

Introduction à XUL

Présentation générale

Présentation générale de XUL et de la plate-forme XPFE

XUL fait partie du framework Mozilla : il n'est pas indépendant de cette plateforme.

XPFE (Cross Platform Front-End) est né du projet de réaliser une plate-forme générique de développement d'applications capables d'accueillir un client de courrier, un éditeur HTML et un navigateur.

XPFE est couplé au moteur de rendu **NG Layout (New Generation Layout)**, que l'on connaît actuellement sous le nom de **Gecko**, pour proposer une plate-forme générique de développement.

XPFE est un framework applicatif : c'est un environnement pour accueillir des applications.

XUL est l'un des langages utilisés au sein de **XPFE** et à ce titre il permet de créer ses propres applications qui pourront ensuite être utilisées avec Mozilla.

XUL signifie (**XML User-interface Language**). **XUL** est une application du XML aux interfaces graphiques : **XUL** permet de décrire la structure de l'application, au même titre qu'un langage de balise décrit en XML décrit la structure d'un document.

XUL est un langage qui permet de construire l'interface graphique. **XUL** utilisé seul ne permet que de réaliser la structure de l'application : le placement des différents éléments, leurs attributs et leur organisation spatiale. Tout ceci permet de réaliser la partie « statique » de l'application.

Pour rendre l'application interactive, il faut décrire les actions et les comportements : ces aspects ne sont pas pris en compte par **XUL** mais reposent sur un langage de script, **JavaScript** en l'occurrence, issu du standard **ECMAScript** (voir <http://www.ecma-international.org/publications/standards/Ecma-262.htm>).

L'apparence d'une application **XUL** est contrôlée par des feuilles de styles **CSS (Cascading Style Sheets)** : avec **CSS** on peut contrôler l'aspect, les couleurs, la taille des éléments de l'application **XUL**, **XUL** en lui-même servant uniquement à décrire la structure de l'interface.

Organisation générale de la plate-forme XPFE

XPFE s'organisent autour de différentes parties dont **XUL** est un élément central.

XUL est intimement lié à **JavaScript** : les événements créés par **XUL** déclenchent un appel de code **JavaScript**. Ce code **JavaScript** interagit alors avec les éléments ou les événements associés à l'interface défini en **XUL**. Par exemple, une fonction JavaScript peut modifier la structure du document **XUL**.

Les interactions entre **XUL** et **JavaScript** se font par l'intermédiaire du **DOM (Document Object Model)**.

Le rendu du document **XUL** est à la charge du moteur de rendu **Gecko** : ce dernier analyse le document **XUL**, dessine sur le système d'affichage du client et prend

également en charge les modifications apportées au document **XUL** via le **DOM**.

La plateforme Mozilla est également constitué de composant **XPCOM** (**Cross Platform COM**)

voir les références suivantes :

- <http://www.mozilla.org/catalog/architecture/xpcom/>,
- <http://xulfr.org/wiki/XpCom>,
- http://dmoz.org/Computers/Programming/Component_Frameworks/XPCOM/,
- <http://fr.wikipedia.org/wiki/XPCOM>.

Un composant **XPCOM** possède une interface fixe qui sert de point de contact pour les autres langages et son implémentation peut varier en fonction de la plate-forme. Ces interfaces sont accessibles par **JavaScript** (via la technologie **XPCConnect**) mais pas seulement. En effet, un composant **XPCOM** peut être accessible à un autre langage. Il existe ainsi **piXPCOM**, **pyXPCOM** ou **rbXPCOM** (voir notamment <http://www.mozilla.org/catalog/architecture/xpcom/> pour des liens sur ces binding et sur le développement et l'utilisation de composant **XPCOM**) qui permet d'accéder respectivement à des composants de Mozilla depuis les langages **PERL**, **Python** et **Ruby**.

Création d'une application XUL

Structure d'une application XUL

On dénombre en général 5 grands types de fichiers XUL :

1. les fichiers pour la mise en page et les éléments,
2. ceux pour les styles,
3. ceux pour les déclarations d'entités (utilisées pour la localisation),
4. ceux pour les scripts,
5. éventuellement des fichiers supplémentaires pour les images ou les données spécifiques à la plateforme.

Travail préparatoire : installation des outils pour développer en XUL

Normalement un simple éditeur de texte suffit pour créer des applications **XUL** avec un navigateur récent supportant la plateforme **Mozilla** (**Firefox**, **Seamonkey**, etc.).

Il est également possible d'installer une application nommé **XULRunner** qui permet d'exécuter une application **XUL** indépendamment d'un navigateur. C'est l'outil que nous allons utiliser ici.

Nous allons installer un environnement de développement **XUL** qui facilitera notre travail de développement.

Dans un premier temps, nous allons installer **XULRunner**.

Sous <ftp://ftp.mozilla.org/pub/mozilla.org/xulrunner/nightly/latest-trunk>, récupérer le fichier « **xulrunner-1.9a1.en-US.win32.zip** ».

L'installation est simple : il suffit de désarchiver le fichier zip dans le repertoire dans lequel vous souhaitez qu'apparaisse le repertoire « **xulrunner** ».

Il reste après à ajouter le chemin vers le repertoire « **xulrunner** » dans la variable système **PATH**.

Pour vérifier que c'est ok, ouvrez une invite de commande DOS et taper **xulrunner -v**.

Quelques liens si vous voulez des informations supplémentaires sur **XULRunner** :

<http://xulfr.org/wiki/XulRunner>

<http://xulfr.org/wiki/XulRunner/Installation>

Nous allons ensuite récupérer l'**IDE Eclipse**. **Eclipse** est un environnement de développement intégré écrit en **Java** et particulièrement adapté pour les développements en ce langage. **Eclipse** est en fait une plateforme de développement et il est possible d'enrichir l'**IDE** de base avec des **plug-in**. Ainsi, il existe de nombreux **plug-in** pour développer dans d'autres langage que **Java** ou pour faciliter l'utilisation de certains outils. Il existe par exemple des **plug-in** pour le développement en **C/C++**, **JavaScript**, **PHP**, **ActionScript**, **XML** et également pour **XUL**.

Nous allons donc installer **Eclipse**. Il est nécessaire qu'une machine virtuelle Java (une « JVM ») récente soit installée sur votre système (un J2SDK supérieur ou égale au 1.4.2) Il faut récupérer le fichier **eclipse-SDK-3.1.2-win32.zip** que vous pourrez trouver à l'URL : <http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.1.2-200601181600/eclipse-SDK-3.1.2-win32.zip> .

Une fois le téléchargement effectué, il suffit de désarchiver le fichier dans le répertoire dans lequel vous souhaitez installer le logiciel.

Il vous suffit de lancer l'exécutable **eclipse.exe** pour lancer Eclipse (je vous recommande de vous créer un raccourci que vous installerez sur le bureau et dans le menu démarrer).

Une fois l'**IDE** ouvert, aller dans le menu **Help > Software Updates > Find and Install...**

Cocher la case « **Search for new features to install** ».

Nous voulons installer EclipseXUL, mais ce dernier a besoin que certains plugins Eclipse soient déjà installé, notamment des éléments que l'on trouve dans le projet WTP (Web Tools Project).

Sélectionner le bouton « **New remote site** » et entrer l'adresse « <http://download.eclipse.org/webtools/updates/> » avec comme nom **WTP update site** par exemple. Puis sélectionner le bouton « **Finish** ».

Ensuite, sélectionner le bouton « **New remote site** » et entrer l'adresse

« <http://eclipsexul.sourceforge.net/update-site> » avec comme nom **EclipseXUL** par exemple. Puis sélectionner le bouton « **Finish** ».

Enfin lancer la recherche, le téléchargement et l'installation de ces plugins. N'oubliez de vérifier que les paramètres réseaux d'Eclipse sont correctement renseignés (dans le menu **Window > Preferences**)

Si l'installation automatique ne fonctionne pas , il faut aller directement aux URL <http://www.eclipse.org/downloads/download.php?file=/webtools/downloads/drops/R-1.0.2-200604200208/wtp-all-in-one-sdk-R-1.0.2-200604200208-win32.zip> et <http://eclipsexul.sourceforge.net/chapter-installation.html> avec votre navigateur favori, télécharger le plug-in et installer « manuellement » les plug-ins (ce qui se résume à désarchiver les fichiers téléchargés dans le répertoire d'Eclipse).

Dans tous les cas, une fois le plug-in télécharger et installer, il faudra redémarrer **Eclipse**.

Une fois Eclipse redémarré, aller sous **Window > Preferences** et sélectionnez **Orangevolt XUL** puis **XULRunner** et indiquer le répertoire dans lequel **XULRunner** a été installé.

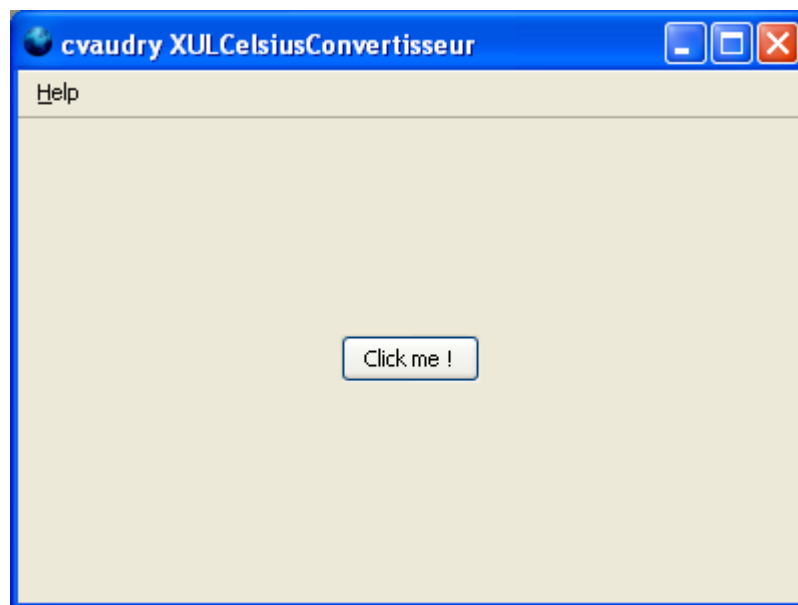
Maintenant si vous sélectionnez le menu **File > New > Other**, dans la fenêtre qui s'ouvre vous avez accès au « répertoire de modèle » Orangevolt et vous pouvez sélectionner « **XUL Application Project** ».

Créez un projet **XUL** du nom de XULCelsiusConvertisseur (par exemple), renseigner le champ « **vendor** » avec votre nom et l'URL vendor avec l'URL de votre site personnel ou à la rigueur avec l'URL de l'ISAIP Dijon ou du Groupe Saint-Joseph (ce n'est pas très important ici). Pensez à écrire l'url complètement avec le http devant, comme dans l'exemple <http://www.isaip.org>.

Pour le numéro de version, laissez 0.1 et pour les version minimum et maximum de Gecko, mettez respectivement 1.8 et 1.9a.

Cliquez sur **Finish** et cliquez sur **Ok** dans la fenêtre modale qui s'ouvre.

Le projet vous génère tout un ensemble de fichier **XUL** et **JavaScript** par défaut, répartis dans un ensemble de répertoires. Opérez un clic droit sur le nom de votre projet (normalement **XULCelsiusConvertisseur**) dans la vue de gauche d'**Eclipse**. Dans le menu contextuel qui apparaît choisir **Run As > XUL Application**. Vous devez normalement voir apparaître la fenêtre suivante au bout de quelques secondes :



Ouvrez dans la vue centrale les 4 fichiers que vous trouverez sous **chrome > content > XULCelsiusConvertisseur** :

1. **XULCelsiusConvertisseur.xul**,
2. **XULCelsiusConvertisseur.js**,
3. **about.xul**,
4. **about.js**.

Ouvrez également le fichier de DTD **XULCelsiusConvertisseur.dtd** qui se trouve sous **locale > fr_FR > XULCelsiusConvertisseur**.

Vous noterez que pour les fichiers avec l'extension « **.xul** », nous avons 2 vues possibles « **Source** » et « **Design** », accessibles via des onglets de même nom en bas à gauche de la vue.

Il sera plus clair dans un premier temps d'utiliser le mode « **Source** » pour définir notre interface.

Une première application XUL vue pas à pas

Dans le fichier XULCelsiusConvertisseur vous devez avoir pour l'instant le source suivant :

```
<?xml version="1.0"?>
<?xmlstylesheet href="chrome://global/skin/" type="text/css"?>

<!DOCTYPE window SYSTEM
    "chrome://XULCelsiusConvertisseur/locale/XULCelsiusConvertisseur.dtd">

<window id = "&app.id;"
        title = "&app.title;"
        width = "400"
        height = "300"
        xmlns = "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
        onload = "onLoad();">

    <script src="XULCelsiusConvertisseur.js"/>

    <menubar>
        <menu label="&menuHelp.label;"
              accesskey="&menuHelp.accesskey;">
            <menupopup>
                <menuitem label="&menuitemAbout.label;"
                           accesskey="&menuitemAbout.accesskey;"
                           oncommand="onAbout();"/>
            </menupopup>
        </menu>
    </menubar>

    <vbox flex="1">
        <spacer flex="1"/>
        <hbox>
            <spacer flex="1"/>
            <button label="Click me !" oncommand="onClickMe();" />

            <spacer flex="1"/>
        </hbox>
        <spacer flex="1"/>
    </vbox>
</window>
```

On note déjà que le fichier XUL est bien un fichier XML.

Nous allons modifier la première ligne en précisant l'encodage de notre fichier : nous allons utiliser l'encodage iso-8859-1 pour l'alphabet des pays européens, ce qui permettra d'identifier les caractères accentués.

La première ligne devient donc :

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

On indique que notre document XUL suit le style général du navigateur ou de l'application qui l'exécute par la ligne :

```
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
```

La ligne :

```
<!DOCTYPE window SYSTEM  
"chrome://XULCelsiusConvertisseur/locale/XULCelsiusConvertisseur.dtd">
```

Nous permet de charger le fichier DTD **XULCelsiusConvertisseur.dtd** qui contient des définitions d'entités qui sont utilisées comme valeur pour des attributs de certains des éléments de notre document XUL :

- **&app.id;** --> **<!ENTITY app.id "test">**
- **&app.title;** --> **<!ENTITY app.title "XULCelsiusConvertisseur">**
- **&menuHelp.label;** --> **<!ENTITY menuHelp.label "Help">**
- **&menuHelp.accesskey;** --> **<!ENTITY menuHelp.accesskey "H">**
- **&menuItemAbout.label;** --> **<!ENTITY menuItemAbout.label "About XULCelsiusConvertisseur">**
- **&menuItemAbout.accesskey;** --> **<!ENTITY menuItemAbout.accesskey "A">**

L'élément racine de notre document XUL est l'élément **<window></window>**

On spécifie l'espace de nommage de cet élément avec l'attribut **xmlns** et sa valeur qui correspond à l'URL <http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul>.

Avec les attributs **width** et **height** on spécifie les dimensions de la fenêtre.

L'attribut **onLoad** permet d'associer une fonction JavaScript à exécuter lors de l'événement « on load » qui correspond au chargement de la page.

La balise **<script/>** nous permet de spécifier le fichier JavaScript, contenant les fonctions qui seront associés aux événements de l'interface.

```
<script src="XULCelsiusConvertisseur.js"/>
```

Le code ci-après, placé dans un élément **<window>...</window>** permet de définir une barre de menus dans la fenêtre.

A l'intérieur de l'élément **<menubar>...</menubar>** on place des menus avec les balises **<menu> ... </menu>**.

Les attributs **label** et **accesskey** de l'élément **menu** permettent respectivement de définir le nom du **menu** et un raccourci-clavier permettant d'activer ce menu directement.

On notera que l'on utilise ici des entités définies dans une DTD séparée comme valeur d'attribut.

L'élément **menupopup** permet de définir le menu déroulant qui s'activera quand on sélectionnera l'item **menu**.

L'élément **<menuItem/>** permet de définir un élément de ce menu déroulant.

L'attribut **oncommand** de **menuItem** permet d'associer une fonction **JavaScript** à déclencher lors de la sélection de cet item dans le menu déroulant.

```

<menubar>
  <menu label="&menuHelp.label;"
        accesskey="&menuHelp.accesskey;">
    <menupopup>
      <menuitem label="&menuitemAbout.label;"
                accesskey="&menuitemAbout.accesskey;"
                oncommand="onAbout();" />
    </menupopup>
  </menu>
</menubar>

```

L'élément fenêtre définit avec les balises **<window>** **</window>** se comporte par défaut comme une boîte verticale.

Ainsi les éléments placés dans la la fenêtre sont placés en étages et prennent toutes la largeur qui leur est offerte.

Ainsi quand nous définissons une interface en XUL nous placerons nos éléments dans des boîtes que nous placerons elles-mêmes éventuellement dans des boîtes ou directement dans la fenêtre qui est elle-même une boîte comme nous l'avons vu.

Un élément vbox (**<vbox>**...**</vbox>**) permet de définir une boîte verticale.

Un élément hbox (**<hbox>**...**</hbox>**) permet de définir une boîte horizontale : dans une boîte horizontale, les éléments prennent toute la hauteur possible et s'entassent de gauche à droite.

Les éléments spacer (**<spacer/>**) un élément **XUL** qui se charge d'occuper tout l'espace disponible dans une boîte. Il possède, comme la plupart des éléments **XUL**, un attribut **flex** qui permet de dimensionner proportionnellement les éléments d'une même boîte.

Quand on ne précise pas d'attribut **flex** pour un élément, ce dernier prend tout l'espace qui lui est nécessaire. Si on précise l'attribut **flex**, pour les éléments dans une même boîte, la valeur de ce **flex**, indique comment les éléments se partagent l'espace restant.

Dans notre projet, pour l'instant, si on considère l'élément **hbox** :

L'élément bouton défini par **<button/>** prend tout l'espace qui lui est nécessaire dans la boîte horizontale. Les 2 éléments **spacer** se répartissent équitablement l'espace restant. Si pour l'un des 2 avait été défini un **flex** de 2 et l'autre un flex de 1, on aurait eu une répartition 2/3 (pour celui ayant **flex="2"**) et 1/3 (pour celui ayant **flex="1"**).

```

<vbox flex="1">
  <spacer flex="1"/>
  <hbox>
    <spacer flex="1"/>
    <button label="Click me !" oncommand="onClickMe();" />
    <spacer flex="1"/>
  </hbox>
  <spacer flex="1"/>
</vbox>

```

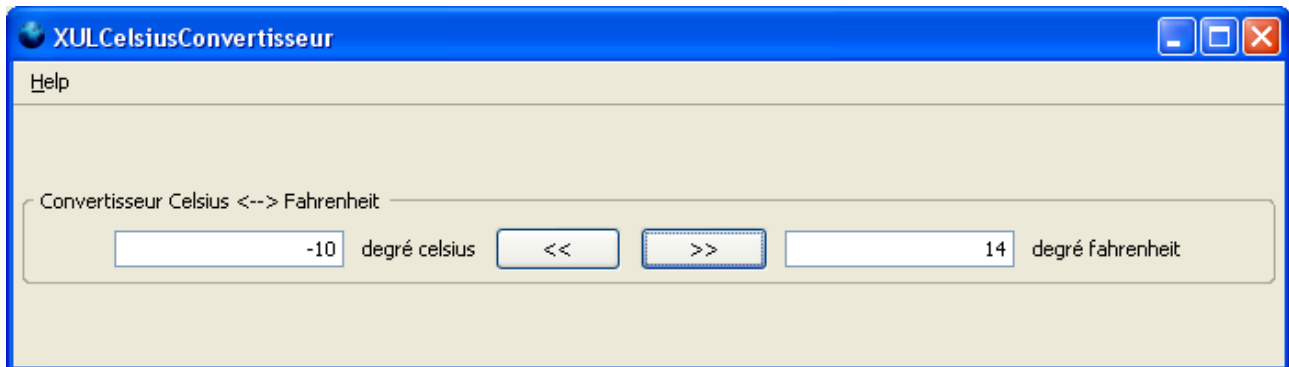
De même dans l'élément vbox, la hbox prend tout l'espace qui lui est nécessaire et les 2 éléments spacer se répartissent la palce qui restent. C'est cette combinaison d'élément dans les boîtes qui nous permet d'avoir le bouton centré dans le document.

Construction de l'application XUL Celsius Convertisseur pas à pas

Nous avons vu un certain nombre d'éléments de base. A partir de ces éléments et de quelques autres, nous allons commencer à construire notre application de conversion Celsius vers Fahrenheit.

On rappelle que pour convertir F degrés Fahrenheit en C degrés Celsius nous avons la formule suivante : $C = 5/9 (F - 32)$.

Ci-après, une copie d'écran de ce que nous souhaiterions obtenir :



Notre application utilise peu d'éléments différents :

- des champs de textes : élément **textbox** --> **<textbox/>**
- des boutons : élément **button** --> **<button/>**
- des descriptions textuels : élément **description** --> **<description/>**
- des labels : élément **label** --> **<label/>**
- des boîtes de groupes : élément **groupbox** --> **<groupbox>...</groupbox>**
- des boîtes horizontales : élément **hbox** --> **<hbox>...</hbox>**
- des espacements : élément **spacer** --> **<spacer/>**

Ces différents éléments XUL possèdent bien sûr des attributs.

Nous allons d'abord réaliser une première version simple :

Nous prévoyons une boîte horizontale dans laquelle nous regrouperons nos différents éléments. Puis nous placerons dans cette boîte, un champ de texte, un label, 2 boutons, un champ de texte et un dernier label.

Voilà ce que cela peut donner :

```
<hbox>
  <textbox id="champ_celsius"/>
  <label value="degré celsius"/>
  <button label="&lt;&lt;"
    id="fahrenheit_vers_celsius"/>
  <button label="&gt;&gt;" id="celsius_vers_fahrenheit"/>
  <textbox id="champ_fahrenheit"/>
  <label value="degré fahrenheit"/>
</hbox>
```

L'attribut **id** permet de définir l'identifiant d'un élément. L'attribut **label** pour un bouton permet de définir le texte qui sera présent dessus. L'attribut **value** pour un élément **label**, **description** ou **textbox** permet de définir un texte pour cet élément.

Une proposition de solution

Ci-après vous trouverez une proposition de code pour les fichiers XULCelsiusConvertisseur.xul et XULCelsiusConvertisseur.js

XULCelsiusConvertisseur.xul

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<!DOCTYPE window SYSTEM
    "chrome://XULCelsiusConvertisseur/locale/XULCelsiusConvertisseur.dtd">
<window
    id = "&app.id;"
    title = "&app.title;"
    width = "700"
    height = "200"
    xmlns = "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
    onload = "onLoad();"

    <script src="XULCelsiusConvertisseur.js"/>

    <menubar>
        <menu label="&menuHelp.label;"
            accesskey="&menuHelp.accesskey;">
            <menupopup>
                <menuitem label="&menuitemAbout.label;"
                    accesskey="&menuitemAbout.accesskey;"
                    oncommand="onAbout();"/>
            </menupopup>
        </menu>
    </menubar>

    <spacer flex="1"/>
    <groupbox>
        <caption label="Convertisseur Celsius &lt;--&gt; Fahrenheit"/>
        <hbox align="center">
            <spacer flex="1"/>
            <textbox id="champ_celsius" style="text-align : right"/>
            <label value="degré celsius" control="champ_celsius"/>
            <button label="&lt;&lt;"
                id="fahrenheit_vers_celsius"
                oncommand="convertir_fahrenheit_vers_celsius();"/>
            <button label="&gt;&gt;"
                id="celsius_vers_fahrenheit"
                oncommand="convertir_celsius_vers_fahrenheit();"/>
            <textbox id="champ_fahrenheit" style="text-align : right"/>
            <label value="degré fahrenheit" control="champ_fahrenheit"/>
            <spacer flex="1"/>
        </hbox>
    </groupbox>
    <spacer flex="1"/>
</window>
```

XULCelsiusConvertisseur.js

```
function convertir_fahrenheit_vers_celsius(event)
{
    var fahrenheit = document.getElementById('champ_fahrenheit').value;
    fahrenheit = parseFloat(fahrenheit);
    var celsius = (5.0 * (fahrenheit - 32.0) ) / 9.0 ;
    document.getElementById('champ_celsius').value = celsius;
}

function convertir_celsius_vers_fahrenheit(event)
{
    var celsius = document.getElementById('champ_celsius').value;
    celsius = parseFloat(celsius);
    var fahrenheit = 32 + (9.0 * celsius) / 5.0 ;
    document.getElementById('champ_fahrenheit').value = fahrenheit;
}

function onLoad()
{
    enableDump( true);

    println( "application initialized.");
}

function onAbout( event)
{
    println( "onAbout activated.");

    window.openDialog( "about.xul", "_blank", "chrome,close,modal");
}

function println( s)
{
    dump( s + "\n");
}

function print( s)
{
    dump( s);
}

function showError(objError)
{
    var sMsg;

    with (objError)
    {
        sMsg = 'NAME: ' + name;
        sMsg += '\nMESSAGE: ' + message;
        sMsg += '\n\nFILE: ' + fileName;
        sMsg += '\nLINE: ' + lineNumber;
    }
}
```

```

    alert(sMsg);
}

function printError(objError)
{
    var sMsg;

    with (objError)
    {
        sMsg = 'NAME: ' + name;
        sMsg += '\nMESSAGE: ' + message;
        sMsg += '\n\nFILE: ' + fileName;
        sMsg += '\nLINE: ' + lineNumber;
    }

    println( sMsg);
}

function enableDump( bool)
{
    const PREFS_CID      = "@mozilla.org/preferences;1";
    const PREFS_I_PREF  = "nsIPref";
    const PREF_STRING   = "browser.dom.window.dump.enabled";
    try
    {
        var Pref = new Components.Constructor(PREFS_CID,PREFS_I_PREF);
        var pref = new Pref();
        pref.SetBoolPref(PREF_STRING, bool);
    }
    catch( e)
    {
        showError( e);
    }
}

```

Rapide webographie

Pour l'instant il y a peu de ressource sur XUL, néanmoins on peut trouver des tutoriels très complets en français à l'URL <http://xulfr.org/xulplanet/> .