UNIVERSITÉ DE LA ROCHELLE - UFR SCIENCES ET TECHNOLOGIE
Licence Professionnelle Création Multimédia

TP Image - le format SVG

I. Introduction

Le **Scalable Vector Graphics** ou SVG, est un format de données conçu pour décrire des ensembles de graphiques vectoriels, et basé sur XML. Ce format est spécifié par le World Wide Web Consortium (W3C), qui, en septembre 2001, publie la "Scalable Vector Graphics (SVG) 1.0 Specification, W3C Recommendation". La version 2 est en cours de définition.
Le SVG a été très vite adopté dans le monde de la cartographie.
Avec SVG, les images deviennent lisibles, et peuvent se manipuler avec du CSS ou du Javascript.

Dessinons un cercle avec l'outil Ellipse d'Adobe Illustrator, et enregistrons-le au format SVG.
Le document peut s'ouvrir dans n'importe quel éditeur de texte :
(ici, en simplifiant un peu le fichier obtenu par AI)

```
<svg version="1.1" baseProfile="basic" id="Calque_1"
	xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
viewBox="0 0 600 600"
	xml:space="preserve">
<circle fill="#FF00FF" stroke="#00FF00" stroke-width="10" cx="200" cy="200" r="150"/>
</svg>
```

Le cercle peut être vu par coup d'oeil de MacOs, ouvert avec Chrome, Safari, IE9, …
Recopier le code dans la balise body d'un fichier HTML et utiliser Brackets.

Remarques :

- Un document SVG est donc « editable » en mode texte (contrairement à des fichiers SWF par exemple), il est compatible avec d'autres technologies : SMIL (Synchronized Multimedia Integration Language), CSS, JavaScript sans nécessiter donc l'apprentissage de nouvelles. Composé de texte, il peut donc être indexé. En cartographie, une recherche sur un nom de lieu pourra être fait sur une carte par exemple (des traductions peuvent être facilement réalisées, etc.)

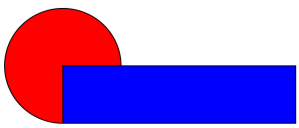
- Un document SVG a l'extension .svg ou .svgz, lorsqu'il est compressé selon la technologie GZIP.

- SVG étant une application de XML, les règles de syntaxe de XML s'appliquent. En particulier :
1. Les balises sont sensibles à la casse : <aa> est différent de <aA>
2. Toutes les balises doivent être fermées <tag>...</tag> ou <tag/>
3. Toute valeur d'attribut, même un nombre, est entre "quotes".
	Exemple : <circle r="70"/>

- Un document SVG est affiché de bas en haut : les figures suivantes dans le fichier sont placées au-dessus de celles qui les précèdent. Exemple :

```
<circle fill="#FF0000" stroke="#000000" cx="120" cy="120" r="60"/>
<rect x="120" y="120" fill="#0000FF" stroke="#000000" width="240" height="60"/>
```

est affiché ainsi :



L'attribut viewBox et sa valeur viewBox="0 0 600 600 » définissent l'apparence initiale du cercle. Ici, le centre étant aux coordonnées 200 200, il sera placé au tiers de la fenêtre. Son rayon de 150 sera 4 fois plus petit que la largeur de la fenêtre. Vérifier l'apparence avec d'autres valeurs de fenêtre.

II. Formes de base

Mon premier SVG, au clavier seulement… Pour cet exercice, on n'utilisera que Brackets ou tout autre éditeur de code.

Exercice 1 : dessiner le symbole « Peace & Love »

On utilisera les éléments *circle* et *line*
(Voir les attributs de ces deux éléments à https://www.w3.org/TR/SVG/shapes.html)

Le cercle devra être de rayon 100 et placé en (300,200). Un peu de trigonométrie sera utile pour placer convenablement les extrémités des 2 lignes obliques, sinon, en quelques itérations, ce sera trouvé…

Remarque : les formes de base en SVG sont circle, ellipse, rect, line, polyline et polygon.
Pour ces 2 derniers, un seul attribut est requis : points.
Exemple : <polygon points="0,0 40,20 20,20 20,40 30,50 40,60" /> Le tracé passe par (0,0) puis (40,20), etc. Contrairement à polyline, polygone rejoint le dernier point au premier.

III. Du style !

Pour enrichir une forme, on peut directement donner des valeurs aux attributs
<rect x="200" y="100" width="600" height="300"
propriété="valeur" propriété="valeur" propriété="valeur" />
Pour le cercle de l'exercice 1, ce pourrait écrire : stroke="black" fill="gold" par exemple.

Il y a trois possibilités, sans surprise, pour définir des styles :

- style « inline »
exemple pour le cercle : style="stroke:black; fill:gold;"
ou encore :
style="fill:#26CD22; stroke:#1F56D2; fill-opacity:0.7; stroke-opacity:0.5"
style s'ajoute aux autres attributs.
Le style inline est le plus simple si ce style n'est jamais réutilisé.

- style sheet interne

```
<style type="text/css">

    .style1 {stroke:black; fill:gold;}

</style>
```

(plus précisément, pour éviter des séquences d'échappement, on trouvera souvent une section d'échappement CDATA :

```
<defs>
  <style type="text/css">
        <![CDATA[
```

```
        .style1 {stroke:pink; fill:yellow;}

        ]]>
  </style>
</defs>
```
Ici, <style> a été inclue en outre dans une balise <defs>, sémantiquement plus propre (non obligatoire).

la référence au style style1 se fait par class="style1"

- feuille de style externe
référencée par
<?xml-stylesheet href="style.css" type="text/css"?>

Exercice 2 : styliser la figure de l'exercice 1 par un effet de dégradé linéaire vertical.
(Voir https://www.w3.org/TR/SVG/pservers.html)

En ce servant de cette référence et de l'exemple ci-dessous,

```
<linearGradient id="gradient1" x1="0%" y1="0%" x2="100%" y2="0%">
<stop offset="0%" style="stop-color:#26CD22;"/>
<stop offset="100%" style="stop-color:#1F56D2;"/>
</linearGradient>
```

définir les attributs de l'élément <linearGradient> (que l'on placera dans la balise <defs>), identifié par id=« gradient1 », et référencé ainsi :

```
<style type="text/css">

      .style1 {stroke:black; fill:url(#gradient1);}

   </style>

<circle
    class="style1"
…
```
de façon à obtenir l'effet suivant :

IV. Les chemins (paths)

Les tracés, ou chemins, ou *paths*, sont les éléments les plus importants car ils sont la façon de représenter n'importe quel tracé au delà des formes de base. Plus compliqués à manier « à la main », ils sont le résultat produit par tous les logiciels graphiques vectoriels (Illustrator, ou Inkscape par exemple dont le SVG est le format natif).
La balise invitant un tracé est <path>. L'attribut définissant le chemin est d.
Un premier « point courant » est défini par la commande M (move to). L permet de tracer une ligne depuis ce point courant jusqu'à un autre point, H et V définissent des horizontales et verticales, Q et T des courbes quadratiques de Bézier, C et S des courbes cubiques de Bézier et A des arcs d'ellipse. Un tracé se termine par Z.
Ainsi, typiquement : <path stroke="orangered" fill="none"
d="M574.2,373.7c-161-77.9-213-372.7,1.3-264.9C793.7,1,728.8,298.4,574.2,373.7z"/>

Dans cet exemple, certaines commandes sont en majuscules, d'autres en minuscules.
En majuscule : ce sont des coordonnées absolues. La même commande en minuscule considère des coordonnées relatives au point précédent. Cela prend moins de place et c'est plus efficace.
(il n'y a pas de différence entre Z et z…)

Exercice 3 :
1) exporter des chemins SVG à partir de Photoshop et d'Illustrator.

Dessiner une étoile à 5 branches.
Avec Illustrator, il faut enregistrer en SVG simplement. Avec Photoshop, l'exportation se fait en sélectionnant un calque donné.
Si un calque contient des pixels, l'exportation conduira à un fichier au poids similaire à un fichier PNG : le fichier SVG contient une balise <image> !
Pour exporter un fichier SVG avec des chemins, il faut créer des formes vectorielles sous Photoshop. Pour une étoile à 5 branches, on utilisera la forme « polygone » avec l'option « étoile ».

Dans tous les cas, observer avec un éditeur le code obtenu et alléger le code éventuellement (en particulier, supprimer l'attribut enable-background et constater qu'il est possible d'arrondir souvent les coordonnées des points à l'entier le plus proche sans effet visuel).

2) écrire à la main un path permettant de tracer un carré rempli, en utilisant des coordonnées absolues, puis en utilisant des coordonnées relatives.

V. Systèmes de coordonnées et transformations

L'espace appelé canevas qui contient le SVG est infini dans les 2 dimensions. Mais il est « rendered » sur l'écran dans une région de dimension finie appelée viewport, à imaginer comme une fenêtre qui permet de voir une partie ou tout du SVG.
Ce viewport peut être défini par les attributs width et height (attention, ne pas mettre ces attributs pour un SVG que l'on souhaite redimensionnable avec la fenêtre du navigateur !) :
exemple : <svg width="800" height="600">. Par défaut, les unités sont en px, mais peuvent être spécifiées en em, ex, px, pt, pc, cm, mm, in.
Le système de coordonnées a pour origine (0,0), le coin supérieur gauche. Mais le viewport peut avoir son système de coordonnées propres.
Le même exemple peut s'écrire <svg width="800" height="600" viewbox="0 0 800 600" > (dans l'ordre : <min-x> <min-y> <width> <height>).
Si on change les 2 dernières valeurs, par exemple :
<svg width="800" height="600" viewbox="0 0 400 300" >, le SVG est rogné à la région Lxh 400 300, et cette région va occuper tout le viewport de largeur x hauteur 800 600.
Si on change les valeurs min-x et min-y, par ex. pour <svg width="800" height="600" viewbox="100 100 400 300 »>, cela équivaut à une transformation de la région par transform="translate(-100 -100)"
Si des changements de rapport L/h sont effectués, il faut pouvoir contrôler les déformations possibles. C'est le rôle de l'attribut preserveAspectRatio. Son usage est parfaitement décrit dans cet article : https://sarasoueidan.com/blog/svg-coordinate-systems/ avec une démo interactive permettant de tester tous les cas possibles.
remarque : un SVG pouvant contenir d'autres SVG, les descendants sont repérés dans le système de coordonnées parent.
les transformations introduisent un nouveau système de coordonnées. Appliquées à un groupe d'éléments <g>, les transformations s'appliquent aux éléments du groupe.

SVG et HTML
Pour insérer un graphique SVG image.svg au sein d'une page Web, au moins 6 méthodes sont possibles :

```
<iframe src="image.svg">Your browser does not support iframes</iframe>
ou
<embed type="image/svg+xml" src="image.svg" />
ou
<img src="image.svg" />
ou comme image de fond CSS :
#myelement
{
        background-image: url(image.svg);
}
```

Les 2 premières méthodes sont à éviter, les 2 suivantes peuvent être utilisées pour insérer des SVG sans interaction.
Pour un contrôle avec CSS et scripts, la balise HTML5 Object est un meilleur choix :

```
<object type="image/svg+xml" data="image.svg">Your browser does not support SVG</object>
```

Enfin, il est tout à fait possible d'insérer directement le SVG dans le fichier HTML :

```
<svg>
..
</svg>
```

Exercice 4 :

Utiliser le graphique créé (la figure Peace & Love stylisée) dans une page web, au sein de la page HTML, et/ou au sein d'une balise object.

VI. Animations (animate et animateTransform)

Il est possible d'animer un élément SVG de différentes manières : par translation, par rotation, par changement d'échelle, ou par inclinaison (sur X ou sur Y).
Une animation doit être définie par un état initial, un état final, le temps de l'animation et sa répétition ou non. Prenons un exemple d'une animation (translation) appliquée à un rectangle :

```
<rect id="Rectangle1" fill="#3A8ACA"  x="0" y="0" width="100" height="200">
  <animateTransform attributeName="transform"
    type="translate"
    from="0 0"
    to="150 300"
    begin="0s"
    dur="2s"
    repeatCount="indefinite"
  />
</rect>
```

Attention, un seul animateTransform par élément. Pour appliquer plusieurs animations, on met les suivantes à la suite, dans un <g> englobant le tout. (on englobe à chaque fois l'élément précédent et son animation dans un groupe <g> et on ajoute l'animation suivante).

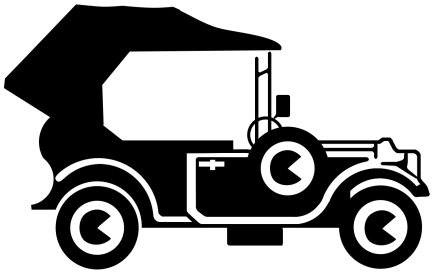
Des animations peuvent démarrer lorsque la précédente (ou une autre) se termine.
Exemple avec animate :

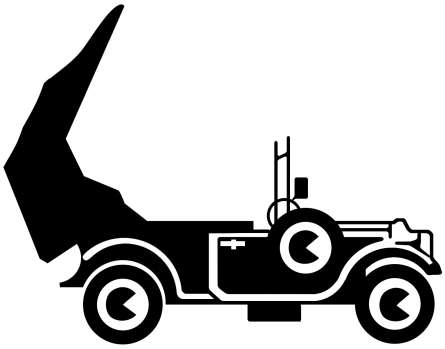
```
<circle cx="200" cy="150" r="10" stroke-width="1" stroke="aqua" stroke-opacity="1" fill="azure">
        <animate attributeName="r"  id="zoom"
        from="10" to="100"
        begin="0s" dur="5s" fill="freeze"/>
        <animate attributeName="stroke-width"
        from="1" to="20"
        begin="0s" dur="5s" fill="freeze"/>
        <animate attributeName="stroke-opacity"
        from="1" to="0"
        begin="zoom.end" dur="5s" fill="freeze"/>
</circle>
```

Remarques :
- il existe animateColor qui fonctionne comme animate mais le from et to concernent des couleurs ! (obsolète, puisque animate le permet)
- pour modifier un attribut sans transition, pendant une certaine durée, on préfèrera set à animate.

Exercice 5 :
Vos parents se souviendront de l'émission de l'ORTF « Histoires sans parole » qui diffusait le soir, des anciens films muets N&B. Le générique a marqué plusieurs générations :
https://www.youtube.com/watch?v=e20UedS7oIA
On se propose de créer une partie de l'animation : le tacot qui se désarticule en heurtant le rocher.
Il va donc falloir disposer d'un voiture en plusieurs morceaux :
- la capote (qui se relève, puis se referme)
- le châssis arrière (dont des parties non visibles lorsque la voiture n'est pas désarticulée)
- le châssis avant (qui se replie en arrière…)

En s'appuyant sur une copie d'écran du générique, reproduire à la plume une voiture dont les différents éléments seront regroupés en 3 groupes.
Chaque groupe <g> peut contenir plusieurs chemins <path> : il sera utile de réaliser des éléments séparés (la roue sera utilisée 3 fois) et il faudra des chemins pour des parties blanches (intérieur des roues, de différentes parties du châssis, qui ne peuvent être transparentes).





Décrire les différentes séquences de l'animation puis écrire le SVG correspondant.

VII. Animations (animateMotion)

animateMotion est une autre balise d'animation de SVG. Elle permet de faire défiler un object le long d'un chemin.
On place cette balise comme enfant direct de l'objet qu'on veut déplacer. Ensuite, on utilise, en plus des attributs habituels, soit l'attribut path qui contient les coordonnées du chemin, soit (le plus souvent) la balise mpath, enfant direct de animateMotion, dont l'attribut xlink:href appelle un path par son id.
Explication par un exemple :

```
<circle cx="20" cy="10" r="10">

<animateMotion begin="10s" dur="10s" repeatDur="indefinite">
        <mpath xlink:href="#chemin"/>
</animateMotion>
</circle>
```

en supposant que le chemin ait été défini auparavant par un path :

```
<path id="chemin" d="M  ….. />
```

Avec l'attribut rotate="auto", l'objet animé est automatiquement dans l'axe de la tangente au déplacement.

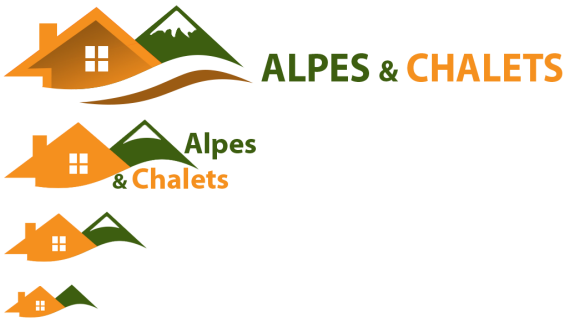
Exercice 6 :

- Utiliser les exercices précédents pour faire déplacer un cercle autour d'un carré.
- Réutiliser la voiture créée précédemment (en changer l'échelle !) et la déplacer le long d'un chemin.

- Compléter l'animation « Histoires sans parole » pour déplacer la voiture jusqu'au rocher où elle se disloque.

VIII. Responsive Web Design et SVG

Nous allons utiliser les capacités du SVG et de ses styles pour créer un logo responsive : un logo sera affiché sur une page web différemment en fonction des caractéristiques de l'écran.
Exemple :



Première idée, on utilise classiquement la méthode des sprites.
HTML :
```
<div style="background-color:#fff" class='logo'></div>
```
CSS :
```
.logo {
  width: 600px;
  height: 300px;
  background: url(images/logo.svg);
  background-position: center top;
}

@media only screen and (min-width: 15em) {
.logo  {background-position: left -900px;}
}
@media only screen and (min-width: 28em) {
.logo  {background-position: left -600px;}
}
@media only screen and (min-width: 45em){
.logo {background-position: left -300px;}
}
@media only screen and (min-width: 60em){
.logo {background-position: left 0px;}
}
```
SVG :
les différentes versions du logo sont espacées verticalement de la valeur adéquate.

Méthode à oublier.

On préférera une feuille de style et la propriété display (ou opacity) Exemple :

```
  <style type="text/css"><![CDATA[
@media only screen and (min-width: 30em) {
   #small {
       display:none }
 }
@media only screen and (min-width: 60em) {
   #large {
       display:none  }
}
@media only screen and (max-width: 30em) {
   #large {
       display:none  }
}
   ]]></style>
```

Exercice 7 :

Réaliser un logo responsive avec au moins 4 variations. Idéalement (c'est tout l'intérêt), réaliser un logo dont les versions simplifiées suppriment des éléments de la version plus complète et les stylisent différemment.
Un exemple très repris sur le Web :

D’autres exemples (travaux d’étudiants) sont fournis sur Moodle.

Annexe : le XML (Source Wikipedia)

L'Extensible Markup Language est un métalangage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG… Elle est reconnaissable par son usage des chevrons (<, >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche…) entre systèmes d'informations hétérogènes (interopérabilité).

Ce qui est spécifique à XML, c'est le choix des chevrons pour identifier les balises, et l'obligation de les fermer. Les mots clés ne sont pas définis par la norme XML, mais par le vocabulaire choisi.
XML se présente en général comme une chaîne de caractères, séparant deux niveaux : du texte à destination des humains, et des balises à destination des machines. La structuration de ce texte par les balises produit un modèle informatique.

Une balise est un nom commode pour désigner les constructions entre deux chevrons (<, >) dans un fichier XML. On distinguera les balises ouvrantes <élément attribut="valeur">, les balises fermantes </élément> (sans attributs et commençant par une barre oblique) et les balises vides <élément attribut="valeur"/> (avec attributs possibles et terminant par une barre oblique). Il ne faut pas confondre les balises avec les éléments. Ces notations permettent de délimiter des éléments (ainsi que leurs attributs), mais les balises ne sont pas des nœuds dans le modèle abstrait du document.

Un attribut est un nom et une valeur. Un nom d'attribut a les mêmes contraintes et possibilités de qualification qu'un nom d'élément. La valeur est un texte sans élément (ni autres nœuds). Un attribut est toujours porté par un élément (balise ouvrante). La valeur peut être vide <element attribut=""/>, mais pas nulle <element attribut>. Un attribut est unique. La répétition d'un attribut de même nom sur le même élément provoquera une erreur de l'interpréteur XML. L'ordre des attributs n'est pas significatif. <element attribut1="valeur1" attribut2="valeur2"/> et <element attribut2="valeur2" attribut1="valeur1"/> sont équivalents pour un interpréteur XML.

En XML, les commentaires XML sont délimités par <!-- et -->. Le contenu d'un commentaire ne sera pas interprété.
<!—C’est du commentaire -->.
La chaîne de caractères -- ne peut apparaître dans le contenu d'un commentaire (un interpréteur XML considère que ce signal annonce la fin d'un commentaire).

Une instruction de traitement XML est une balise particulière qui s'ouvre par un chevron ouvrant et un point d'interrogation, et qui se ferme avec un point d'interrogation et un chevron fermant <?clé valeur?>. Les lettres accolées au chevron ouvrant forment la clé jusqu'au premier espace. La suite peut contenir des espaces, de la ponctuation et forme la valeur. Cette valeur n'a pas de syntaxe imposée par XML, ce sont les applications qui les interprètent pour leur besoin.

En XML, le prologue XML est constitué de la déclaration XML <?xml version="1.0" encoding="UTF-8"?>, et de la déclaration de type de document (DOCTYPE). La déclaration XML est obligatoire à partir de la version 1.1. La déclaration DOCTYPE avait une grande importance en SGML. Elle attache le document traité par un interpréteur à son schéma (DTD, Document Type Definition, « Définition de type de document ») afin de le valider et d'interpréter certains raccourcis (les entités). Désormais, il existe plusieurs langages de validation, et parfois plusieurs manières de les attacher. La déclaration DOCTYPE n'a plus la même importance.

Encodage
Par défaut, le texte est traité comme de l'Unicode (UTF-8). XML permet de spécifier d'autres encodages dans le prologue pour des raisons historiques.

Espaces
En XML, espaces et sauts de lignes sont équivalents, autrement dit, un document peut perdre son indentation en restant identique pour les traitements, sauf instructions particulières (exemple : bloc préformaté avec l'attribut @xml:space="preserve").

Malformations
Comme tout langage informatique, XML est fondé sur des caractères qui ont un sens particulier pour la machine. La force de XML est d'avoir réduit le nombre de ces caractères au minimum, afin que le texte prime sur le métatexte (priorité des données sur les instructions). Dès qu'un analyseur XML (ex : navigateur Web) rencontre un chevron ouvrant (<), les caractères qui suivent sont interprétés comme une balise. Ceci pose évidemment des problèmes dans une inéquation mathématique if (a < 10) print("Unité"). La balise < 10)… n'ayant pas un nom d'élément correct (présence d'une espace, commence par un chiffre). C'est un problème pour tous les langages informatiques, qui introduisent alors des caractères d'échappement.
Il faut au moins un caractère pour signaler que les caractères qui suivent doivent être interprétés autrement. En XML, c'est le rôle de l'esperluette (&) qui introduit des entités. Une entité est un nom XML encadré d'une esperluette et d'un point virgule.
Dans un texte, le signe inférieur doit être remplacé par < (lesser than). Du coup, l'esperluette acquiert elle aussi un statut particulier, et doit aussi être échappée. Les esperluettes d'un texte doivent être remplacées par l'entité & (ampersand, « esperluette » en français). Mentionnons ici 3 autres entités nécessaires à XML, > (greater than, « supérieur » en français) pour >, " (quote, « guillemet » en français) pour ", ' (apostrophe) pour '.
Sections d'échappement
Une section d'échappement XML <![CDATA[…]]> permet de contenir n'importe quel texte, avec tous les caractères spéciaux XML <>&, sans qu'il soit nécessaire d'échapper ces caractères avec des entités. Elles sont souvent utilisées dans des documents qui contiennent des syntaxes informatiques avec de nombreux chevrons et esperluettes (ex : JavaScript). Ces sections d'échappement permettent de garder le texte original, en gardant conforme le flux XML.

Utilisations et langages dérivés
- langage de balisage ; (ex XHTML)
- format de données ; (Ex RDF, Dublin Core)
- langage de description de format de document ; (Ex DTD)
- langage de représentation ; (Ex MathML, Office Open document, SVG, SMIL)

- langage de programmation ; (Ex Mozilla, Flex)
- protocole de communication. (Ex SOAP)

Langages associés

Les langages associés à XML sont des syntaxes qui ne sont pas en XML mais très attachées à XML. CSS illustrera bien la notion. Il peut être contenu dans un attribut (@xhtml:style), dans un élément (<xhtml:style>), ou relié à un document XML par une instruction de traitement (<?xml-stylesheet href="common.css" type="text/css"?>). XPath fournit un autre exemple de spécification entièrement destinée à XML, mais qui est justement sans éléments ou attributs, afin d'être associé à un langage XML (XSLT).