

TECHNOLOGIES
XML
EXTENSIBLE MARKUP LANGUAGE
PARTIE 1

Objectifs

2

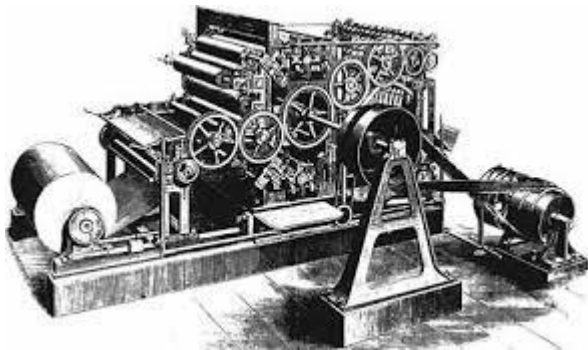
- Historique de XML et Interêt
- Comprendre la syntaxe XML et les modèles sous-jacents.
- Etudier et mettre en œuvre quelques outils (langages) pour la manipulation de XML.

Plan

3

- Introduction
- Définition de Document Type(DTD)
- Schéma XML
- XPATH
- Le langage de style XSLT
- Programmation XML(DOM & SAX)
- XQUERY
- BD XML

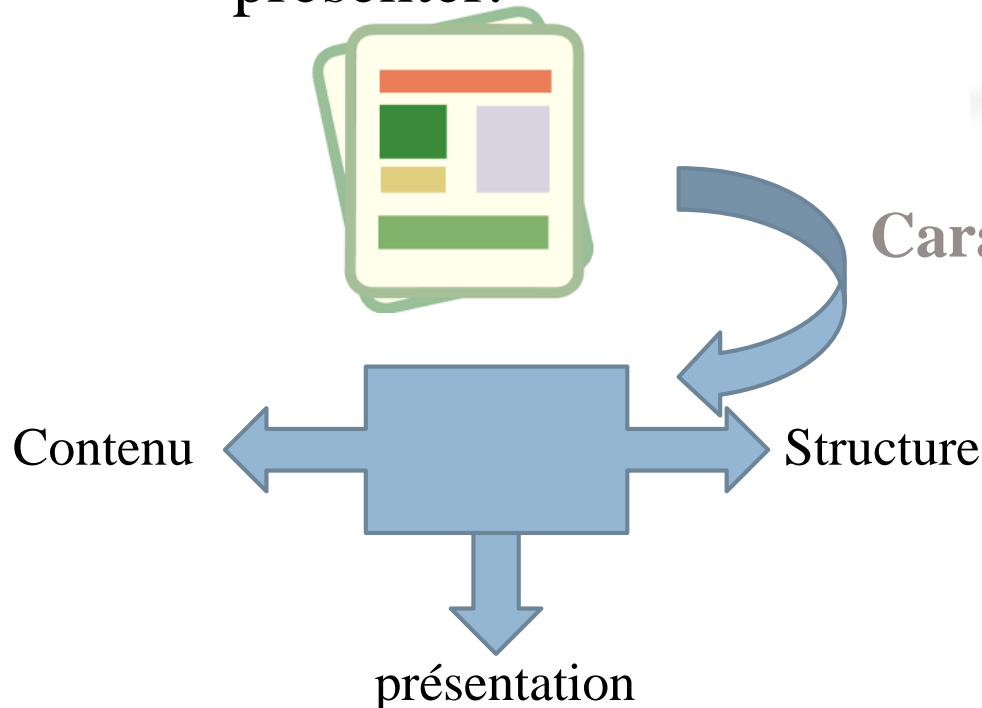
- ***Markup*** signifiait au début les symboles utilisés par l'imprimeur pour la mise en page et l'affichage:
 - marques, annotations manuscrites placées sur un document pour préciser à l'imprimeur comment il doit être présenté





Les documents électroniques prenaient de plus en plus de l'ampleur (avant 1967)

- Usage des macros spécifiques pour pouvoir les présenter.





Caractérisé

Markup 

Historique

6

- en 1967 William Tunnicliffe a proposé pour la première fois une approche pour séparer le fond du document par rapport à sa forme **GenCode**
 - propres notations de contrôle, souvent spécifiques à l'appareil cible
- 
- en 1969 des chercheurs de IBM(C.Goldfarb, E.Mosher et R.Lorie) ont présenté **GML** (*Generalized Markup Language*) basé sur le Gencode
 - faciliter l'échange d'informations entre les outils d'édition, de formatage et de recherche de documents
 - Le GML possédait déjà un concept de « type de documents » similaire aux documents DTD (**Document** Type Definition)
- 

Historique(suite)

7

- en 1974 Goldfarb a présenté SGML (Standard Generalized Markup Language)
 - publié en 1986 comme norme ISO (ISO 8879:1986).
 - est un langage de description de données qui divise un document en deux parties : la DTD (**Document** Type Definition) et les données elles-mêmes.
 - c'est un standard ouvert indépendant de toute plateforme ou constructeur.
 - Les fichiers SGML sont stockés sous forme de textes ASCII et peuvent donc être utilisés par n'importe quelle machine.

SGML

8

- ❑ SGML par sa performance a été utilisé par les militaires américains , le secteur aéronautique et par d'autres domaines nécessitant la gestion de documentations
- ❑ SGML sépare le contenu, la présentation et la structure d'un document
 - ❑ gère les présentations avec possibilité de multiples présentations et plusieurs formats (Postscript, Latex, HTML...) , l'échange des documents et la garanti de la pérennité

SGML critiques

9

- ❑ Très lourd et complexe
- ❑ Une grande rigueur est demandée à l'entrée des documents
- ❑ Liens hypertextes possibles mais complexes
 - Réservé aux professionnels de la documentation
 - Difficilement extensible au Web



HTML

- Avec le développement de l'Internet et pour la présentation des pages Web une application très légère du SGML a vu le jour appelée HTML (développé par Tim Berners-Lee) en 1990.

Avantages :

- Simple
- Intègre les liens hypertexte d'une manière facile
- facile d'intégrer la DTD HTML dans les navigateurs (Netscape, Explorer...)

Limites:

- Langage non extensible
 - figé (avec un nombre limité de balise)
- La sémantique du contenu est perdue.
 - ➔ On ne sait pas interpréter des données fournies en HTML
 - C'est un langage de balisage de présentation et non de contenu

Nouvelles utilisation du Web

11

- **Quelques applications :**
 - **Commerce électronique et échange de données informatisées (EDI):** les entreprises veulent échanger des informations et les intégrer dans leurs propres systèmes d'information
 - **Moteur de recherche:** besoin d'interprétation des données transmises pour les indexer efficacement
 - **Gestion électronique de documents (GED):** besoin d'indexer, d'organiser et de structurer le contenu des documents accessibles

Nouveaux besoins

12

- les données doivent être représentées indépendamment d'une machine donnée ou d'une application
- Besoin de transformer facilement les données d'un format à un autre d'une manière rapide et à moindre coût
- ...



- HTML est un langage pour présenter des informations à l'écran. il ne permet pas l'échange et le traitement (autre que l'affichage) des données.
- SGML est très complexe pour l'utiliser sur internet



XML

XML n'est pas le remplaçant de HTML:

- XML sert à structurer l'information
- HTML a pour but de la présenter

Pourquoi XML?

14

- **Un document XML est un format orienté texte** : il circule facilement
- **XML n'est pas lié à un mode d'utilisation** : C'est un métalangage et chacun peut définir son propre langage
 - SMIL(Synchronized Multimedia Integration Language): Permet l'intégration de données multimédias (image, vidéo, audio,...) en décrivant le déroulement temporel et spatial des différents composants intégrés
 - SVG (Scalable Vector Graphic) :utilisé pour décrire des graphiques en 2 dimensions
 - WML (Wireless Markup Language):conçu spécifiquement pour le WAP (protocole pour accès à Internet via un appareil de transmission sans fil) afin de pouvoir afficher sur un écran de téléphone mobile
 -
- **Beaucoup d'outils de manipulation** : XPath, DOM, SAX, XSLT, XQuery, etc

Exemples de domaine d'utilisation

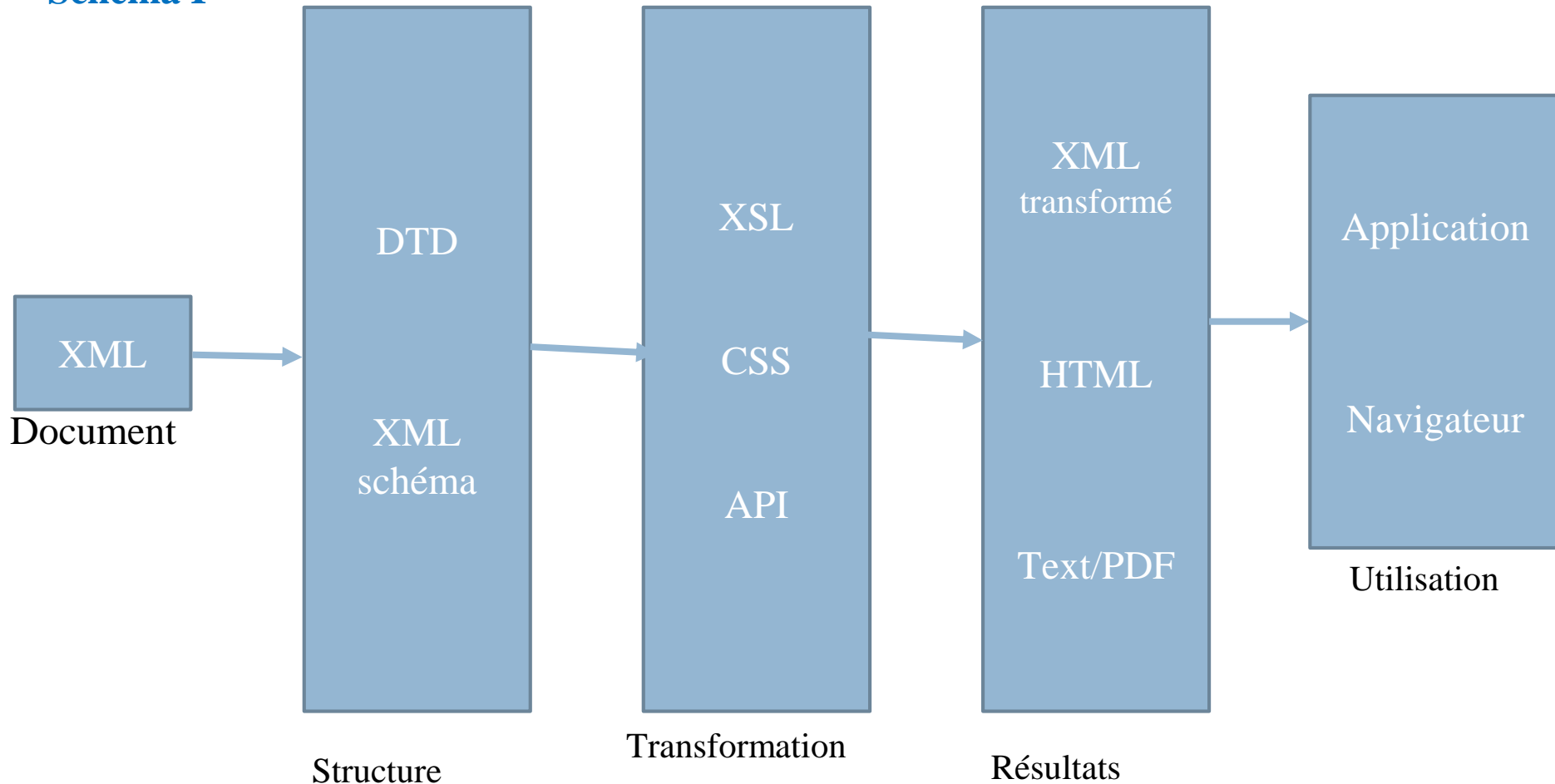
15

- Multimédia (MPEG 7 est basée sur XML)
- Intégration d'applications d'entreprise (EAI) avec la transformation des données vers XML (adaptateurs XML)
- Web Services : ensemble de protocoles et de normes informatiques utilisés pour échanger des données entre les applications
 - Le langage WSDL : description des services en XML
 - Le protocole SOAP : format des messages échangés en XML
- Web Sémantique: ensemble de technologies et de normes utilisés pour rendre le contenu Web accessible et utilisable par des logiciels
 - RDF (Resource Description Framework) qui est un modèle de description des ressources Web et leurs métadonnées afin de permettre le traitement automatique de telles descriptions
 - OWL (Web Ontology Language) : C'est un formalisme basé sur RDF qui fournit des moyens pour définir des ontologies Web structurés

Cycle de vie d'un document XML

16

Schéma I



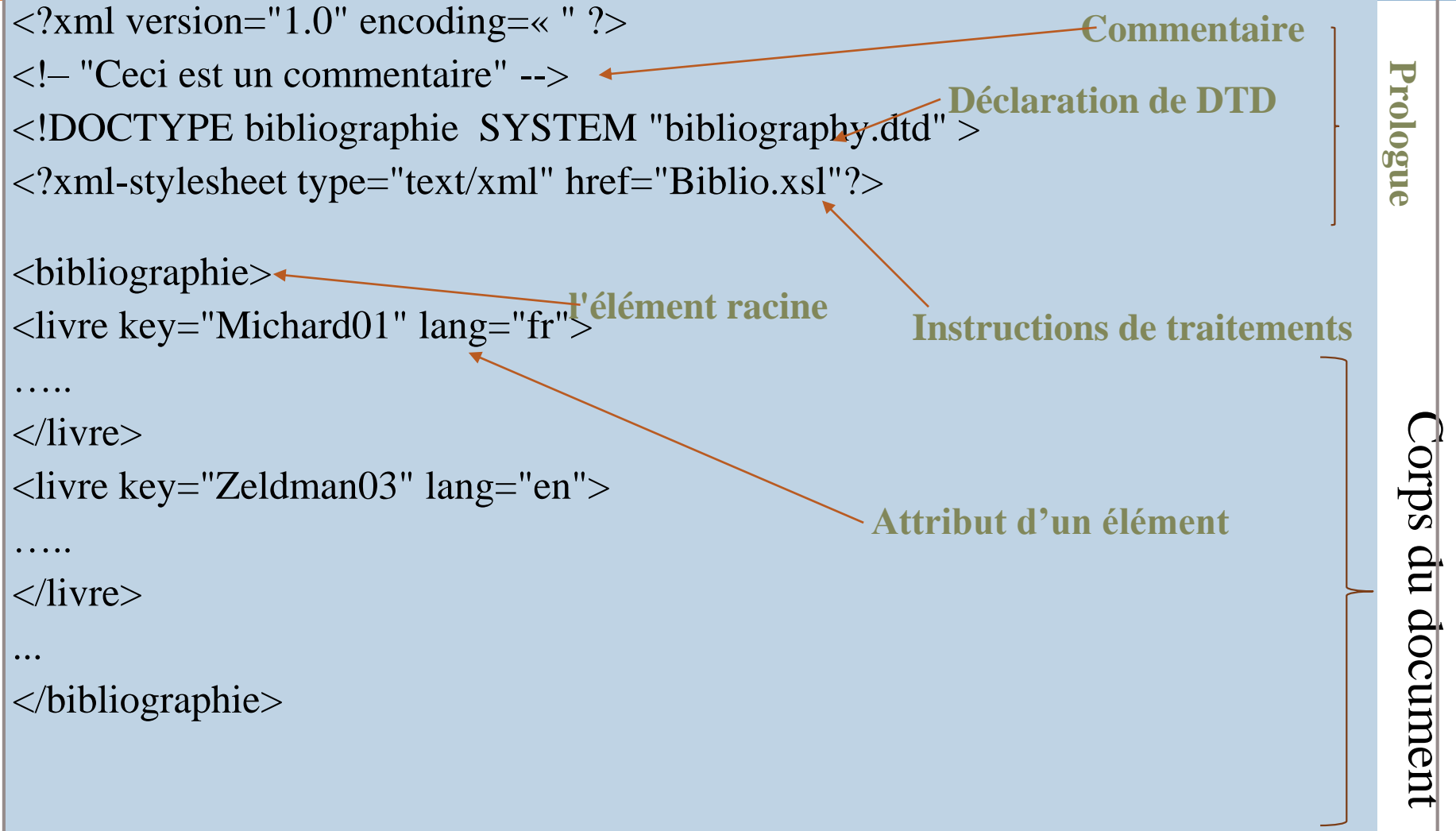
Structure d'un document XML

18

| | | |
|------------------------------------|---|----------|
| <code><?xml ... ?></code> |] | Prologue |
| <code>...</code> |] | |
| <code><root-element></code> |] | |
| <code>...</code> | | Corps |
| <code></root-element></code> |] | |

Exemple XML

19



Le prologue: entête

20

`<?xml version="1.0" encoding="UTF-8" standalone="no" ?>`

- Version="1.0" : version de la spécification XML utilisée.
Il existe aussi la version 1.1, mais la version 1.0 est la plus utilisée;
- Le jeu de caractères (*encoding*)
EX:
 - ISO-8859-1: convient pour la plupart des langues latines ou occidentales (anglais, français, allemand, espagnol...)
 - UTF-8 (par défaut) : indispensable pour les autres langues (japonais...)
- Standalone: faire savoir à l'avance à l'analyseur syntaxique XML si le document est autonome ou pas.
 - c'est-à-dire s'il existe des déclarations externes qui affectent le document (DTD externe, Entité paramétré...)
 - Standalone= "yes" indique au parseur que le document est indépendant
 - Standalone= " no " indique qu'il y a des déclarations de balisage externes.
- Le prologue peut contenir d'autre éléments: les commentaires, la déclaration de la DTD et les instructions de traitement

Le prologue : Commentaire

21

□ Commentaires

- Ils se positionnent n'importe où dans le corps du document XML où dans le prologue.

<!-- Ceci est un commentaire -->

- Un commentaire ne doit pas précéder le prologue
- Un commentaire ne peut être placé à l'intérieur d'une balise
- Il ne doit pas avoir de « -- » dans le texte du commentaire
- Un commentaire ne doit pas apparaître dans une balise CDATA (pas d'analyse et d'interprétation du contenu)

Le prologue: Instruction de traitement

22

Indique aux navigateurs
d'appliquer la feuille de style
myCss.css avant de la présenter
au lecteur

- Les instructions (processing instructions) permettent de traiter les documents avant de les présenter au lecteur
- Ils sont délimités par les chaînes de caractères '<? ' et '?>'

<?xml-stylesheet type="text/css" href=" myCss

Nom de l'IT

Indique aux robots si la
page peut être indexée et
si les liens peuvent être
explorés

- Le nom de l'IT permet à l'application de déterminer si l'instruction lui est destinée

<?robots index="yes/no" follow="yes/no">

Prologue: Déclaration de DTD externe

23

Déclaration de type de document (à l'aide d'un fichier annexe appelé DTD - *Document Type Definition*)

Ex:

```
<!DOCTYPE bibliography SYSTEM "bibliography.dtd" >
```

Les éléments

24

Composants de base d'un document XML

`<title>XML langage et applications</title>`

↑
balise ouvrante

↑
Contenu

↑
balise fermante

- Chaque élément peut contenir du texte simple (comme un fichier), d'autres éléments (comme un répertoire), ou un mélange des deux.
- Élément sans contenu (vide) :
`<identification isbn= "0-7357-1201-8" />`
- Toute balise rencontrée est analysée

Attributs

25

- Contient des informations sur l'élément

<livre key="Michard01" lang="fr">

nom

valeur

- Un élément peut avoir plusieurs attributs
- Les valeurs d'attributs:
 - Un nombre, une chaîne de caractères, une adresse,...

Attributs Prédéfinis

26

Il existe en XML quatre attributs prédéfinis:

- **xml:id**: permet d'associer un identificateur à tout élément indépendamment de toute DTD ou de tout schéma.
- **xml:lang**: spécifie la langue utilisée par les données de l'élément
 - Ex: `<p xml:lang="fr">Bonjour</p>`
 - `<p xml:lang="en-US">Hi</p>`
- **xml:space** (default|preserve): spécifie si les espaces doivent être préservés dans le contenu de l'élément

Attributs Prédéfinis (Suite)

27

■ **xml:base**

- L'attribut de `xml:base` permet de préciser l'URI de base d'un élément
- Par défaut, l'URI de base d'un élément est hérité de son parent.
- Permet de résoudre les références relatives à des URIs au sein du document XML.

```
<document xml:base="http://example.com/docs/">  
  <section>  
    <a href="section2.html"> Section suivante </a>  
  </section>  
</document>
```

Choisir élément ou attribut??

28

- Les éléments sont utilisés pour encapsuler des sous-ensembles de données,
- Les attributs sont généralement employés pour fournir des informations complémentaires sur un élément et non pour contenir des données brutes.

Utilisez les attributs si des données utilisés sont de type simple et :

- que vos informations requièrent une valeur fixe ou par défaut ;
- que vos informations requièrent des informations d'un élément existant;
- que la taille de votre fichier XML constitue un facteur critique; les attributs ont en effet généralement tendance à occuper quelques octets de moins que les éléments.
- Si on a pas de souci concernant l'ordre

Section CDATA: section littérale

29

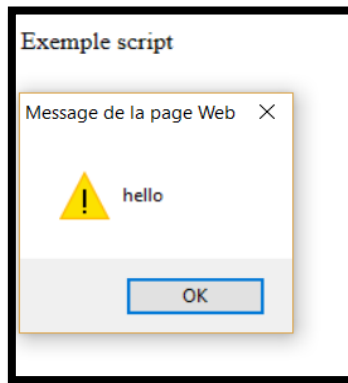
- CDATA signifie "Character data" = données textuelles
- Intérêt: Insérer du texte non interprété par le processeur XML
- La syntaxe:
 <![CDATA [
 Texte non analysé,
]]>

NB: on ne peut pas imbriquer deux sections CDATA

CDATA vs COMMENTAIRE ??

30

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>commentaire</title>
  </head>
  <body>
    <p>Exemple script</p>
    <script type="text/javascript">
      alert('hello');
    </script>
  </body>
</html>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>commentaire</title>
  </head>
  <body>
    <p>Exemple script</p>
    <!--
      <script type="text/javascript">
        alert('hello');
      </script>
    -->
  </body>
</html>
```

Exemple script

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>CDATA</title>
  </head>
  <body>
    <p>Exemple CDATA</p>
    <![CDATA[
      <script type="text/javascript">
        alert('hello');
      </script>
    ]]>
  </body>
</html>
```

Exemple CDATA

```
<script type="text/javascript"> alert('hello'); </script>
```

XML: Règles de syntaxe

31

- L'élément racine doit être unique;
- Le nom de l'élément commence par une lettre ou par _
- Si le nom est composé d'un seul caractère il doit être dans [a-zA-Z]
- Pas d'espaces ni de tabulations;
- Tous les éléments doivent être fermés dans l'ordre de leur ouverture;
 - `<livre> <auteur> <livre/> <auteur/>` : Faux
- Les balises XML sont sensibles à la casse;
- Les valeurs des attributs doivent être entre guillemets;
- *Un attribut d'un élément ne peut avoir qu'une seule valeur;*
- Ne pas utiliser de caractères réservés à XML dans le texte du document : `<`, `>` et `&` ; ces caractères pourront être respectivement obtenues à l'aide des entités suivants:
`<`, `>` et `&` ;
- Un document XML a pour extension .xml



Conventions de nommage

32

- Employer des minuscules pour les attributs et les éléments
- Évitez les accents dans les noms d'attributs et d'éléments
- Séparer les noms composés par `_`, `-`, `..`, majuscule

Espace de nom: Besoin

33

□ Exemple:

<cours>

<titre> Technologie Web</titre>

<intervenant>

<nom>AZIM</nom>

<prénom>Mohamed</prénom>

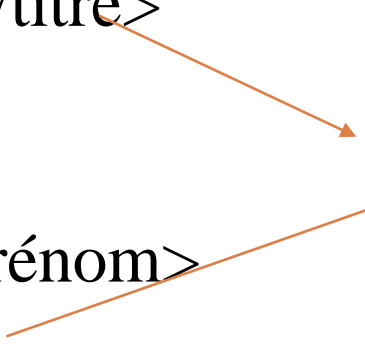
<titre>Vacataire</titre>

</intervenant>

.....

</cours>

Comment distinguer
entre les deux titres ??



Espace de nom: Besoin

34

Mélanger plusieurs vocabulaires au sein d'un même document xml



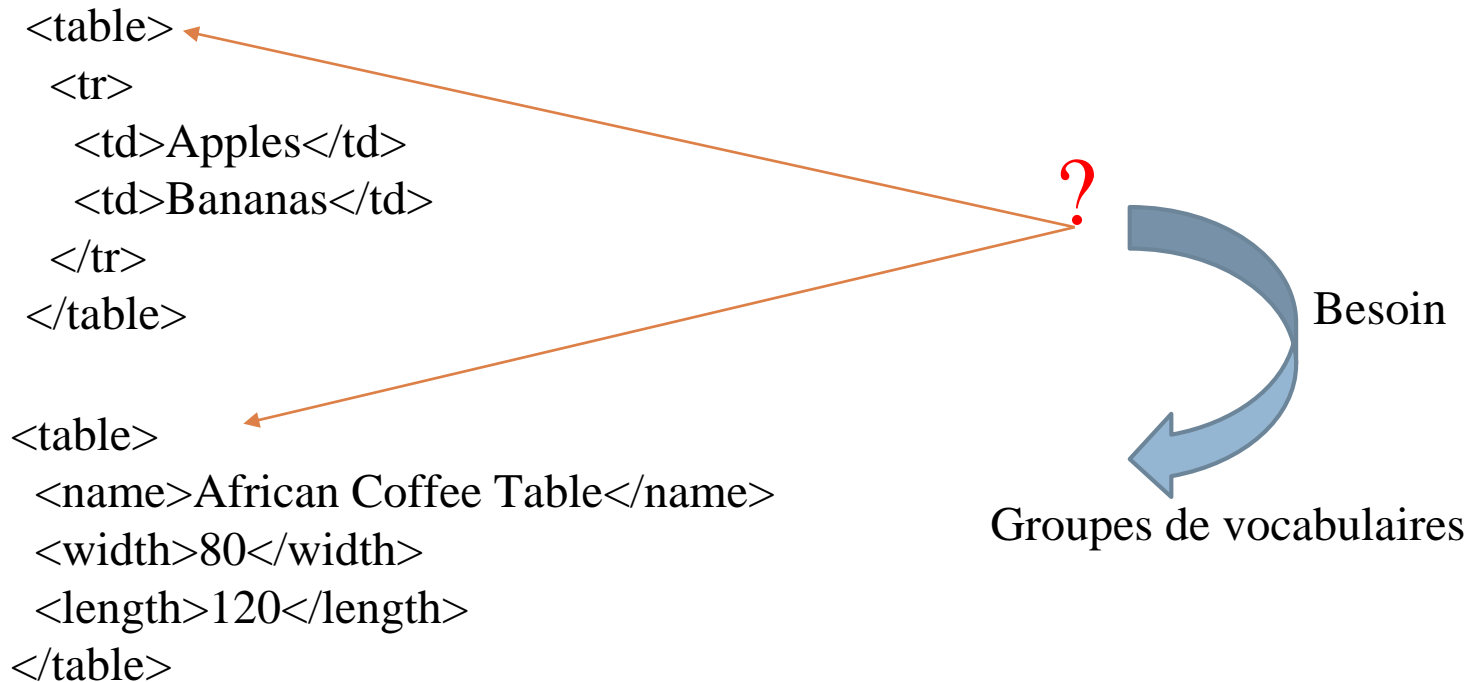
Identifier la provenance de chaque élément et de chaque attribut pour le valider correctement



Espace de nom

Exemple

35



Espace de nom

36

- Les espaces de nommage XML représentent une méthode simple pour la qualification des noms des éléments et attributs utilisés dans les documents XML, en les associant à des espaces de nommage identifiés par des *adresses URI [w3C]*

Espace de nom

37

- Pour identifier les espaces de nom, *le W3C a utilisé les identificateurs du Web, c'est à dire les **URI** (Uniform Resource Identifier).*
- Ex : URIs qui identifient des vocabulaires:
 - xhtml : <http://www.w3.org/1999/xhtml>
 - Schéma XML : <http://www.w3.org/2001/XMLSchema>
 - DocBook (langage sémantique pour la documentation technique): <http://docbook.org/ns/docbook>
 - Dublin Core(schéma de métadonnées générique qui permet de décrire des ressources numériques ou physiques et d'établir des relations avec d'autres ressources): <http://purl.org/dc/elements/1.1/>

Vocabulaire qui permet de décrire une page xhtml

Espace de nom: Syntaxe

38

- Espaces de noms sans préfixe:
 - **xmlns**="mon_uri"

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Ma page</title>
...
</head>
</html>
```

- La déclaration de l'espace de noms par défaut se fait au moyen de l'attribut
- Elle s'applique à l'élément dans lequel elle est effectuée et à tous ses enfants

Espace de nom: Syntaxe

39

□ Les espaces de noms avec préfixe

xmlns:prefix="mon_uri"

```
<cours xmlns:cours="http://www.ensias.ma/cours">
```

```
  <cours:titre>Technologie Web</cours:titre>
```

```
<intervenant
```

```
  xmlns:personne="http://www.ensias.ma/personnel">
```

```
    <personne:nom> AZIM </personne:nom>
```

```
    <personne:prenom> Ahmed </personne:prenom>
```

```
    <personne:titre> Vacataire</personne:titre>
```

```
  </intervenant >
```

```
</cours>
```

Espace de nom (Suite)

40

- Un espace de nom déclaré sans préfixe devient l'espace de nom par défaut de tous les éléments descendants
- Les éléments non qualifiés appartiennent à l'espace de nom le plus interne

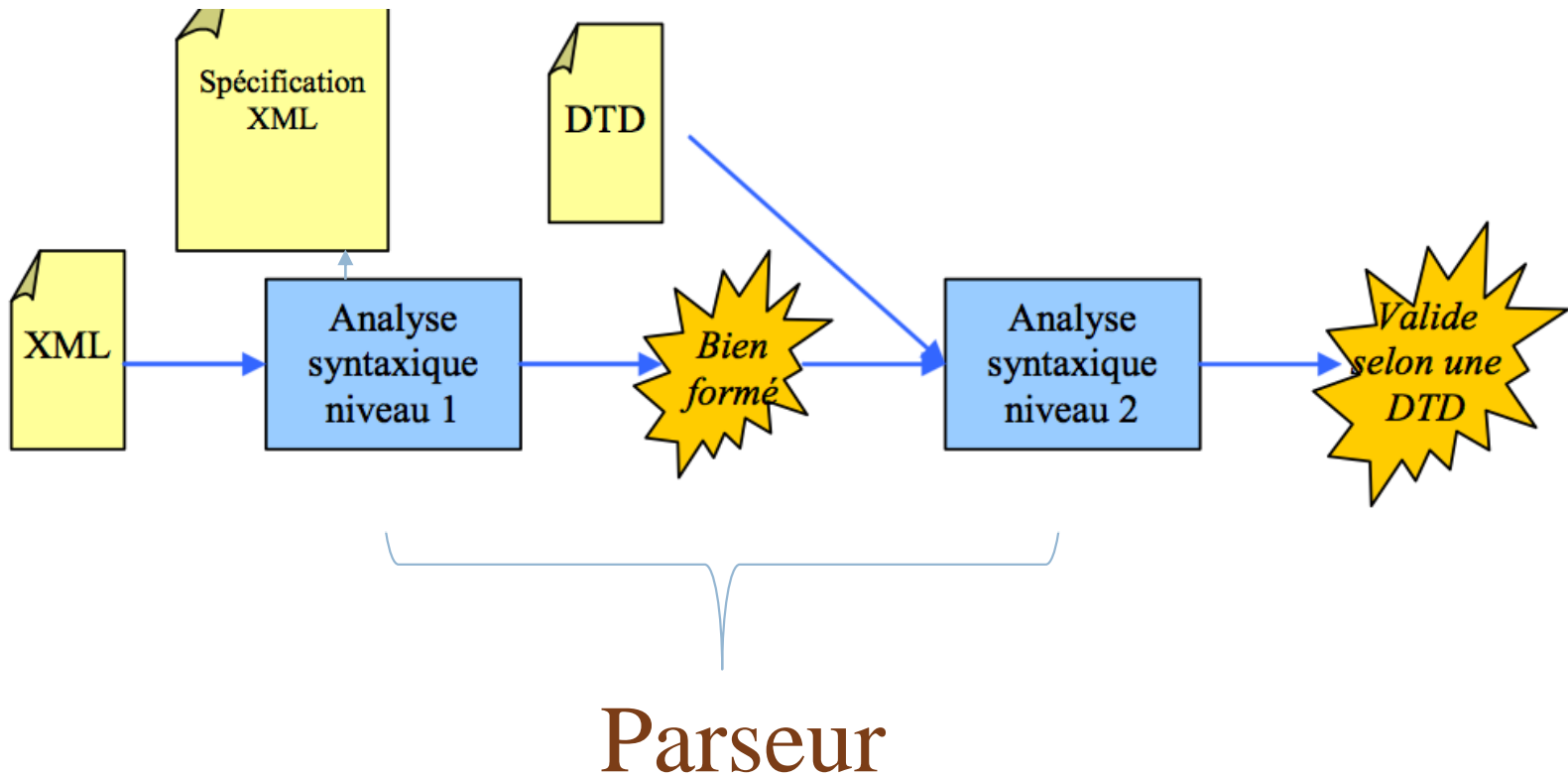
Document bien formé et valide

41

- Le document est **bien formé** s'il respecte les règles de base définies dans la spécification XML.
- Le document est **valide** s'il est bien formé et conforme à une grammaire (DTD ou schéma)

Validation de document

42



Parseurs

43

- XML permet de structurer et de représenter les données
 - Il ne permet pas d'exploiter directement les données
- Un analyseur syntaxique XML (ou parseur), permet de **récupérer dans une structure** hiérarchique XML, **des balises**, leur **contenus**, leurs **attributs** pour les rendre accessibles.
 - EX: Les navigateurs intègrent un parseur → affichage du fichier xml sous forme d'un arbre
- Deux types d'approches sont possibles:
 - *Hiérarchique* : manipulation d'un document XML après l'avoir représenté en mémoire sous la forme d'un arbre d'objets (DOM)
 - *Événementiel*: manipulation d'un document XML au fur et à mesure de sa lecture, sans avoir à le charger en totalité en mémoire (SAX)
- Deux types de parseurs XML:
 - *les parseurs non validant* qui vérifient que le document XML est bien formé (respecte la syntaxe XML de base)
 - *les parseurs validant* qui permettent de vérifier la conformité d'un document XML avec la DTD ou schema XML

Atelier 1

44

Création d'un document XML