Animations en XML

Pierre-Alain Reynier
http://www.lif.univ-mrs.fr/~preynier/XML/

Quels objectifs ?

- Décrire en XML des documents permettant de créer des animations, des interactions avec l'utilisateur

- Principal cadre d'application : le Web

- Intérêt du XML : (comme SVG)
  - insertion dans le monde XML
  - textes indexables

Plusieurs possibilités :

- SMIL : Synchronized Multimedia Integration Language
  adapté à la présentation de contenus multimedia

- SVG animé
  permet d'animer les objets créés en SVG

- XUL : XML User-interface Language
  permet de construire des interfaces utilisateur graphiques avec des balises XML

Nous allons voir :

- SMIL : présentation (très) sommaire

- SVG animé : présentation + détaillée
  le modèle d'animations est proche de celui utilisé dans SMIL

- XUL : présentation + détaillée

SMIL

Synchronized Multimedia Integration Language

SMIL : pourquoi ?

- Le web est très disparate :
  - nombreux formats et médias
  - le web n'a pas été pensé multimédia

- Problèmes d'intégration :
  - entre les médias
  - avec le web

SMIL : objectifs

- Ajouter des possibilités de synchronisation de données multimédia au web

- Mécanisme de référencement des médias

- Langage ouvert, éditable à la main

SMIL est / n'est pas

- SMIL est :
  une collection d'éléments et d'attributs XML que l'on utilise pour décrire la coordination spatiale et temporelle d'un ou plusieurs objets multimédia.

- SMIL n'est pas :
  - un remplaçant de Flash
  - un remplaçant de Mpeg4
  - un remplaçant de HTML Dynamique
  - un remplaçant des applets Java

SMIL : quelques références

- La documentation W3C :
  http://www.w3.org/AudioVideo/
- Test officiels :
  http://www.w3.org/2001/SMIL20/testsuite/
- Hot News !
  01/12/08 : SMIL 3.0 :
  http://www.w3.org/TR/2008/REC-SMIL3-20081201/

SVG animé

Scalable Vector Graphics

SVG animé : introduction

- 2 façons d'ajouter de l'animation à SVG :
  - utiliser des attributs et des éléments propres à SVG
  - utiliser JavaScript + DOM

- Possibilité de définir des animations et des interactions avec l'utilisateur

- La syntaxe est très proche de celle de SMIL.

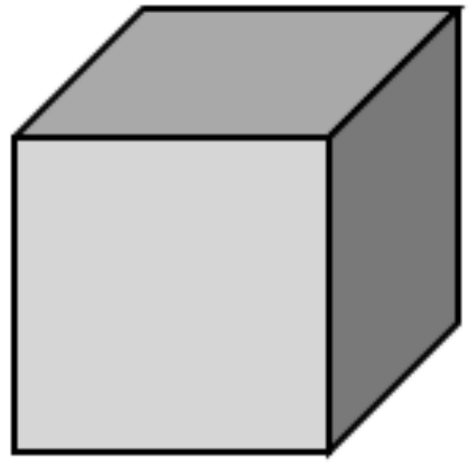
SVG animé : plan

- Animations générale : animate
- Animations spécifiques :
  - couleurs
  - transformations
- Interactions

SVG animé : animate

- On considère le graphisme suivant :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE svg SYSTEM "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="500" height="500" xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <symbol id="cube" stroke="black" stroke-line-join="bevel"
stroke-width="2">
      <rect width="100" height="100" fill="#ccc" x="1" y="42" />
      <polygon points="1,42 42,1 142,1 101,42 1,42" stroke-
width="2" fill="#999" />
      <polygon points="101,42 142,1 142,101 101,142 101,42"
fill="#666" />
    </symbol>
  </defs>
  <use xlink:href="#cube" x="150" y="150" />
</svg>
```



SVG animé : attributs

- On modifie la valeur de l'attribut y afin de faire monter et descendre le cube :

```
<use xlink:href="#cube" x="150" y="150">
  <animate attributeName="y" dur="2s" values="150;
    140; 130; 120; 110; 100; 110; 120; 130; 140; 150" />
</use>
```

- Effet attendu : le cube, dans une durée de 2 secondes, parcourt les points décrits.
- N'importe quel attribut peut être animé de cette façon, à l'exception des attributs de couleur et de transformation (voir plus loin)

SVG animé : attributs

- Par défaut, le comportement est obtenu en calculant une interpolation linéaire des valeurs données
- Il est possible de faire des boucles : attribut `repeatCount`
- Il est possible de spécifier des dates de début et de fin : attributs `begin` et `end`
- Possible enfin de synchroniser plusieurs animations...

SVG animé : from et to

- On peut spécifier l'intervalle d'évolution par le raccourci `from` et `to` :

```
<animate attributeName="x" dur="3s" from="150"
to="100"/>
```

- Le comportement est obtenu en calculant une interpolation linéaire entre ces deux valeurs.

SVG animé : fill

- Par défaut, à l'issue de l'animation, la figure revient à son état initial.
- Pour changer ce comportement et conserver l'état final, on utilise `fill="freeze"` :

```
<animate attributeName="x" dur="3s" from="150"
to="100" fill="freeze"/>
```

SVG animé : répétition

- attribut `repeatCount` : répétition un nombre déterminé de fois
- Valeur `indefinite` pour répétition un nombre infini de fois
- Exemple :

```
<animate attributeName="y" dur="3s" values="150;
140; 130; 120; 110; 100; 110; 120; 130; 140; 150"
repeatCount="3"/>
```

SVG animé : Couleurs

- élément `animateColor`
- attribut `attributeType` défini à la valeur CSS (par défaut XML)
- Autrement, même syntaxe
- Exemple :

```
<animateColor attributeName="fill"
attributeType="CSS" begin="2s" dur="3s" from="grey"
to="blue" fill="freeze" />
```

SVG animé : Transformations

- élément `animateTransform`
- attribut `type` définit le genre de trasnformation
- Quelques exemples :

- translation :

```
<animateTransform attributeName="transform"
type="translate" from="0,0"  to="100,100" ... />
```

- redimensionnement :

```
<animateTransform attributeName="transform"
type="scale" from="1"  to="4 2" ... />
```

SVG animé : Superpositions

- Par défaut une animation écrase les animations précédentes
- Ce comportment peut être modifié en renseignant l'attribut `additive` à la valeur `sum`

SVG animé : Interactions

- 2 modes d'utilisation
- Mode simple :
  on définit un graphisme avec nom (`id`) : `button`
  on associe le début ou la fin d'une animation à une interaction entre cette zone et la souris par la séquence `begin="bouton.click"` ou encore `begin="bouton.mouseover"`
- Mode + complexe : JavaScript

SVG animé : Interactions

- Mode + complexe : JavaScript :
- Balise `script` : définition de fonctions JavaScript
- Appel à ces fonctions via l'attribut `on_nomEvt_` : `onmouseover="elargir_cercle(evt)"`

```
<svg width="8cm" height="8cm">
<script type=
"text/ecmascript">
 function elargir(evt){
var cercle = evt.getTarget();
cercle.setAttribute("r", 50);}

 function reduire(evt){
 var cercle = evt.getTarget();
 cercle.setAttribute("r", 25);}
</script>
```

```
<circle cx="150" cy="100"
r="25" fill="red"
onmouseover="elargir(evt)"
onmouseout="reduire(evt)"/>
<text x="150" y="175"
style="text-anchor:middle;">
Passer la souris sur le
cercle pour changer sa taille.
</text>
</svg>
```

SVG animé : Conclusion

- Langage simple pour décrire des animations simples
- Quelques possibilités d'interactions avec la souris (clic, double clic, passage, départ...)
- Quelques possibilités supplémentaires (synchronisation...)

SVG animé : Qqs références

- Site de la W3C : http://www.w3.org/Graphics/SVG/
- Support SVG dans Mozilla/Firefox : https://developer.mozilla.org/fr/SVG
- Site d'Adobe sur SVG : http://www.adobe.com/svg/

XUL

XML User Interface Language

XUL : introduction

- XUL veut dire : XML User-interface Language
- XUL se prononce “zool”
- Historique : XUL a été créé pour faciliter le développement du navigateur Mozilla
- XUL doit permettre de développer facilement une interface graphique agréable multi-plateformes

XUL : fonctionnement

- repose sur le moteur Gecko de Mozilla
- peut être utilisé via :
  - un navigateur Gecko (Mozilla, Firefox)
  - Internet Explorer avec un plug-in
  - un outil indépendant : XulRunner
- peut intégrer d'autres formats XML : XHTML, SVG, MathML...
- souvent utilisé pour des extensions Firefox
- interaction avec JavaScript

XUL : pour faire quoi ?

- Champs de saisie tels que des boîtes de textes et des cases à cocher
- Barres d'outils avec boutons et autres contenus
- Menus dans des barres de menus ou des menus surgissants (contextuels)
- Boîtes de dialogues à onglets
- Arbres de données hiérarchiques ou tabulaires
- Raccourcis claviers

XUL : structure d'un fichier

- XUL est une application XML :
  - espace de nommage :

`xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"`

  - ensemble d'éléments (balises) et d'attributs

- Balise principale : `window`
  Attributs : `id`, `width`, `height`, `title`

XUL : quelques fonctions

- Fenêtres : window
- Boîtes : box, hbox, vbox
- Boutons (liens JavaScript) : button
- Textes : textbox
- Images : image
- Listes : listbox
- Menus : menubar
- Barre d'outils : toolbar
- Barre de progression : progressmeter
- Liste hiérarchique : tree, treecol...
- Fichiers RDF
- Code HTML : html
- Evénements

XUL : window

- On a déjà vu les attributs de la balise window : id, title, width, height

- Ouverture d'une fenêtre XUL via JavaScript :

```
window.open("hello.xul", "hello",
"chrome,width=400,height=300");
```

XUL : exemple window (html)

Définition d'une fonction JavaScript :

```
<script language="JavaScript">
  function winopen()
  {
  window.open("hello.xul", "hello",
  "chrome,width=400,height=300");
  }
</script>
```

Appel de la fonction dans du code HTML :

```
<form method="get" name="win"> <input value="Open"
name="opwin" onclick="winopen();" type="button">
</form>
```

XUL : boîtes

- Balise générale : `box`
- Enchaînement horizontal de textes : `hbox`
- Enchaînement vertical de textes : `vbox`
- Regroupement de boîtes : `groupbox`
- Effets d'ombrage par superposition : `stack`

XUL : exemples de boîtes

```
<box>
  <hbox>
    <box>
      <description
      value="Panel gauche" />
    </box>
    <vbox>
      <box>
        <description
    value=" Panel droite haut" />
      </box>
      <box>
        <description
    value=" Panel droite bas" />
      </box>
    </vbox>
  </hbox>
</box>
```

Panel gauche Panel droite haut

Panel droite bas

XUL : boutons

- Balise générale : `button`
- Principaux attributs :
  `label` : afficher un texte
  `image` : associer une image au bouton
  `type` : définit le genre de bouton (menu, menu-button)
  `oncommand` : déclenche une fonction javascript

XUL : exemple de bouton

```
<script>
function clicking(txt)
{
   alert(txt);
}
</script>

<box>
<button label=" Sauver"
   oncommand="clicking('clic');" />
<button label="Load" type="menu"/>
</box>
```

XUL : boîte de texte

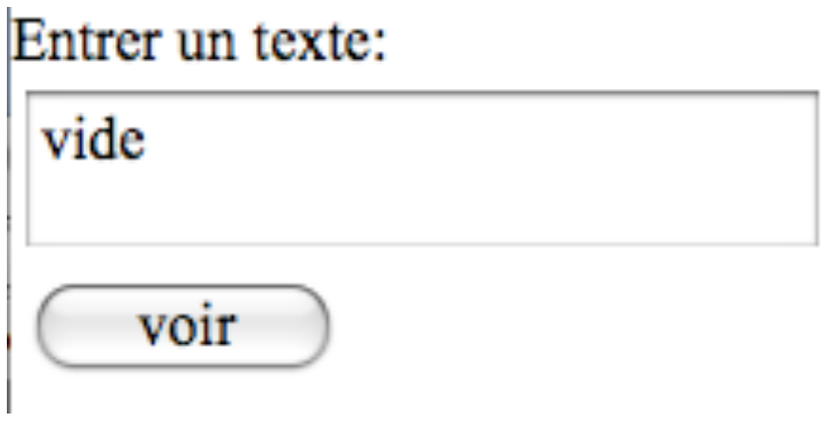
- Balise générale : `textbox`
- Principaux attributs (self-explained) : `multiline, rows, value, maxlength, size...`
- Types de textbox (en plus du default) : `autocomplete, password, timed`
- Balise `label` : s'utilise en conjonction avec `textbox` via l'attribut control (cf exple)

XUL : exple de boîte textuelle

```
<label control="tb" value="Entrer un texte:" />

<textbox id="tb" multiline="true" rows="1"
size="40" value="vide" />

<button label="voir" oncommand="alert(
document.getElementById('tb').value);" />
```



XUL : images

- Insertion d'images : `image`

```
<image src="photo.jpg" />
```

- Utilisation pour un bouton :

```
<button label="Sauver" image="save.gif"
orient="vertical"  />
```

XUL : listes et tableaux

- Listes : `listbox, listitem`
- Tableaux : des balises en plus !

`listhead, listheader` : en-têtes
`listcols, listcol` : définit les colonnes
`listitem, listcell` : définit les cellules

XUL : exple de tableau

```
<listbox>
  <listhead>
   <listheader label="Nom"></listheader>
   <listheader label="Age"></listheader>
   <listheader label="Ville"></listheader>
  </listhead>
  <listcols>
   <listcol flex="1"></listcol>
   <listcol flex="1"></listcol>
  </listcols>
  <listitem>
   <listcell label="Julia"></listcell>
   <listcell label="19"></listcell>
   <listcell label="Boston"></listcell>
  </listitem>
 <listitem>...</listitem>
  <listitem>...</listitem>
</listbox>
```

Nom	Age	Ville
Julia	19	Boston
Sandra	25	London
Sharon	35	Paris

XUL : menus

- On inclut le tout dans un `toolbox`
- Barre horizontale de menu : `menubar`
- Liste d'éléments de menus : `menu`
- Elément déroulant : `menupopup`
- Elément de menu : `menuitem`
- Attributs de `menuitem` proches de ceux de `button`

XUL : exple de menu

```
<toolbox flex="1">
  <menubar>
     <menu label="Fichiers">
        <menupopup>
            <menuitem id="openMenu" label="Ouvrir"
onCommand="openFun" />
            <menuitem id="closeMenu" label="Fermer" />
            <menuseparator />
            <menuitem id="exitMenu" label="Quitter" />
        </menupopup>
     </menu>
     <menu label="Aide" >
        <menupopup>
            <menuitem label="Manuel" />
        </menupopup>
     </menu>
 </menubar>
</toolbox>
```

XUL : barres d'outils

- On inclut le tout dans un `toolbox`
- Barre horizontale d'outils : `toolbar`
- Boutons : `toolbarbutton`
- Icônes : `toolbargrippy`
- Elément dans la barre : `toolbaritem`
- Gestion d'événements : voir plus loin

XUL : exple de barre d'outils

```
<toolbox>
   <toolbar>
      <toolbarbutton id="openButton" image="open.gif"  />
      <toolbarbutton id="saveButton" image="save.gif"  />
      <toolbarseparator />
      <toolbaritem>
         <description value="Entrer une URL " />
         <textbox>
         </textbox>
      </toolbaritem>
   </toolbar>
</toolbox>
```

XUL : barre de progression

- Elément principal : `progressmeter`
- Valeur initiale : `value`
- Possibilité de modifier la valeur d'avancement avec une fonction JavaScript :

```
<hbox width="400" >
   <progressmeter id="MonPM" width="400"
     value="0" />
</hbox>
<script>
   var mpm = document.getElementById("MonPM");
   function setProgress(y){mpm.value = y;}
</script>
```

XUL : arbres

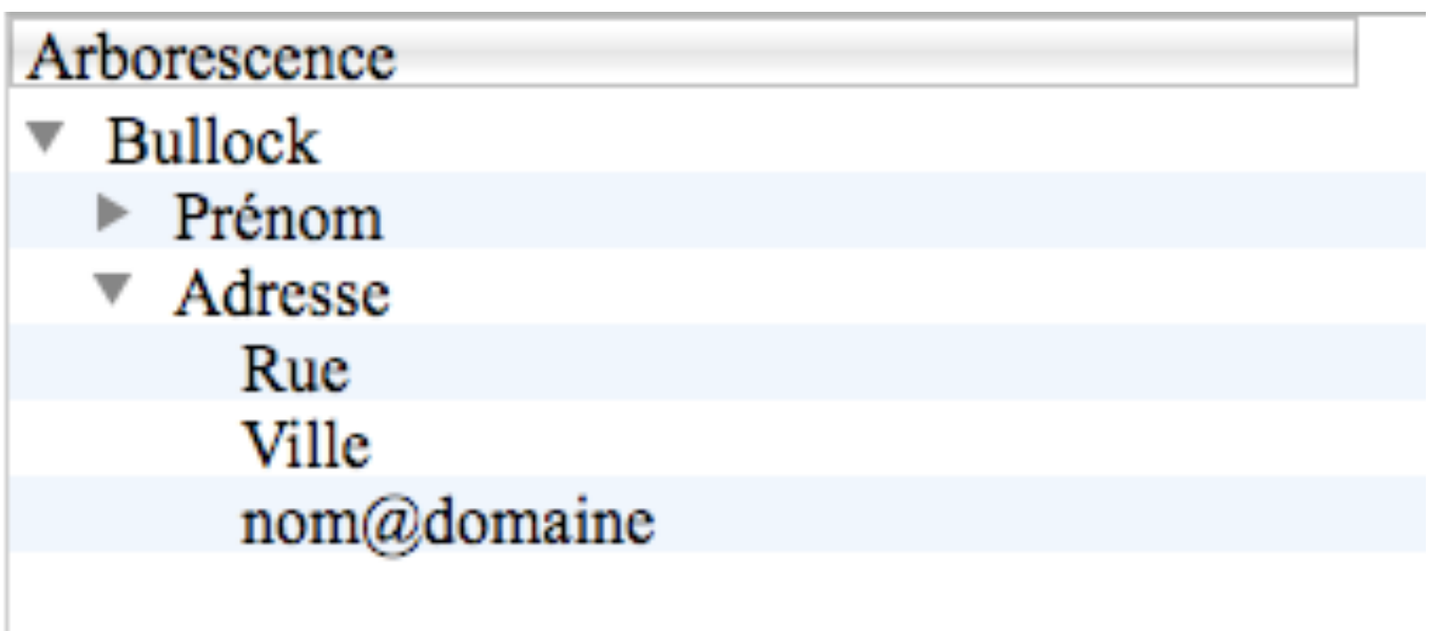
- L'objet `tree` est une composante arborescente :
  - destinée à contenir et afficher des données chargées dynamiquement
  - le contenu peut varier au cours de l'utilisation
  - peut prendre la forme d'une liste, d'une table ou d'une arborescence
- Contenu différent de l'affichage :
  - partie affichée : l'objet `tree`
  - contenu dynamique `treeview`

XUL : arbres

- Eléments de `tree` :
  `treecols, treecol` : colonnes
  `treechildren` : début d'arborescence
  `treeitem` : un élément de l'arborescence
  `treerow` : une ligne de la table
  `treecell` : une cellule
- Rq : un `treeitem` contient un `treerow` plus (evt) des `treechildren`.
- Contenus dynamiques : plus complexe, utilisation de fichiers RDF

XUL : exple d'arborescence

```
<tree flex="1">
 <treecols >
  <treecol label="Arborescence" primary="true"
   width="320" />
 </treecols>
  <treechildren>
   <treeitem container="true" open="true" >
    <treerow>
      <treecell label="Bullock" />
    </treerow>
    <treechildren>
     <treeitem container="true" open="true">
      <treerow>
       <treecell label="PrÃ©nom" />
      </treerow>
          [...]
     </treeitem>
    </treechildren>
   </treeitem>
  </treechildren>
</tree>
```



XUL : événements DOM

- Permet d'associer un événement à un objet DOM de l'interface graphique
- L'objet est reconnu par son id
- L'id est saisi par `getElementByID` :

```
var x = document.GetElementById("bx");
x.addEventListener("command", mafonction, true);
```

- Evénements possibles : `command`, `mousemove`, `click`, `close`...

XUL : événements DOM

- L'objet event : en pratique, chaque gestionnaire d'événements dispose d'un unique argument qui contient un objet event.
- Permet de définir des fonctions plus facilement :

```
function mafonction(event){
alert("Coordonnées: " + event.clientX + " " +
event.clientY);
}
button.addEventListener("click", mafonction,
true);
```

XUL : conclusion

- Langage XML pour des interfaces utilisateurs
- Associé à JavaScript et DOM pour associer facilement des actions aux commandes de l'interface
- Styles redéfinissables, mais style Firefox de base très agréable :

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

XUL : quelques références

- Un site avec beaucoup d'exemples : http://www.hevanet.com/acorbin/xul/top.xul

- Le tutoriel Mozilla très complet : https://developer.mozilla.org/Fr/Tutoriel_XUL

- Un tutoriel plus succinct : http://www.xul.fr/tutoriel/