Langages pour les documents Web

1. Rappels HTML (HyperText Markup Language)
2. DTD (Document Type Definition)
3. CSS (Cascading Style Sheets)
4. XML (eXtensible Markup Language)

Adaptations aux documents Web de
SGML (Standard Generalized Markup Language) — 1986
= marquage de documents permettant d'enrichir le contenu
en *structure, style, sémantique*.

1. Rappels HTML

- Conçu en même temps que HTTP par Tim Berners-Lee au CERN en 1990
- Rendu populaire en 1993 par Mosaic (NCSA)
- Evolution :
  - Novembre 1995 : HTML 2.0 défini par IETF (ISOC)
  - Janvier 1997 : HTML 3.2 défini par W3C
  - Décembre 1997 : HTML 4.0 défini par W3C
    - internationalisation (caractères, langue)
    - accessibilité (agents non graphiques)
    - feuilles de style (séparation structure/présentation)
    - scripting (documents dynamiques/interactifs)
    - nouveau modèle de tableaux
    - documents composés (frames)
- Référence : http://www.w3.org/

Exemple de document HTML

```
<!DOCTYPE HTML PUBLIC
 "-//W3C/DTD HTML 4.0//EN"
 "http://www.w3.org/TR/REC-html40/strict.dtd">

<HTML>
 <HEAD>
  <!-- Titre de mon document -->
  <TITLE>Exemple de document HTML</TITLE>
  <META NAME="Autheur" CONTENT="MS">
 </HEAD>

 <BODY BGCOLOR="white" LINK="blue" VLINK="maroon">
  <IMG SRC="logo.gif" ALT="Logo">
  <BR>
  <H1> Bonjour&nspb;! </H1>
 </BODY>
</HTML>
```


Concepts de base

- Eléments :
  - Définissent la structure (en-tête, paragraphe, ...).
  - Définis par : nom, étiq. début, contenu, étiq. fin.

    `<BODY> Bonjour </BODY>`
  - Les étiquettes peuvent être optionnelles (p.e. P).
  - Si contenu vide, pas d'étiquette de fin (p.e. IMG).
  - Le nom est “case-insensitive”.
- Attributs :
  - Propriétés associées aux éléments.
  - Définis par : nom, type de la valeur, valeur par défaut.

    `COLOR="white" SIZE=+1`
  - Le nom est “case-insensitive”.
  - La valeur est “case-insensitive” ou “case-sensitive”.
- *La structure du documents est un arbre !*

  `<I> Ceci est <B>incorrect</I> ! <B>`

Définition du HTML formalisée par une DTD.

2. Document Type Définition (DTD)

= méthode de définition pour tous les langages à marquages (“markup languages”).

Il spécifie :

- le modèle de chaque élément
- ses attributs
- l’ordre et la quantité des sous-éléments

Document DTD = ensemble de déclarations de la forme :

```
<!KEYWORD NAME_PARAM PARAM1 PARAM2 ...>
```

où

- `KEYWORD` spécifie la nature de la définition, p.e. : `DOCTYPE`, `ENTITY`, `ELEMENT`, `ATTLIST`, ...
- `NAME_PARAM` est un nom (d’élément ou de macro)
- `PARAM1`, `PARAM2`, ... sont les valeurs associées à l’élément défini

Mots clé principaux (1/3)

`DOCTYPE` spécifie la DTD à utiliser pour un document écrit dans le langage $\mathcal{L}$.

- est normalement mis au début du document
- le nom indique l’élément racine du document
- Exemple pour HTML :

```
<!DOCTYPE HTML PUBLIC
   "-//W3C/DTD HTML 4.0//EN"
   "http://www.w3.org/TR/REC-html40/strict.dtd">
```

`ENTITY` spécifie une macro-définition (au sens C).

- deux sortes : “générale” (utilisée préfixée par &) ou “paramètre”(utilisée préfixée par %)
- Exemple pour HTML :

```
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
<!ENTITY % coreattrs
 "id          ID              #IMPLIED
  class       CDATA           #IMPLIED
  style       %StyleSheet;    #IMPLIED
  title       %Text;          #IMPLIED" >
```

Mots clé principaux (2/3)

`ELEMENT` définit un élément du langage $\mathcal{L}$ avec ses options et ses contraintes.

- Comporte trois parties :
  - le nom de l'élément
  - la spécification des étiquettes de début et fin (− obligatoire, O optionel)
  - le contenu de l'élément = expr. régulière formée avec les caractères suivants :
    - `EMPTY` (vide)
    - `?` (optionel), − (sauf ...)
    - `*` et + (répéter ...)
    - `,` (séquence) et `&` (ensemble)
    - `|` (choix)
- Exemple pour HTML :

```
<!ELEMENT BODY O O (%flow;)* --definition list-->
<!ELEMENT UL   - - (LI)+>
<!ELEMENT IMG  - O EMPTY>
<!ELEMENT DL   - - (DT|DD)+ --definition list-->
```


Mots clé principaux (3/3)

`ATTLIST` définit la liste d'attributs possibles pour un élément du langage $\mathcal{L}$.

```
<!ATTLIST NomElement
   NomAttr1 TypeAttr1 ValDefaut1
   ...
   NomAttrN TypeAttrN ValDefautN>
```

- `TypeAttr` `CDATA`, `ID`, type énuméré

```
<!ATTLIST HR
   ...
   id    ID                    #IMPLIED
   class CDATA                 #IMPLIED
   align (left|center|right)   #IMPLIED
   size  %Pixels;              #IMPLIED
 >
```

- `ValDefaut` spécifie une valeur pour l'attribut si elle n'est pas définie explicitement.Types possibles :
  - `CDATA` une chaîne de caractères
  - `#FIXED` fixe une valeur
  - `#IMPLIED` valeur optionnelle
  - `#REQUIRED` valeur obligatoire

Exemple DTD

Extrait de la définition de HTML 4.0

```
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
<!ENTITY % coreattrs
 "id          ID              #IMPLIED
  class       CDATA           #IMPLIED
  style       %StyleSheet;    #IMPLIED
  title       %Text;          #IMPLIED" >
<!ELEMENT UL - - (LI)+>
<!ATTLIST UL %attrs; >
<!ELEMENT IMG - O EMPTY>
<!ATTLIST IMG
  %attrs;
  src         %URI;           #REQUIRED
  alt         %Text;          #REQUIRED
  longdesc    %URI;           #IMPLIED
  name        CDATA           #IMPLIED
  height      %Length;        #IMPLIED
  width       %Length;        #IMPLIED
  usemap      %URI;           #IMPLIED
  ismap       (ismap)         #IMPLIED  >
```


3. Feuilles de style (CSS)

HTML permet de structurer un document sans se soucier de la forme qu'il prendra (sa présentation).

Préciser les effets de présentation = ajouter un style :

- A l'aide d'un fichier externe, dans l'en-tête

  ```
  <LINK REL="stylesheet" TYPE="text/css"
        HREF="mstyle.css">
  ```

- En spécifiant directement le style dans l'en-tête :

  ```
  <STYLE TYPE="text/css">...
  </STYLE>
  ```

  ou pour être indépendant de langage :

  ```
  <META HTTP-EQUIV="Content-Style-Type"
        CONTENT="text/css"> ...
  </META>
  ```

- En utilisant les attributs de style de HTML :

  ```
  <H1 COLOR="red"><FONT SIZE=+2>Titre</FONT></H1>
  <P STYLE="color:red">Paragraphe</P>
  <P CLASS="important">Paragraphe important</P>
  <P ID="c01">Un paragraphe identifié</P>
  ```

CSS

- = langage pour la spécification de style pour les documents HTML
- Permet une séparation claire de la structure et du style, donc une plus grande indépendance du médium (graphique, ascii) utilisé.
- “CSS level 2” est une recommandation W3C (mai 1998)
  - spécification http://www.w3.org/TR/REC-CSS2
  - validation http://www.w3.org/css-validator/
- Nombreux navigateurs n’interprètent pas correctement les feuilles de style CSS :-(
- Mais “CSS level 3” en préparation ! http:/www.w3.org/Style
- Règle de style en deux parties :
  - un sélecteur : lien entre HTML et CSS
  - une déclaration : liste de paires propriété, valeur

  Exemple :

```
H1 { color: red; }
```


Commentaires

Identiques au C.

Cascade de styles

= combinaison de feuilles de style pour faciliter leur partage.

```
@import url(http://www.style.org/marine);
H1 { color: red; } /* surcharge du style importé */
```

Priorité : utilisateur < auteur. On peut modifier la priorité avec le mot clé `important`.

```
H1 { color: red ! important; }
```

Propriétés

- Police de caractères : font-family, font-style, font-size,...
- Couleurs et fonds : color, background,...
- Propriétés du texte : word-spacing, text-align,...
- Propriétés des boîtes : margin, padding, border, width,...
- Catégories : display, list-style, white-space,...

Unités de mesure

- Longueurs : relatives (em, ex, px), absolues (in=2,54cm, cm, mm, pt=1/72in, pc=12pt)
- Pourcentages : 120%
- Couleurs : red, #FF0000, rgb(255,0,0)

Groupements

- De sélecteurs :

```
H1, H2, H3 { font-family: helvetica; }
```

- De déclarations :

```
H1 {    font-weight: bold;
        font-size: 12pt;
        line-height: 14pt;
        font-family: helvetica;
        font-variant: normal;       }
```

- De valeurs :

```
H1 { font: bold 12pt/14pt helvetica; }
```

Héritage

- En général, le style est hérité du père.
  Exemple : EM est en bleu dans

```
<H1 STYLE="color: blue">Titre
   <EM>important</EM></H1>
```

- Style par défaut donné par BODY

```
body { color: black;
       background: url(texture.gif) white; }
```


Sélecteurs

Permettent de nuancer les styles d'un élément.

- Sélecteur de classe (CLASS)

```
H1.important { color: #FF0000; }
<H1 CLASS="important">Attention !</A1>
```

  Plusieurs éléments peuvent partager une classe pour laquelle sont définis les attributs :

```
.important { color: rgb(255, 0, 0); }
```

- Sélecteur par identificateur (ID) unique

```
#c01p { letter-spacing: 0.3em; }
<P ID="c01p">Paragraphe serré.</P>
```

  Equivalent à (car l'ID est unique) :

```
P#c01p { letter-spacing: 0.3em; }
```

- Sélecteur contextuel
  - contexte descendant :

```
H1 EM { color: red; }
```

  - contexte descendant direct (fils) :

```
H1 > EM { color: red; }
```

  - contexte multiple :

```
UL LI    { font-size: small; }
UL UL LI { font-size: x-small; }
```

  Si conflit, le sélecteur le plus spécifique l'importe.

4. XML

Inconvénients de HTML :

- Nombre d'attributs et d'éléments limité
- Extension ⇒ révision définition HTML
- Conversions difficiles vers HTML

Principe :

Java = "program once, run everywhere",
XML = "write once, publish everywhere"

XML = langage pour traiter les documents et les données structurés.

- Recommandation W3C (février 1998, octobre 2000) http://www.w3.org/TR/REC-xml
- XML comparé à SGML :
  - XML est un sous-ensemble de SGML, permet la distribution de documents SGML sur le Web.
  - Le parsing de SGML nécessite une DTD, mais pas le parsing d'XML où le DTD sert uniq. à la validation.
  - Les étiq. de fin sont obligatoires et les éléments sans contenu finissent par l'étiq. /


```
<defclass name="Car">
   <defattribute name="mark" type="string" />
   <defattribute name="age" type="integer" />
</defclass>
<defobject id="123">
   <class>Car</class>
   <attribute name="mark">Renault</attribute>
   <attribute name="age">1992</attribute>
</defobject>
```

- XML comparé à HTML :
  - XML de plus bas niveau (syntaxe abstraite).
  - HTML est une application (syntaxe concrète) SGML.
  - HTML a une sémantique, mais pas XML.
  - XML précise uniq. la structure, pas le style.
  - Amélioration du système de liens hypertexte.

```
<slide>
<title>Ceci peut être lu par tout parser XML</title>
<ul>
<li> XML de bas niveau </li>
<li> HTML une application </li>
</ul>
</slide>
```

Galaxie XML

- XML, XML namespace
- XML Schema : décrire la structure d'un document et le typage (remplace DTD)
- XPath, XPointer : langage de chemins pour naviguer dans les documents
- XLink : liens hypertexte
- RDF (Resource Description Framework) : meta-information pour annoter des documents
- RDF Schema : typage pour RDF

Traitement de XML

- DOM (Document Object Model) : API d'accès aux documents et aux données XML.
- XSL (Extensible Stylesheet Language) : feuilles de style
  - XSLT (XSL Transformation)
  - FO (Formatting Object)
- XML Query : langage d'interrogation pour XML

Avantages

- Réutilisation des informations.
- Publier indépendamment des médias.
- Imprimé et numérique à partir de la même source.
- Standard pour la représenter les données : on peut changer d'outils.
- Indépendant de formats propriétaires.
- XML+XSL peut remplacer formats de traitement de texte et logiciel de publication.
- Améliorer la recherche d'informations sur le Web :
  - recherche sur des balises et des DTD spécifiques
  - recherche sur le Web comme dans une BD.
- Une communauté peut définir une norme d'échange d'informations :
  - se mettre d'accord sur une DTD
  - échanger les données XML, sans la DTD

Type d'application XML

- Client Web médiateur de plusieurs sources hétérogènes.
- Décharger le serveur, charger le client d'une partie du traitement :
  - présentation de données
  - calculs, transformations
- Présenter différentes vues des mêmes données, sans recharger à chaque fois.
- Adapter le format de présentation à l'utilisateur.
- Définir d'autres langages de marquage :
  - XHTML
  - MathML
  - SMIL (Synchronized Multimedia Integration Language)
  - TIM (Telecom Interchange Markup)
  - CML (Chemical Markup Language)
  - ThML (Theological Markup Language)

Exemple document XML :

```
<?xml version="1.0"?>
<!DOCTYPE bibliography SYSTEM
          "http://somewhere.net/book.dtd">
<bibliography>
<book>
  <author>
    <firstname>Håkon Wium</firstname>
    <lastname>Lie</lastname>
  </author>
  <title>Cascading Style Sheets,
         designing for the Web</title>
  <publisher>Addison Wesley</publisher>
</book>
</bibliography>
```

Type DTD correspondant (book.dtd) :

```
<!ELEMENT bibliography (book*)>
<!ELEMENT book (author, title, publisher, year?)>
<!ELEMENT author (firstname, lastname)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
```

XML Schema respecte une syntaxe de type XML.