Introduction et HTML

Technologies du Web 1

Jean-Christophe Routier
Licence 1 SESI
Université Lille 1

Introduction

Objectifs :

- Présentation des bases de la création de documents web par la découverte des technologies « côté client »
  - création de la page, contenu et forme : HTML/CSS
  - programmation : javascript
- approfondi et prolongé par l'UE **Technologies du Web 2** du S4 Informatique

`http://portail.fil.univ-lille1.fr/ls2/tw1`

Ce cours n'est pas...

- ... un catalogue exhaustif des fonctionnalités HTML et CSS
- ... un cours de « création graphique » de page web
- ... un cours complet sur javascript
  - pas de « programmation objet »
  - pas d'utilisation de frameworks existants
- ... un cours complet sur les technologies web
  - pas de programmation côté serveur (php, J2EE, etc.)
  - pas de bases de données
  - pas d'« *ajax* »

Bon alors...

on va faire quoi ?

A l'issue de ce module vous devriez...

- **Être capables de concevoir des documents web dans le respect des standards.**
  - Connaître les principaux standards du web : (X)HTML 5, CSS, Javascript, DOM
  - Maîtriser la notion de séparation contenu / forme / dynamicité
  - Savoir
    - modéliser un document sous forme arborescente et traduire ce modèle en un document (X)HTML 5
    - réaliser la mise en forme en utilisant le langage CSS
    - rendre le document dynamique et le manipuler via l'interface DOM / javascript
  - Être conscient de l'importance du respect des normes
  - Maîtriser le processus de rédaction et de validation des documents.
- **Savoir utiliser des outils de développement adaptés**
  - inspecteurs de structure HTML et de style CSS,
  - debugger javascript

A l'issue de ce module vous devriez...

- **Savoir développer des programmes en javascript et connaître les bases de ce langage.**
  - syntaxe
  - spécificité du langage
  - réutiliser ce qui a été vu en ocaml (*InitProg* et *API1*)
  - fonctionnalités « de base » du langage
  - produire des pages web dynamiques
- **Connaître les bases de la programmation événementielle.**
  - principes
  - mise en œuvre
- **Être capables de rechercher des informations dans un document et de les exploiter**
  - spécifications de standards (HTML, CSS)
  - sites de références

Fonctionnement

- quelques cours d'amphi, pas nécessairement chaque semaine
  - notions d'ordre général
- 2 TD d'1h30 par semaine, tous en « salle machines »
  - apprendre par la pratique
  - lire les documentations proposées
  - savoir rechercher une information, l'analyser et l'utiliser
  - explorer, essayer
- réalisation de « mini-projets »
- exercices à rendre régulièrement, pris en compte dans l'évaluation

De nombreux éléments seront acquis via les exercices.

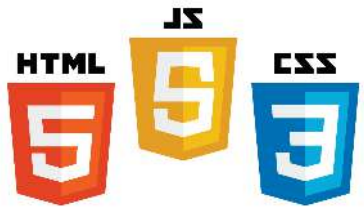
Evaluation :

contrôles en TD + devoir final + exercices rendus

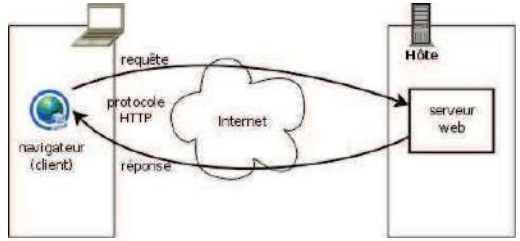
Internet et le web

- **Internet** : un réseau mondial (supranational) de machines interconnectées
  - un réseau de réseaux
  - des protocoles de communication : **TCP-IP**
  - nombreuses applications : courrier électronique, transfert de fichiers (ftp), messagerie instantanée, peer-to-peer, **World Wide Web**
- le **Web** (Tim Berners-Lee, 1989) :
  - système d'information réparti en « *pages web* » = **documents web**
  - basé sur la notion d'**hypertexte** et la notion d'**hyperliens** permettant de naviguer entre les documents web
    ↪ les documents forment un graphe = « *toile* »
  - des protocoles de commmunication **HTTP**, **HTTPS**
  - des adresses pour nommer les documents : **URL**
  - des langages pour créer les documents : **HTML**, **CSS**, **javascript**
  - des **navigateurs** qui interprètent les documents

- **W3C** « *World Wide Web Consortium* »
  - organisme de **normalisation** chargé de promouvoir la compatibilité des technologies du web
  - « *un seul web partout et pour tous* »
- **HTML5** = HTML5 + CSS3 + javascript



Client-Serveur



Documents web

- nécessité de respecter les normes/standards, **interopérabilité** (différents clients)
  - **validation** (XHTML5 et CSS)
- **séparation contenu-forme**
  - contenu : **XHTML**, structure du document
  - forme : **CSS**, présentation, style
  - exemple : sans style ⊙ – style 1 ⊙ – style 2 ⊙
- avantages :
  - adaptation aux différents supports, accessibilité
  - facilité de maintenance, évolutivité

HTML

HTML
HyperText Markup Language

- un langage de description de document
- permet de structurer le contenu d'un document
- langage de balisage parenthésé

XHTML :

- une « version » d'HTML conforme au langage XML
- XML : eXtensible Meta Language
- XHTML5 ≡ HTML5
  - mêmes éléments et mêmes attributs
  - une syntaxe plus rigoureuse
    plus facilement manipulable par des outils logiciels

Langage
vocabulaire + syntaxe + sémantique

HTML :

- **vocabulaire** = **éléments** prédéfinis, identifiés par des **balises**
  - ouvrante/fermante : `<element>`/`</element>`
  - insensible à la casse : `<eLEmENt>` = `<eLeMeNT>`
    mais XHTML ⇒ **minuscules**
- **sémantique** = rôle/sens des éléments
  - `<h1>` : un titre de premier niveau
  - `<p>` : un paragraphe de texte
  - `<img>` : une image stockée dans un fichier externe
  - `<time>` : une heure ou une date
- **syntaxe** = règles d'écriture du document

Structure d'un document

Structure minimale :

- un **DOCTYPE**
- un élément **racine** <html>
- un élément **entête** <head>
  - un élément **titre** <title>
  - déclaration de l'**encodage** de caractères utilisé
- un élément **corps** <body>


```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

  <!-- entête du document -->
  <head>
    <title>Document HTML 5 minimal</title>
    <meta charset="UTF-8"/>
  </head>

  <!-- corps du document -->
  <body>
     <!-- on placera ici le contenu de la page -->
  </body>
</html>
```


Validation

La **validation** permet de vérifier la correction syntaxique.

http://validator.w3.org/nu

La validation d'un document produit doit être **systématique**

options à adapter pour validation XHTML5, cf. TD.

Règles d'écriture : parenthésage

Les éléments non vides doivent toujours avoir une **balise ouvrante** <element> et une **balise fermante** </element>.
Le **contenu** de l'élément se trouve entre ces balises.

ex : *contenu du paragraphe*

Les éléments vides ont une seule balise terminée par /> : <vide/>
Ils n'ont pas de contenu.

ex :

Règles d'écriture : Imbrication/Emboitement

Le contenu d'un élément peut contenir d'autres éléments « imbriqués ».

```
<p>
  début
  <code>
      emboité1
      <strong>
         emboité2
      </strong>
      emboité1
  </code>
  suite
</p>
```


Règles d'écriture : Imbrication/Emboitement

Les emboitements obéissent à des règles, tout n'est pas possible.

ex :

- un élément <p> ne peut pas contenir un élément <h1>
- un élément <ul> contient nécessairement au moins un élément
- un élément <li> est nécessairement emboité dans un élément
- etc.

Règles d'écriture : Imbrication/Emboitement

Les éléments ne doivent pas se chevaucher : premier ouvert, dernier fermé.

Ceci est **interdit** :

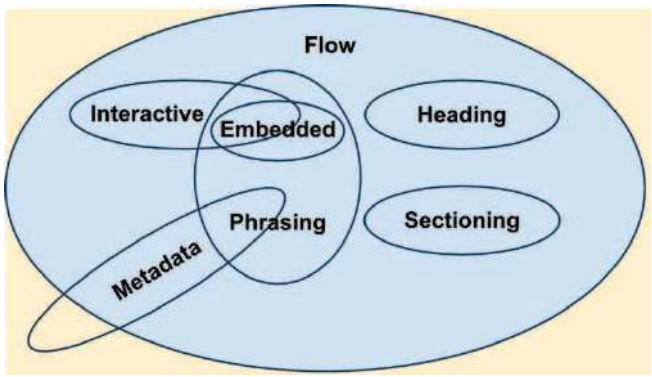
```
<p> debut <code> emboité ? </p>  suite <code>
```


Attributs

- définit une propriété pour un élément
- se note dans la balise ouvrante de l'élément, en minuscules et sa valeur est entre guillemets "
- certains sont communs à tous les éléments
  - `title` information textuelle sur le contenu de l'élément
  - `class` associe une ou plusieurs classes à l'élément, une classe peut être partagée par plusieurs éléments
  - `id` nomme un élément de manière **unique** dans le document
  - **etc.**
- d'autres sont spécifiques à un élément
  - `src` pour l'élément <img> désigne la source de l'image

```
<img src="images/HTML5_Logo.png"
     class="gauche encadree"
     id="logo"
     alt="le logo HTML 5" />
```

Différentes catégories d'éléments



cf. Mozilla Developer Network

- éléments de **flux**
  - contiennent généralement du texte ou du contenu intégré
- éléments de **phrasé**
  - définissent le texte et le balisage qu'il contient
  - des séquences de contenu phrasé constituent des paragraphes
- éléments **intégrés**
  - importent une ressource définie en dehors du document
- éléments de **titre**
- éléments **sectionnants**
  - permettent une décomposition du document

Sémantique des éléments

- chaque élément possède une sémantique qui définit son usage
- lors de la création d'un document faire des choix pertinents d'éléments

Savoir rechercher l'information (documentation), la lire et l'exploiter

Sections

Eléments sectionnants

`<section>` section générique dont le contenu est cohérent thématiquement

`<article>` contenu autonome dans un document qui doit pouvoir être réutilisé indépendamment

`<aside>` information connexe au contenu principal « voisin »

`<nav>` contient des liens de navigation vers des fragments du document ou vers d'autres documents (« menu »)

Entête et pied

`<header>` introduction à un document, une section, un article. Peut contenir un titre, un logo, etc.

`<footer>` pied de page, d'une section, d'un article, etc. Peut contenir des informations, des liens annexes, etc.

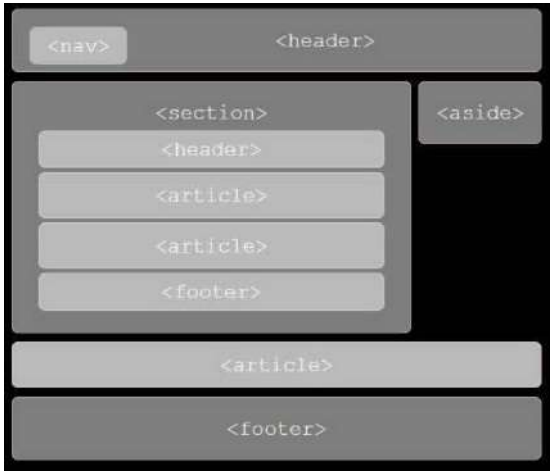

```
<body>
  <header>
     <nav> </nav>
  </header>

  <aside> </aside>

  <section>
     <header> </header>
     <article> </article>
     <article> </article>
     <footer> </footer>
  </section>

  <article> </article>

  <footer> </footer>
</body>
```



⊙ MDN ⊙ FIL

Titres

Titres

`<h1>` à `<h6>` titres, par ordre décroissant d'importance

exemple

Eléments de flux

Ce sont des éléments structurants.

Quelques exemples

`<p>`	paragraphe, ne peut contenir que des éléments de type phrasé
`<ol>, <ul>`	listes
`<table>`	tableaux
`<figure>`	contenu indépendant accompagné d'une légende et référencé dans le texte. Le plus souvent une image, un diagramme, etc.
etc.	

Découverte par la pratique en TD.

Eléments de phrasés

Ne peuvent contenir que du contenu de type phrasé.

Quelques exemples

`<em>`	un élément de texte important, mis en valeur,
`<strong>`	un élément de texte très important, encore plus mis en valeur,
`<code>`	une portion de code de programme
`<kbd>`	une entrée au clavier
etc.	

Découverte par la pratique en TD.

exemple

Hyperliens

`<a>`

L'élément `<a>` permet de définir des liens externes ou internes au document. La cible du lien est définie par l'attribut `href` dont la valeur est une **URL**.

- élément de flux ou de phrasé, selon la catégorie du contenu

```
<a href="http://fil.univ-lille1.fr"
   target="_blank" title="le site du FIL"> le FIL </a>
```

URL

Uniform Resource Locator : identifiant des ressources web
= « adresses web »

- URL absolue :
  - `http://fr.wikipedia.org/wiki/Uniform_Resource_Locator`
  - `mailto:prenom.nom@univ-lille1.fr?subject=[TW1]question`
- URL relative : `../images/firefox.png`

Eléments génériques

** et **

`<div>` élément de flux générique, permet d'organiser le contenu du document

`<span>` élément de phrasé générique, permet de distinguer une portion de texte

- pas de sémantique spécifique a priori
- sémantique définie implicitement par le rédacteur, via les attributs `class` et `id`
- prendront leur importance avec les CSS

Hors catégorie

Certains éléments n'appartiennent à aucune catégorie car ils apparaissent nécessairement emboité dans un autre élément.

Quelques exemples

`<li>` un élément dans une liste `<ul>` ou `<ol>`

`<figcaption>` la légende d'une `<figure>` plus en valeur,

`<tr>, <td>` les cellules d'une `<table>`

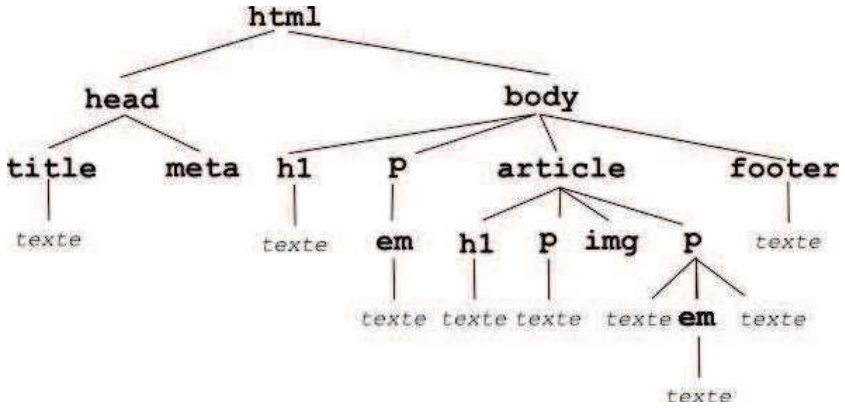
etc.

Des arbres...

Vous pensez peut-être qu'un arbre c'est ça

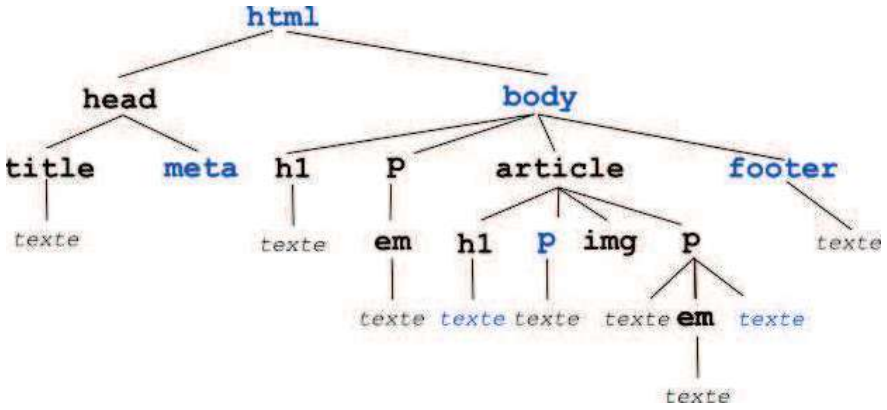
mais en fait non...

Un arbre c'est ça :



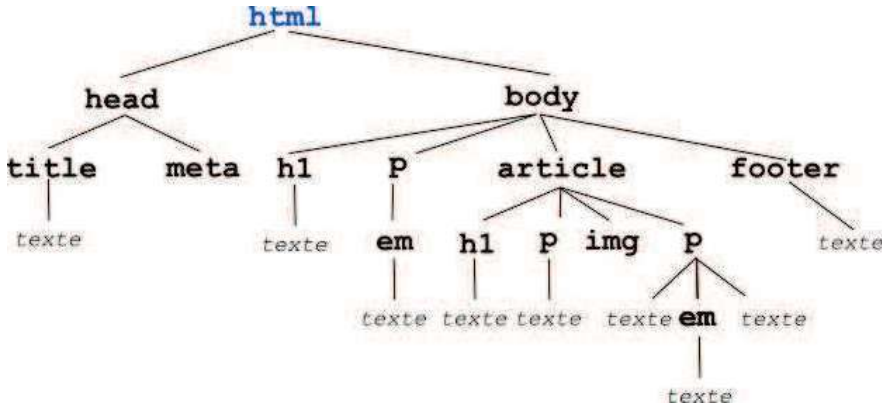
Vocabulaire sur les arbres

Des **nœuds** de l'arbre.



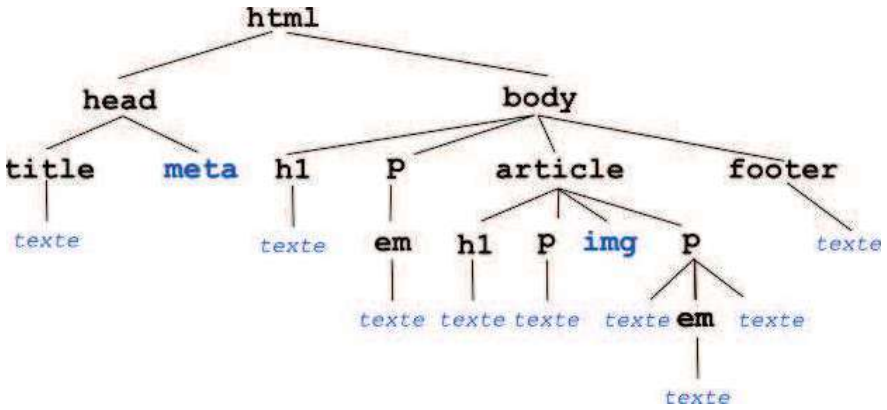
Vocabulaire sur les arbres

Le nœud **racine** de l'arbre



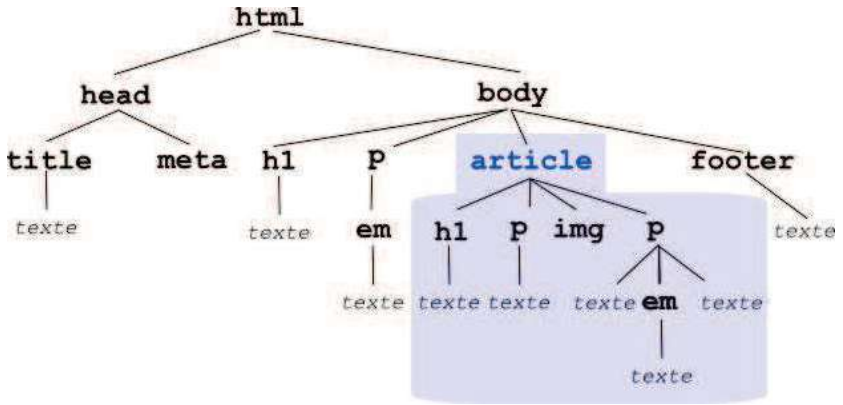
Vocabulaire sur les arbres

Les nœuds **feuilles** de l'arbre



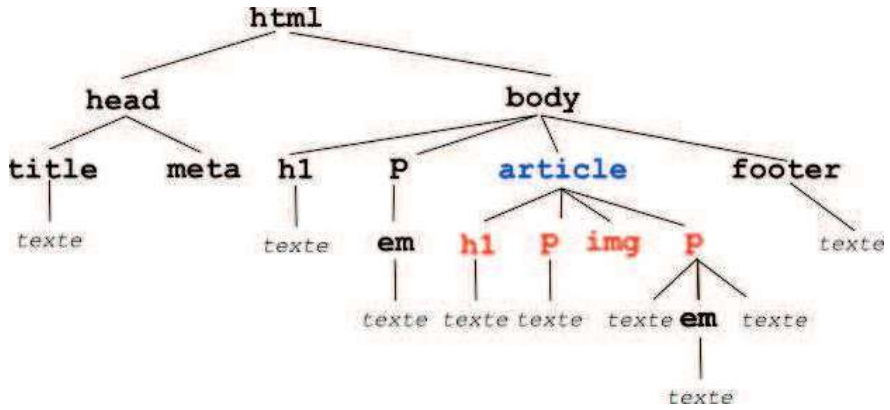
Vocabulaire sur les arbres

sous-arbre de racine le nœud `article`.
Les nœuds du sous-arbre sont les **descendants** du nœud `article`



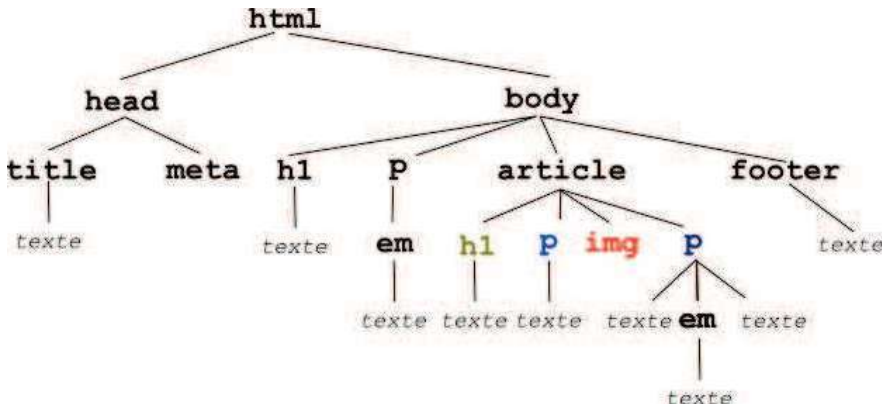
Vocabulaire sur les arbres

Le nœud `article` est le **père** des noeuds `h1`, `p`, `img` et `p`.
Les nœuds `h1`, `p`, `img` et `p` sont les **fils** du nœud `article`.
= descendants de premier niveau.



Vocabulaire sur les arbres

Les nœuds `h1`, `p`, `img` et `p` sont des nœuds **frères**.
Le nœud `img` **suit** le premier nœud `p`.
Le nœud `h1` **précède** le premier nœud `p`.

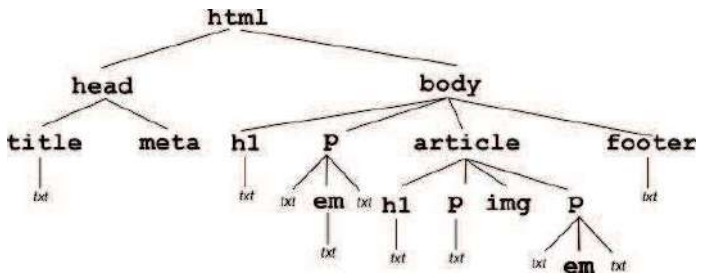


Structure arborescente d'un document HTML

Arbre DOM

On parle d'arbre **DOM** (*Document Object Model*) d'un document.

Les nœuds de l'arbre sont les éléments.



Le plus souvent on ignore les nœuds *txt*.

exemple ⊙

Structure arborescente d'un document HTML

- on parcourt le document séquentiellement
- chaque élément est un nouveau nœud
- si l'élément <elt> est emboité dans l'élément <boite>, le nœud **elt** est fils du nœud **boite**
- les nœuds des éléments emboités à un même niveau sont frères.
  Ils se suivent « de gauche à droite » selon leur ordre d'apparition dans le document

exemple ⊙