XML

Jean-Baptiste Camps & Simon Gabay

Définition

XML

Définition

XML= e**X**tensible **M**arkup **L**anguage

Définition

XML= e**X**tensible **M**arkup **L**anguage

Language

- Langage…

Language

- Langage…

- de programmation (C, Java, Python…)

  →créer des programmes, des applications, des systèmes d'exploitation

Language

- Langage…

- de programmation (C, Java, Python…)

  →créer des programmes, des applications, des systèmes d'exploitation

- de requêtes (SQL, XQuery…)

  →interroger des données

Language

- Langage…

- de programmation (C, Java, Python…)

  →créer des programmes, des applications, des systèmes d'exploitation

- de requêtes (SQL, XQuery…)

  →interroger des données

- de description (SGML, HTML, XML)

  →décrire et interroger des données

Language

- Langage de description

Language

- Langage de description
- Recommendation W3C

Language

- Langage de description
- Recommendation W3C

- W3C=World Wide Web Consortium

Language

- Langage de description
- Recommendation W3C

- W3C=World Wide Web Consortium
- Langage standardisé

Language

- Langage de description
- Recommendation W3C

- W3C=World Wide Web Consortium
- Langage standardisé
- Depuis 1998

Language

- Langage de description
- Recommendation W3C
- Ouvert
- Non-propriétaire

Language

- Langage de description
- Recommendation W3C
- Ouvert

- Non-propriétaire
- Indépendant

Language

- Langage de description
- Recommendation W3C
- Ouvert

- Non-propriétaire
- Indépendant

  →Stable

Language

- Langage de description
- Recommendation W3C
- Ouvert

- Non-propriétaire
- Indépendant

  →Stable

  →Interopérable

Markup Language

Markup Language

- Format de donnés

Markup Language

- Format de donnés

- Liste de personnes

Dupond
    Paul

Henchoz
    Jeanne

Lambert
    Paul-Henri

Markup Language

- Format de donnés

- Liste de personnes

Dupond
    Paul

Henchoz
    Jeanne

Lambert
    Paul-Henri

- Liste de personnes en JSON

```
{"personnes":[
  { "Nom":"Dupond", "Prenom":"Paul" },
  { "Nom":"Henchoz", "Prenom":"Jeanne" },
  { "Nom":"Lambert", "Prenom":"Paul-Henri" }
]}
```

Markup Language

- Format de donnés
- Balisage (*markup*)

- Liste de personnes

Dupond
Paul

Henchoz
Jeanne

Lambert
Paul-Henri

- Liste de personnes en JSON

```
{"personnes":[
  { "Nom":"Dupond", "Prenom":"Paul" },
  { "Nom":"Henchoz", "Prenom":"Jeanne" },
  { "Nom":"Lambert", "Prenom":"Paul-Henri" }
]}
```

Markup Language

- Format de donnés
- Balisage (*markup*)

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

- Liste de personnes en JSON

```
{"personnes":[
  { "Nom":"Dupond", "Prenom":"Paul" },
  { "Nom":"Henchoz", "Prenom":"Jeanne" },
  { "Nom":"Lambert", "Prenom":"Paul-Henri" }
]}
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

- Base de données vs narration

J'ai vu hier <personne> <prenom>Paul</prenom> <nom>Dupond</nom> </personne>dans une rue de <ville>Genève</ville>

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

- Base de données vs narration

J'ai vu hier <personne> <prenom>Paul</prenom> <nom>Dupond</nom> </personne>dans une rue de <ville>Genève</ville>

→compréhensible par l'homme comme la machine

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Trop ancien (1998) ?

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Trop ancien (1998) ?
- Comment encoder ce texte…

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- … en JSON ?

```
{"personnes":[
  { "Nom":"Dupond", "Prenom":"Paul" },
  { "Nom":"Henchoz", "Prenom":"Jeanne" },
  { "Nom":"Lambert", "Prenom":"Paul-Henri" }
]}
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données

- Un texte quelconque

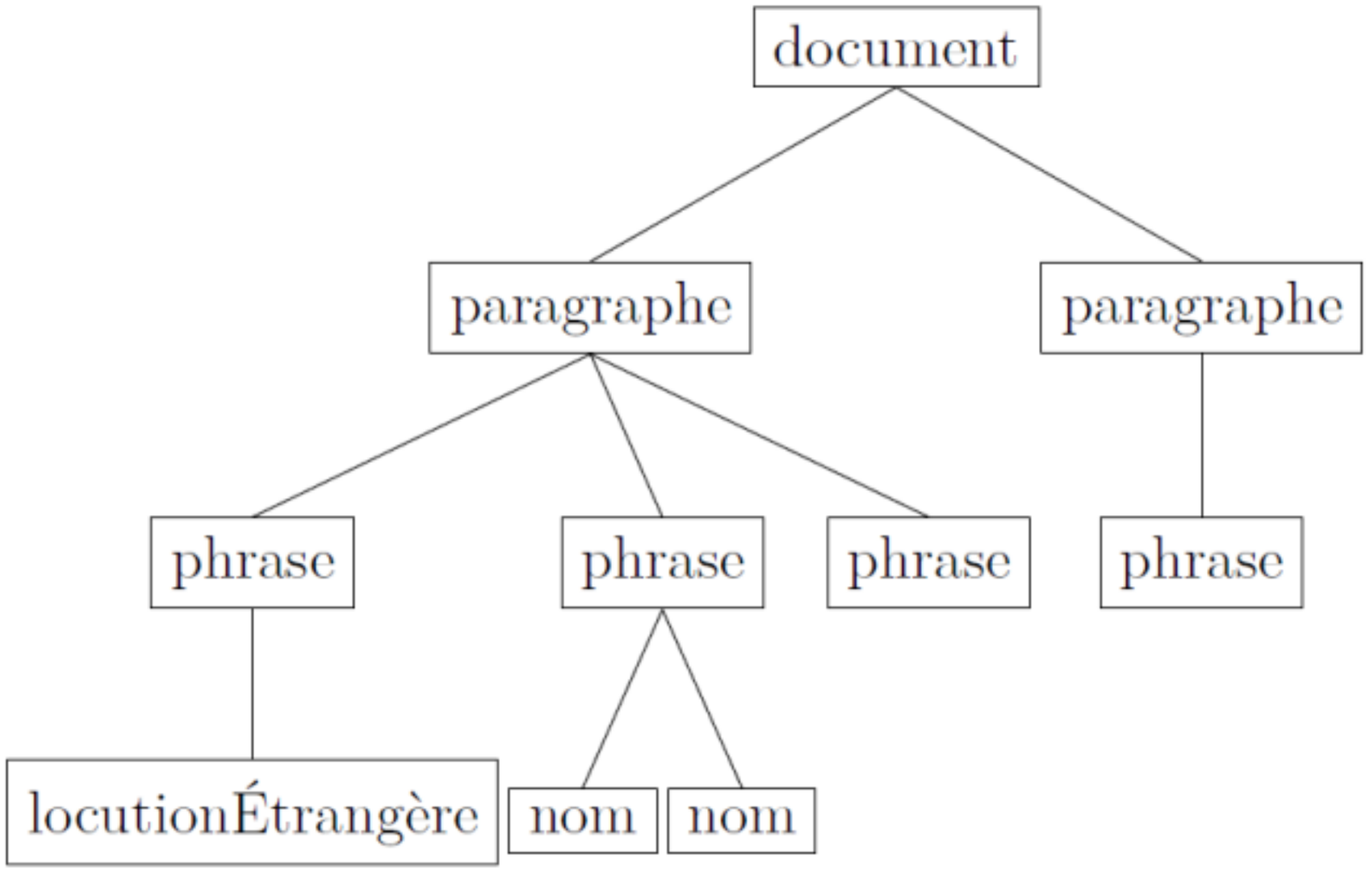
On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- En XML

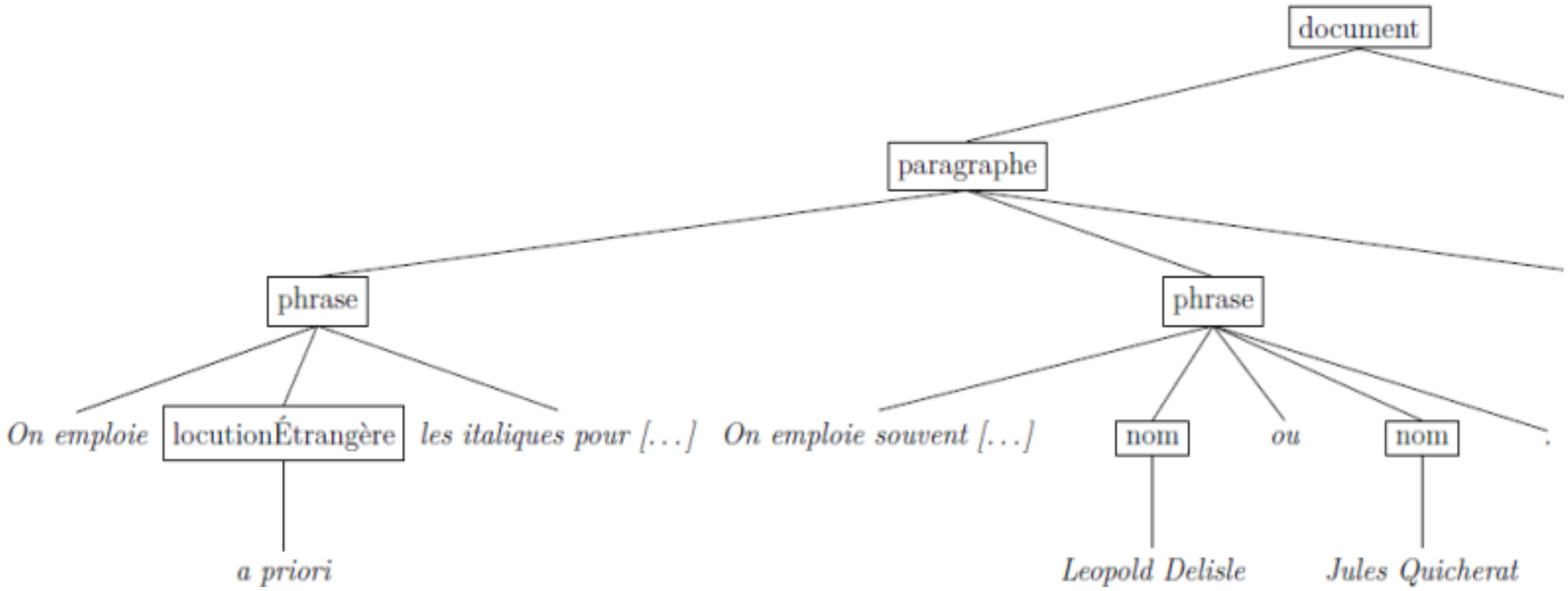
```
<document>
  <paragraphe>
    <phrase>On emploie<locutionEtrangère>a priori</locutionEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</phrase>
    <phrase>On emploie souvent les petites capitales pour les noms propres, comme <nom>Léopold Delisle</nom>ou <nom>Jules Quicherat</nom>.</phrase>
    <phrase>On emploie en revanche généralement le gras pour des raisons coupables.</phrase>
  </paragraphe>
</document>
```

→Structure arborescente composée de nœuds avec un nœud racine

Markup Language



Markup Language



Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données

- Un texte quelconque

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- En XML

```
<document>
  <paragraphe>
    <phrase>On emploie<locutionEtrangère>a priori</locutionEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</phrase>
    <phrase>On emploie souvent les petites capitales pour les noms propres, comme <nom>Léopold Delisle</nom>ou <nom>Jules Quicherat</nom>.</phrase>
    <phrase>On emploie en revanche généralement le gras pour des raisons coupables.</phrase>
  </paragraphe>
</document>
```

→Utiliser le texte encodé comme base de données

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Structure des données
- Type d'encodage

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (présentationnel)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 2

```
<p>On emploie<locEtrangère>a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 2 (descriptif)

```
<p>On emploie<locEtrangère>a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?
- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 2 (sémantique)

```
<p>On emploie<locEtrangère>a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 2 (sémantique)

```
<p>On emploie<locEtrangère>a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 3

```
<p>On emploie<locEtrangère langue="latin">a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 2 (sémantique)

```
<p>On emploie<locEtrangère>a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 3 (sémantique)

```
<p>On emploie<locEtrangère langue="latin">a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Structure des données
- Type d'encodage

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1 (procédural)

```
<p>On emploie<italique>a priori</italique>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 2 (sémantique)

```
<p>On emploie<locEtrangère>a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

- Solution 3 (sémantique)

```
<p>On emploie<locEtrangère langue="latin">a priori</locEtrangère>les italiques pour les locutions et termes empruntés à d'autres langues.</p>
```

→Séparer le fond de la forme

Markup Language

- Format de donnés
- Balisage (*markup*)
- Pour différents types de documents
- Un langage dépassé ?

- Structure des données
- Type d'encodage
- Granularité

- Comment encoder ?

On emploie a priori les italiques pour les locutions et termes empruntés à d'autres langues. On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat. On emploie en revanche généralement le gras pour des raisons coupables.

- Solution 1

On emploie souvent les petites capitales pour les noms propres, comme Léopold Delisle ou Jules Quicherat.

- Solution 2 (sémantique)

On emploie souvent les petites capitales pour les noms propres, comme <personne>Léopold Delisle</personne> ou <personne>Jules Quicherat</personne>.

- Solution 3 (sémantique)

On emploie souvent les petites capitales pour les noms propres, comme <personne><prenom>Léopold</prenom><nom>Delisle</nom></personne> ou <personne><prenom>Jules </prenom><nom>Quicherat</nom></personne>.

eXtensible Markup Language

- Extensible

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

=

```
<persons>
  <person>
    <name>Dupond</name>
    <firstName>Paul</firstName>
  </person>
  <person>
    <name>Henchoz</name>
    <firstName>Jeanne</firstName>
  </person>
</persons>
```

eXtensible Markup Language

- Extensible

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

=

```
<persone>
  <persona>
    <nome>Dupond</nome>
    <cognome>Paul</cognome>
  </persona>
  <persona>
    <nome>Henchoz</nome>
    <cognome>Jeanne</cognome>
  </persona>
</persone>
```

eXtensible Markup Language

- Extensible

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

=

```
<1>
  <A>
    <α>Dupond</α>
    <β>Paul</β>
  </A>
  <A>
    <α>Henchoz</α>
    <β>Jeanne</β>
  </A>
</1>
```

eXtensible Markup Language

- Extensible

- Liste de personnes en XML

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

=

```
<listPerson>
  <person>
    <surname>Dupond</surname>
    <forename>Paul</forename>
  </person>
  <person>
    <surname>Henchoz</surname>
    <forename>Jeanne</forename>
  </person>
</listPerson>
```

eXtensible Markup Language

- Extensible
- Métalanguage

eXtensible Markup Language

- Extensible
- Métalanguage
- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
  - Mon langage

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)

  - Mon langage

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
</personnes>
```

  - TEI

```
<listPerson>
  <person n="1">
    <surname>Dupond</surname>
    <forename>Paul</forename>
  </person>
</listPerson>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)

  - Mon langage

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
</personnes>
```

  - TEI

```
<listPerson>
  <person n="1">
    <surname>Dupond</surname>
    <forename>Paul</forename>
  </person>
</listPerson>
```

→Pas de jeu de balises prédéfini

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
  - Bien formé

```
<personnes>
  [personne n="1"]
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
- Bien formé

```
<personnes>
  <personne n="1">
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
  - Bien formé

```
<personnes>
  [personne n="1"]
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
  - Bien formé

```
<personnes>
  [personne n="1"
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
  - Bien formé

```
<personnes>
  [personne n="1"]
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
  - Bien formé et valide

```
<personnes>
  [personne n="1"]
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

eXtensible Markup Language

- Extensible
- Métalanguage

- Permet de définir différents langages avec chacun leur grammaire propre (EAD, XHTML, TEI…)
- C'est un ensemble d'éléments et de règles sur ce que doit être un document
  - Bien formé et valide

```
<personnes>
  [personne n="1"]
    <nom>Dupond</surname>
    <prenom>Paul</prenom>
  </personne>
  <personne n=2>
    <nom>Henchoz</nom>
  </personne>
  <personne n="3">
    <nom>Lambert</nom>
    <prenom>Paul</prenom>-Henri
  </personne>
</personnes>
```

Un document XML

- Composants

    1. Éléments

- Un <element> porte un nom (ou *gi* pour identification générique)

```
<element>
<personne>
<nom>
<paragraphe>
```

- Il contient une balise d'ouverture ET de fermeture

<nom>Victor Hugo</nom> ✓
<nom>Victor Hugo ✗

- Il peut être vide (autofermant)

<blanc/>=<blanc></blanc> ✓

- Il peut contenir d'autres éléments

```
<auteur>
  <prenom>Victor </prenom>
  <nom>Hugo</nom>
</auteur>
```

Un document XML

- Composants
    1. Éléments
    2. Attributs

- Un élément peut porter un @attribut avec un nom et une "valeur"

```
<nom type="auteur">Victor Hugo</nom> ✓
<nom type='auteur'>Victor Hugo</nom> ✓
<nom type=auteur>Victor Hugo</nom> ✗
```

- La "valeur" peut contenir des lettres et des chiffres

```
<nom naissance="1802-02-26">Victor Hugo</nom> ✓
```

- Un élément vide peut porter un attribut

```
<blanc raison="pageTrouée"/> ✓
```

Un document XML

- Composants
    1. Éléments
    2. Attributs
    3. Entités de caractères

- Caractères sensibles au balisage
- Supérieur à (<) et aussi le chevron d'un <element>
    - 1 < 2
- Esperluette (&)
    - Laurel & Hardy
- Unicode
    - 1 <
    - Laurel & Hardy

Un document XML

- Composants
    1. Éléments
    2. Attributs
    3. Entités de caractères
    4. CDATA (Character DATA)

- Contenu entre <![CDATA[et]]>
- Les règles de XML ne s'y appliquent pas

```
<![CDATA[
   #Ici, je peux mettre toutes les éperluettes que je veux et
tous les chevrons, youpi    # &&&&&& <<<<<<<
   # Je peux par exemple intégrer un script en Perl
   while (1){
      print "Devinez le nombre compris entre 1 et 100 :\t ";
      my $tent = <STDIN>;
      if ($nombreADeviner == $tent){
            print "Bravo !\n";
            last
      }
       elsif ( ($tent == ' ') || ($tent =~ m/quitter|sortir/ )){
            print "Au revoir !\n";
            last
       }
      elsif ($tent < $nombreADeviner){
            print "Plus haut !\n"
       }
      elsif ($tent > $nombreADeviner){
            print "Plus bas !\n"
       }
}
   ]]>
```

Un document XML

- Composants
  1. Éléments
  2. Attributs
  3. Entités de caractères
  4. CDATA (Character DATA)
  5. Instructions de traitement

- Contenues entre <? et ?>
- Permettent de faire appel à des applications extérieures
  - Pour mettre en page

```
<?xml-stylesheet type="text/css" href="X.css"?>
```

  - Pour renvoyer aux règles du doc

```
<?xml-model href="ODD/out/Modele.rng"
type="application/xml"schematypens="http://
relaxng.org/ns/structure/1.0"?>
```

- Utilisé pour la déclaration XML en tête de document

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

Un document XML

- Composants
  1. Éléments
  2. Attributs
  3. Entités de caractères
  4. CDATA (Character DATA)
  5. Instructions de traitement
  6. Commentaire

- Contenu entre <!-- et -->
- Permet de laisser des notes à l'intérieur du code

```
<personnes>
  <personne>
    <nom>Dupond</nom>
    <prenom>Paul</prenom>
  </personne>
  <personne>
    <nom>Henchoz</nom>
    <prenom>Jeanne</prenom>
  </personne>
  <personne>
    <nom>Lambert</nom>
    <prenom>Paul-Henri</prenom>
  </personne>
  <!-- ajouter Michel Legrand -->
</personnes>
```

Un document XML

- Composants
- Règles
    1. Composition des noms

- XML est sensible à la casse
- Noms peuvent être composés de caractères alphanumériques, des soulignements (_), traits d'unions (-), et points (.)

<balise_balise> ✓
<balise-balise> ✓
<balise.balise> ✓
<balise.balise-balise_123> ✓

- Ils ne doivent pas débuter par un nombre, un trait d'union…

<1balise> ✗ <.balise> ✗
<!balise> ✗ <?balise> ✗
<balise/balise> ✗ <balise,balise> ✗
<…balise> ✗

- Ne pas finir par des points d'interrogation ou d'exclamation

<balise!> ✗
<balise?> ✗

Un document XML

- Composants
- Règles
    1. Composition des noms
    2. Utilisation des attributs

- La valeur de l'attribut est entre guillemets simples ('attribut') ou doubles ("attribut")

```
<nom type="auteur">Victor Hugo</nom> ✓
<nom type='auteur'>Victor Hugo</nom> ✓
<nom type=auteur>Victor Hugo</nom> ✗
```

- Il ne peut y avoir deux fois le même attribut pour le même élément

```
<nom type="eleve" type="homme">Max</nom> ✗
<nom metier="eleve" genre="homme">Max</nom> ✓
```

Un document XML

- Composants
- Règles
  1. Composition des noms
  2. Utilisation des attributs
  3. Pas d'entrecroisement

- Imbrication mais pas entrecroisement

```
<auteur><nom>Victor Hugo</nom></auteur> ✓
<auteur><nom>Victor Hugo</auteur></nom> ✗
```

- Cas problématique

```
<poeme>
  <vers>Je parle français <english>but also English</vers>
  <vers>in this poem</english>de deux vers</vers>
</poeme>
```

- Exercice

```
1. <exp>du texte</exp>
2. <exp><seg1>du</seg1><seg2>texte</seg2></exp>
3. <exp><seg1>du<seg2></seg1>texte</seg2></exp>
4. <exp type='text'>du texte</exp>
5. <exp type="text">du texte</exp>
6. <exp type=text>du texte</exp>
7. <exp type = "text">du texte</exp>
8. <exp type="text">du texte<exp/>
9. <exp type="text">du texte<blanc/></exp>
10. <exp type="text">du texte< /exp>
11. <exp type="text">du texte</Exp>
```

Un document XML

- Composants
- Règles
    1. Composition des noms
    2. Utilisation des attributs
    3. Pas d'entrecroisement
    4. Place des commentaires

- Pas de commentaire à l'intérieur d'une balise

```
<auteur>
  <prenom>Victor </prenom>
  <nom>Hugo</nom>
  <!-- commentaire --> ✓
</auteur>

<auteur>
  <prenom>Victor </prenom>
  <nom>Hugo</nom <!-- commentaire -->> ✗
</auteur>
```

Un document XML

- Composants
- Règles
    1. Composition des noms
    2. Utilisation des attributs
    3. Pas d'entrecroisement
    4. Place des commentaires
    5. Fermeture des balises

- Une balise ouverte doit être fermée

<nom>Victor Hugo</nom> ✓
<nom>Victor Hugo ✗

- Cas particulier : les balises autofermantes

<blanc/>=<blanc></blanc> ✓

→Nœud vide

Un document XML

- Composants
- Règles
  1. Composition des noms
  2. Utilisation des attributs
  3. Pas d'entrecroisement
  4. Place des commentaires
  5. Fermeture des balises
  6. Caractères sensibles au balisage
     - < devient <
       - 1 < 2
     - & devient &
       - Laurel & Hardy

Un document XML

- Composants
- Règles
    1. Composition des noms
    2. Utilisation des attributs
    3. Pas d'entrecroisement
    4. Place des commentaires
    5. Fermeture des balises
    6. Caractères sensibles au balisage
    7. Autre

- Une seule racine contient tout le document
- Les noms sont liés avec un namespace (espace de nom) pour labelliser le vocabulaire duquel dérive un ensemble d'éléments et d'attributs

Un document XML

- Composants
- Règles

→Le document est alors bien formé, mais est-il valide ?

- Un document XML doit respecter…
  - … des règles qui nous sont imposées…

    →Le document est alors bien formé
  - … des règles que nous nous imposons en plus

    →Le document est alors valide
    - Ces règles supplémentaires constituent une grammaire

Un document XML

- Composants
- Règles

→Le document est alors bien formé, mais est-il valide ?

- Un schéma permet de spécifier

1. Le nom des éléments
2. Les noms, types et valeurs par défaut des attributs
3. Le règles d'imbrication des éléments
4. Les règles concernant le contenu des éléments

Un document XML

- Composants
- Règles

→Le document est alors bien formé, mais est-il valide ?

- Un schéma permet de spécifier

- Exemple

  <personne> doit obligatoirement contenir un <prenom> et un <nom> dans cet ordre. <prenom> contient un attribut @genre de valeur "f" ou "m".

```
<personne>
  <prenom genre="m">Victor </prenom>
  <nom>Hugo</nom>
</personne> ✓

<personne>
  <prenom genre="homme">Victor </prenom>
  <nom>Hugo</nom>
</personne> ✗

<personne>
  <nom>Hugo</nom>
  <prenom genre="m">Victor </prenom>
</personne> ✗

<personne>
  <nom>Hugo</nom>
</personne> ✗
```