, une méthode parseText qui enverra la requête Ajax et une méthode previewParser qui s'occupera d'appeler la méthode parseText.

Un petit schéma pour être plus clair :



La requête Ajax

Nous allons coder la méthode envoyant la requête Ajax : il faut récupérer le texte du textarea et l'envoyer en post vers le fichier PHP. La réponse sera insérée dans la div prev.

Code : JavaScript

```
[...]
//Parse le texte via PHP pour la visualisation
parseText : function(text) {
 var prev = this.preview;
 var req = new Request({
  url : 'parse.php',
  method : 'post',
  onComplete: function() {
   prev.set('html', this.response.text);
  }
 });
 req.send('text='+escape(text));
},
[...]
```

Analyse du code

Ma méthode reçoit donc le texte issu du champ en paramètre, on envoie la requête avec le texte. Une fois la réponse récupérée, on l'insère dans la div prev. Sur le fichier PHP, il faudra donc récupérer une variable post $_POST['text'], que nous allons « parser » grâce à un objet PHP.

Le *parser* PHP

Si vous ne connaissez pas PHP, allez voir le tuto de M@teo21. Nous allons créer tout d'abord un objet pour parser notre texte en format HTML. Notre classe va donc transformer les tags zCode en balises HTML.
Prenons un fichier parser.class.php, notre objet va contenir quatre attributs par défaut et des méthodes de « parsage » pour nos tags.

Code : PHP - parser.class.php

```php
<?php
/**
* @author franckysolo
* @see Tuto ZForm avec mootools - Site du Zéro
* @since 24/09/2010
* @revision 13/02/2011
*/
class ZCodeParser
{
 /**
* Tags autorisés dans le ZForm
* @var array
*/
 private $_authorizedTags = array(
  'attention', 'erreur', 'infos', 'question',
  'liste' , 'listnum', 'puce',
  'tableau', 'ligne', 'colonne',
  'citation', 'titre', 'image', 'clip',
  'couleur', 'police', 'taille',
  'gras', 'italique', 'souligne', 'barre',
  'position', 'flottant', 'marge',
  'url', 'urlancre', 'ancre'
 );
 /**
* Tags parsés avec str_replace
* @var array
*/
 private $_replaceTags = array
 (
  //Simple Tag
  '<gras>'  => '<span style="font-weight: bold;">',
  '</gras>'  => '</span>',
  '<souligne>'  => '<span style="text-decoration: underline;">',
  '</souligne>'  => '</span>',
  '<barre>'  => '<span style="text-decoration: line-through;">',
  '</barre>'  => '</span>',
  //Div
  '<attention>'  => '<div class="parser-attention">',
  '</attention>'  => '</div>',
  '<erreur>'  => '<div class="parser-erreur">',
  '</erreur>'  => '</div>',
  '<infos>'  => '<div class="parser-infos">',
  '</infos>'  => '</div>',
  '<question>'  => '<div class="parser-question">',
  '</question>'  => '</div>',

  '<italique>'  => '<em>',
  '</italique>'  => '</em>',
  '<marge>'  => '<blockquote>',
  '</marge>'  => '</blockquote>',
  '<liste>'  => '<ul>' ,
  '</liste>'  => '</ul>' ,
  '<listnum>'  => '<ol>',
  '</listnum>'  => '</ol>',
  '<puce>'  => '<li>',
  '</puce>'  => '</li>',
  '<tableau>'  => '<table>',
  '</tableau>'  => '</table>',
  '<ligne>'  => '<tr>',
  '</ligne>'  => '</tr>',
  '<colonne>' => '<td>',
  '</colonne>' => '</td>',

  //Attr Tag
  '<titre value="h1">'  => '<h1>',
  '</titre>'  => '</h1>',
```

```
        '<titre value="h2">'  => '<h2>',
        '</titre>'    => '</h2>',
        '<titre value="h3">'  => '<h3>',
        '</titre>'    => '</h3>',

        '<position value="gauche">'  => '<div style="text-align: left;">',
        '<position value="center">'  => '<div style="text-align: center;">',
        '<position value="droite">' => '<div style="text-align: right;">',
        '<position value="justifier">'  => '<div style="text-align: justified;">',
        '</position>'     => '</div>',

        '<flottant value="gauche">'  => '<div style="float: left;">',
        '<flottant value="droit">' => '<div style="float: right;">',
        '</flottant>'     => '</div>',

        '<taille value="xx-large">' => '<span style="font-size: xx-large;">',
        '<taille value="x-large">' => '<span style="font-size: x-large;">',
        '<taille value="large">' => '<span style="font-size: large;">',
        '<taille value=="medium">' => '<span style="font-size: medium;">',
        '<taille value="xx-small">' => '<span style="font-size: xx-small;">',
        '<taille value="x-small">' => '<span style="font-size: x-small;">',
        '<taille value="small">' => '<span style="font-size: small;">',
        '</taille>'       => '</span>',

      );

      /**
     * Pattern Regex parsage avec preg_replace
     * @var array
     */
      private $_patterns = array(
        '`\<image\>(.+)\<\/image\>`iU',
        '`\<ancre value=\"(.+)\"\>\</ancre\>`iU',
        '`\<urlancre value=\"(.+)\"\>(.+)\</urlancre\>`iU',
        '`\<url value=\"(.+)\"\>(.+)\</url\>`iU',
        '`\<couleur value=\"(\#[a-fA-F0-9]{6})\"\>(.+)\<\/couleur\>`isU',
        '`\<police value=\"(.+)\"\>(.+)\<\/police\>`isU',
        '`\<citation auteur=\"(.+)\">(.+)\<\/citation\>`isU',

      );
      /**
     * Balises HTML de remplacement. Parsage avec preg_replace
     * @var array
     */
      private $_remplacement = array(
        '<img src="$1" alt="Image utilisateur" />',
        '<a name="$1"></a>',
        '<a href="#$1">$2</a>',
        '<a href="$1">$2</a>',
        '<span style="color:$1">$2</span>',
        '<span style="font-family: $1">$2</span>',
        '<div class="title-citation">Citation : $1</div><div class="citation">$2</div
      );

      /**
     * Enlève les \n sur tableau et liste et protège contre les failles XSS
     * @param string $text
     * @return string
     */
      public function cleanText($text)
      {
        $parseValues = array();
        //On « préparse » nos balises autorisées pour éviter la transformation des
    chevrons avec htmlspecialchars
        foreach($this->_authorizedTags as $key => $value) {
          $parseValues[$key] = '`<([\/]?'.$value.')( )*([^>]*)?>`i';
        }
        $text = preg_replace_callback($parseValues, array(&$this, '_preparse'), $text
        //On applique la protection à tout le texte
        $text = htmlspecialchars($text, ENT_NOQUOTES);
```

```
        //On « déparse » nos balises autorisées
        foreach($this->_authorizedTags as $key => $value) {
         $parseValues[$key] = '`\[([\/]?'.$value.')( )*([^\]]*)?\]`i';
        }
        $text = preg_replace_callback($parseValues, array(&$this, '_postparse'),
      $text);
        $text = preg_replace_callback
        (
         '`(\<script(?:.*)\>(?:.+)\<\/script\>)`isU',
         create_function('$text', 'return htmlspecialchars($text[0]);'),
         $text
        );
        //On enlève les retours à la ligne ajoutés pour l'indentation dans le champ d
      texte
        $pattern = '`\<(?:(tableau|listnum|liste))\>(\n*)(?:.+)\<\/(?
      :(tableau|listnum|liste))\>`isU';
        $text = preg_replace_callback($pattern, array(&$this, 'cleanBr'), $text);
        return $text;
       }

       /**
      * Supprime le retour à la ligne de mise en forme
      * @param string $text
      * @return string
      */
       public function cleanBr($text)
       {
        return str_replace(array("\n", "\r"), '', $text[0]);
       }
       /**
      * Parse le texte avec str_replace
      * @param string $text
      * @return string
      */
       public function parseReplace($text)
       {
        return str_replace(array_keys($this->_replaceTags), array_values($this-
      >_replaceTags), $text);
       }

       /**
      * Parse le texte avec preg_replace
      * @param string $text
      */
       public function parseRegex($text)
       {
        $array = array_combine($this->_patterns, $this->_remplacement);
        foreach($array as $pattern => $replacement){
         $text = preg_replace($pattern, $replacement, $text);
        }
        return $text;
       }

       /**
      * Parse le texte
      * @param string $text
      */
       public function parse($text) {
        $text = stripslashes($text);
        $text = $this->cleanText($text);
        $text = $this->parseReplace($text);
        $text = $this->parseRegex($text);
        $text = $this->parserObjectTag($text);
        $text = nl2br($text);
        return $text;
       }

       /**
      * Tag clip => <object><param value=value /></object>
      * @param string $text
```

```
    */
     public function parserObjectTag($text) {
      $text = preg_replace
      (
       '`\<clip=\"(video/x-msvideo|video/x-ms-wmv|x-shockwave-
flash|video/mpeg):([\d]{3}):([\d]{3}):(0|1):(false|true|\s)*\"\>(.+)\</clip\>`i
       '<object type="application/$1" data="$6" width="$2" height="$3"
style="margin:auto; background-color: black;">
<param name="movie" value="$6" />
<param name="autostart" value="$4" />
<param name="autoplay" value="$5" />
<param name="quality" value="high" />
<param name="loop" value="false">
<param name="wmode" value="transparent" />
</object>', $text
      );
      return $text;
     }

     /**
    * Convertit les < > en [ ] des balises autorisées
    * @param array $matches
    */
     private function _preparse($matches)
     {
      return '['.$matches[1].$matches[2].$matches[3].']';
     }

     /**
    * Convertit les [ ] en < > des balises autorisées
    * @param array $matches
    */
     private function _postparse($matches)
     {
      return '<'.$matches[1].$matches[2].$matches[3].'>';
     }
    }
    ?>
```

Analyse du code

Nous utiliserons str_replace pour nos tags contenus dans le tableau parseTag. Pour les autres Tags on utilisera des regex. Je vous renvoie au cours sur les regex pour les captures et remplacements de mots, ici le but n'est pas d'étudier les regex mais de parser notre texte.
La méthode cleanText me permet d'éliminer les retours à la ligne ajoutés pour la bonne indentation des tags du ZForm, mais aussi protège contre les failles XSS.

La visualisation

Enfin, nous allons activer l'auto-visualisation grâce à la méthode previewParser. Notre méthode parseText JavaScript envoie la requête au fichier parser.php.
Créons donc ce fichier :

Code : PHP - parser.php

```
<?php
include 'parser.class.php';
$text = $_POST['text'];
$parser = new ZCodeParser();
echo $parser->parse($text);
```

Dernière chose à accomplir, la méthode previewParser va appeler la méthode parseText.

Code : JavaScript

```
//Visualisation dans la div id=prev
previewParser : function() {
 var txt = this.input.value;
 this.parseText(txt);
}
```

Et si je clique ou j'écris quelque chose, je n'ai toujours pas d'aperçu !!! 

Il faut déclarer notre méthode pour qu'elle fonctionne, nous allons l'ajouter à l'insertion des tags et sur l'événement keyup du le champ de texte.

Finalement, vous aurez les fichiers suivants :

- les images des boutons dans le dossier public/images/boutons/ ;
- un fichier zform.js contenant notre ZForm en JavaScript dans le dossier public/js/ ;
- un fichier parser.class.php contenant notre *parser* PHP à la racine ;
- un fichier parser.php pour la requête Ajax à la racine ;
- un fichier zform.css dans le dossier public/css/ ;
- notre index.html à la racine.

Voici le code complet de ZForm js :

Code : JavaScript - zform.js

```
/**
* @author franckysolo
* @see Tuto ZForm avec mootools - Site du Zéro
* @since 24/09/2010
* @revision 13/02/2011
*/

var ZForm = new Class({

    Implements : Options,
    options : {
        textId : 'text', // id du textarea
        preview : 'prev', // id de la div de prévisualisation
        buttons : '.splButtons', // les boutons simples
        dlgButtons : '.dlgButtons', // les boutons ouvrant une boite
de dialogue
        listButtons : '.listButtons', // les boutons listes ouvrant
un menu de choix
        slider : '.sliderBox', // les boites contenant un menu de
choix
        sliderLink : '.slider-link',  // les liens du menu de choix
        closeDelay : 500 // temps avant fermeture de la boite de
menu en milliseconde
    },
    /**
* @param options
*/
    initialize : function(options) {
        this.setOptions(options);
        this.input = $(this.options.textId); //
getElementById(textId)
```

```
        this.preview = $(this.options.preview); //
getElementById(preview)
        this.timer = 0;
        //Buttons
        this.buttons = $$(this.options.buttons);  //tableau de tous
les boutons simples
        this.dlgButtons = $$(this.options.dlgButtons);//tableau de
tous les boutons ouvrant une boite de dialogue
        this.sliders = $$(this.options.slider);//tableau de toutes
les boites contenant un menu de choix
        this.listButtons = $$(this.options.listButtons);//tableau de
toutes les listes ouvrant un menu de choix
        this.sliderLink = $$(this.options.sliderLink);//tous les
liens de menu de choix
        this.initListButtons();
    },

    //Écoute les événements
    listeningEvent : function() {
        this.input.focus();
        this.eventButtons();
        this.eventMessageBox();
        this.eventList();
        this.eventLinkList();
        //Preview du parser
        if(undefined !== this.input) {
            this.input.addEvent('keyup', function() {
                this.previewParser();
            }.bind(this));
        }
    },

    initListButtons : function() {
        if(undefined !== this.sliders) {
            //On fait une boucle sur toutes les boites à menu
            this.sliders.each(function(slider){
                //Je récupère l'attribut name de l'image
                var img = slider.getPrevious('img');
                var x = img.getPosition().x.toInt();
                var y = img.getPosition().y.toInt() +
img.getSize().y.toInt();
                //Positionne les box par rapport au bouton cliqué
                slider.setStyles({
                    'position' : 'absolute',
                    'left' : x,
                    'top' : y,
                    'display' : 'none',
                    'visibility' : 'hidden'
                });
            });
        }
    },

    /**
* Events sur les boutons simples
*/
    eventButtons : function() {
        //Si c'est I.E. on utilise plutôt l'event mousedown
        var event = (Browser.ie) ? 'mousedown' : 'click';
        //On écoute tous les boutons simples
        this.buttons.each(function(button){
            //Si un event est détecté
            button.addEvent(event, function(){
                //On récupère l'attribut data-bt de l'image
                var name = button.getProperty('data-bt');
                //On appelle la méthode d'insertion de tags avec les
bons paramètres
                this.insertTag('<'+name+'>', '</'+name+'>',
this.input.getSelectedText());
            }.bind(this));
```

```
            }, this);
        },

        //Insertion des tags
        insertTag : function(startTag, endTag, select) {
            //On donne le focus à notre champ de texte
            this.input.focus();
            //S'il n'y a pas de sélection
            if(select === null || undefined === select) {
                //On insère nos tags autour du curseur
                this.input.insertAroundCursor({'before' : startTag,
'after' : endTag});
            } else {
                //Sinon on insère nos tags au curseur
                this.input.insertAtCursor(startTag + select + endTag,
false);
            }
            this.input.focus();
            this.previewParser();
        },

        /**
* Lien ouvrant une fenêtre box pour insérer une ou des valeurs au
choix
*/
        eventMessageBox : function() {
            var event = (Browser.ie) ? 'mousedown' : 'click';
            this.dlgButtons.each(function(bt){
                var link = $(bt);
                var text = value = '';
                if(undefined !== link) {
                    link.addEvent(event, function(){
                        var name = link.getProperty('data-bt');
                        var startTag = endTag = '';
                        var select = this.input.getSelectedText();
                        switch(name) {
                            case 'tableau':
                                startTag = this.setTable();
                                text = null;
                            break;

                            case 'liste':
                            case 'listnum' :
                                startTag = this.setList(name);
                                text = null;
                            break;

                            case 'image' :
                                text = (select !== '') ? select :
prompt("Tapez le nom d'emplacement de l'image : ");
                                startTag = this.getSimpleTag(name, text);
                            break;

                            case 'clip':
                                startTag = this.setObject(name);
                            break;

                            case 'citation' :
                                value = prompt("Tapez le nom de l'auteur
de la citation :");
                                text = null;
                                startTag = this.getTagValue(name, value,
'auteur');
                            break;

                            case 'url' :
                                value = prompt("Tapez l'adresse du lien :
");
                                text = (select === '') ? prompt("Tapez le
texte du lien : "): select;
```

```
                                startTag = this.getTagValue(name, value);
                            break;

                            case 'urlancre' :
                                value =  (select !== '') ? select :
prompt("Tapez le nom de l'ancre : ");
                                text = prompt("Tapez le texte du lien :
");
                                startTag = this.getTagValue(name, value);
                                break;

                            case 'ancre' :
                                alert('Lier avec un lien de
type:'+"\n"+'<urlancre value="nomdeLancre">Texte du lien</url>');
                                value = (select !== '') ? select :
prompt("Tapez le nom de l'ancre : ");
                                text = prompt("Tapez le texte du lien :
");
                                startTag = this.getTagValue(name, value);
                            break;
                        }
                        endTag = '</'+name+'>';
                        this.insertTag(startTag, endTag, text);
                    }.bind(this));
                }
            }, this);
        },
        /**
* Formate un tag à la forme <name attribut="value">text
*/
        getTagValue : function(name, value, attribut) {
            var tag = '';
            //Si aucun nom d'attribut n'est assigné
            if('' === attribut || undefined === attribut) {
                attribut = 'value'; // L'attribut par défaut sera value
            }
            //Si aucune valeur d'attribut n'est assignée
            if (value === '') {
                tag += '<'+name+' '+attribut+'="null">'; // attribut a
pour valeur null

            } else {
                //Sinon on construit notre tag d'ouverture
                tag = '<'+name+' '+attribut+'="'+ value +'">';
            }
            return tag;
        },
        /**
* Formate un tag à la forme <tag>text
*/
        getSimpleTag : function(name, text) {
            var tag = '';
            //Si aucune sélection
            if (text === '') {
                tag = '<'+name+'>null';         //null entre les balises
            } else {
                tag = '<'+name+'>'; //sinon on ajoute juste le tag
            }
            return tag;
        },
        /**
* Tableau
*/
        setTable : function() {
            var tag = '';
            var lineNumber = rowNumber = 0;
            lineNumber = prompt('Tapez le nombre de lignes que contiendra
le tableau :','Nombre de lignes');
            rowNumber = prompt('Tapez le nombre de colonnes que
contiendra le tableau :','Nombre de colonnes');
```

```
            for ( var i = 0; i < lineNumber; i++) {
                tag += "\t<ligne>\n";
                for (var j = 1; j <= rowNumber; j++) {
                    var text = prompt("Ligne:" + (i + 1)+ "\nTapez le
texte de cette colonne : ", "Texte colonne "+ j);
                    if (text) {
                        tag += "\t\t<colonne>" + text + "</colonne>\n";
                    } else {
                        break;
                    }
                }
                tag += "\t</ligne>\n";
            }
            tag = "<tableau>\n" + tag;
            return tag;
        },
        /**
* Liste et liste numérique
*/
        setList : function(name) {
            var i = 1;
            var tag = '';
            var insertText = prompt (
                "Tapez le texte de la puce : (Si vous voulez arrêter,
cliquez sur Annuler) ",
                "Puce " + i
            );
            while (insertText) {
                i++;
                if (insertText) {
                    var puce = "\t<puce>" + insertText + "</puce>\n";
                    tag = "" + tag + "" + puce + "";
                    insertText = prompt (
                        "Tapez le texte de la puce : (Si vous voulez
arrêter, cliquez sur Annuler) ",
                        "Puce " + i
                    );
                } else {
                    break;
                }
            }
            tag = "\n<"+name+">\n" + tag;
            return tag;
        },

        setObject : function(name) {
            var tag = '';
            var app = prompt("Saisissez le type de l'application au
choix:\n\r flash|wmv|avi|mpeg");
            var video = prompt("Saisissez le chemin du clip");
            var w = prompt("Saisissez la largeur");
            var h = prompt("Saisissez la hauteur du clip");
            var autoStart = '0';
            var autoPlay = 'false';
            var type = '';
            switch(app) {
                case 'flash':
                     type =  'x-shockwave-flash';
                    break;
                case 'wmv':
                     type =  'video/x-ms-wmv';
                    autoPlay = '';
                    break;
                case 'avi':
                     type =  'video/x-msvideo';
                    autoPlay = '';
                    break;
                case 'mpeg':
                     type =  'video/mpeg';
                    break;
```

```
            }
            if(h ==='' || h > 600) h = 200;
            if(w === '' || w > 600) w = 150;
            if(app == '') {
                alert("Vous devez saisir le nom de l'application");
                app = prompt("Saisissez le type de l'application au
choix:\n\r flash|wmv|avi|mpeg");
            } else {
                if(video != '') {
                    tag =
'<'+name+'="'+type+':'+w+':'+h+':'+autoStart+':'+autoPlay+'">'+video;
                } else {
                    tag =
'<'+name+'="'+type+':'+w+':'+h+':'+autoStart+':'+autoPlay+'">';
                }
            }
            return tag;
        },

        // Cache les boites à menu
        hide :  function(box) {
            var fx = new Fx.Morph(box, {duration : 800, transition:
Fx.Transitions.linear});
            fx.start({
                opacity : [1,0],
                display : 'block',
                visibility : 'hidden'
            });
        },
        // Affiche les boites à menu
        show : function(el, box) {
            var fx = new Fx.Morph(box, {duration : 800, transition:
Fx.Transitions.linear});
            fx.start({
                left : el.getPosition().x.toInt(),
                display : 'block',
                visibility : 'visible',
                opacity : [0,1]
            });
        },
        // Event sur les boites à liste
        eventBoxList : function(el) {
            var box = el.getNext('div');
            if(undefined !==  box) {
                this.show(el, box);
                box.addEvents({
                    'mouseleave': function(){
                        this.timer = this.hide.delay(this.closeDelay,
this, box);
                    }.bind(this),
                    'mouseenter': function(){
                        clearTimeout(this.timer);
                    }.bind(this)
                });
            }
        },
        // Event sur les liens de la liste
        eventLinkList : function() {
            if(undefined !== this.sliderLink) {
                //On boucle sur tous les lien slider-link
                this.sliderLink.each(function(link){
                    //Si on clique
                    link.addEvent('mousedown', function(){
                        //On récupère l'attribut data-bt de l'image
                        var name =
link.getParent('div').getPrevious('img').getProperty('data-bt');
                        //On insère notre tag avec la valeur de
l'attribut rel du lien cliqué
                        this.insertTag('<'+name+'
value="'+link.get('rel')+'">', '</'+name+'>');
```

```
                        this.previewParser();
                    }.bind(this));
                }, this);
            }
        },
        /**
    * Events sur les boutons sélecteurs
    */
        eventList : function() {
            //Événements sur un bouton sélecteur
            this.listButtons.each(function(bt){
                if(undefined !== bt) {
                    bt.addEvents({
                        'mouseenter' : function(){ //On affiche la box
    et on écoute ses events
                            this.eventBoxList(bt);
                        }.bind(this),
                        'mouseleave' : function() { //On cache la box
                            this.timer = this.hide.delay(this.closeDelay,
    this, bt.getNext(this.options.slider));
                        }.bind(this)
                    });
                }
            }, this);
        },

        //Parse le texte via PHP pour la visualisation
        parseText : function(text) {
            var prev = this.preview;
            var req = new Request({
                url : 'parse.php',
                method : 'post',
                evalResponse : true,
                evalScripts : true,
                onComplete: function() {
                    prev.set('html', this.response.text);
                }
            });
            req.send('text='+escape(text));
        },
        //Visualisation dans la div id=prev
        previewParser : function() {
            var txt = this.input.value;
            this.parseText(txt);
        }
    });
```

Testez la démo en ligne !!!

Idées d'améliorations

Il reste quelques optimisations à faire, des fonctionnalités non implémentées, par exemple l'ajout de *smiley* ou la réduction et l'agrandissement du champ de texte grâce à deux boutons.
Voici, une exemple d'implémentation plus approfondis utilisant un parsage Xml, Xsl via Ajax :

IEditor 1.0
c'est un projet Open-Source pour y participer contactez-moi par mp.

Liens utiles

- Mootools
- Tester et déboguer vos scripts JS

- Comparaison des frameworks JavaScript

Ce tutoriel se termine, en espérant vous avoir un peu éclairés sur l'utilisation de Mootools.

franckysolo