Université Claude Bernard Lyon 1

e-Miage C216

REGROUPEMENT 2
17 OCTOBRE 2017

LIONEL MÉDINI

Rappel : objectifs pédagogiques du module

- Conception de pages web
  - Structuration
  - Mise en forme
- Scripting côté serveur
  - Génération dynamique de pages
- Couplage serveur web / base de données
- Prise en compte du protocole HTTP
- Scripting côté client

Rappel : planning des 3 devoirs

- Devoir 1
  - Conception de pages web statiques
  - Scripting côté serveur
- Devoir 2
  - Couplage serveur web / BD
  - Découverte du scripting côté client
- Devoir 3
  - Utilisation approfondie du protocole HTTP
  - Mécanismes de requête asynchrone

Plan : syntaxe et programmation XML

- Découverte de XML
  - Présentation et syntaxe
  - Notion d'espaces de noms XML
- Programmation XML
  - Le langage de feuilles de style XSL
    - Xpath
    - XSLT
  - L'API DOM (Document Object Model)
- Utilisation dynamique côté client (AJAX)
  - Utilisation du DOM en JavaScript
  - Mécanismes de requêtes asynchrones

(X)HTML et XML

- XML : principes de base
  - DTD de SGML (beaucoup plus concis)
  - Restrictions de syntaxe et non de contenu
  - Au même titre que SGML, c'est un méta-langage de description des données
    - ⇒ Ça ne sert à rien
    - ⇒ Ça permet de définir des « applications » pour faire ce qu'on veut avec
      - Visualiser des pages web
      - Décrire des images
      - Échanger des données techniques…
  - V 1.0 : fév. 1998 -> 04 fév. 2004 (3ème édition)
  - V 1.1 : 04 février 2004

(X)HTML et XML

- XML : composants
  - XML (syntaxe) : documents « bien formés »
  - DTD/schémas XML : documents « valides »
  - Processeur (parser) XML : analyse et traitement
  - DOM : modèle arborescent des données d'un document pour un langage de programmation
  - SAX : accès « simple » aux données (programmation événementielle)
  - Espaces de noms (« namespaces ») : interopérabilité des applications
  - XBase, XPointer, XLink : mécanismes de liens
  - XSL : mécanismes de transformation

HTML, XML et XHTML

- Syntaxe XML : un document XML bien formé est composé
  - D'un prologue, contenant :
    - Éventuellement une déclaration XML

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone="yes"?>
```

    - Éventuellement une déclaration de type de document

```
<!DOCTYPE Nom_de_l_élément_racine Type_de_source
  emplacement1 emplacement2 [sous-ensemble interne de
  DTD]>
```

  - D'un élément « racine », composé :
    - Soit d'une balise ouvrante, éventuellement d'un contenu et d'une balise fermante

```
<NomElement attribut1="valeur1" attribut2="valeur2"...>
  contenu
</NomElement>
```

HTML, XML et XHTML

- Syntaxe XML : un document XML bien formé est composé
  - D'un prologue, contenant :
    - Éventuellement une déclaration XML

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone="yes"?>
```

    - Éventuellement une déclaration de type de document

```
<!DOCTYPE Nom_de_l_élément_racine Type_de_source
  emplacement1 emplacement2 [sous-ensemble interne de
  DTD]>
```

  - D'un élément « racine », composé :
    - Soit d'une balise d'élément vide

```
<Nom_element_vide att1="val1" att2="val2"... />
```

HTML, XML et XHTML

- Syntaxe XML : remarques
  - Structure
    - Un contenu est composé de texte et / ou d'autres éléments appelés fils de l'élément courant
    - Le texte est appelé « données caractères analysables » ou PCDATA (pour Parsed Character DATA).
    - Le contenu autorisé pour le PCDATA dépend du type d'encodage choisi
    - Le fait d'avoir un unique élément racine, des éléments fils, eux-mêmes décomposables, etc. définit une **structure arborescente**
    - Pas de chevauchement de balises entre un élément père et un élément fils
  - Caractères spéciaux
    - Les caractères "<" (inférieur) et "&" (esperluète) sont interdits dans les contenus. On aura recours aux entités "`<`" et "`&`".
    - L'usage de ">" (supérieur) ou des guillemets simples ou doubles peut également être perturbant. Dans ce cas, on a recours à "`>`", "`&apos;`" et "`&quot;`".
    - Si l'on veut vraiment utiliser les caractères "<" ou "&", il est possible de définir une balise sous forme de zone de caractères non analysés (CDATA), sous la forme :

```
<![CDATA [ texte comprenant des caractères interdits ]]>
```

URI, URL et URN

- URI : Uniform Resource Identifier
  - Objectif : identifier de façon unique une ressource sur le web
  - 2 possibilités :
    - Dire où elle se trouve
      - Donner son URL (Uniform Resource Locator)

Format	protocole ":" chemin "/" nom de fichier "/" requête
Exemple	http://www.monsite.fr/monAppli/index.php?param1=toto
Enregistrement	Auprès de l'entité DNS concernée (délégation ICANN)
Remarque	Permet d'accéder réellement à la ressource

    - Dire comment elle s'appelle
      - Donner son URN (Uniform Resource Name)

URI, URL et URN

- URI : Uniform Resource Identifier
  - Objectif : identifier de façon unique une ressource sur le web
  - 2 possibilités :
    - Dire où elle se trouve
      - Donner son URL (Uniform Resource Locator)
    - Dire comment elle s'appelle
      - Donner son URN (Uniform Resource Name)

Format	"URN:" NID (namespace identifier) ":" NSS (namespace specific string)
Exemple	URN:ISBN:0-395-36341-1
Enregistrement	NID auprès de l'IANA (Internet Assigned Numbers Authority)
Remarque	Choix plus « libre », et correspondant mieux à la définition d'un espace de noms

URI, URL et URN

- URI : Uniform Resource Identifier
  - Objectif : identifier de façon unique une ressource sur le web
  - 2 possibilités :
    - Dire où elle se trouve
    - Dire comment elle s'appelle

  - ➔ Syntaxe générique : « scheme » ":" autorité ":" chemin ":" requête ":" fragment
  - ➔ Un URI est uniquement un identificateur, qui n'a pas de sens en soi
  - ➔ Il ne signifie rien pour le processeur XML, qui le transmet tel quel à l'application

  - Remarque : d'un point de vue pratique, les URL sont plus sures afin d'éviter les conflits entre les espaces de noms

Espaces de noms XML

- Position du problème
  - Liberté de choix des noms de balises et des attributs XML
  - ⇒ Conflits et polysémie entre ces noms/attributs
  - Besoin d'associer plusieurs applications dans un même document
  - ⇒ « Préfixage » des noms de balises par l'URI de l'application concernée

Espaces de noms XML

- Noms qualifiés (qualified names)
  - Noms de balises appartenant à des espaces de noms
  - Syntaxe : PrefixeDEspaceDeNoms:PartieLocale
  - Exemple : `<xsl:stylesheet>`
  - Le préfixe fait référence à un URI
  - Les noms d'attributs peuvent également être préfixés
- Association d'un préfixe à un URI
  - Attribut xmlns
  - Exemple : `<xhtml:html xmlns:xhtml="http://www.w3.org/1999/xhtml">`
- Remarques
  - Portée : l'élément porteur de l'attribut `xmlns`
  - Bien entendu, un document XML peut contenir des éléments se référant à plusieurs espaces de noms
  - Le préfixe en lui-même n'a aucune signification
  - En interne, le parser passe à l'application des « noms pleinement qualifiés », où le préfixe est remplacé par la valeur de l'URI

Espaces de noms XML

- Espace de noms par défaut
  - Pas de préfixe d'espace de noms
  - Exemple : `<html xmlns="http://www.w3.org/1999/xhtml">`
- Annulation d'espaces de noms
  - Valeur de l'attribut xmlns vide : `xmlns=""`
- Exemple

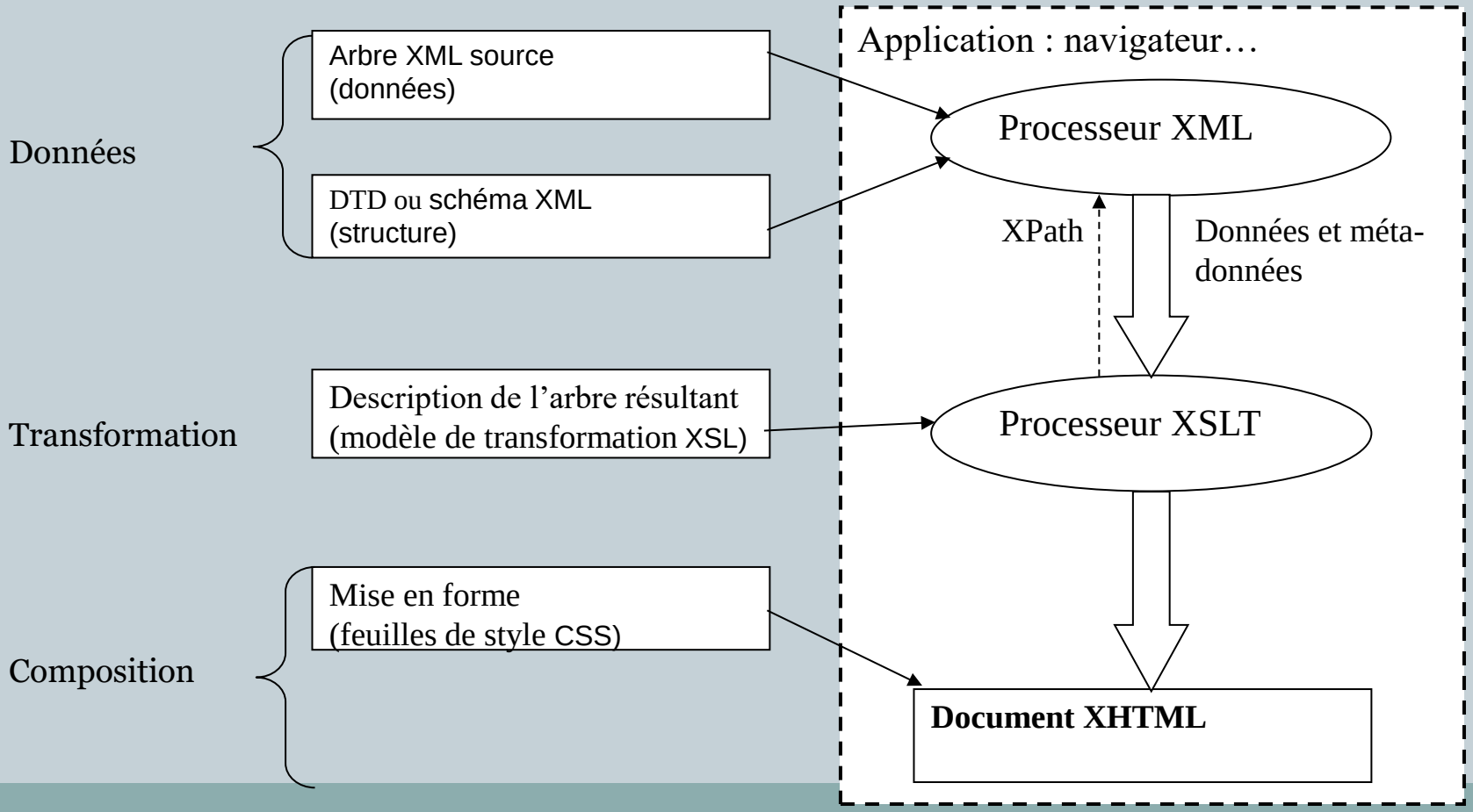
```
<?xml version="1.0"?>
<CV xmlns="http://www.univ-
lyon1.fr/etds/CV/english"
    xmlns:xhtml="http://www.w3.org
/1999/xhtml">
  <person>
    <civil_status>
      <title>Mr.</title>
    </civil_status>
    ...
  </person>
  <xhtml:html>
    <xhtml:head>
      <xhtml:title>CV of a
student</xhtml:title>
    </xhtml:head>
    <xhtml:body>
    ...
    </xhtml:body>
  </xhtml:html>
</CV>
```

Transformation d'arbres XML : XSL

- Caractéristiques de XSL
  - Officiellement : XML Stylesheet Language
  - En pratique, ça ne sert à rien d'appliquer des éléments de style à un document XML
  - Mais XSL fournit un mécanisme très puissant pour transformer un arbre XML
    - En un autre arbre XML (échange de données)
    - En un arbre XHTML (visualisation des données XML)
    - En un texte simple (fichier non destiné à une application utilisant un parser XML)
    - En un document papier formatté (XSL-FO)

Transformation d'arbres XML : XSL

- Utilisation la plus courante de XSL



Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Permet de pointer vers les données de l'arbre XML
      - pour le parcours de documents XML
      - pour le test de valeurs associées aux contenus ou aux attributs d'éléments
    - Ne respecte pas la syntaxe XML
      - pour ne pas « perturber » l'analyse des feuilles de style XSLT par le parser XML
    - 2 versions
      - Xpath 1.0 : principes de base
      - Xpath 2.0 : plus proche de Xquery
      - Dernière recommandation : http://www.w3.org/TR/xpath20/

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Notion de nœud
      - Tout type de données (élément, attribut, PI, fragment)
      - Racine du document : '/'
      - Les éléments sont identifiés par leurs noms
      - Les attributs sont identifiés par '@' suivi du nom de l'attribut
    - Notion de chemin de localisation
      - Absolu : à partir de la racine de l'arbre XPath
      - Relatif : à partir du nœud contextuel
      - Récursif : à partir du nœud contextuel, mais seulement vers le bas

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Axes de navigation
      - Déplacements dans l'arbre XPath
      - Syntaxe : Nom_D_Axe::Nom_De_Noeud
      - Recommandation : http://www.w3.org/TR/xpath20/#axes

Nom d'axe	Description	Exemple d'utilisation/ syntaxe abrégée
self	Nœud contextuel	self::node() ou ./node() ou .
child	Enfants du nœud contextuel	child::Etat_civil ou Etat_civil (défaut)
descendant	Tout enfant, petit enfant etc. du nœud contextuel	descendant::Etat_civil
descendant-or-self	Comme descendant + le nœud contextuel lui-même	descendant-or-self:: Etat_civil ou .//Etat_civil
parent	Parent du nœud contextuel	parent::Prenom ou ../Prenom
ancestor	Tout parent, grand parent etc. du nœud contextuel	ancestor::Prenom
ancestor-or-self	Comme parent + le nœud contextuel lui-même	ancestor-or-self::Prenom
following-sibling	Tous les frères suivants du nœud contextuel (vide si le nœud est un attribut)	following-sibling::Nom
preceding-sibling	Tous les frères précédents du nœud contextuel (vide si le nœud est un attribut)	preceding-sibling::Prenom
following	following–sibling + descendants de tous les nœuds frères suivants	following::Nom
preceding	preceding–sibling + descendants de tous les nœuds frères précédents	preceding::Prenom
attribute	Attributs du nœud contextuel	attribute::id ou./@id
namespace	Tous les nœuds appartenant au même espace de noms que le nœud indiqué	namespace::xhtml:div

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Opérateurs et fonctions
      - Expression de caractéristiques de sélection complexes
      - Communs avec XQuery
      - Recommandation à part entière : http://www.w3.org/TR/xquery-operators/

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Opérateurs et fonctions
      - Accesseurs
        - Pour récupérer un élément d'un nœud
        - Exemples : node-name(), string(), base-uri()
      - Génération d'erreurs
        - error()
      - Génération de traces
        - trace()
      - Constructeurs
        - Pour les types de données XML spécifiques
        - Exemple : MonType()
      - Casting entre types de données

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Opérateurs et fonctions
      - Traitements spécifiques à certains types de données
        - numériques
        - chaînes
        - URI
        - données temporelles (heures, date et durée)
        - noms qualifiés
        - notations
        - données binaires (base64Bynary, hexBinary)
        - nœuds
        - séquences (de nœuds)
        - contexte

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Structure d'une requête dans un document XML
      - Test
        - Définit le type de nœud à sélectionner
        - Par son nom (élément / attribut…)
        - Par une fonction Xpath
        - Exemples
          - /CV/Personne/@nom
          - .//div
          - ancestor-or-self::Personne
          - /descendant::*

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XPath
    - Structure d'une requête dans un document XML
      - Prédicat (facultatif)
        - Permet de sélectionner des nœuds par leurs contenus
        - Définition des caractéristiques recherchées
          - Entre crochets " [" … "] "
          - Opérateurs de comparaison ou fonctions
        - Exemple
          - //div[@id="toto"]
          - /descendant-or-self::div[attribute(id) = "toto"]
          - //MonElement[position() mod 2 = 0 or value() = "toto"]

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : principes de base
    - Description de l'arbre résultant (programmation déclarative)
    - Application XML définissant des « balises de transformation »
      - ⇒ Référence à un espace de noms spécifique « `xsl:` »
    - Balises spécifiques interprétées par un processeur XSLT
    - Structuration par modèles (« templates ») de contenus
      - Définissant le traitement à appliquer à un élément repéré par une expression XPath
      - Imbriqués grâce à des balises d'application de templates

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : structure
    - Élément racine :

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

    - Éléments de premier niveau (cardinalité=0 ou 1)
      - `<xsl:output>` : définit le type d'arbre de sortie
        - Attribut `method` : 3 valeurs possibles (`text`, `html`, `xml`)
        - Autres attributs : `version`, `encoding`, `standalone`, `indent`…
      - `<xsl:include>` et `<xsl:import>` : permettent d'inclure d'autres feuilles de style
        - Attribut `href` : URI de la ressource à inclure
        - Différence entre les deux : règles de priorités

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : syntaxe
    - Éléments de premier niveau (cardinalité=0 ou 1)
      - `<xsl:strip-space>` et `<xsl:preserve-space>` : gestion des espaces dans l'arbre résultant (resp. suppression et conservation)
        - Attribut `elements` : noms des éléments concernés séparés par des espaces
      - `<xsl:template>` : modèle racine de l'arbre de sortie
        - Attribut `match` : désigne le nœud XPath concerné par le modèle (au premier niveau, toujours "/")
        - Contient la racine de la déclaration de l'arbre de sortie
      - Autres éléments (key, variable, attribute-set, param) : voir la recommandation

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les templates
    - Définition
      - Modèles simples : `<xsl:template match="noeud_XPath">`
        - L'expression XPath qui définit le nœud peut inclure un filtre
        - Ce nœud devient le nœud contextuel dans le template
      - Modèles nommés : `<xsl:template name="nom_XML">`
    - Appel
      - Modèles simples :

        ```
        <xsl:apply-templates select="expr_XPath" />
        ```

        - L'expression XPath est un chemin de localisation qui désigne le nœud
      - Modèles nommés :

        ```
        <xsl:call-template name="nom_XML" />
        ```

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Génération de contenus XML
      - `<xsl:element name="p" namespace="xhtml">Contenu de l'élément (ici: un paragraphe XHTML)</xsl:element>`

        Remarque : `<xsl:element>` n'est nécessaire que lorsque le nom de l'élément à générer doit être calculé
      - `<xsl:attribute name="href" namespace="xhtml">Contenu de l'attribut (ici : référence XHTML)</xsl:attribute>`

        Remarque : `<xsl:attribute>` se place dans l'élément auquel il se rapporte
      - `<xsl:text>Contenu textuel quelconque.</xsl:text>`

        Remarque : `<xsl:text>` ne sert qu'au formatage du texte (gestion des espaces…)
      - Tout autre élément XML bien formé est accepté

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Traitement de contenus de l'arbre XML source
      - `<xsl:value-of select="expr_XPath" />`
        - Permet d'obtenir la valeur d'un nœud (élément ou attribut)
        - L'expression XPath est un chemin de localisation
        - Elle désigne un nœud à partir du nœud contextuel
      - `<xsl:copy-of select="expr_XPath" />`
        - Permet de recopier dans l'arbre destination toute une partie de l'arbre source
        - L'expression XPath fonctionne comme précédemment
      - `<xsl:copy />`
        - Permet de copier uniquement un élément sans ses sous-éléments

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Structures de contrôle
      - `<xsl:if test="expr_XPath">Contenu conditionnel</xsl:if>`
        - Le contenu conditionnel peut être composé d'autres éléments (`<xsl:value-of select="expr_XPath" />`)
      - `<xsl:for-each select="expr_XPath">Contenu répété</xsl:for-each>`
        - Cet élément est redondant avec `<xsl:apply-templates />` mais rend la feuille de style moins lisible

Transformation d'arbres XML : XSL

- Les deux composants de XSL
  - XSLT : les éléments
    - Structures de contrôle
      - `<xsl:choose>`

```
<xsl:choose>
        <xsl:when test="expr_XPath1">
                Contenu conditionnel 1
        </xsl:when>
        <xsl:when test="expr_XPath2">
                Contenu conditionnel 2
        </xsl:when>
        ...
        <xsl:otherwise>
                Contenu conditionnel n
        </xsl:otherwise>
</xsl:choose>
```

Transformation d'arbres XML : XSL

- Exemple
  - Date de 2002
  - À ouvrir uniquement avec IE ou une application *ad hoc*
  - http://www710.univ-lyon1.fr/~lmedini/LIA/LSI/Exam/Septembre/

Transformation d'arbres XML : XSL

- Quelques exemples d'outils
  - Dans un navigateur
    - IE : processeur XSLT incorporé à MSXML (contrôle ActiveX)
    - Gecko, WebKit : objet window.XSLTProcessor
  - En JavaScript :
    Google AJAXSLT
  - Dans HTML-Kit (éditeur HTML WYSIWYG) :
    plugin ErgXSLT
  - En Java, intégré à JAXP 2 (J2EE puis JEE5) :
    TrAX Transformation API for XML (javax.xml.transform, xsltc)
  - En C / en PHP :
    libxslt (PHP 5)

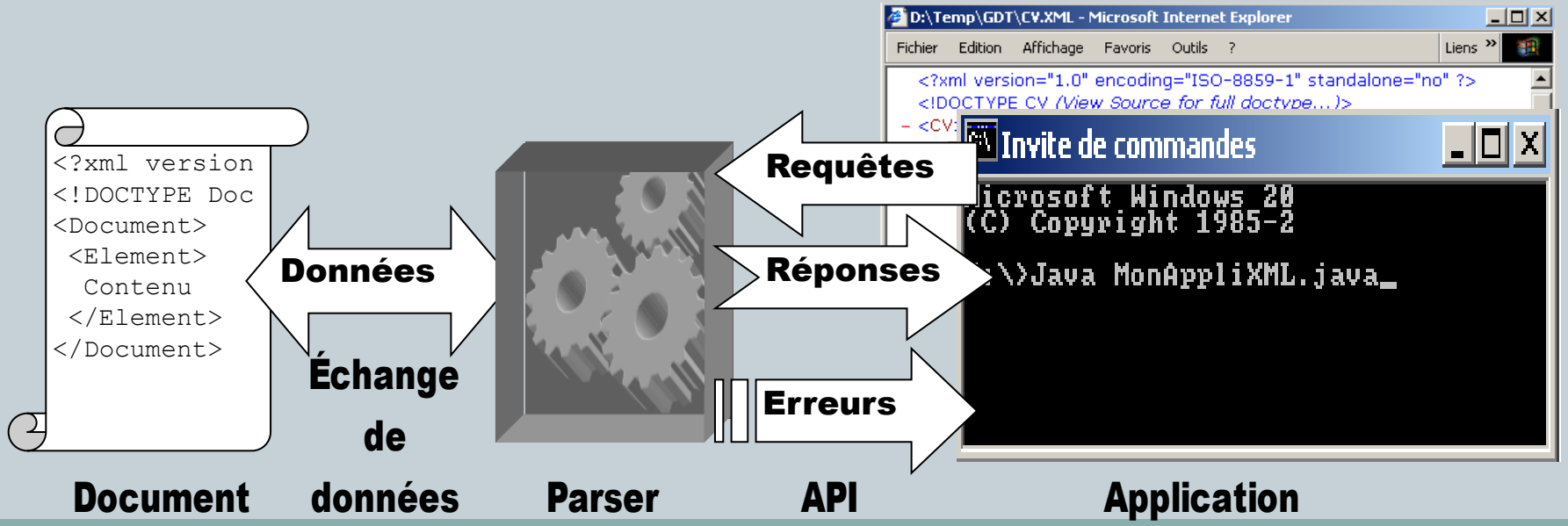
Outils de programmation avec XML

- Définitions
  - Qu'est-ce qu'un parser ?
    - « Un module logiciel [...] utilisé pour lire les documents XML et pour accéder à leur contenu et à leur structure. »
  - Qu'est-ce qu'une application ?
    - « On suppose qu'un processeur XML effectue son travail pour le compte d'un autre module, appelé l'application. »

http://babel.alis.com/web_ml/xml/REC-xml.fr.html#dt-xml-proc

Outils de programmation avec XML

- Communications entre parsers et applications
  - Rappel : Application Programming Interface
    - Outils
    - Protocole de communication
  - Schéma des échanges de données



L'API DOM (Document Object Model)

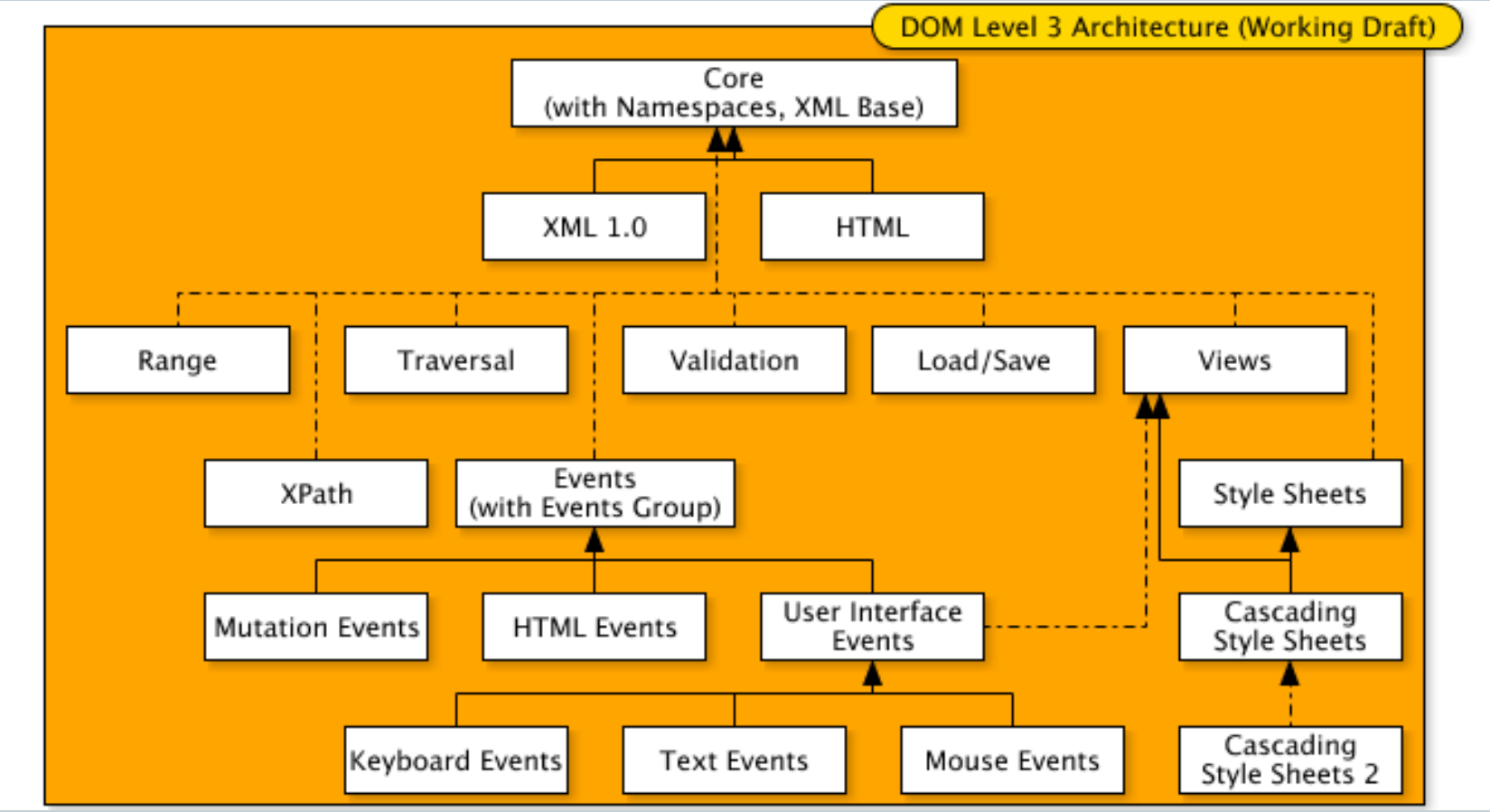
- Généralités
  - Modèle objet de document
  - Motivations
    - Rendre les applications W3 dynamiques
    - Accéder aux documents HTML et XML depuis un langage de programmation
  - Utilisations courantes
    - Intégré aux navigateurs
    - Utilisé en programmation comme API XML
  - Origine : DOM working group (W3C)
    - Début : 1997 ; fin : …
    - But : standardiser les tentatives existantes

L'API DOM (Document Object Model)

- Principes fondamentaux
  - Représentation arborescente d'un document
    - Tout le document est chargé en mémoire
    - Navigation dans la structure arborescente
    - Représentation des nœuds par des *interfaces*
      - Propriétés
      - Méthodes
  - Recommandations sous forme de niveaux
    - Niveau 0 : avant…
    - Niveau 1 : octobre 1998
    - Niveau 2 : depuis novembre 2000
    - Niveau 3 : depuis janvier 2004

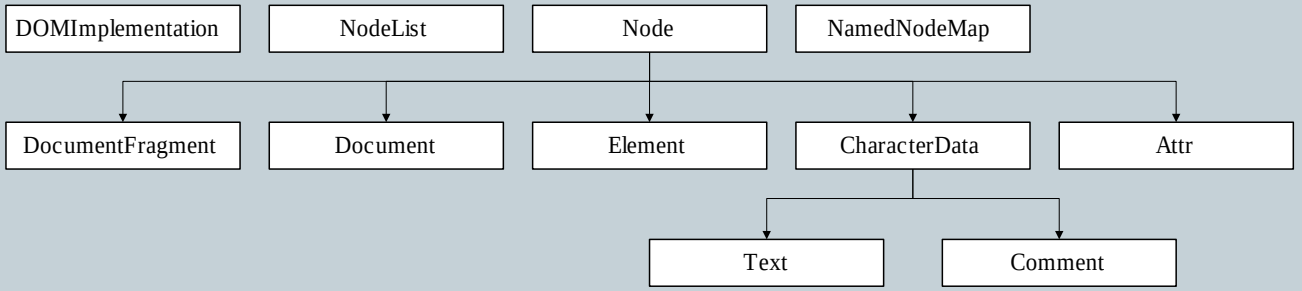
L'API DOM (Document Object Model)

- Fonctionnalités

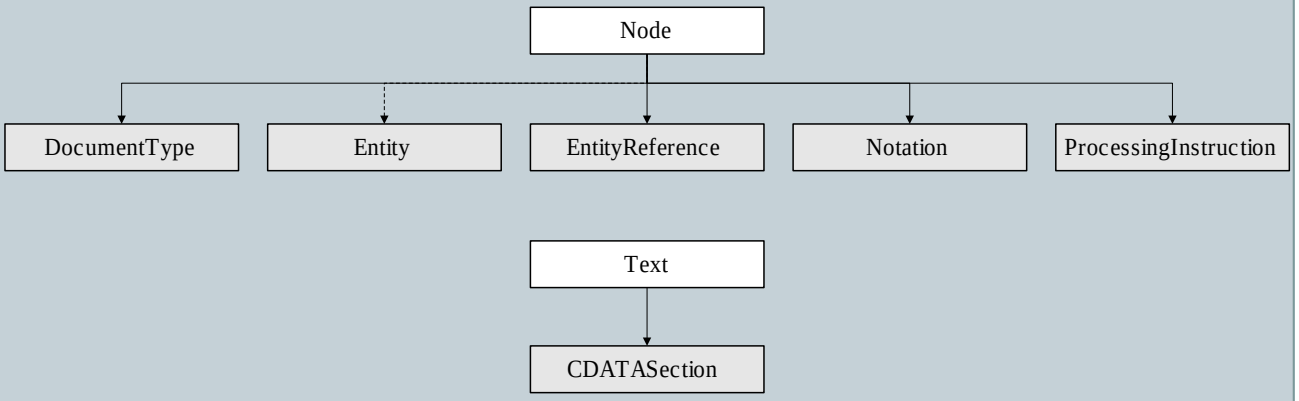


L'API DOM (Document Object Model)

- Modules
  - DOM Core :



  - DOM XML :



L'API DOM (Document Object Model)

- Interfaces DOM (Core et XML) les plus utilisées
  - Node : tout type de nœud de l'arbre DOM
    - Constantes
      Tous les types de nœuds définis (exemple : node.ELEMENT_NODE)
    - Propriétés
      nodeName, nodeType, NodeValue, ChildNodes, textContent
    - Méthodes
      insertBefore(), replaceChild(), removeChild(), appendChild(), cloneNode()
  - NodeList : comme son nom l'indique…
    - Propriétés : length, item(i)

L'API DOM (Document Object Model)

- Interfaces DOM (Core et XML) les plus utilisées
  - Document : le nœud racine de l'arbre DOM
    - Dérive de Node
    - Propriétés
      doctype, documentElement, encoding
    - Méthodes
      createElement(name), createTextNode(), createAttribute(name), getElementById(id), getElementsByTagName(name)
  - DocumentFragment : partie d'un document ; *cf.* Node

L'API DOM (Document Object Model)

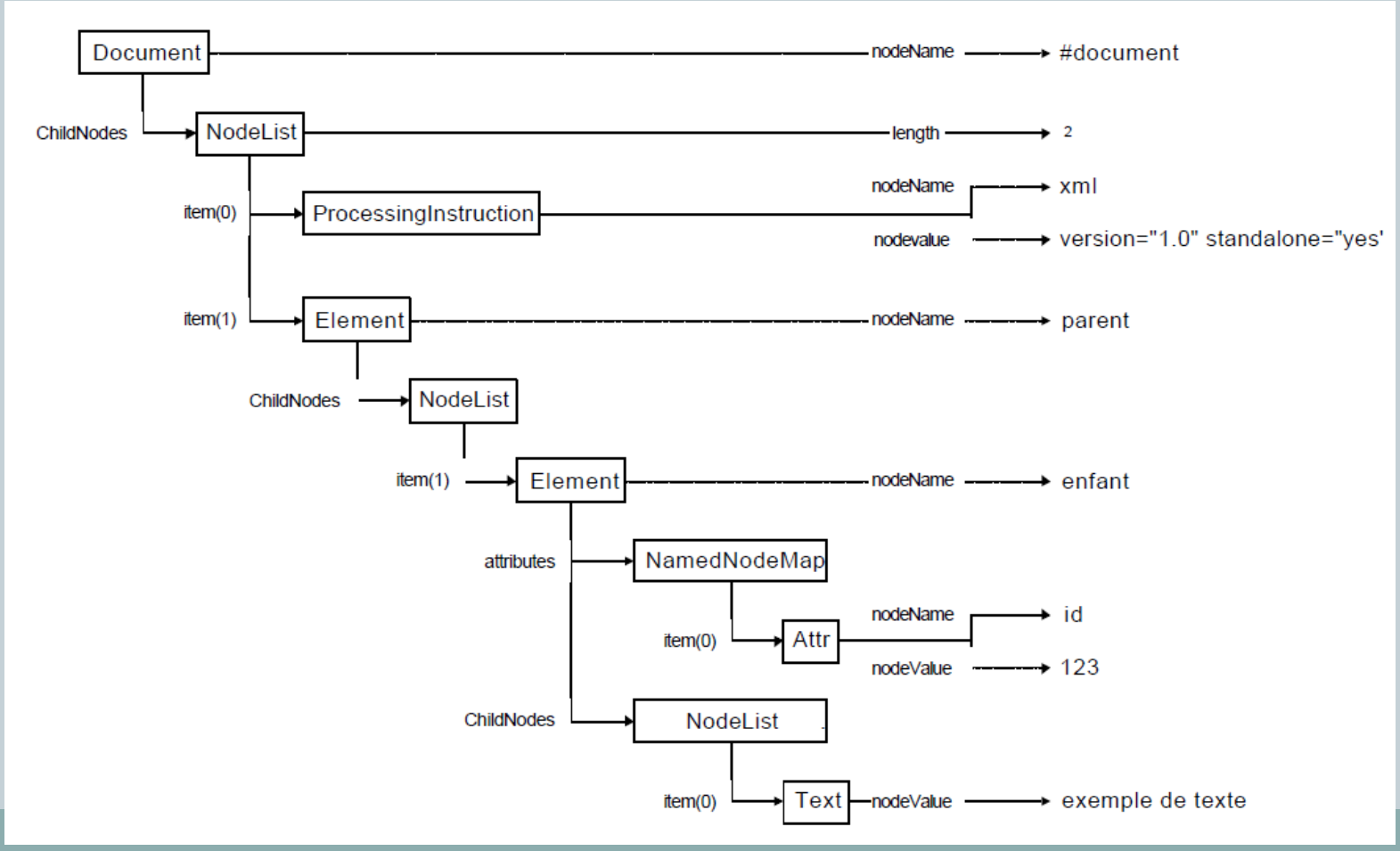
- Interfaces DOM (Core et XML) les plus utilisées
  - Element : un élément, au sens HTML ou XML
    - Propriété : tagName
    - Méthodes
      getAttribute(name), setAttribute(name, value), hasAttribute(name), getAttributeNode(name), setAttributeNode(node), removeAttribute(name), removeAttributeNode(node),
  - Attr : un attribut…
    - Propriétés : name, value, ownerElement

L'API DOM (Document Object Model)

- Interfaces DOM (Core et XML) les plus utilisées
  - Text : nœud textuel (sous nœud d'un élément)
    - Propriétés
      data, length (héritées de CharacterData)
    - Méthodes
      appendData(), insertData(), deleteData(), replaceData(), substringData() (héritées de CharacterData)
      replaceWholeText()

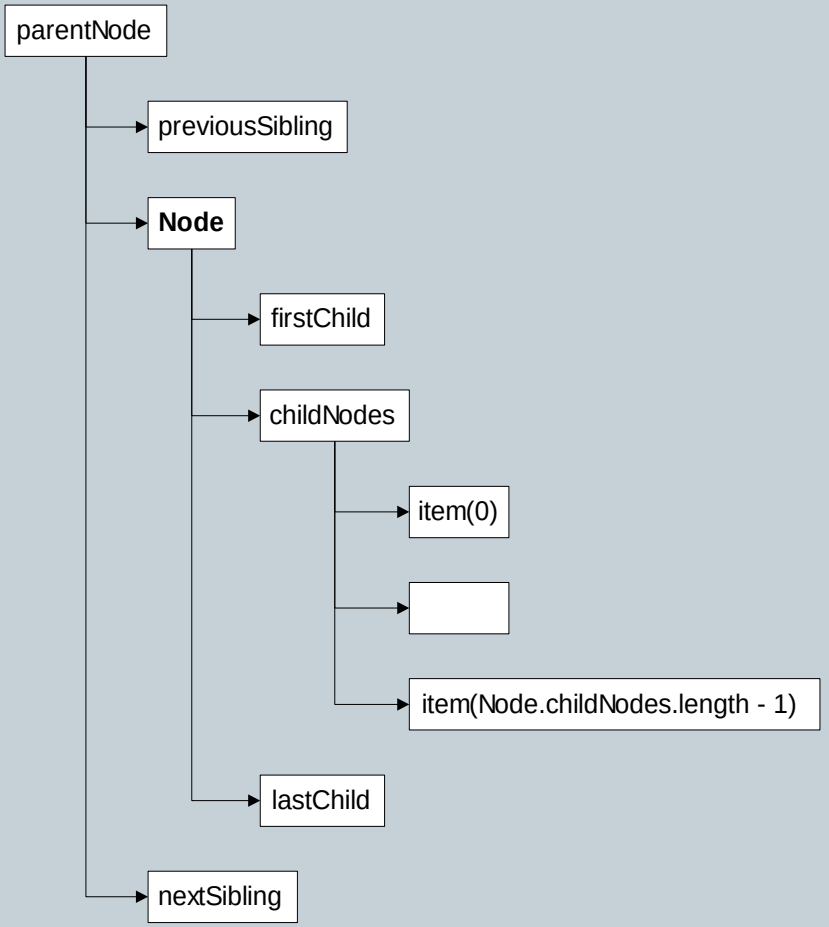
L'API DOM (Document Object Model)

- Hiérarchisation des interfaces d'un document XML



L'API DOM (Document Object Model)

- Déplacement dans une arborescence DOM (interfaces du module Core)



L'API DOM (Document Object Model)

- Conclusion sur le DOM
  - Utilisation du DOM XML en JavaScript
    - Utilisation des accesseurs pour accéder aux propriétés
    - DOM XML relativement standardisé sur les navigateurs récents
      - Exemple : document.getElementById()
    - En revanche, DOM HTML plus dépendant du navigateur
      - Exemple : monElement.innerHTML += …;
- Références sur le DOM
  - http://www.w3.org/DOM/
  - http://www.w3schools.com/dom/

Asynchronous Javascript And XML (AJAX)

- Introduction
  - Généralités
    - Applications web avec interface utilisateur
    - Déporter un maximum de code sur le client
      - Réduction des ressources consommées côté serveur
      - Réduction de la bande passante réseau
  - Applications Web AJAX les plus connues
    - Google (Mail, Map, Earth…)
    - Suggestions automatiques
    - Traitement de texte
    - …
  - Exemple
    - http://www.standards-schmandards.com/exhibits/ajax/

Asynchronous Javascript And XML (AJAX)

- Fonctionnement
  - Requête asynchrone au serveur dans une fonction JavaScript (déclenchée par un événement quelconque)
  - Transfert asynchrone de données en XML
  - Traitement dynamique côté client
    - Affichage (inclusion au document HTML, transformation XSLT…)
    - Logique applicative (fonctions JavaScript dédiées)
- Spécificité de la technologie AJAX
  - Requête asynchrone sur un document XML *via* un
    - Objet XMLHttpRequest (Mozilla)
    - Contrôle ActiveX XMLHTTP (IE)

Asynchronous Javascript And XML (AJAX)

- Fonctionnement
  - Étapes d'une communication AJAX côté client
    - Envoi de la requête
      - Créer un objet requête
      - Spécifier les éléments de la requête
        - URL, méthode , headers HTTP, paramètres
      - Lui associer un gestionnaire d'événement
      - L'envoyer
    - Réception de la réponse
      - À chaque changement d'état de la requête, tester si l'état est « ready »
      - Traiter les données reçues
        - Ajout à l'interface, transformation XSL…

Asynchronous Javascript And XML (AJAX)

- Fonctionnement
  - Étapes d'une communication AJAX côté serveur
    - Que doit faire un serveur Web à la réception d'une requête asynchrone AJAX ?

Asynchronous Javascript And XML (AJAX)

- Exemple de code : création d'un objet requête

```
var req = null;

function getRequest()
{
   if (window.XMLHttpRequest)
   {
      req = new XMLHttpRequest();
   }
   else if (typeof ActiveXObject != "undefined")
   {
      req=new ActiveXObject("Microsoft.XMLHTTP");
   }
   return req;
}
```

Safari / Mozilla

Internet Explorer

Asynchronous Javascript And XML (AJAX)

- Exemple de code : chargement asynchrone

```
function GetDataUsingAJAX (HttpMethod, url, params, elt)
{
   if(req != null)
   {
      // méthode avec paramètres
      req.onreadystatechange = function() {stateChange(elt)};
      // méthode sans paramètre
      // req.onreadystatechange = stateChange;

      req.open(HttpMethod, url, true);
      req.setRequestHeader("Accept", "application/xml");
      req.send(params);
    }
}
```

Association de la fonction de gestion de l'état

Asynchronous Javascript And XML (AJAX)

- Exemple de code : gestion de l'état

```
function stateChange (elt)
{
   if(req.readyState == 4) {
      if (req.responseXML != null) {
         var docXML= req.responseXML;
      } else {
         var docXML= req.responseText;
         docXML=parseFromString(docXML);
      }
      var docXMLresult = traiteXML(docXML);
      var str = (new XMLSerializer()).serializeToString(docXMLresult);
      document.getElementById(elt).innerHTML += str;
   }
}
```

READY_STATE_COMPLETE

Asynchronous Javascript And XML (AJAX)

- Exemple de code : transformation XSLT

```
//Après chargement asynchrone des documents XML et XSLT
function transform XSLT (XMLDoc, XSLDoc, id)
{
   if(XMLDoc == null || XSLDoc == null) {return;}
   try {
      if (window.ActiveXObject)
      {
         var target = document.getElementById(id);
         target.innerHTML = xml.transformNode(xsl);
      }
```

Internet Explorer

Asynchronous Javascript And XML (AJAX)

- Exemple de code : transformation XSLT

```
    } else if (window.XSLTProcessor) {
        var fragment;
        var xsltProcessor = new XSLTProcessor();
        xsltProcessor.importStylesheet(xsl);
        fragment = xsltProcessor.transformToFragment(xml, document);
        var target = document.getElementById(id);

        target.appendChild(fragment);
    }
  } catch (e) {
    return e;
  }
}
```

Safari / Mozilla

- Exemple de transformation XSLT côté client :

http://liris.cnrs.fr/lionel.medini/enseignement/TER/

Asynchronous Javascript And XML (AJAX)

- Implémentation de la logique applicative
  - Programmation d'un ensemble de fonctions JavaScript
    - Réécriture de fonctions existantes
    - Mélange de la logique métier et des fonctionnalités techniques
    - Pas forcément à l'épreuve des changements technologiques
    - Réutilisabilité moyenne
    - Code parfois un peu « fouillis »

Asynchronous Javascript And XML (AJAX)

- implémentation de la logique applicative
  - ➔ Pour de grosses applications
    - Utilisation de frameworks
      - Programmation dans un autre langage
      - Génération du code JavaScript
    - Exemples d'outils
      - openAjax (IBM) : Dojo, Rico ; Ruby / Ruby on Rails (RoR)
      - Plugin Eclipse : Rich Ajax Platform
      - … Et il y en a beaucoup d'autres