Humanités numériques : introduction à l'édition de textes et d'instruments de recherche (XML)

E. Rouquette

Cours 1 – 09 novembre 2023

Introduction

Présentation du cours

Ressources du cours : https://github.com/Enimie/AP2_XML

Présentation du cours

Ressources du cours : https://github.com/Enimie/AP2_XML

Objectifs du cours

- Comprendre les enjeux liés aux XML
- Savoir lire et composer un document XML de base
- Connaître les technologies liées à XML
- Apprendre les éléments de base dans les langages TEI et EAD
- Apprendre à transformer et à exploiter ses documents XML (XSLT, XPath)

Présentation du cours

Ressources du cours : https://github.com/Enimie/AP2_XML

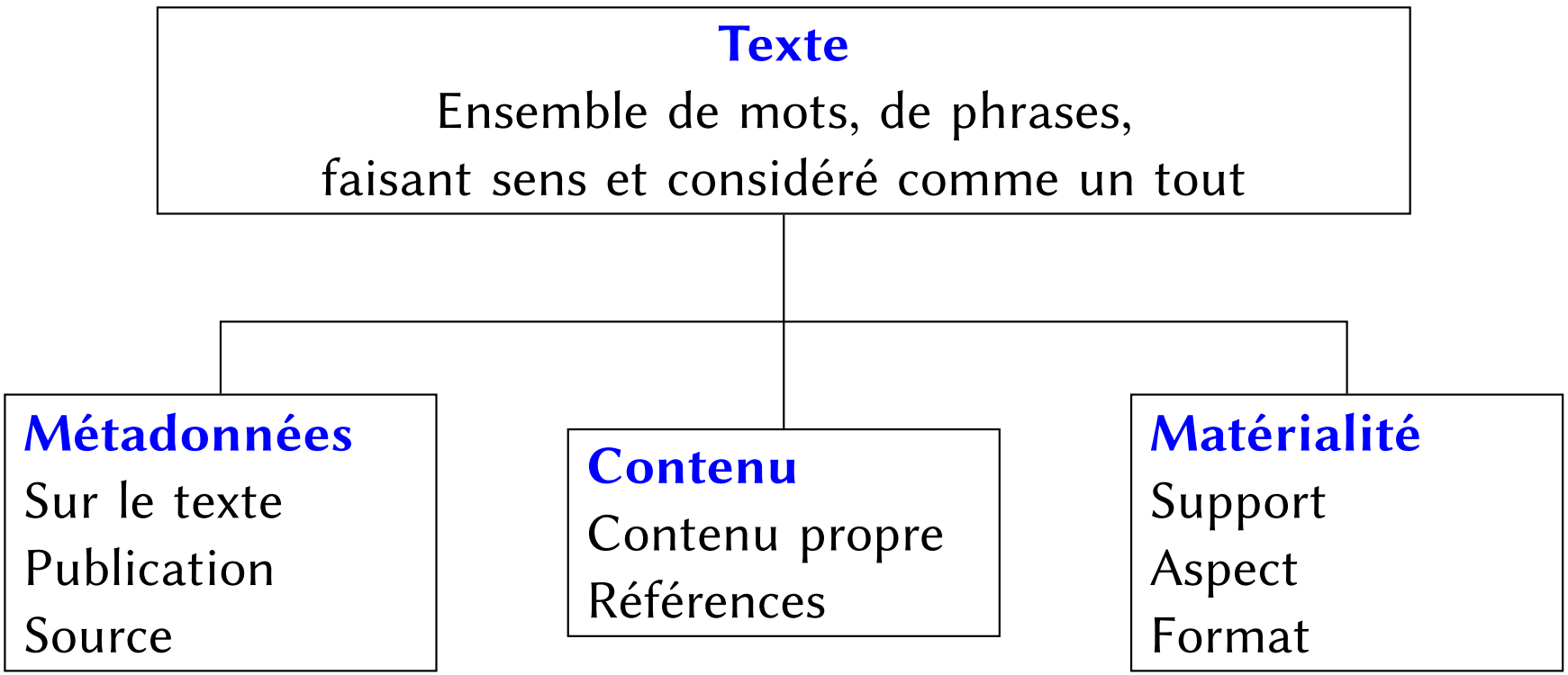
Objectifs du cours

- Comprendre les enjeux liés aux XML
- Savoir lire et composer un document XML de base
- Connaître les technologies liées à XML
- Apprendre les éléments de base dans les langages TEI et EAD
- Apprendre à transformer et à exploiter ses documents XML (XSLT, XPath)

5 séances de deux heures :

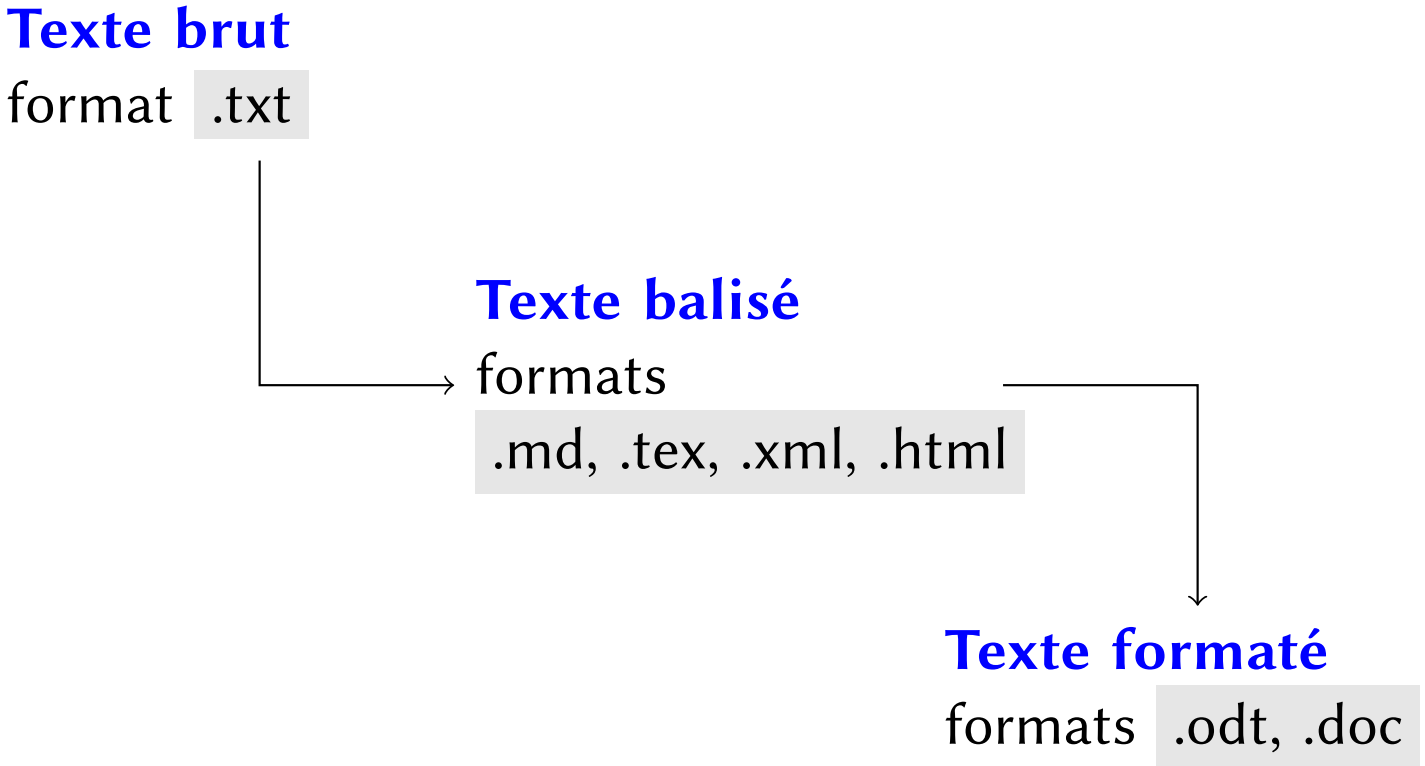
- Deux séances sur le langage et les schémas XML
- Deux séances sur l'édition scientifique avec XML-TEI
- Une séance sur l'EAD et sur XPath

Qu'est-ce qu'un texte ?



Qu'est-ce qu'un texte ?

Le texte informatisé



Qu'est-ce qu'un texte ?

Texte brut

```
Le 15 septembre 1840, vers six heures du matin, la
 ↪  Ville-de-Montereau, près de partir, fumait à gros
 ↪  tourbillons devant le quai Saint-Bernard.
```

Texte balisé en XML

```
<paragraphe>Le <date>15 septembre 1840</date>, vers
 ↪  six heures du matin, la
 ↪  <nomBateau>Ville-de-Montereau</nomBateau>, près
 ↪  de partir, fumait à gros tourbillons devant le
 ↪  quai
 ↪  <nomLieu>Saint-Bernard</nomLieu>.</paragraphe>
```

Texte formaté

Le 15 septembre 1840, vers six heures du matin, la *Ville-de-Montereau*, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Les langages à balises

- Markdown
- LaTeX
- html
- XML

→ **Mise en forme** (typographique) *vs* **mise en sens** (sémantique)

Les langages à balises

- Markdown
- LaTeX
- html
- XML

Texte balisé en markdown

```
# Première partie
## I
Le 15 septembre 1840, vers six heures du matin, la
 ↪  *Ville-de-Montereau*, près de partir, fumait à
 ↪  gros tourbillons devant le quai Saint-Bernard.
```

Les langages à balises

- Markdown
- LaTeX
- html
- XML

Texte balisé en LaTeX

```
\part{Première partie}
\chapter{I}
Le 15 septembre 1840, vers six heures du matin, la
 ↪  \emph{Ville-de-Montereau}, près de partir, fumait
 ↪  à gros tourbillons devant le quai Saint-Bernard.
```

Les langages à balises

- Markdown
- LaTeX
- html
- XML

Texte balisé en html

```
<h1e>Première partie</h1>
<h2>I</h2>
<p>Le 15 septembre 1840, vers six heures du matin, la
 ↪  <i>Ville-de-Montereau</i>, près de partir, fumait
 ↪  à gros tourbillons devant le quai
 ↪  Saint-Bernard>.</p>
```

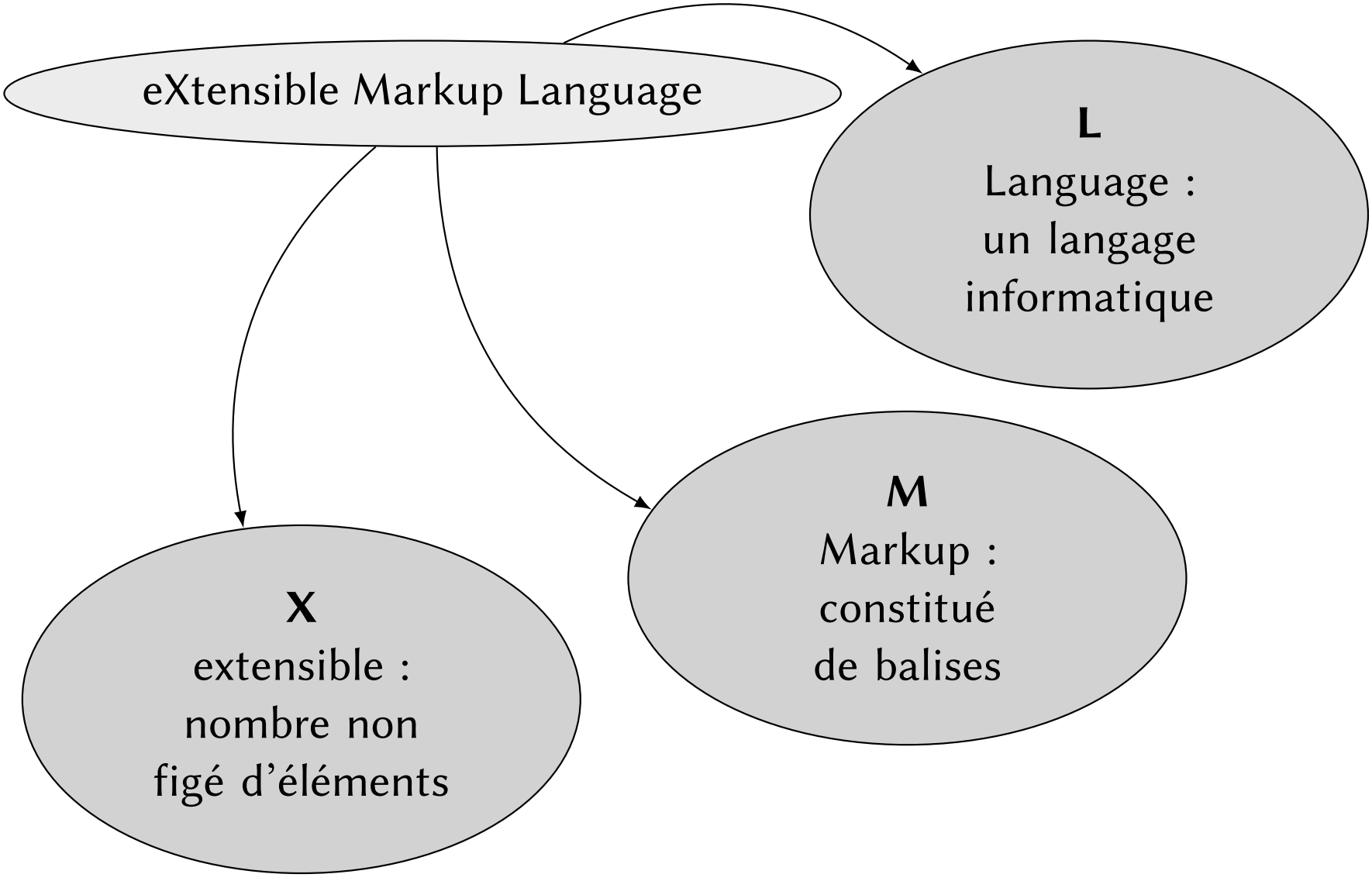
Les langages à balises

- Markdown
- LaTeX
- html
- XML

Texte balisé en XML

```
<titrePartie>Première partie</titrePartie>
<numeroChapitre>I</numeroChapitre>
<paragraphe>Le <date>15 septembre 1840</date>, vers
 ↪  six heures du matin, la
 ↪  <nomBateau>Ville-de-Montereau</nomBateau>, près
 ↪  de partir, fumait à gros tourbillons devant le
 ↪  quai
 ↪  <nomLieu>Saint-Bernard</nomLieu>.</paragraphe>
```

XML : présentation



Historique

Contexte :

- convergence numérique
- échange de l'information
- Multiplication des supports

→ **faciliter l'échange** de contenus complexes (arbres, textes enrichis, etc.) entre différents systèmes informatiques (**interopérabilité**)

Historique

Quelques dates

1970 **SGML**, Standard Generalized Markup Language, créé par IBM pour échanger les données.

1989-1992 **HTML**, HyperText Markup Language, par Tim Berners-Lee, spécialisé dans l'affichage des données.

1996 création, par un groupe de travail, de XML pour décrire les données

1998 **XML** devient une **norme du W3C**

1999 reformulation d'HTML 4 conformément au principe d'XML → **XHTML** 1.0

Concepts-clefs

Pérennité

Besoin de formats qui durent dans le temps pour ne pas avoir sans cesse à convertir les fichiers au fil du temps.

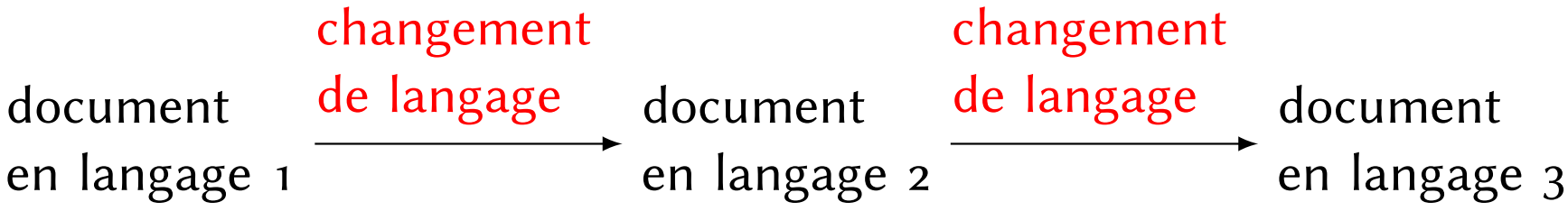
Interopérabilité

Besoin de formats interopérables, c'est-à-dire qui puissent être échangés et réutilisés, également pour ne pas avoir à convertir sans cesse ses données

W3C (World Wide Web Consortium)

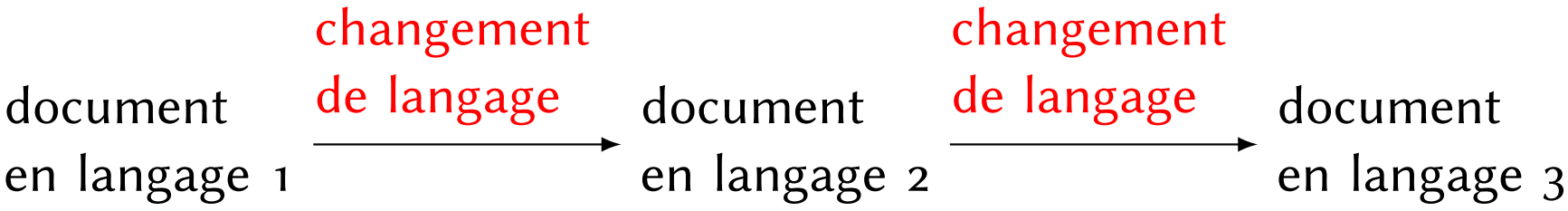
Le W3C est un consortium qui vise la standardisation des technologies du Web. Il promeut des standards pour favoriser l'interopérabilité et la pérennité des données.

Pérennité



Lorsqu'un langage n'est pas pérenne...

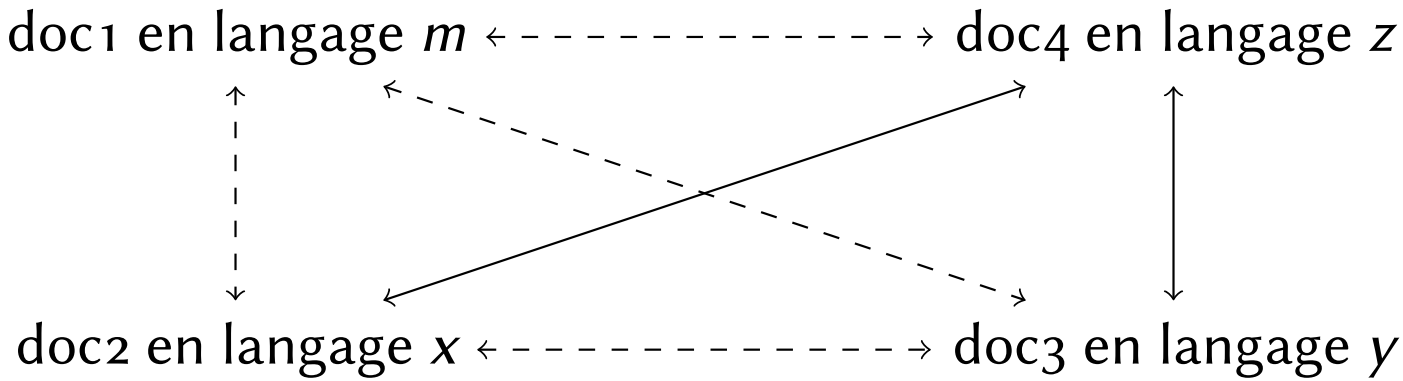
Pérennité



Lorsqu'un langage n'est pas pérenne...

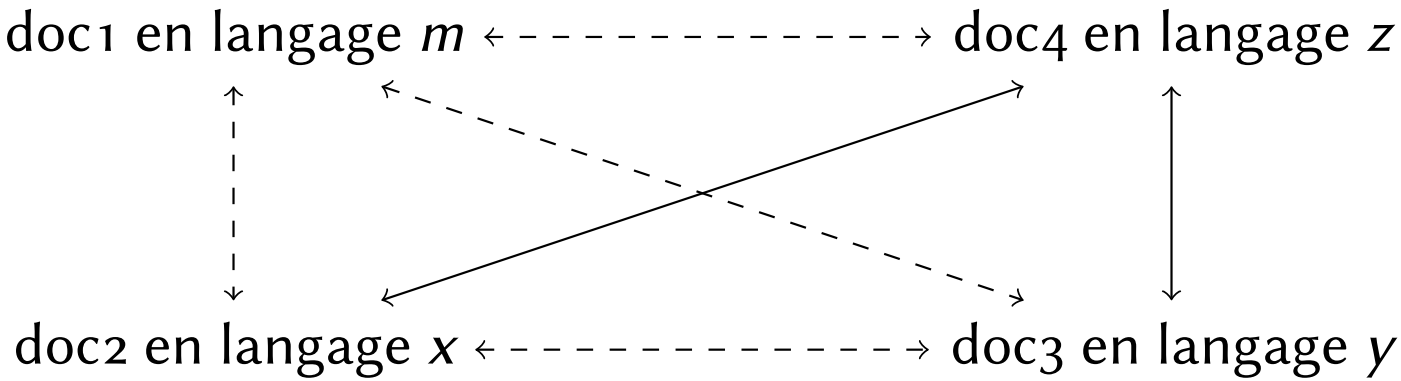
Lorsqu'un langage est pérenne...

Interopérabilité

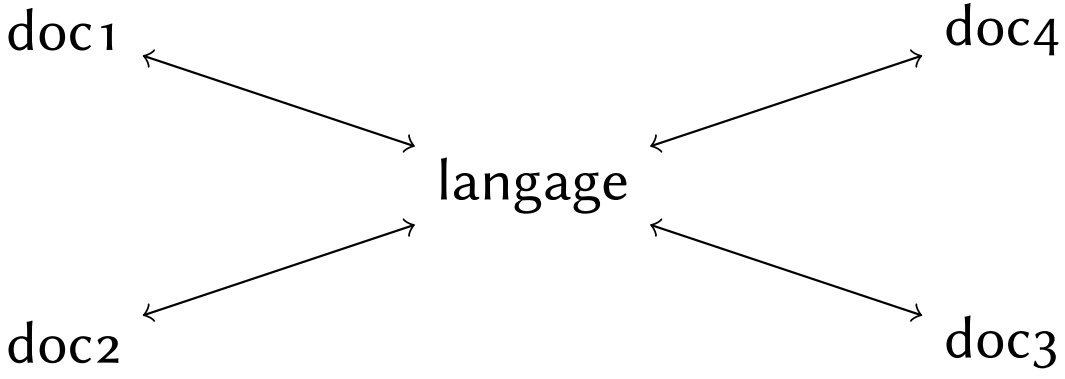


Quand il n'y a pas de langage interopérable...

Interopérabilité



Quand il n'y a pas de langage interopérable...



Quand il y a un langage interopérable

Qu'est-ce qu'XML ?

- XML ne « fait » rien !
- XML est un langage de structuration des données :
  - Les informations (données) sont contenues dans des balises
  - XML décrit les données
  - XML stocke les données
- Tout type de données :
  - Textes
  - Textes structurés
  - Image
  - ...

Mais surtout du texte !

Avantages de XML

Avantages internes

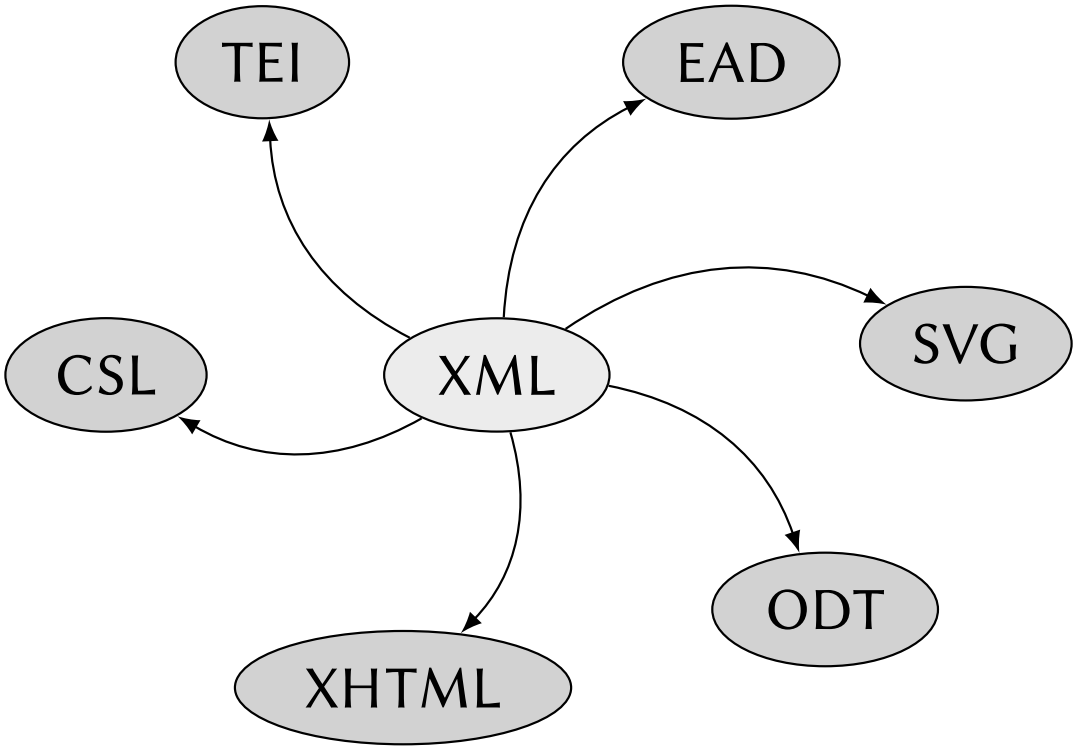
- séparation entre fond et forme, contenu et présentation
- langage simple : du texte
- lisible par l'humain et la machine
- langage structuré
- souple car extensible
- composable (plusieurs langages au sein d'un même document)

Avantages externes

- indépendant des plateformes
- standard du W3C : longévité et interopérabilité
- adaptable à tous les types de description qu'on veut mener (on peut créer des langages)

Omniprésence de XML

XML est un métalangage.



Omniprésence de XML

- standards XML utilisés dans de nombreuses bases de données (formats BnF : METS, MODS...)
- langage XML/EAD utilisé dans la description des fonds d'archives et manuscrits
- langage XML/TEI largement utilisé dans les projets de recherche en SHS (éditions numériques).

Liste d'éditions numériques sur le site www.digitale-edition.de
Exemple d'édition numérique en TEI : http://stendhal.demarre-shs.fr/index2.php?show=pageapage&id=0

Outils liés à XML

XPath	**sélection** des données
XPointer	**lien** entre les données
XQuery	**interrogation** des données
XSL	**transformation** des données
schémas	**standardisation** de la validation des documents

Ce que fait ressortir XML

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
<titre>L'Éducation sentimentale</titre>
<auteur>Gustave Flaubert</auteur>
<texte>
<partie><titrePartie>Première partie</titrePartie>
<chapitre><numeroChapitre>I</numeroChapitre>
<paragraphe>Le <date>15 septembre 1840</date>, vers
 ↪  six heures du matin, la
 ↪  <nomBateau>Ville-de-Montereau</nomBateau>, près
 ↪  de partir, fumait à gros tourbillons devant le
 ↪  quai <nomLieu> Saint-Bernard</nomLieu>.
</paragraphe>
</chapitre>
</partie>
</texte>
</document>
```

Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la *Ville-de-Montereau*, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la *Ville-de-Montereau*, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Des métadonnées

```
...
<titre>L'Éducation sentimentale</titre>
<auteur>Gustave Flaubert</auteur>
...
```

Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la *Ville-de-Montereau*, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Une structure

`<partie>`, `<chapitre>`, `<paragraphe>`

Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la *Ville-de-Montereau*, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Des informations sémantiques sur le texte

`<date>`, `<nomBateau>`, `<nomLieu>`

Les balises structurent l'information

- elles peuvent encadrer une information de type texte, avec une balise ouvrante et une balise fermante

  ```
  <balise>texte</balise>
  ```

- elles peuvent être auto-fermantes

  ```
  <balise/>
  ```

- elles peuvent contenir d'autres balises

  ```
  <balise1>
  <balise2>contenu</balise2>
  <balise3/>
  </balise1>
  ```

Les balises permettent de stocker et d'interroger l'information

- Obtenir à partir d'un même fichier plusieurs sorties différentes
- Interroger le contenu d'un texte à partir de son sémantisme (recherche de noms de personne, de nom de lieu, de date, etc)

Forme des balises

Balise ouvrante :
Chevron gauche - nom de la balise - chevron droit

<balise>

Forme des balises

Balise ouvrante :
Chevron gauche - nom de la balise - chevron droit
Balise fermante :
Chevron gauche - slash - nom de la balise -chevron droit

<balise>Contenu de la balise</balise>

Forme des balises

Balise ouvrante :
Chevron gauche - nom de la balise - chevron droit
Balise fermante :
Chevron gauche - slash - nom de la balise -chevron droit
Balise autofermante (balise « borne », *milestones*)
Chevron gauche - nom de la balise - slash - chevron droit

<balise>Contenu de la balise</balise>
<balise/>

Forme des balises

Balise ouvrante :
Chevron gauche - nom de la balise - chevron droit
Balise fermante :
Chevron gauche - slash - nom de la balise -chevron droit
Balise autofermante (balise « borne », *milestones*)
Chevron gauche - nom de la balise - slash - chevron droit

<balise>Contenu de la balise</balise>
<balise/>

Exemple :

```
<paragraphe> début d'un paragraphe au milieu duquel
 ↪  il y a un changement de page <pagebreak/> suite
 ↪  du paragraphe </paragraphe>
```

Syntaxe

Le prologue

```
<?xml version="1.0" encoding="UTF-8"?>
```

Un document XML commence obligatoirement par une **déclaration XML (ou prologue)**. Il indique la version de la recommandation XML qui sert de base à l'écriture du fichier (il n'y a eu à cette date qu'une seule version, la version 1.0), ainsi que l'encodage de caractères utilisé.
Cette déclaration permet à la machine de comprendre que le document en question est un document XML : c'est une **instruction pour le processeur**

Les éléments

Un élément est le contenu d'une balise :

```
<element>contenu</element>
```

Élément vide (*milestone*) :

```
<element/>
```

Les éléments

Règles (principe de conformité) :

Nom Un élément doit être entouré de deux balises de même nom (sensible à la casse)

Les éléments

Règles (principe de conformité) :

Nom Un élément doit être entouré de deux balises de même nom (sensible à la casse)

Clôture Un élément ouvert doit être fermé obligatoirement

Les éléments

Règles (principe de conformité) :

Nom Un élément doit être entouré de deux balises de même nom (sensible à la casse)

Clôture Un élément ouvert doit être fermé obligatoirement

Imbrication Un élément ne peut pas en chevaucher un autre

Les éléments

Règles (principe de conformité) :

Nom Un élément doit être entouré de deux balises de même nom (sensible à la casse)

Clôture Un élément ouvert doit être fermé obligatoirement

Imbrication Un élément ne peut pas en chevaucher un autre

⚠ On ne peut pas écrire

```
<element/>contenu</Element>
```

ni :

```
<element/>contenu
```

ni :

```
<element/><element2>contenu</element></element2>
```

Les éléments

XML est un métalangage ; donc :

- il n'est **pas doté de balises spécifiques** : on peut inventer ses propres balises.

`<p>contenu</p>` ✓
`<par>contenu</par>` ✓
`<paragraph>contenu</paragraph>` ✓

- il sert d'**architecture** pour les langages XML/TEI, XML/EAD qui, eux sont dotés d'une grammaire précise (listes de balises et de valeurs, manière de les agencer)

`<p>contenu</p>` ✓
`<par>contenu</par>` ✗

Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

- une numérotation

```
<paragraphe n="1">contenu du paragraphe 1</p>
<paragraphe n="2">contenu du paragraphe 2</p>
```

Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

- une numérotation

```
<paragraphe n="1">contenu du paragraphe 1</p>
<paragraphe n="2">contenu du paragraphe 2</p>
```

- des types liés à la structure du document

```
<titre type="principal">L'Éducation sentimentale</titre>
<titre type="partie">Première partie</titre>
```

Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

- une numérotation

```
<paragraphe n="1">contenu du paragraphe 1</p>
<paragraphe n="2">contenu du paragraphe 2</p>
```

- des types liés à la structure du document

```
<titre type="principal">L'Éducation sentimentale</titre>
<titre type="partie">Première partie</titre>
```

- des types liés au sémantisme des éléments

```
... devant le quai <nomLieu>Saint-Bernard>
```

Les attributs et leur valeur

Utilisation :

Spécification Utilisé pour spécifier un élément

Les attributs et leur valeur

Utilisation :

Spécification Utilisé pour spécifier un élément

Situation Placés après le nom de l'élément dans la balise ouvrante

Les attributs et leur valeur

Utilisation :

Spécification	Utilisé pour spécifier un élément
Situation	Placés après le nom de l'élément dans la balise ouvrante
Répétabilité	Un même attribut ne peut pas être réutilisé dans la même balise

Les commentaires

Un autre type d'élément peut apparaître dans les documents XML, les commentaires :

```
<!-- Un commentaire -->
```

Utilisation :

Les commentaires

Un autre type d'élément peut apparaître dans les documents XML, les commentaires :

```
<!-- Un commentaire -->
```

Utilisation :

Syntaxe Les commentaires sont placés entre les marques <!-- (ouvrante) et --> (fermante)

Les commentaires

Un autre type d'élément peut apparaître dans les documents XML, les commentaires :

```
<!-- Un commentaire -->
```

Utilisation :

Syntaxe Les commentaires sont placés entre les marques <!-- (ouvrante) et --> (fermante)

Rôle Ils servent à à faire des remarques sur les usages de l'encodage, sur les choses à faire, etc. Ils sont destinés à un lecteur humain

Exercice

Bien formé ou mal formé ?

```
<par>du texte</par>
<par><article>du</article><nom>texte</nom></par>
<par><article>du <nom></article>texte</nom></par>
<par><article>du <nom>texte</nom></article></par>
<par type="texte"> du texte</par>
<par type=texte> du texte</par>
<par type="texte"> du texte<par/>
<par type="texte"> du texte<nom>nom</par>
<par type="texte"> du texte</par>
<segment type="texte" type="nombre"> du texte</par>
```

L'arborescence

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
<titre>L'Éducation sentimentale</titre>
<auteur>Gustave Flaubert</auteur>
<texte>
<partie><titrePartie>Première partie</titrePartie>
<chapitre><numeroChapitre>I</numeroChapitre>
<paragraphe>Le <date>15 septembre 1840</date>, vers
 ↪  six heures du matin, la
 ↪  <nomBateau>Ville-de-Montereau</nomBateau>, près
 ↪  de partir, fumait à gros tourbillons devant le
 ↪  quai <nomLieu>Saint-Bernard</nomLieu>.
</paragraphe>
</chapitre>
</partie>
</texte>
</document>
```

L’arborescence

Un élément racine

```
<document>
...
</document>
```

L’élément racine est obligatoire ; il contient tous les autres éléments

L’arborescence

Les éléments enfants de l’élément racine

```
<document>
<titre>...</titre>
<auteur>...</auteur>
<texte>...</texte>
</document>
```

L'arborescence

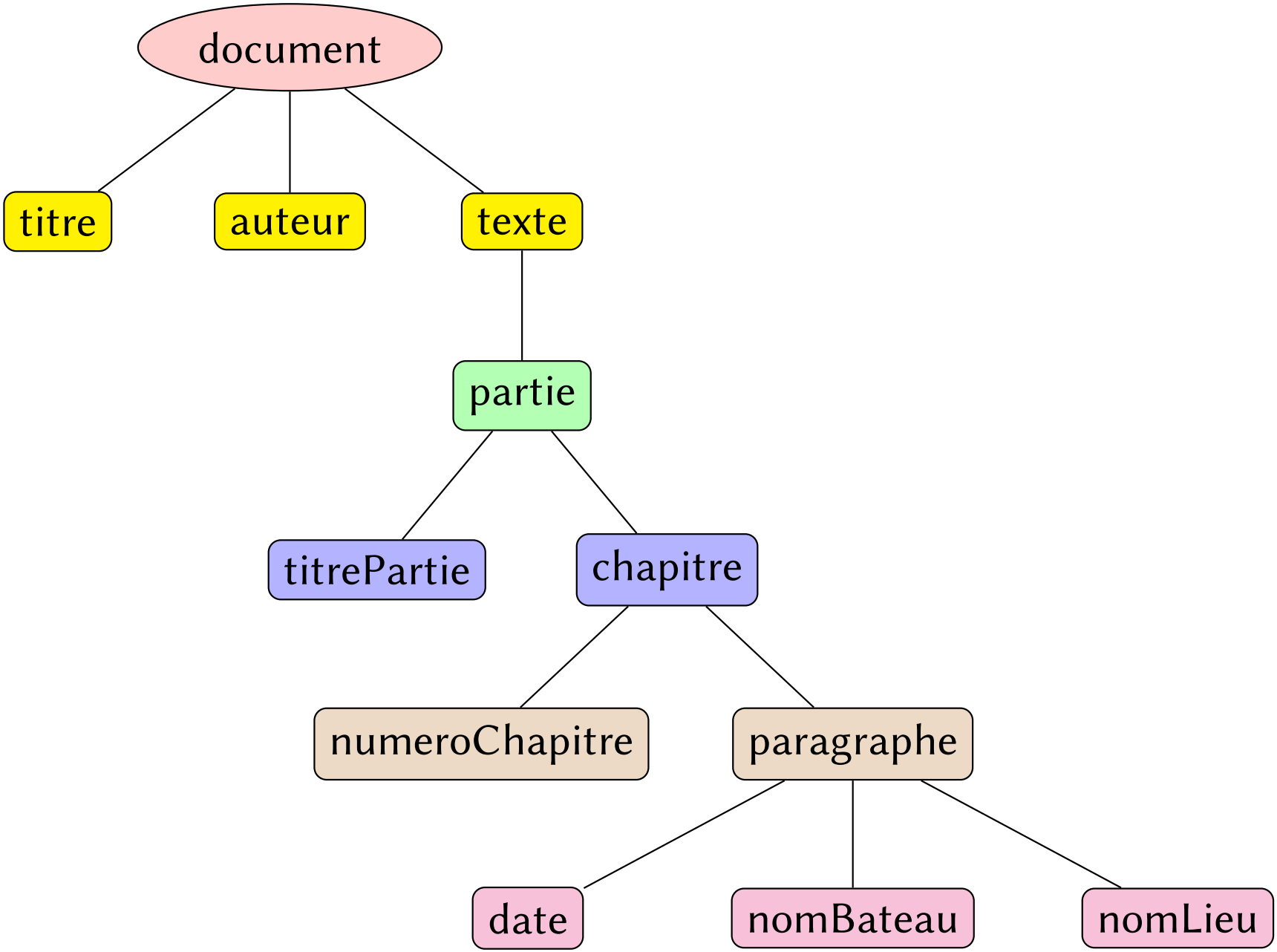
Les éléments enfants de l'élément racine

```
<document>
<titre>...</titre>
<auteur>...</auteur>
<texte>...</texte>
</document>
```

...Qui eux-mêmes peuvent avoir des éléments enfants

```
<texte>
<partie>
<chapitre>
<paragraphe>...</paragraphe>
</chapitre>
</partie>
</texte>
```

L'arborescence



Récapitulatif

Le langage XML **est** :

- un langage à **balises**
- un langage à **structuration arborescente**, qui fonctionne avec des nœuds parents et enfants
- un **métalangage**

Récapitulatif

Le langage XML **est** :

- un langage à **balises**
- un langage à **structuration arborescente**, qui fonctionne avec des nœuds parents et enfants
- un **métalangage**

Le langage XML **doit** :

- avoir une **balise racine**
- correspondre au principe de **conformité** (pas de chevauchement)

Créer ses premiers documents XML avec Oxygen

Quelques règles sur le nom des éléments

- **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`

Quelques règles sur le nom des éléments

- **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs

Quelques règles sur le nom des éléments

- **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs
- les **majuscules** sont utilisées pour **distinguer deux mots** dans un élément : `<metadonneesDoc>`

Quelques règles sur le nom des éléments

- **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs
- les **majuscules** sont utilisées pour **distinguer deux mots** dans un élément : `<metadonneesDoc>`
- les majuscules **ne doivent pas commencer** un élément : `<MetadonneesDoc>`

Quelques règles sur le nom des éléments

- **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs
- les **majuscules** sont utilisées pour **distinguer deux mots** dans un élément : `<metadonneesDoc>`
- les majuscules **ne doivent pas commencer** un élément : `<MetadonneesDoc>`
- Les noms d'éléments ne doivent pas non plus commencer par un chiffre.

Exercice : encoder un texte court

1. Encodez le poème *L'Adieu* d'Apollinaire que vous trouverez ici : https://gallica.bnf.fr/ark:/12148/bpt6k1083760/f70.item (le texte est page 70).
2. Tracez l'arborescence de votre document XML

Exercice : encoder un texte plus long

1. Encodez la première page des *Souvenirs d'histoire locale* sur Charles-Louis-Jospeh Destable, par Charles Cerf, que vous trouverez sur Gallica : `https://gallica.bnf.fr/ark:/12148/bpt6k96742375`
   Cliquez sur « version texte (OCR) » pour obtenir le texte brut
2. Tracez l'arborescence de votre document XML
3. Observez comment oXygen valide ou invalide un document

Exercice : encoder un texte plus long

- Ne pas oublier le prologue XML
- Encoder soigneusement les métadonnées : celles du document encodé, et celles du document XML lui-même
- Encoder la structure du texte
- Encoder les particularités du document et tout élément qui paraîtra pertinent
- Indenter le fichier pour qu'il soit plus lisible

Pour cela :

- Réfléchir en amont à vos choix d'encodage
- Partir de la structure globale pour aller ensuite vers une structuration plus fine
- Trouver des noms d'éléments parlant
- Utiliser des attributs et des valeurs d'attributs