

Langages Web 1



Langages de Sérialisation

Florent Nicart

Université de Rouen

2016–2017

Problématique

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

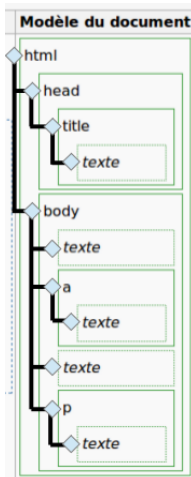
Interactions

Lang.&prot.
Système

- Un moyen STANDARD de structurer l'information ?
- Pour quoi faire ? Affichage, échange, stockage ?
- Contenu : du texte riche / multimédia / dynamique,
- représentant un document simple ou la vue dynamique d'une application.
- Comment représenter du contenu tel qu'il soit lisible partout ? (indépendamment de la plate-forme de lecture, i.e. terminal + logiciel).
- Web = information texte structurée
- Quel(s) modèle(s) utiliser pour structurer l'information ?

Un modèle arborescent

- Ces structures récursives peuvent être représentées mathématiquement par des arbres.



- l'inclusion des éléments est représentée par la relation parent–enfant dans l'arbre,
- les séquences (concaténation) sont représentées par les noeuds de même niveau (même parent),
- **Sérialisation** : produire une représentation linéaire dans un langage dit **descriptif**.
- **Désérialisation** : reconstruire la structure à partir d'un flux linéaire.

Les langages à balises

Un concept générique

Langages

Modèle

SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

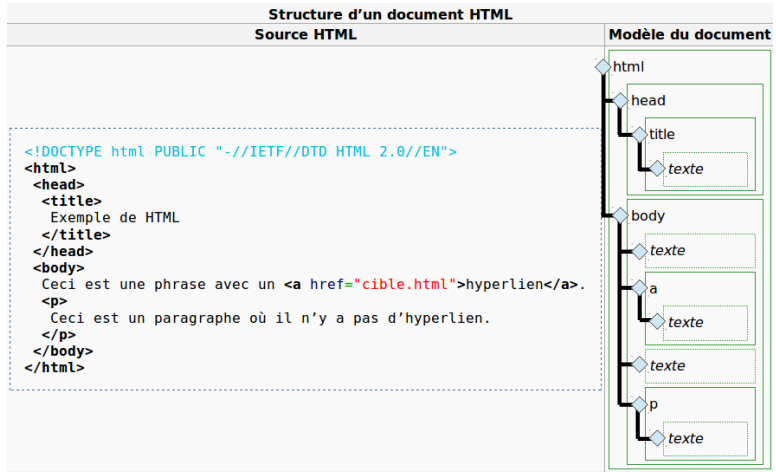
- Idée : utiliser des balises (*markups*) pour délimiter les éléments et leur contenu lors de la sérialisation,
- le langage descriptif sera formé de la collection des types de balises et de leurs attributs (html4, html5, svg, etc.),
- la famille des langages ***ML** (les Markup Languages ¹) partage une syntaxe commune pour décrire ces balises.

1. À ne pas confondre avec les méta-langages ML comme OCaML.

Les langages à balises

Exemple : HTML

Un arbre et sa sérialisation en HTML :



Les langages à balises

Un méta langage

La syntaxe² commune des langages à balises de la famille

★ML :

<élément [*attribut="valeur"*]* [*> | />*]

</élément>

- Tokens syntaxiques du méta-langage :

< > /> </ = " '

- On note : ces tokens appartient à la table ASCII.
- Éléments de l'instance applicative :

élément, attribut

- On note : pas de récursivité dans ce méta-langage.
⇒ un simple analyseur lexical suffit pour le décodage.

2. Les éléments en vert servent uniquement à décrire la syntaxe.

Les langages à balises

Un méta langage

La syntaxe² commune des langages à balises de la famille

★ML :

<élément [*attribut="valeur"*]* [*> | />*]

</élément>

- Tokens syntaxiques du méta-langage :

< > /> </ = " '

- On note : ces tokens appartient à la table ASCII.

- Éléments de l'instance applicative :

élément, attribut

- On note : pas de récursivité dans ce méta-langage.
⇒ un simple analyseur lexical suffit pour le décodage.

2. Les éléments en vert servent uniquement à décrire la syntaxe.

Les langages à balises

Un méta langage

La syntaxe² commune des langages à balises de la famille

★ML :

<élément [*attribut="valeur"*]* [*> | />*]

</élément>

- Tokens syntaxiques du méta-langage :

< > /> </ = " '

- On note : ces tokens appartient à la table ASCII.
- Éléments de l'instance applicative :

élément, attribut

- On note : pas de récursivité dans ce méta-langage.
⇒ un simple analyseur lexical suffit pour le décodage.

2. Les éléments en vert servent uniquement à décrire la syntaxe.

Les langages à balises

Un méta langage

La syntaxe² commune des langages à balises de la famille

★ML :

<élément [*attribut="valeur"*]* [*> | />*]

</élément>

- Tokens syntaxiques du méta-langage :

< > /> </ = " '

- On note : ces tokens appartient à la table ASCII.
- Éléments de l'instance applicative :

élément, attribut

- On note : pas de récursivité dans ce méta-langage.
⇒ un simple analyseur lexical suffit pour le décodage.

2. Les éléments en vert servent uniquement à décrire la syntaxe.

Les langages à balises

Langage applicatif

Langages

Modèle

SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- Cette syntaxe est insuffisante et sert à définir des langages applicatifs (ou DSL ³)
- Pour être réellement utile, nous avons besoin d'une définition récursive et donc, d'une grammaire.
- Pour les langages à balises, une grammaire consiste à :
 - donner la liste des éléments, des attributs et leurs types,
 - indiquer les règles de combinaison entre élément, ex :`<div>` accepté à l'intérieur de `p` mais pas `img`.

Grammaires hors-contexte

- Il existe une bijection naturelle entre *arbres* et *langages parenthésés*,
- En particulier les langages qui imposent que leurs instances soient *bien formées* correspondent aux langages de *Dyck*,

$$([()])() \rightarrow ([[]])() \rightarrow ([[]])() \rightarrow ()() \rightarrow () \rightarrow \varepsilon$$

- Contre exemple : ' ()) ('.
- Ces langages appartiennent à la famille des *grammaires hors-contexte* (reconnaissable par un automate à pile) : l'alphabet est constitué de symboles appairés deux-à-deux (les balises ouvrantes/fermantes).
- Nous allons maintenant discuter de ces langages⁴.

4. Nous verrons plus tard un formalisme objet pour manipuler l'arbre du document : *DOM*

Les origines : GML

- Le langage **GML** (*Generalized Markup Language*) a été créé en 1969 chez IBM dans le but de séparer la représentation des documents des caractéristiques des imprimantes.
- Sa normalisation ISO en 1986 donnera **SGML** (*Standard Generalized Markup Language*).
- *SGML est un méta-langage permettant de créer de nouveaux langages (à balises) : des instances de SGML.*
- *Ce sont des langages descriptifs (par opposition à un langage de programmation) : ils décrivent des états et des relations.*

Les principes de SGML

Langages

Modèle

SGML

HTML

XML

XHTML

HTML5

Syntaxe et DTD

ELEMENT

ATTLIST

ENTITY

Rendu

Modes de rendu

Enjeu

Interactions

Lang.&prot.

Système

SGML de séparer les différents aspects de la représentation d'un document :

- la grammaire du langage de structuration utilisé : la **DTD** (*Document Type Definition*),
- la mise en forme : les **feuilles de styles**,
- les **données** qui seront structurées conformément à la *DTD*.

Structuration / Présentation

Exemple de présentation :

blabla

```
<p><font ...>je suis un paragraphe de commentaire  
</font></p>
```

blabla

Exemple structuré (mais permettant exactement le même présentation) :

blabla

```
<p class="commentaire">je suis un paragraphe de  
commentaire.</p>
```

blabla

Structuration / Présentation

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

Parfois la différence peut être plus subtile.
Exemple de présentation :

Je `<i>voulais</i>` vous dire `quelquechose de
très important!`

Exemple structuré (mais permettant exactement le même
présentation) :

Je `voulais` vous dire `quelquechose
de très important!`

Avantages du découplage

Découpler la structuration et la présentation permet :

- de séparer les aspects métiers au niveau de la conception (developpeur/web Designer) ;
- d'améliorer la portabilité et l'accessibilité : la même source de données peut être rendue de différentes manières (écran, impression, liseuse pour non-voyant, etc.) ;
- une plus grande flexibilité : évolutions indépendantes des données et de la mise en forme (Écran, impression, affichage sur terminaux mobiles, import, etc.) ;

Mise en forme

Langages

Modèle

SGML

HTML

XML

XHTML

HTML5

Syntaxe et DTD

ELEMENT

ATTLIST

ENTITY

Rendu

Modes de rendu

Enjeu

Interactions

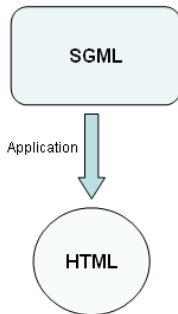
Lang.&prot.

Système

- Dans ce chapitre, nous nous consacrerons à l'aspect structuration de HTML et à sa syntaxe (DTD).
- Nous laisserons de côté les aspects de mise en forme qui seront traité au chapitre suivant avec les feuilles de styles : *CSS* (Cascading Style Sheets).

Naissance de HTML


- **SMGL** sera largement utilisé par IBM, Airbus et la SNECMA, des arsenaux militaires, ... et le CERN.



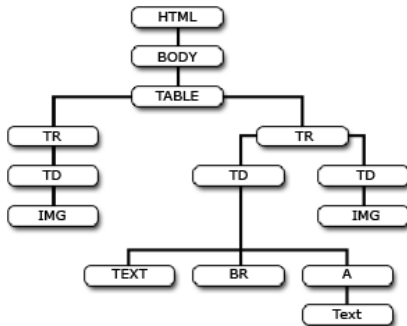
- C'est donc tout naturellement que Tim Berner-Lee conçoit **HTML** (*HyperText Markup Language*) à partir de SGML lorsqu'il crée le Web à la fin des années 80.

Évolutions de HTML

Un rapide historique

août 1991	Apparition de HTML.
1993	Contributions de NCSA Mosaic : ajout de <i>IMG</i> (GIF et XBM) et des formulaires.
1994	Netscape ajoute des éléments de présentation : attributs, polices, alignement, clignotement ; Contraire aux principes de SGML
1995	Tim Berner-Lee (maintenant au MIT) fonde le W3C ⁵ , publication du premier standard : HTML 2.0 (RFC 1866 )
1996	Apparition de CSS (<i>Cascading Style Sheets</i>)
1997	Publication de HTML 3.2 et HTML 4.0 . Ajout de la variante <i>strict</i> , des cadres (<i>frameset</i>).
1999	Corrections mineures.
~2001	Les navigateurs commencent seulement à supporter les DTD HTML (premiers brouillons : 1992).

Documents HTML



- Arbre d'un document et sa sérialisation HTML.
- Les noeuds sont appelés des **éléments**, les marqueurs des **balises**.

```
1 <html>
2 <body>
3 <table>
4 <tr>
5 <td>
6 
7 </td>
8 </tr>
9 <tr>
10 <td>
11 text<br>
12 <a href=...>
13 text
14 </a>
15 </td>
16 <td>
17 
18 </td>
19 </tr>
20 </table>
21 </body>
22 </html>
```

Éléments et balises

```
1 <html>
2 <head>
3 <title>Exemple 1</title>
4 </head>
5 <body>
6 ...
7 </body>
8 </html>
```

- Les **balises ouvrantes** (ex : `<head>`, `<title>`, ...) et **fermantes** (ex : `</head>`, `</title>`, ...) délimitent des **éléments** dits **non-vides** (resp. l'en-tête, le titre, ...).
- Ces éléments peuvent être imbriqués.

Éléments et balises

- Les **éléments vides** sont représentés par une seule balise. Ex :

```
1 <br>  
2 
```

- Les *éléments vides* ne contiennent pas d'autres éléments.
- Remarque : contrairement à ce que l'on peut lire parfois, la syntaxe

```
1 <br />  
2 
```

pour les balises auto-fermantes n'est pas correcte en HTML (uniquement en XHTML).

Attributs et valeurs

- Les *balises ouvrantes* et *auto-fermantes* peuvent recevoir des attributs et leur valeur. Ex :

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <body>
3 <p class="introduction">blabla blabla blabla blabla blabla blabla blabla
  blabla blabla blabla.</p>
4 
5 </body>
6 </html>
```

- Les valeurs des attributs seront (de préférence) toujours placées entre guillemets simples ou doubles.

Types de documents

- On distingue plusieurs types de documents en fonction de leur langage (*HTML*, *XHTML*) ou de leur contenu (*body* ou *frameset*).
- L'élément **DOCTYPE** permet de choisir la « classe » de la présente instance et doit être placé avant l'élément *HTML*.
- Il spécifie la version du langage et donne un lien vers sa DTD (*Document Type Definition*).
- La DTD **transitional** inclut tous les éléments pour assurer une compatibilité ascendante avec les anciennes versions :

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD_HTML_4.01_Transitional//EN" "http://www.w3.org/TR/HTML4.01/loose.dtd">
```


Types de documents

- La DTD **strict** exclut tous les éléments dépréciés (comme `font` ou `align`) :

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD_HTML_4.01//EN" "http://www.w3.org/TR/HTML4.01/strict.dtd">
```

- La DTD **frameset** est identique à transitional mais remplace `body` par `frameset` et ajoute `frame` et `noframe` :

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD_HTML_4.01_Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

Structure d'un document

Un document est composée de deux éléments principaux :

- l'**en-tête** (*header*), contenant des méta-informations,
- et le **corps** (*body*), contenant les données du document.

```
1 <!DOCTYPE ... .dtd">
2 <html>
3   <head>
4     ...
5   </head>
6   <body>
7     </body>
8 </html>
```

Structure d'un jeu de cadres

- Un **jeu de cadre** (*frameset*) permet un découpage exclusivement horizontal (*cols*) ou vertical (*rows*) de la zone d'affichage.
- Un *frameset* ne contient pas de données lui-même (pas de balise *body*) mais la balise *noframe* peut en contenir.

```
1 <!DOCTYPE ... .dtd">
2 <html>
3   <frameset cols="25%,*,25%">
4     <frame src="frame_a.htm"/>
5     <frame src="frame_b.htm"/>
6     <frame src="frame_c.htm"/>
7   <noframes>
8     Sorry, your browser does not handle frames !
9   </noframes>
10 </frameset>
11 </html>
```

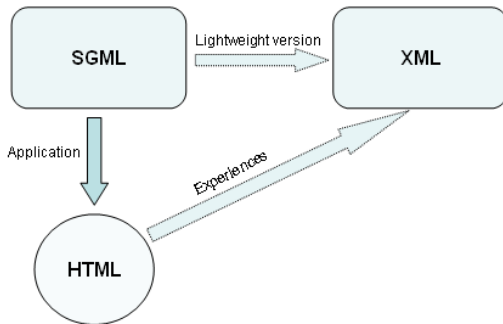
Jeux de cadres imbriqués



```
1 <!DOCTYPE ... .dtd">
2 <html>
3   <frameset ROWS="200,*">
4     <frame SRC="frame1.htm">Premiere_ligne
5     <frameset COLS="200,*">
6       <frame SRC="frame2.htm">
7       <frame SRC="frame3.htm">
8     </frameset>Seconde_ligne
9   </frameset>
10 </html>
```

Le langage XML

- *SGML* est puissant mais considéré comme trop strict et difficile à mettre en œuvre.
- Le 10 février 1998, le W3C publie les spécifications de **XML** (*eXtensible Markup Language*), un langage à balises simple et extensible.



Le langage XML

Conçu pour la description structurée

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- *XML* est un comme un sous-ensemble de *SGML*.
- Dédié à l'échange d'information sur le Web,
- il ne contient aucune directive pour l'affichage,
- Contient les spécifications :
 - **XML** : *eXtensible Markup Language* ;
 - **XLL** : *eXtensible Linking Language* ;
 - **XSL** : *eXtensible Style Language* incluant **XSLT**
(*eXtended Stylesheet Language Transformations*) et
XSF-FO(*eXtended Stylesheet Language - Formatting
Object*) ;
 - **XUA** : *XML User Agent*

Principes de XML

- 1 XML doit être directement utilisable sur internet ;
- 2 XML doit supporter une grande variété d'applications ;
- 3 XML doit être compatible avec SGML ;
- 4 le traitement des documents XML doit être simple ;
- 5 le nombre de fonctionnalités additionnelles doit être minimal, idéalement zéro ;
- 6 les documents XML doivent être lisibles par un humain et raisonnablement clairs ;
- 7 la conception d'une structure XML doit se faire rapidement ;
- 8 la structure d'un document XML doit être non-ambigue et concise ;
- 9 la création de documents XML doit être simple ;
- 10 la concision dans le nommage des balises XML est d'importance minimale.

Instances de XML

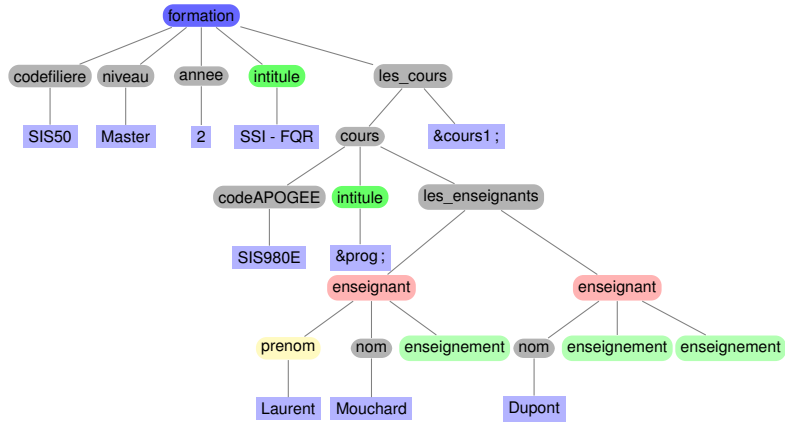
Un exemple de données structurées

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- fichier XML décrivant une formation -->
<!DOCTYPE formation SYSTEM "formation.dtd">
<?xml-stylesheet type="text/css" href="formation.css"?>
<formation>
  <codefiliere>SIS50</codefiliere>
  <niveau>Master</niveau>
  <annee>2</annee>
  <intitule>SSI - FQR</intitule>
  <lescours>
    <cours>
      <codeAPOGEE>SIS908E</codeAPOGEE>
      <intitule>&prog;</intitule>
      <lesenseignants>
        <enseignant>
          <prenom>Tryphon</prenom>
          <nom>Tournesol</nom>
          <enseignement duree="12" nature="cours"/>
        </enseignant>
        <enseignant>
          <nom>Dupont</nom>
          <enseignement duree="18" nature="TD"/>
          <enseignement duree="30" nature="TP"/>
        </enseignant>
      </lesenseignants>
    </cours>
    &cours1;
  </lescours>
</formation>
```

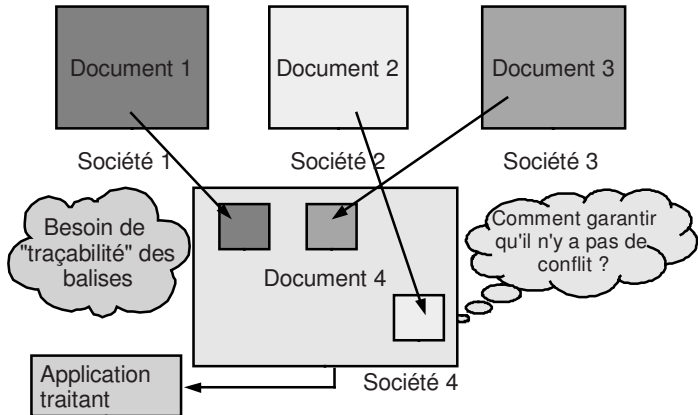

Instances de XML

Un exemple de données structurées



Inclusion en XML

- Un document racine peut en incorporer plusieurs autres utilisant des définitions différentes :



Inclusion en XML

Les espaces de noms

- Permettent d'inclure des définitions externes (au langage du document courant) en évitant les collisions de noms.
- Permettent de donner un même nom a des attributs utilisés dans des contextes différents.
- Le nom d'un espace de noms doit être unique, on utilise en général une URL.
- Un espace de nom s'applique à un élément (et sa descendance) grâce au pseudo-attribut `xmlns`.

```
1 <chapitre xmlns="http://www.alexandrebrillant.com">
2 <paragraphe>
3 ...
4 </paragraphe>
5 </chapitre>
```

- `chapitre` et `paragraphe` appartiennent à l'espace de nom donné.

Résumé sur XML

- XML est un moyen de structurer de l'information pour la sauvegarder ou l'échanger entre programmes.
- Exemples : graphiques *SVG (Scalable Vector Graphics)*, OpenDocument (Libre Office), bases de données textuelles, services Web, etc.
- *XPath* est le langage de « requêtes » sur l'arbre pour exprimer les traitements,
- même la présentation est un procédé externe pris en charge par *XSL* (conversion HTML/PDF/...).
- *XML* sera d'avantage étudié dans le module *Langages Web 2*.

Le langage XHTML

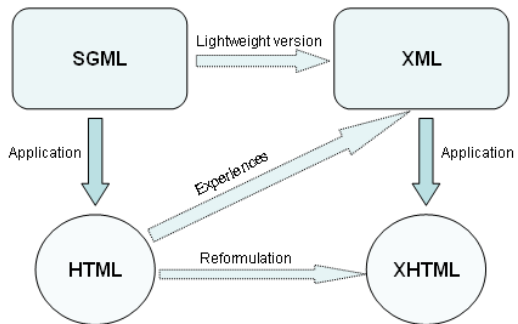
Les défauts de HTML

- Les analyseurs (*parseur*) *HTML* sont permissifs et ne permettent pas de disposer d'un analyseur standard fiable et efficace (traitement sur des terminaux légers),
- *HTML* échoue dans la problématique de séparation du fond de la forme.
- Bien qu'il permette d'inclure des portions dans des langages de scripting (élément *SCRIPT*), *HTML* ne permet pas d'inclure directement des ressources exprimées dans un dialecte *XML* ou *SGML*.
- Exemple : inclusion de graphiques au format *SVG*, formules mathématiques en *MathML*, etc.

Le langage XHTML

Raffinement de HTML, application de XML

- Le 26 janvier 2000, le W3C publie la recommandation **XHTML 1.0**,
- puis **XHTML 1.1** le 31 mai 2001.
- *XHTML* reprend *HTML 4.01* à l'identique en adaptant sa syntaxe aux règles de *XML*



Le langage XHTML

Typage

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- Doctypes pour *xhtml* :

```
1 <!DOCTYPE html
2     PUBLIC "-//W3C//DTD_XHTML_1.0_Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!DOCTYPE html
6     PUBLIC "-//W3C//DTD_XHTML_1.0_Transitional//EN"
7     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
8
9 <!DOCTYPE html
10    PUBLIC "-//W3C//DTD_XHTML_1.0_Frameset//EN"
11    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

- Comme c'est une application *XML*, le doctype peut être précédé de :

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

- et le type *MIME* devient : `application/xhtml+xml`;

Le langage XHTML

Spécificités syntaxiques

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

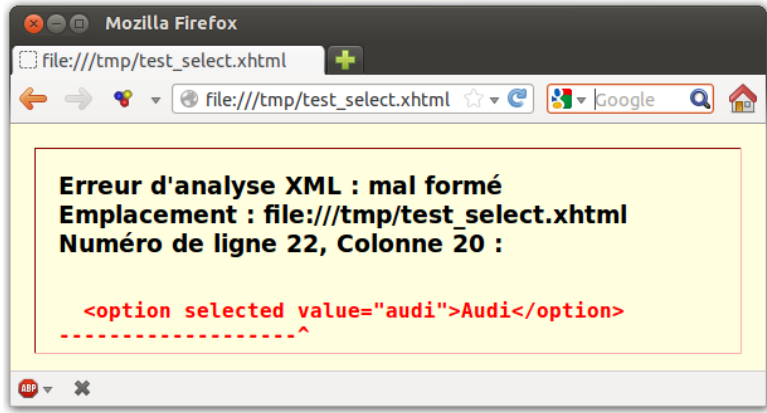
Lang.&prot.
Système

- Puisque *XHTML* est une application de *XML*, toutes les balises doivent être fermées. Ex : ``, `<meta .../>`, `
`, ...
- Contrairement à *HTML*, *XHTML* est sensible à la casse : les noms des éléments s'écrivent en minuscules.
- Les guillemets sont obligatoires pour les valeurs des attributs.
- Les raccourcis syntaxiques sont interdits. Ex : `<option selected ... >` doit être remplacé par `<option selected="selected" ... >`

Le langage XHTML

Comportement d'analyse

- Avec *XHTML*, l'analyseur syntaxique des navigateurs fonctionne de manière stricte : une erreur de syntaxe provoquera l'arrêt du chargement du document :



Le langage XHTML

Autres conséquences

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTRIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

En plus de l'analyse au chargement du document, *XHTML* apporte des changements de comportement pour :

- *JavaScript* : certaines fonctions deviennent sensibles à la casse ; `document.write()` n'existe pas ; `innerHTML` n'insère plus de contenu invalide.
- *CSS* : tous les sélecteurs *CSS* deviennent sensibles à la casse ;

Le langage XHTML

Exemple

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html
3     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
4         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
5 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
6   <head>
7     <meta http-equiv="Content-Type" content="application/xhtml+xml; charset=utf-8"
8       />
9     <title>Virtual Library</title>
10   </head>
11   <body>
12     <p>Moved to <a href="http://example.org/">example.org</a>.</p>
13   </body>
14 </html>
```

Le langage XHTML

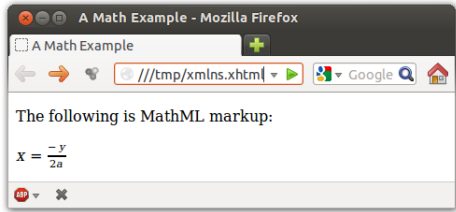
Extensibilité

- À l'instar d'*XML*, le langage *XHTML* permet d'inclure des données structurées dans un autre dialecte (basé sur *XML*).
- Pour les faire cohabiter, les espace de noms sont utilisés.
- L'espace de nom est appliqué à un élément grâce à l'attribut `xmlns`,
- les descendants de cet élément sont considéré comme appartenant à l'espace de nom.
- La valeur de l'espace de nom doit être unique, on utilise souvent une URL. Note : contrairement à une DTD, le document pointé par l'URL n'a aucune importance.

Le langage XHTML

Exemple d'utilisation d'espace de nom

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
  xhtml1/DTD/xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5   <title>A Math Example</title>
6 </head>
7 <body>
8   <p>The following is MathML markup:</p>
9   <math xmlns="http://www.w3.org/1998/Math/MathML">
10     <mrow>
11       <mi>x</mi><mo>=</mo>
12       <mfrac>
13         <mrow>
14           <mo>-</mo>
15           <mi>y</mi>
16         </mrow>
17         <mrow>
18           <mn>2</mn>
19           <mi>a</mi>
20         </mrow>
21       </mfrac>
22     </mrow>
23   </math>
24 </body>
25 </html>
```



Le langage XHTML

Validation

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD


ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- Le doctype d'*XHTML* indique toujours une *DTD* qui sert, non seulement à la validation de l'instance, mais aussi à son analyse (*parseur XML*).
-  La description de la grammaire via une DTD atteint ses limites : le validateur du W3C ne peut valider un document *XHTML* utilisant des *namespaces* personnalisés.
- L'apparition des *namespaces* est postérieur à la *DTD* qui n'avait pas prévu leur usage.
- Le monde *XML* lui préférera dorénavant les **schémas de définition** – *XML Schema Definition (XSD)*

Le langage HTML5

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

HTML



Le langage HTML5

La génèse

- **août 2002** : le W3C publie un brouillon de XHTML 2.0 qui rompt la compatibilité avec les versions précédentes (dépréciation de balises très utilisées) et abandonne le développement de *HTML*.
- **4 juin 2004** : en désaccord, le *WhatWG* (*Web Hypertext Application Technology Working Group*) est formé par Mozilla, Apple et Opera pour travailler sur la prochaine version de *HTML*.
- **avril 2007** : une première spécification est présentée au W3C qui l'accepte comme point de départ du nouveau groupe de travail *HTML*. *XHTML 2.0* est reconnu comme trop ambitieux et laissé de côté.

Le langage HTML5

La genèse

- **2008** : publication d'un premier brouillon public.
- **2011** : dernier appel à la communauté pour confirmer la justesse de la spécification.
- **2012** : retour au brouillon de travail. Le *WhatWG* continue de travailler sur *HTML5* comme *Living standard*.
- **2014** : la version finale de *HTML 5.0*.

Timeline des versions futures :

	2012	2013	2014	2015	2016
HTML 5.0	Candidate Rec	Call for Review	Recommendation		
HTML 5.1	1st Working Draft		Last Call	Candidate Rec	Recommendation
HTML 5.2 ^[28]					

Le langage HTML5

Différences avec les versions précédentes

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- Des nouvelles règles d'analyse, *HTML5* n'est plus une application de *SGML*.
- La grammaire est décrite dans la spécification, il n'y a plus de DTD.
- *HTML5* est orienté *sémantique* : des nouveaux éléments avec une sémantique plus forte que `div` et `span` **comme** `article`, `aside`, `footer`, `header`, `section` **et** `nav`.
- Les formats *MathML* et *SVG* peuvent être inclus en ligne de manière native.

Le langage HTML5

Différences avec les versions précédentes

- **Des nouveaux éléments** : article, aside, audio, canvas, command, datalist, details, embed, figcaption, figure, footer, header, hgroup, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, summary, time, video, wbr.
- **De nouveaux types *input*** : color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week.
- **De nouveaux attributs globaux** : contenteditable, contextmenu, spellcheck, draggable, hidden, role, aria-*, data-*.
- **Des éléments dépréciés** : acronym, applet, basefont, big, center, dir, font, frame, frameset, isindex, noframe, s, strike, tt, u.

Le langage HTML5

Le DOCTYPE

- Le doctype précise le langage, mais plus de version ni de DTD :

```
<!DOCTYPE html>
```

- Document minimum en *HTML 5* :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Title of the document</title>
5 </head>
6
7 <body>
8 The content of the document.....
9 </body>
10
11 </html>
```

Le langage HTML5

Syntaxe XHTML5

- Puisque *XHTML 2.0* a été abandonné, pourquoi parle-t-on de *XHTML5*???
- Réponse il s'agit simplement d'une sérialisation *XML* d'éléments décrits dans la spécification *HTML 5*.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE html>
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr" dir="ltr">
4   <head>
5     <meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF
      -8" />
6     <title>Exemple</title>
7   </head>
8   <body>
9     <!-- Contenu de la page respectant la syntaxe XML. -->
10   </body>
11 </html>
```

Le langage HTML5

Les API

La plus grande évolution est l'ajout d'API permettant d'étendre les fonctionnalités un dialogue efficace avec le système :

- multimedia avec les élément `video`, `audio`, `device`,
- dessin avec `canvas`,
- la géolocalisation avec l'objet `geolocation`,
- interaction avec les fichiers (*File API*),
- gestion native du glisser/lâcher,
- événements envoyés par le serveur (*Push*),
- communication inter-document (*Web Messaging*),
- communication en temps réel (*Web Sockets*),
- stockage local (*Web Storage*),
- bases de données (*Indexed Database and Web SQL Database*),
- application hors-ligne, ...

Respect du standard



- Le respect du standard HTML (et donc de sa syntaxe) a été pendant longtemps un problème entre les différents navigateurs.
- Le respect du standard garantit un affichage correct du document (c'est à dire conforme à ce que son auteur a voulu exprimer).
- Le processus de validation contribue⁶ à atteindre cet objectif (<http://validator.w3.org/>),
- grâce à la notion de DTD, supportée depuis la première version de HTML (mais ignorée des navigateur jusqu'à environ 2001).

6. En produisant une analyse unique pour un document. Voir plus loin pour les aspects liés au rendu.

Document Type Definition

- XML (et SGML) sont des langages génériques extensibles permettant de créer de nouveaux langages (ou *instances*, ou encore *classes de documents*),
- c'est-à-dire qu'ils comportent très peu d'éléments et que chaque instance (HTML, XHTML, SVG, ODF, ...) définit son propre espace de noms, sa propre grammaire.
- La **DTD** (*Document Type Definition*) est un document (lui même utilisant une syntaxe à balises) définissant cette grammaire.
- La DTD associée à un document sera utilisée pour sa validation mais aussi, dans le cas de HTML, ... pour son rendu !

Document Type Definition

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

La *DTD* permet de définir les trois (principaux) éléments du langage :

- les **éléments**, qui correspondront donc à des balises,
- les **attributs**, qui seront les attributs à l'intérieur des balises,
- les **entités**, que nous utiliserons principalement pour définir des caractères ou des macro-définitions,

en décrivant leur relation.

Définition d'un élément 1/2

Déclaration générique d'un élément

`<!ELEMENT nom [O/-] [O/-] spec>`

nom	Nom de l'élément décrit
[O/-]	balise ouvrante/fermante optionnelle
spec	EMPTY, ANY, #PCDATA, description descendance ou mélange.

- **EMPTY** : élément sans contenu
`<!ELEMENT img EMPTY>`
- **ANY** : tout élément de la DTD
`<!ELEMENT container ANY>`
- **PCDATA** : texte interprété (Parsed Character DATA)
`<!ELEMENT TEXTAREA - - (#PCDATA) ->`

Définition d'un élément 2/2

Description de la descendance

<code>e11,e12</code>	<code>e11</code> suivi de <code>e12</code> , dans cet ordre
<code>e11 e12</code>	<code>e11</code> XOR <code>e12</code> (pas les deux)
<code>(...)</code>	regroupe les éléments présents entre parenthèses
<code>e1+</code>	au moins une fois <code>e1</code>
<code>e1*</code>	zéro au plusieurs fois <code>e1</code>
<code>e1?</code>	zéro ou une fois <code>e1</code>

Exemple en XML :

```
<!ELEMENT enseignant (prenom?, nom,  
enseignement+)>  
  <!ELEMENT prenom (#PCDATA)>  
  <!ELEMENT nom (#PCDATA)>  
  <!ELEMENT enseignement EMPTY>
```

Un élément `enseignant` contient éventuellement un `prenom`, suivie d'exactly un `nom`, suivie d'au moins un `enseignement`.

Définition d'attributs 1/2

Déclaration générique d'une liste d'attributs

```
<!ATTLIST elem attr type mode>
```

elem	nom de l'élément contenant l'attribut
attr	nom de l'attribut
type	type de données de l'attribut : <code>CDATA</code> ⁷ , <code>NMTOKEN</code> ⁸ , liste de valeurs, <code>ID</code> (clé, de type <code>NMTOKEN</code>) ou <code>IDREF</code> (référence à clé)
mode	<code>#REQUIRED</code> : attribut obligatoire
	<code>#IMPLIED</code> : attribut optionnel
	<code>#FIXED</code> : attribut fixé, non modifiable

```
<!ELEMENT enseignement EMPTY>
```

```
<!ATTLIST enseignement  
    duree CDATA #REQUIRED  
    nature (cours|TD|TP) #REQUIRED>
```

7. Chaîne quelconque

8. Un seul mot, commençant par une lettre et contenant lettres, chiffres et caractères de ponctuation

Définition d'attributs 2/2

Les clés et références aux clés

- Contenu du fichier DTD :

```
<!ELEMENT individu (#PCDATA)>
  <!ATTLIST individu id ID #REQUIRED>
  <!ATTLIST individu pere IDREF #IMPLIED>
  <!ATTLIST individu mere IDREF #IMPLIED>
```

id	attribut de type clé
pere	attribut de type référence à une clé
mere	attribut de type référence à une clé

- Contenu d'une instance XML :

```
<individu id="id0">père</individu>
<individu id="id1">mère</individu>
<individu id="id2" pere="id0" mere="id1">fils</individu>
<individu id="id3" pere="id0"
mere="id1">fille</individu>
```

Définition d'entités 1/2

Cinq entités prédéfinies

```
<!ENTITY lt    "&#38;#60;"> <!ENTITY gt    "&#62;">  
<!ENTITY amp   "&#38;#38;"> <!ENTITY apos  "&#39;">  
<!ENTITY quot  "&#34;">
```

Entités générales

- déclaration d'une entité générale :

```
<!ENTITY prog "Programmation des  
Nouvelles Technologies">
```
- utilisation d'une entité générale :

```
<intitule>&prog;</intitule>
```

Définition d'entités 2/2

Entités externes

- déclarations d'une entité externe :

```
<!ENTITY logo SYSTEM "http://www.site.fr/logo.png"  
        NDATA png>
```

```
<!ENTITY cours1 SYSTEM "cours1.xml">
```

- utilisation d'une entité externe :

```
<logo>&logo;</logo>  
&cours1;
```

Entités paramètres

- déclaration d'une entité paramètre :

```
<!ENTITY % nature " (cours|TD|TP) ">
```

- utilisation d'une entité externe :

```
<!ATTLIST enseignement  
        duree CDATA #REQUIRED  
        nature %nature; #REQUIRED>
```

Exemples d'entités (strict)

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

```
1 <!ENTITY % ContentType "CDATA"  
2   — media type, as per [RFC2045] —>  
3 <!ENTITY % ContentTypes "CDATA"  
4   — comma-separated list of media types, as per [RFC2045] —>  
5 <!ENTITY % Charset "CDATA"  
6   — a character encoding, as per [RFC2045] —>  
7 <!ENTITY % URI "CDATA"  
8   — a Uniform Resource Identifier,  
9   see [URI] —>  
10 <!ENTITY % Datetime "CDATA" — date and time information. ISO date format —>  
11 <!ENTITY % Script "CDATA" — script expression —>  
12 <!ENTITY % StyleSheet "CDATA" — style sheet data —>  
13 <!ENTITY % Text "CDATA">  
14 ...
```


Exemples d'entités (strict)

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

```
1 <!-- Parameter Entities -->
2 <!ENTITY % head.misc "SCRIPT|STYLE|META|LINK|OBJECT" — repeatable head elements
   —>
3 <!ENTITY % heading "H1|H2|H3|H4|H5|H6">
4 <!ENTITY % list "UL|OL">
5 <!ENTITY % preformatted "PRE">
```

Exemples d'entités (strict)

Attributs génériques :

```
<!ENTITY % coreattrs
    "id            ID                #IMPLIED -- document-wide unique id --
     class         CDATA             #IMPLIED -- space-separated list of classes --
     style        %StyleSheet;      #IMPLIED -- associated style info --
     title        %Text;            #IMPLIED -- advisory title --"
>

<!ENTITY % i18n
    "lang         %LanguageCode; #IMPLIED -- language code --
     dir         (ltr|rtl)       #IMPLIED -- direction for weak/neutral text --"
>

<!ENTITY % events
    "onclick      %Script;         #IMPLIED -- a pointer button was clicked --
     ondblclick   %Script;         #IMPLIED -- a pointer button was double clicked--
     onmousedown  %Script;         #IMPLIED -- a pointer button was pressed down --
     onmouseup    %Script;         #IMPLIED -- a pointer button was released --
     onmouseover  %Script;         #IMPLIED -- a pointer was moved onto --
     onmousemove  %Script;         #IMPLIED -- a pointer was moved within --
     onmouseout   %Script;         #IMPLIED -- a pointer was moved away --
     onkeypress   %Script;         #IMPLIED -- a key was pressed and released --
     onkeydown    %Script;         #IMPLIED -- a key was pressed down --
     onkeyup      %Script;         #IMPLIED -- a key was released --"
>

<!ENTITY % attrs "%coreattrs; %i18n; %events;">
```

Exemples d'entités (strict) – Fin

Balises de texte / flux :

```
<!ENTITY % fontstyle "TT|I|B|BIG|SMALL">

<!ENTITY % phrase "EM|STRONG|DFN|CODE|SAMP|KBD|VAR|CITE|ABBR|ACRONYM">

<!ENTITY % special "A|IMG|OBJECT|BR|SCRIPT|MAP|Q|SUB|SUP|SPAN|BDO">

<!ENTITY % formctrl "INPUT|SELECT|TEXTAREA|LABEL|BUTTON">

<!-- %inline; covers inline or "text-level" elements -->
<!ENTITY % inline "#PCDATA | %fontstyle; | %phrase; | %special; | %formctrl;">

...

<!--===== HTML content models =====>
<!-- HTML has two basic content models:
           %inline;      character level elements and text strings
           %block;       block-like elements e.g. paragraphs and lists
-->
<!ENTITY % block
    "P | %heading; | %list; | %preformatted; | DL | DIV | NOSCRIPT |
    BLOCKQUOTE | FORM | HR | TABLE | FIELDSET | ADDRESS">

<!ENTITY % flow "%block; | %inline;">
```

Standard et rendu

La conformité par rapport au standard repose sur deux aspects :

- l'analyse syntaxique,
- le rendu.

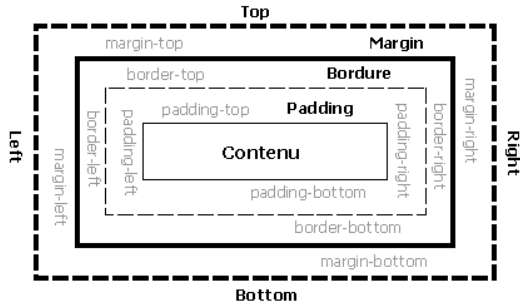
La DTD spécifie la grammaire et ne concerne donc que le premier point.

Le rendu, lui est décrit dans une spécification et repose sur son implémentation.

Sans respect du standard, pour un même document valide, il peut y avoir des rendus différents.

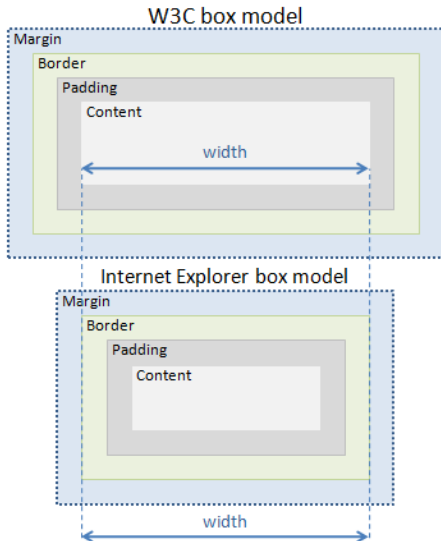
Des différences de rendu ?

- Outre le fait que certains éditeurs de navigateurs ajoutaient des balises propriétaires,
- certains navigateurs faisaient un rendu non conforme de certains éléments.
- En particulier, le *modèle de boîte*, qui permet de calculer les dimensions de la plupart des éléments (Tableaux, div, etc.).



- Calculs des dimensions différents,
- IE 5.x ne connaît que le modèle Micro\$oft
- IE 6 intègre les deux modèles ...

Modèles de boîtes



Le DocType switching

ou mode Quirks

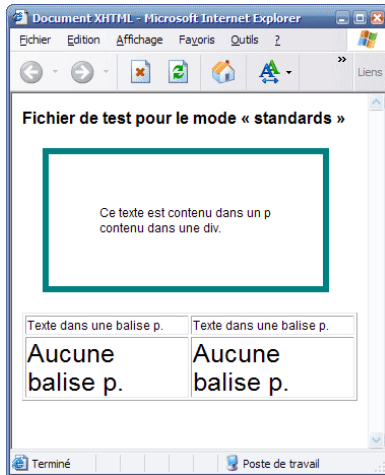
- Les navigateurs récents qui respectent le standard offrent souvent un rendu en *mode compatibilité* appelé aussi *mode quirks*.
- Dans ce mode, les navigateurs reproduisent volontairement certains « défauts » d'affichage répandus pour afficher correctement d'anciennes pages.
- Ce mode est généralement déclenché par l'absence d'élément *doctype* dans le document (*doctype switching*).
- Le rendu en mode de compatibilité n'est pas un standard⁹.

9. What happens in Quirks Mode : <http://www.cs.tut.fi/~jkorpela/quirks-mode.html>

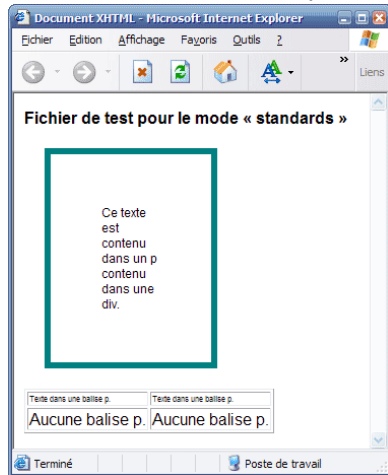
Différences de rendu

Quirks vs standard

Rendu mode standard



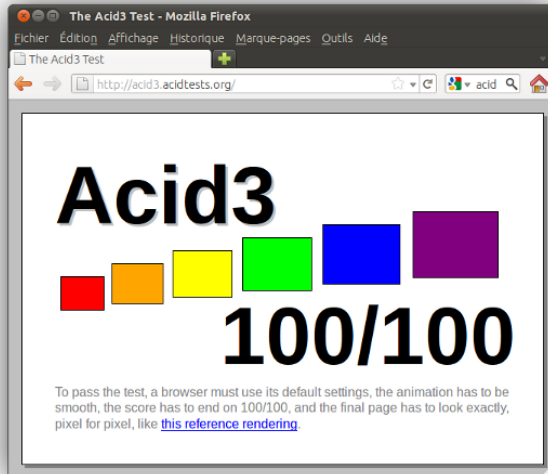
Rendu mode quirks



... notables !

Test Acid3 pour navigateurs

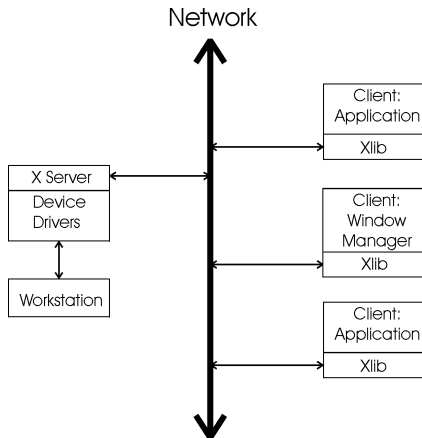
Test de compatibilité de rendu



Respect du standard

Les enjeux

- Le respect des standards est devenu une priorité pour tous les acteurs (même pour MS).
- La raison principale : **les applications web !**
- Retour vers une vieille architecture : XWindow, HTML+HTTP remplacent le protocole X.



X Window System Architecture

... en particulier pour les webapps

Éviter d'avoir à écrire ça :

```
1 <!--[ if IE 5]> <link rel="stylesheet" href="
   styles / internet-explorer-5.css" type="text /
   css" media="screen" /> <![endif]-->
```

```
2
3 <!--[ if IE 6]> <link rel="stylesheet" href="
   styles / internet-explorer-6.css" type="text /
   css" media="screen" /> <![endif]-->
```

```
4
5 <!--[ if IE 7]> <link rel="stylesheet" href="
   styles / internet-explorer-7.css" type="text /
   css" media="screen" /> <![endif]-->
```

```
6 ...
```

Applications web

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

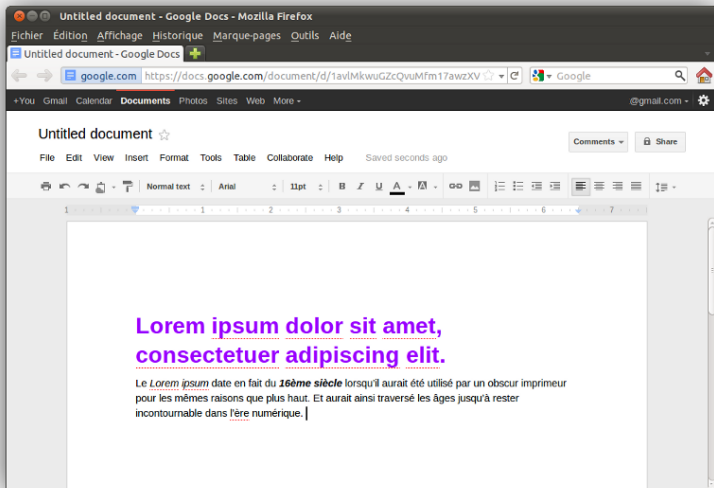
ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système



Applications web

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

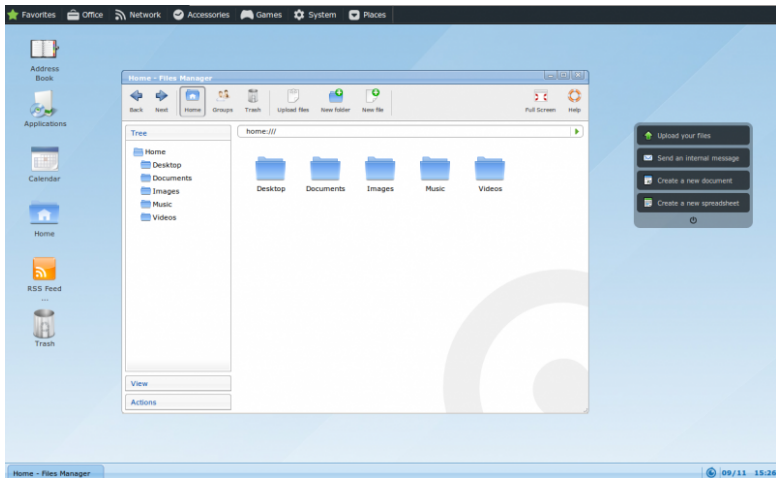
ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système



Interactions

avec les autres langages et protocoles

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- HTML comporte des dépendances implicite ou explicite avec :
 - le protocole HTTP,
 - le langage CSS
 - le langage JavaScript
- Note : ce n'est pas réciproque.

Interactions

avec le protocole HTTP

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

Les documents HTML sont implicitement ou explicitement liés au protocoles HTTP :

- par les URL : liens cliquables (``), les images (``) relations avec des ressources externes les images (`<link rel="...">`)
- Formulaires qui produisent une requête HTTP dont certains paramètres sont contenus dans le corps du documents : cible (`action`), méthode du protocole (`method`), encodage des données envoyées (`enctype`).

Interactions

avec le langage CSS

Le rendu d'un document HTML se fait à travers l'emploi de feuilles de styles implicites ou explicites :

- tout élément a un style par défaut,
- ce style peut être redéfini pour tout élément (par attribut `style=" . . . "` ou incorporation d'une feuille de style) grâce au langage CSS (*Chapitre 3*).

En effet, pour chaque document chargé, le navigateur :

- crée les éléments décrits dans le document,
- associe à chacun d'eux des propriétés¹⁰ de rendu : modèle de boîte, couleur de tracé, couleur de fond, style, etc.

Chacun est altérable.

10. de manière quasi-orthogonal

Interactions

avec le langage JavaScript

Les documents HTML peut incorporer implicitement ou explicitement du code JavaScript (*Chapitre 4*) :

- par inclusion de snippets ou de fichier externe (balise `<script>` dans l'en-tête),
- affectation de code à des événements : `<p onclick="alert('coucou') ">` .

Pour chaque document chargé, le navigateur associe :

- une machine virtuelle au document : un espace de variables, de fils d'exécution, etc ...
- des événements interceptables à chaque élément (dans l'arbre) produit.

Ce langage dispose d'une API pour opérer des transformations sur l'arbre du document (*Chapitre 5*).

Interactions

avec le serveur : les formulaires

Sans utiliser AJaX, les formulaires sont le seul moyen de retourner l'état d'une vue au serveur.

`<form ...>`

- **action** : url de la cible qui recevra les données
- **accept-charset** :
- **autocomplete** :
- **enctype** : encodage du formulaire avec POST :
`application/x-www-form-urlencoded`,
`multipart/form-data`, `text/plain`.
- **method** : méthode HTTP à utiliser (GET ou POST).
- **novalidate** : permet l'envoi sans validation.
- **target** : fenêtre qui recevra la réponse (`_blank`,
`_self`, `_parent`, `_top`).
- ...

Interactions

avec le serveur : les formulaires

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>List of inputs</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body>
  <form action="http://localhost:3333" method="post" enctype="multipart/form-data">
    <input type="text" name="txt" value="Some&nbsp;text">
    <input type="password" name="pass" value="Secret"><br>
    <textarea placeholder="Type here !" id="tar"></textarea>
    <select name="Car">
      <optgroup label="Swedish Cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
      </optgroup>
      <optgroup label="German Cars">
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
      </optgroup>
    </select>
    <input type="file" name="uploadedfile">
    <input type="submit" value="Go">
  </form>
</body>
</html>
```

Interactions

les formulaires : éléments d'entrées

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système

- **Types de d'éléments classiques** : `text`, `password`, `hidden`, `radio`, `checkbox`, `button`, `reset`, `submit`, `image`, `file`.
- **Nouveaux types d'HTML5** : `tel`, `url`, `email`, `search`, `date`, `time`, `datetime`, `datetime-local`, `month`, `week`, `number`, `range`, `color`

Éléments sémantiques

Et utilisation par le système

Langages

Modèle
SGML
HTML
XML
XHTML
HTML5

Syntaxe et DTD

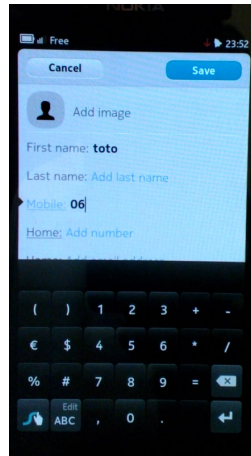
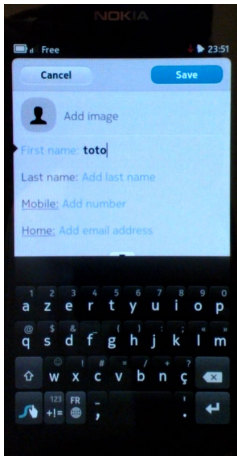
ELEMENT
ATTLIST
ENTITY

Rendu

Modes de rendu
Enjeu

Interactions

Lang.&prot.
Système








Interactions

avec le système du client



HTML, à travers le navigateur web, interagit avec des composants du système du client. HTML augmente ce niveau d'interactivité :

- les widgets d'interface graphique : les champs de saisie formulaires,
- le système de fichier : champs de type `file`, IAPI `File`,
- des bases de données : web Storage et Indexed DB,
- les sockets réseau : les `WebSocket`,
- la carte graphique(GPU) : l'API Canvas (2D, 3D),
- le GPS : API de géolocalisation,

Quelques références

- HTTL 2.0 : RFC 1866 
- HTML 4.01 : HTML 4.01 Specification 
- HTML 5 A vocabulary and associated APIs for HTML and XHTML 
- HTML 5 differences from HTML 4 
- HTML Living Standard specifications 

Pour les TP :

- The W3C Markup Validation Service :
<http://validator.w3.org> 
- HTML Reference - W3Schools :
<http://www.w3schools.com/tags/> 
- Recommended list of Doctype declarations :
<http://www.w3.org/QA/2002/04/valid-dtd-list.html> 