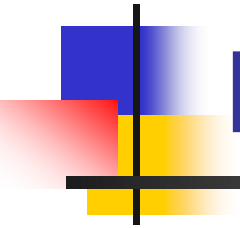
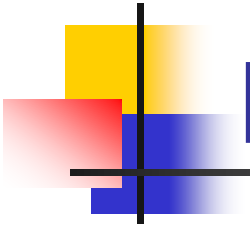


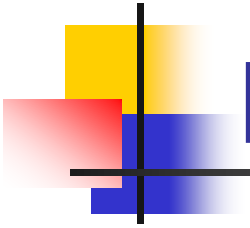
Programmation Web

Introduction



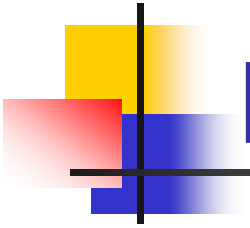
Le Web

- Le World-Wide Web (ou WWW, ou Web)
  - Très grand système d'information
    - réparti sur un ensemble de sites connectés par le réseau Internet
    - constitué de documents hypertextes
      - textes, images, sons, videos, etc.
  - Chaque site propose des documents qui sont transmis sur le réseau par l'intermédiaire d'un **programme serveur**
  - Ce programme serveur dialogue avec un **programme client** qui prend le plus souvent la forme d'un navigateur



Le Web

- Le dialogue
  - Le dialogue entre un programme serveur et un programme client s'effectue selon des règles précises qui constituent un *protocole*
    - *Le protocole du Web est HTTP*
  - *D'autres protocoles sont également utilisés*
    - *FTP pour la transmission de fichiers*

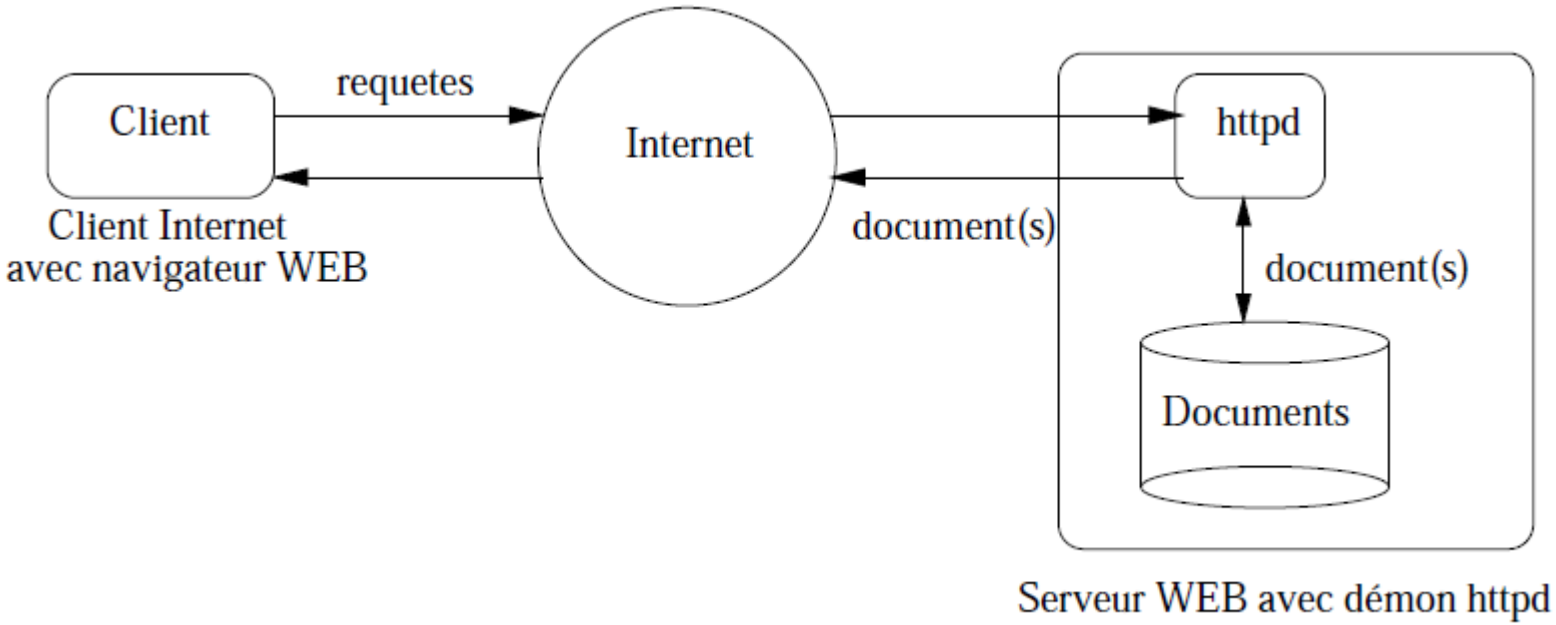


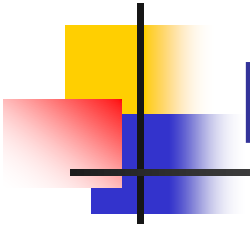
L'architecture utilisée

- Client / Serveur
  - Un site est constitué, matériellement, d'un ordinateur connecté à Internet, et d'un programme tournant en permanence sur cet ordinateur, le *serveur*
  - Le programme serveur est en attente de requêtes transmises à son attention sur le réseau par un programme client
  - Quand une requête est reçue, le programme serveur
    - l'analyse afin de déterminer quel est le document demandé
    - recherche ce document
    - et le transmet au programme client
  - Apache est le serveur HTTP le plus utilisé

Architecture Client Serveur

- Architecture web
  - Un petit démon (programme : httpd) attend les requêtes pour les analyser





Le démon HTTPD

- Requête au niveau du client
  - Le client spécifie un fichier comme par ex. ceci :

    Fichier
  - qui indique en fait :
    1. La machine sur laquelle se trouve le fichier: www.iro.umontreal.ca
    2. le "path" ou chemin pour atteindre le fichier: ~vaucher/test.html
    3. le "protocole": http
  - Ceci se traduit en interne par une requête GET avec des paramètres précis indiquant le serveur, le port… sur lequel il faut mettre les informations

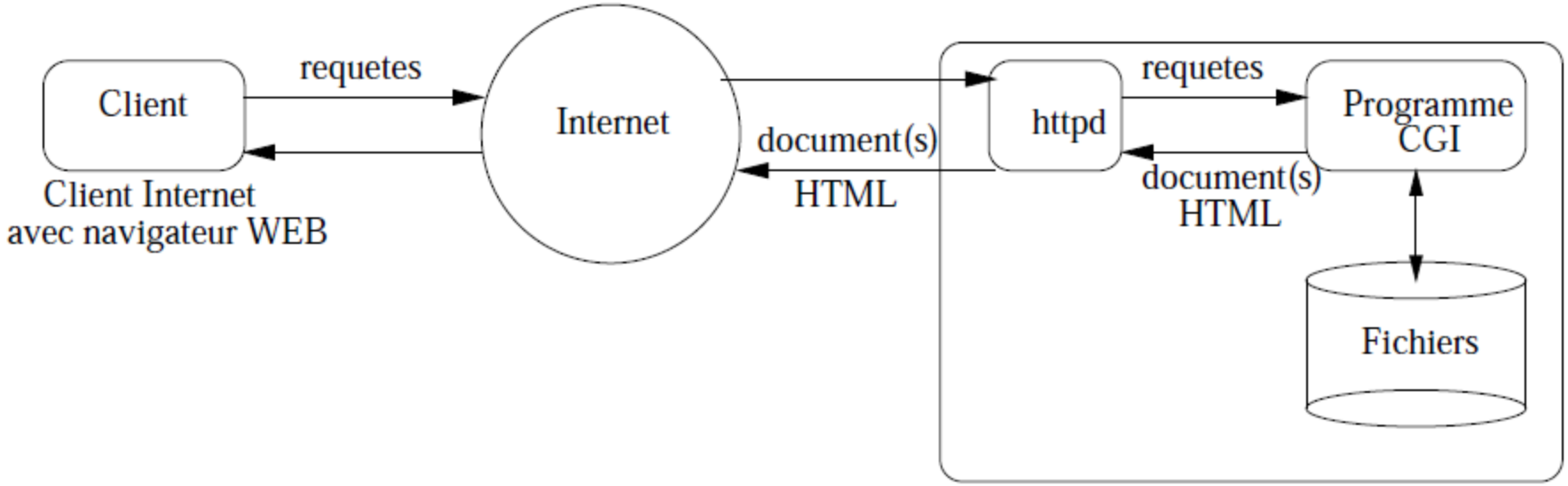
- Suite à cette requête,
  - le browser va ouvrir un socket TCP (structure de communication entre serveurs, avec un numéro d'IP) vers le serveur dont l'adresse est donnée par:
    - host : www.iro.umontreal.ca
    - port : 80 (port par défaut du serveur HTTP)
  - et écrire la requête sur le socket
- Ensuite,
  - Le browser va attendre la réponse du serveur
  - La réponse comprendra deux parties :
    1. un HEADER
    2. données du fichier proprement dit, avec indication du contenu par le paramètre : Content-Type
  - Exemple :
    - Content-Type : text/html; charset=ISO-8859-1

- Le rôle du Content-Type
  - La manière dont le browser va présenter le contenu du fichier retourné dépend du "Content-Type"
  - Pour du HTML avec "Content-type: text/html"
    - Le fichier devra être formaté
  - Pour du texte pur (text/html)
    - le fichier sera affiche tel quel
  - Pour des images ou du son
    - le browser devra faire appel à divers PLUG-IN spécialisés chacun dans un type de fichier

- Le serveur HTTP
  - C'est un simple programme qui ouvre un "Server Socket" sur le port 80 et attend des connexions de la part des clients
  - A chaque ouverture de socket, un serveur crée un Thread (un processus, un exécutable..;) distinct qui lit la commande et en extrait le 'path' pour trouver le nom du fichier à retourner
  - Dans le cas normal (fichier existant et accessible en lecture), le serveur répond en écrivant sur le socket le HEADER suivi du contenu du fichier; et finalement, il ferme le socket

Vers une architecture orientée davantage vers les services

- Le protocole CGI (Common Gateway Interface)
  - On a fait un ajout minime à HTTP pour permettre l'implantation d'une relation client/serveur plus élaborée où les requêtes comprennent des paramètres et les fichiers retournés sont générés dynamiquement suite à l'exécution de programmes

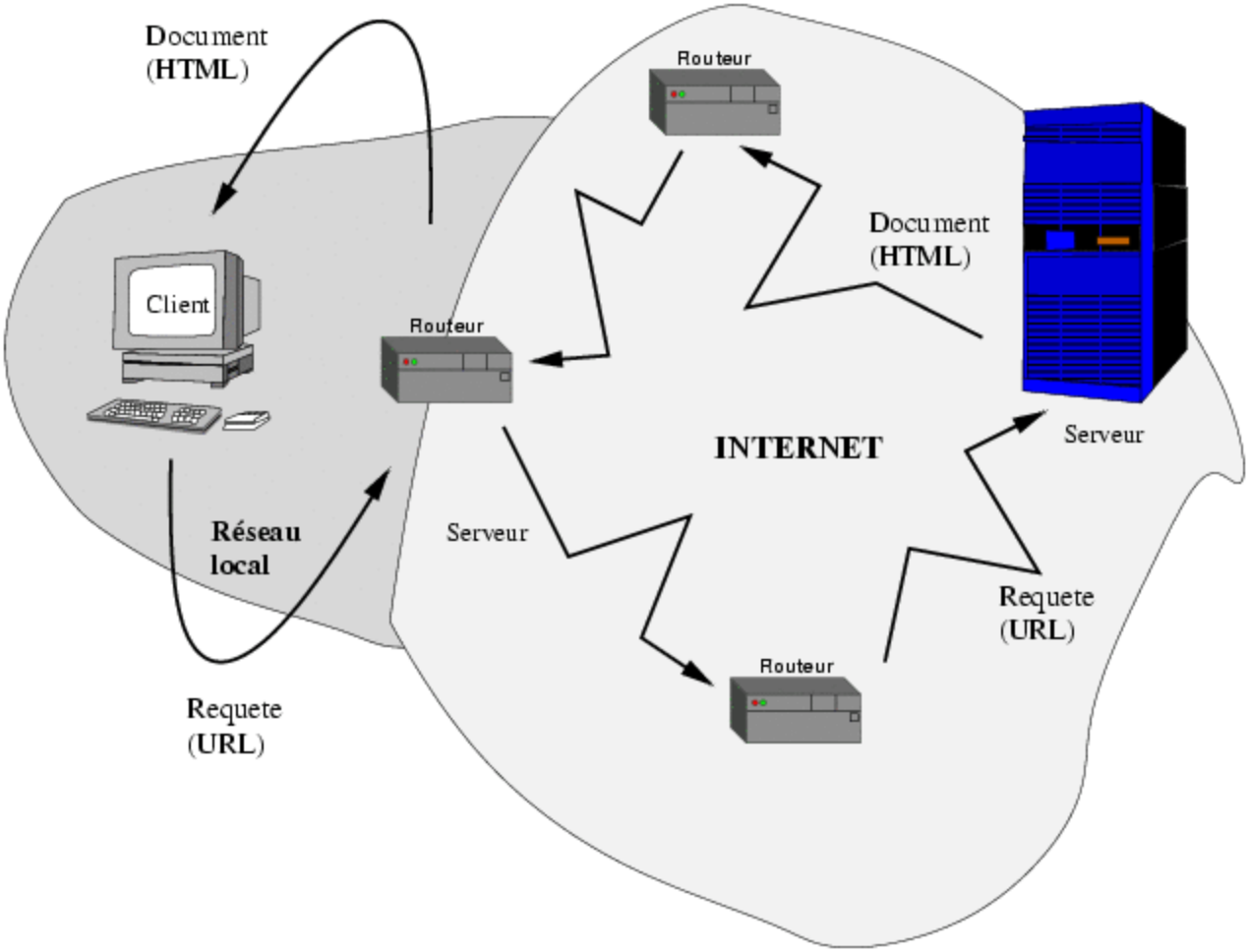


- Comment cela se passe avec le CGI ?
  - On peut indiquer que le fichier spécifié dans la requête GET ne doit pas être retourne tel quel;
    - mais plutôt que
  - c'est un PROGRAMME qui doit être exécuté et que c'est son OUTPUT qui doit être retourné
  - On peut aussi donner des paramètres ou des données qui seront passés au programme
  - Exemple

    http://www-perso.iro.umontreal.ca/~vaucher/SCRIPTS/echo.cgi?a=1&b=tutu
    - Les paramètres sont après le ? et le &
  - Il y a donc une réelle collaboration entre le HTTPD et le CGI
    - pour se passer les paramètres en respectant les conventions

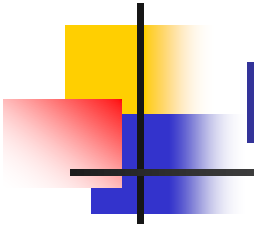
Le réseau

- L'architecture (on reviendra dessus plus tard)



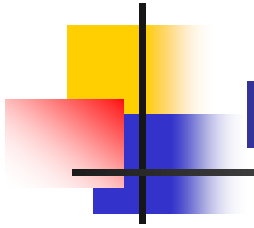
La technologie dédiée à la construction d'applications Web

- AJAX (***Asynchronous Javascript and XML)***
  - Manière de construire des applications Web et des sites web dynamiques basés sur diverses technologies Web ajoutées aux navigateurs
    - JavaScript, CSS, XML, DOM et XMLHttpRequest
  - Les applications Ajax fonctionnent sur tous les navigateurs Web qui mettent en œuvre les technologies décrites précédemment



Intérêt de la technologie Ajax

- Une **application web** permet à l'utilisateur de naviguer dans son application et de faire des mouvements :
  - envoi de formulaires, navigation, etc.
- Dans une application qui n'utilise pas la technologie Ajax
  - A chaque mouvement, une requête est envoyée au serveur HTTP, qui l'interprète et renvoie une nouvelle page à l'utilisateur (ces requêtes sont dites «synchrones »)
  - Ce mouvement conduit au rechargement intégral de la page, d'où rallongement du temps de réponse
- Une application qui utilise la technologie Ajax
  - envoie des requêtes au serveur HTTP pour récupérer uniquement les données nécessaires (ces requêtes sont dites « asynchrones »)



Intérêt de la technologie Ajax (suite)

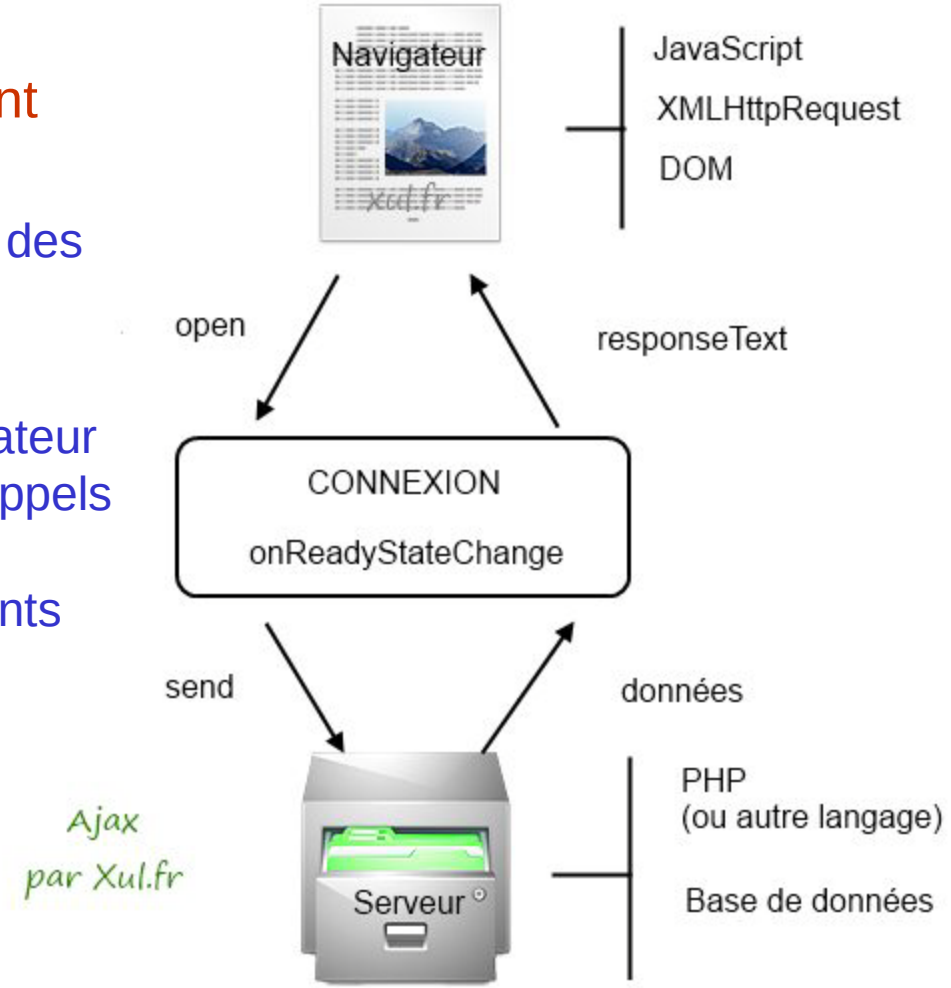
- C'est ce qui permet d'être réactif et donc rapide
  1. On utilise les feuilles de style (CSS3) pour afficher les nouvelles informations reçues par le serveur

     ➔ Des sélecteurs sont mis en alerte pour afficher ce qu'il faut
  2. On utilise JavaScript côté client pour interpréter la réponse et effectuer les traitements nécessaires

     ➔ De ce fait, l'application est à chaque fois focalisée sur ce qui est nouveau à traiter

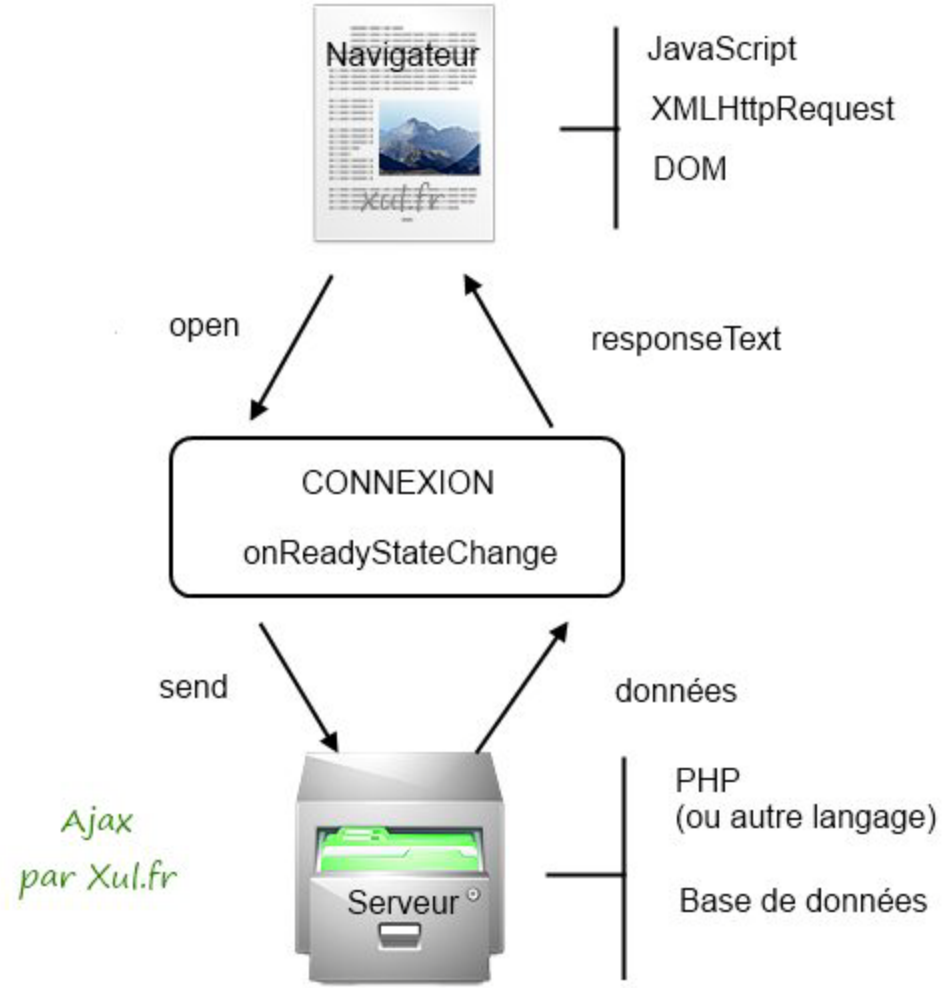
Comment cela fonctionne ? (1)

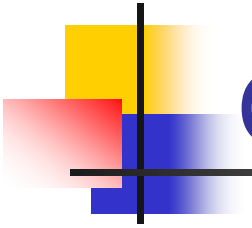
- Ajax utilise un modèle de programmation comprenant
  - Une présentation
    - des formulaires avec des boutons d'interaction
  - Des événements
    - des actions de l'utilisateur qui provoquent des appels à des fonctions JS associées aux éléments de la page



Comment cela fonctionne ? (2)

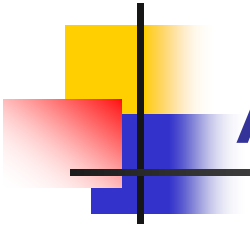
- Les fonctions JavaScript
  - identifient les éléments de la page grâce au DOM
  - communiquent avec le serveur par l'objet XMLHttpRequest





Comment cela fonctionne ? (3)

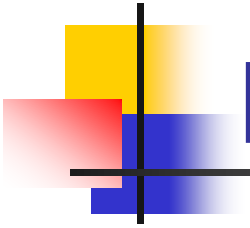
- Pour recueillir des informations sur le serveur, XMLHttpRequest dispose de deux méthodes :
  - **open** : établit une connexion
  - **send** : envoie une requête au serveur
- Les données fournies par le serveur seront récupérées dans les champs de l'objet XMLHttpRequest :
  - **responseXml** pour un fichier XML ou
  - **responseText** pour un fichier de texte brut



Applications Web

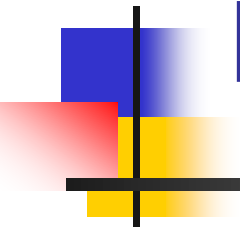
- Rôle de XML
  - Pour transférer les données entre le JavaScript et PHP par exemple, on a besoin d'adopter des conventions, une sorte de protocole respecté des deux côtés, pour que JavaScript puisse interpréter ce qui est retourné par le serveur

    ➔ c'est XML qui a d'abord été utilisé pour ça
  - Mais aujourd'hui il existe d'autres formats plus simples à utiliser, notamment JSON, qui peuvent remplacer XML



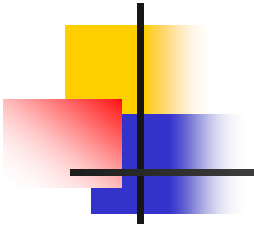
Le Web

- Documents web
  - Les documents échangés sur le Web peuvent être de types très divers
  - Le principal type de ressource est le document hypertexte, un texte dans lequel certains mots, ou groupes de mots, sont des liens, ou ancres, donnant accès à d'autres documents
  - Le langage qui permet de spécifier des documents hypertextes, et donc de fait le principal langage du Web, est HTML



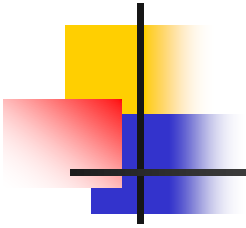
Langages de structuration

HTML4 …HTML5



HTML

- Définition
  - HTML est un langage de **description** de contenu
  - Il est le seul qui est interprété par un navigateur Web standard pour produire de l'affichage
  - Il existe d'autres standards plus riches mais marginaux (XUL par exemple)
  - Le HTML décrit le document et ses constituants



HTML

- Principes du langage :
  - Un langage à balises :
    - Balises à dimension 0 : **
, <img>**
    - Balises à dimension 1 : **<a>, **
  - Les éléments ainsi formés peuvent se contenir les uns les autres :

```
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

  - Des attributs plus ou moins obligatoires pour préciser des propriétés des éléments :

```
<a href="http://...">un lien</a>
```

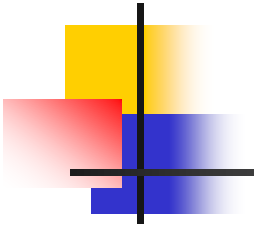
HTML

- Les balises et le texte
  - Le texte du document est ce qui est « entre » ou autour des balises
  - Les balises servent à déterminer la « fonction » du texte dans une structure
  - Exemple

    ce texte est un lien
  - Les balises peuvent simplement altérer la « forme »
  - Exemple

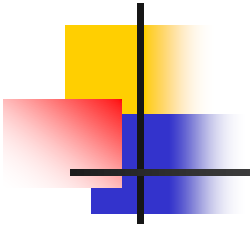
    en gras, *en italiques*
  - Les primitives premières du HTML ont une sémantique « documentaire »

    <p>, <h1>, <cite>, <address>, <blockquote>, <ul>, <ol>,



HTML

- L'Hypermédia, l'agencement et les échanges
  - Un deuxième jeu important du HTML concerne l'établissement des hyperliens :
    - <a>, <link>, <base>
  - Un troisième jeu important du HTML concerne la gestion de l'organisation visuelle (l'agencement) :
    - <table>, <thead>, <tbody>, <tr>, <td>, <colgroup>, <cols>, , , <iframe>, <frameset>, <frame>
  - Un dernier jeu du HTML concerne la récupération de données à travers des formulaires :
    - <form>, <input>, <select>, <textarea>, <label>



HTML

- La constitution du document
  - Un document HTML est un document HTML :
    - <html>...</html>
  - ou pas ? :
    - ** ... **
  - Un document est de l'information + de la méta-information :
    - **<html><head>...meta...</head><body>…information...</body></html>**
    - Le HEAD : sert à enregistrer des informations complémentaires (mots-clefs, feuilles de styles applicables, des scripts à utiliser, des chargements annexes, etc.)
    - Le BODY : contient la structure de la partie « visible » du document

HTML

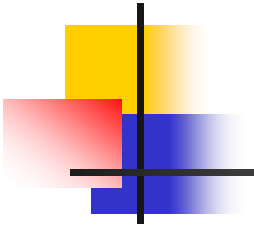
- Les deux écritures
  - La forme sérialisée :
    - **<html><head>...meta...</head><body>...information...</body></html>**
  - La forme « structurée » :

```
<html>
    <head>
      ...meta…
    </head>
    <body>
      ...information…
    </body>
</html>
```

HTML

- La constitution d'un document linéaire
  - Un document textuel se construit selon un principe documentaire historique (des titres, des sous-titres et des paragraphes) :

```
...
<body>
<h1>Titre 1 : Première partie</h1>
<h2>Titre 2</h2>
<p> contenu </p>
<h2>Titre 2</h2>
<p> contenu </p>
<h1>Titre 1 : Deuxième partie</h1>
...
```



HTML

- La constitution d'un document agencé
  - Un document agencé utilise des tables ou des blocs
    - La stratégie par table :
      - Convient bien aux interfaces « formelles » et structurées (applications informatiques, applications de gestion, données structurées, données tabulaires).
      - A des inconvénients : « Accessibilité », coût de calcul de l'agencement quand trop de tables sont imbriquées
    - La stratégie par bloc :
      - Est issue du monde de la PAO. Elle convient aux informations « scénarisées »
      - A des inconvénients : difficulté de lecture de la structure, difficulté de maîtrise du calage graphique, impose la connaissance des « feuilles de style CSS »

- L’agencement en tables

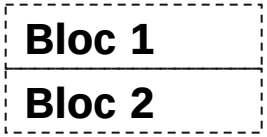
```
...
<table>
  <thead>
    <tr>
      <th>Titre 1</th>
      <th>Titre 2</th>
    </tr>
  </thead>
...
```

```
...
<tbody>
    <tr>
      <td>
      </td>
      <td>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        Cellule fusionnée
      </td>
    </tr>
  </thead>
</table>
...
```

TITRE1	TITRE2
Cellule fusionnée	

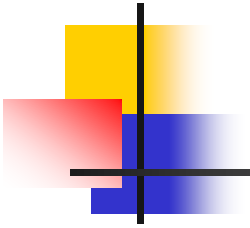
L’agencement en blocs

```
...
<div>
<p> bloc 1 </p>
</div>
...
<div>
<span>Bloc 2</span>
</div>
```



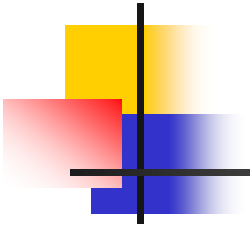
- Les DIV représentent des éléments « bloc »
- Les SPAN représentent une frontière identifiable autour d’un texte « en ligne »
- Les DIV doivent être agencés à l’aide de primitives de feuille de style :

```
<div style="width: 100px;
  height: 80px; position:
  relative; left: 30px;">
</div>
```



HTML

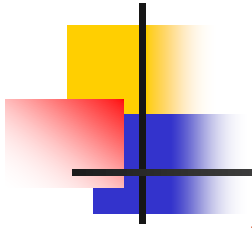
- L’échange de données avec le serveur
  - En réception : réponse à une requête (mode habituel)
  - En émission : envoi de données par
    - l’URL : c'est le mode « GET » dans lequel on passe des variables
    - un formulaire : en « GET » mais plus généralement en « POST », permet de passer des variables issues de saisies d’utilisateur



HTML

- Le formulaire HTML
  - Un « élément-bloc » encadrant des champs d'entrée de donnée :

```
...
<form name="formulaire" method="POST"
  action="view.php">
   ... Champs de saisie ...
   <input type="submit" name="go_btn"
  value="zyva!!">
</form>
...
```



HTML

- Les éléments de formulaire :
  - Champ de texte :

```
<input type="text" name="nom"
  value="valeur initiale">
```

  - Case à cocher :

```
<input type="checkbox" name="nom"
  value="valeur_envoyee" [CHECKED] >
```

  - Boutons radio :

```
<input type="radio" name="nom_de_groupe"
  value="valeur_envoyee" [CHECKED] >
```

  - Plusieurs radio portant le même nom de groupe fonctionnent en exclusion

HTML

- Les éléments de formulaire (suite) :
  - Liste de sélection :

```
<select name="nom" [MULTIPLE] >
<option value="code">libellé</option>
<option value="code" [SELECTED] >libellé</option>
</select>
```

  - Zones de texte :

```
<textarea name="nom" cols="30" rows="8">Contenu
  initial</textarea>
```

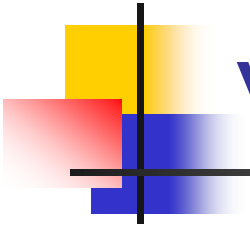
  - Champ caché :

```
<input type="hidden" name="nom" value="valeur">
```

HTML

- La présentation des formulaires
  - Elle combine en général les éléments de formulaire et les tables :

```
<form name="formulaire" action="view.php">
<table width="100%">
    <tr>
        <td align="right">Libellé</td>
    </tr>
    <tr>
        <td align="left">
            <input type="text" name="nom"
  size="30">
        </td>
    </tr>
</table>
</form>
```

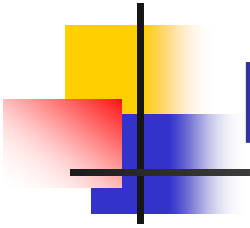


Vers HTML5

- HTML4
  - Plusieurs tentatives d'amélioration : XHTML… abandonnées
- HTML5
  - Définir un langage unique qui peut être écrit dans la syntaxe HTML et XML
  - Définir des modèles de traitement permettant de favoriser des implémentations interopérables
  - Améliorer le balisage des documents
  - Introduire des APIs pour créer des applications Internet riches (AIR)

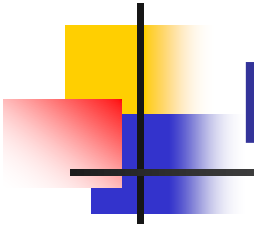
Nouveautés

- Quoi de neuf chez les balises ?
  - **Balises visant à structurer le contenu**
    - Utiles pour la navigation et le référencement
    - Facilitant le travail de l'analyseur pour repérer les éléments
      - Barre de navigation, menu, etc.
  - **Balises de media**
    - Introduction de <video> et <audio> pour le multimédia
      - sont une énorme claque à Flash et Silverlight
    - Désormais, c'est au navigateur de gérer les éléments multimédia et de les lire, ce qui optimise l'utilisation des ressources (plus besoin de plugin)
      - permet de contrôler la lecture par du JavaScript, et plein d'autres interactivités
    - Peuvent être mises en forme avec des CSS ➔ devient simple de faire un lecteur avec un peu de CSS et de JS !



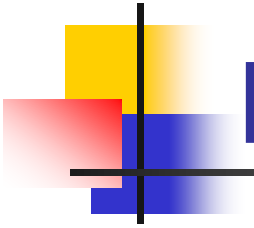
Nouveautés

- **Balises sémantiques**
  - <mark> permet de « marquer » un texte, comme par exemple des mots recherchés
  - <time> permet de spécifier un temps donné au navigateur
  - <meter> spécifie une valeur numérique de mesure quelconque
  - <progress> spécifie une progression
- **Balises d'interactivité**
  - <detail> permet de spécifier un détail sur une partie de texte comme une aide ou une définition mais qui ne sera affiché qu'au survol ou sur une autre action
  - <menu> et <command> : menus interactifs



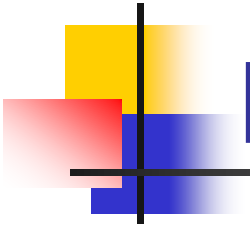
Nouveautés

- **Balise de modélisation de données**
  - <datagrid> permet d'insérer une représentation de données en modèle de grille (arbre, tableaux, listes, etc.)
    - Cette balise englobe un élément de liste (<table>, <select>, /<ul>, etc.) dont elle récupère les données initiales
  - Il est alors possible en utilisant les méthodes fournies par HTMLDataGridElement de rendre dynamiques ces données, par exemple par des appels AJAX, lors de tri, de filtres, etc.



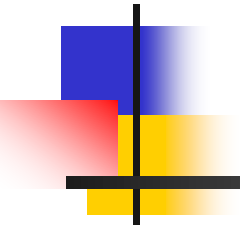
Nouveautés

- **Une balise pour le dessin vectoriel**
  - <canvas> : éléments graphiques 2D en JS à la volée
- **Un stockage DOM**
  - DOM a été étendu pour ajouter une méthode de stockage d'informations permettant d'enregistrer des données de manière permanente dans le navigateur
- **La géolocalisation**
  - API spécifique permettant aux pages web d'interroger le navigateur sur la position de l'utilisateur
- **Des Workers**
  - Tâches de fond qui vont s'exécuter en parallèle du programme JavaScript principal ➔ mieux gérer la surcharge de JavaScript ➔ plus de problème de ralentissement à cause d'un JS trop gourmand



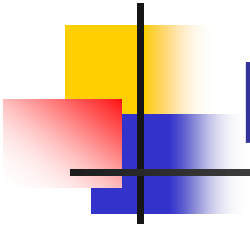
Nouveautés

- La Concurrence
  - HTML 5 va surtout concurrencer les plugins de navigateurs, notamment Flash et Silverlight
    - En fait, l'intérêt des plugins est avant tout de palier un manque, surtout au niveau de la gestion du multimédia dans le web (la vidéo en premier)
    - Le HTML 5 permet de gérer ça de manière plus légère, plus sémantique et plus accessible
  - En parallèle au développement de HTML 5, il y a aussi celui du XHTML 2… à suivre



HTML5

Quelques éléments du langage

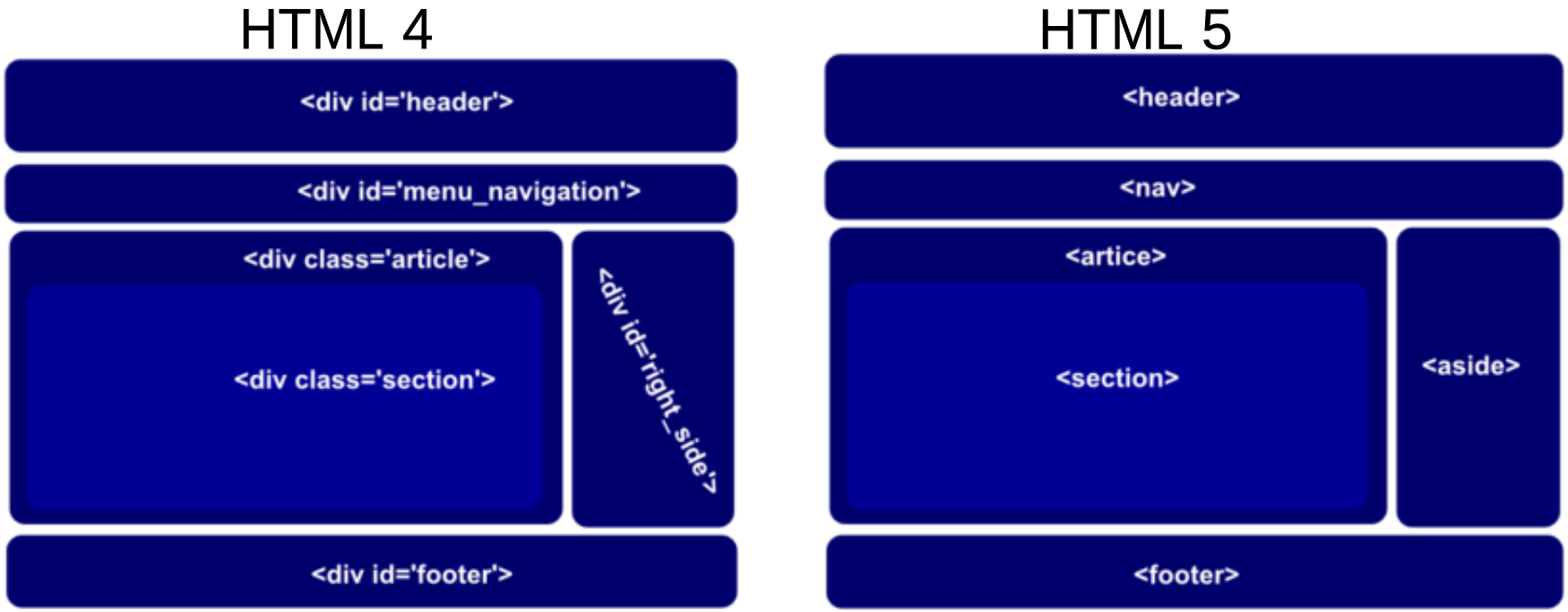


HTML5

- Structure de la page
  - La page représente l'identité ou la signature d'un site
  - Son architecture est importante aussi bien pour l'internaute pour s'y retrouver que pour le navigateur pour localiser l'information
  - Pour le Web2.0, cette architecture est essentielle
  - Plusieurs éléments doivent apparaître comme
    - L'entête pour préciser le thème
    - Le menu pour la navigation
    - Les articles ou billets (pour un blog)
    - Le pied de page contenant quelques données sur l'identité du Webmaster, la date de création ou de mise à jour, etc.

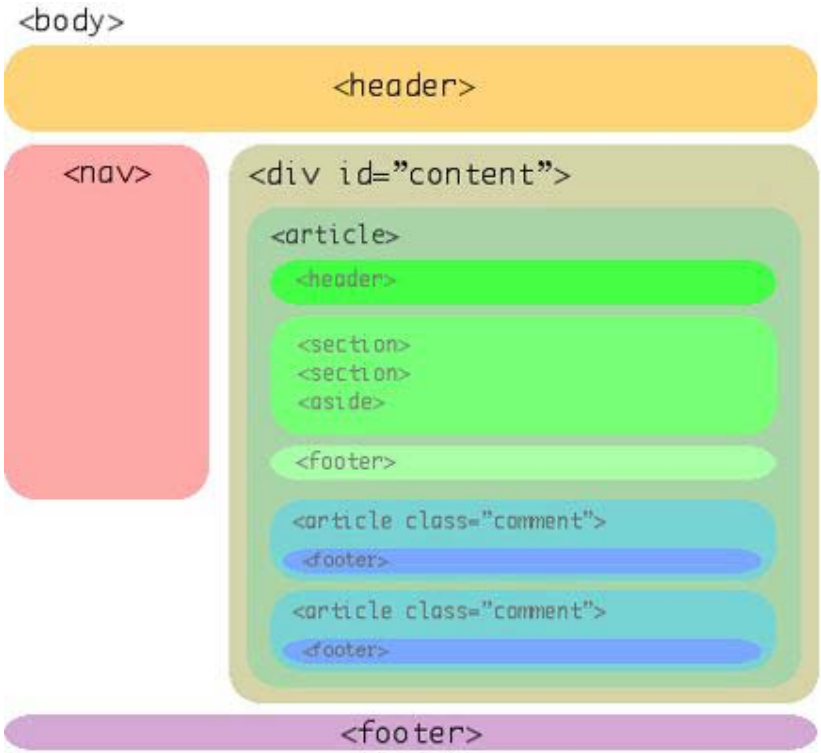
HTML5

- Structure de la page
  - Avant, tout était structuré de manière identique par des <DIV>
  - HTML5 introduit de nouvelles balises dédiées à la description de la sémantique du contenu : header, nav, article, section, aside, et footer

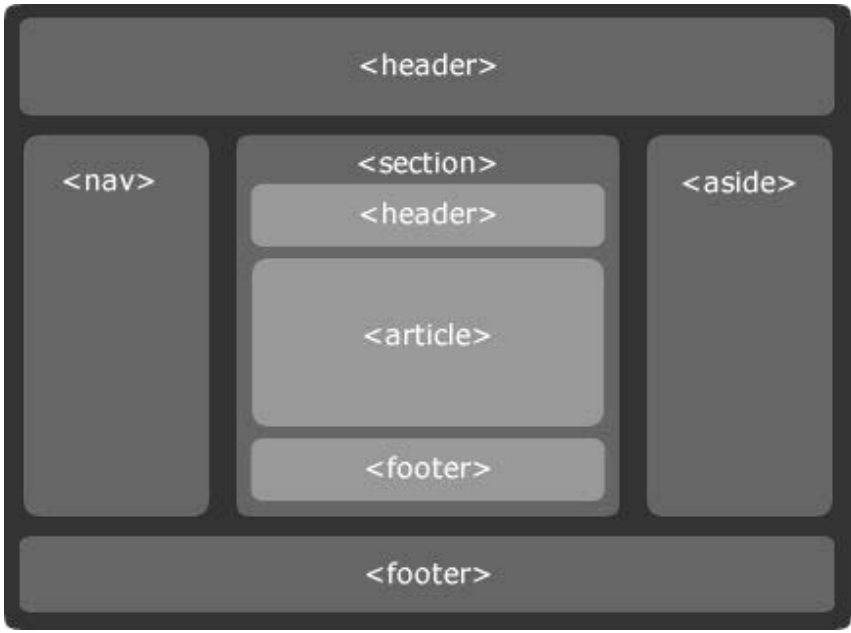


HTML5

- D’autres exemples de structures



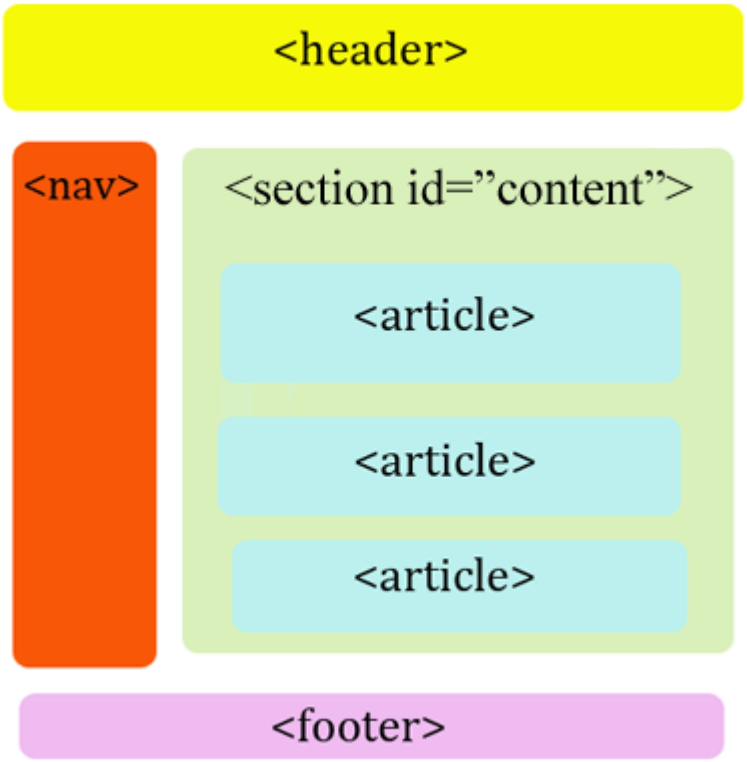
Romy tetue



Smashing Magazine

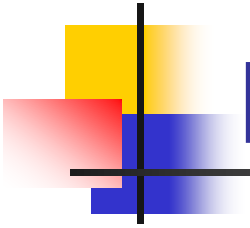
HTML5

- D’autres exemples de structures



HTML5 Doctor

Structure d’un billet de blog



HTML5

- Structure du fichier HTML5
  - Le fichier HTML5 doit démarrer avec cette instruction :

    <!doctype html>
  - montrant qu'il y a qu'un seul doctype
  - Un DOCTYPE (contraction pour "Document Type Declaration") informe le validateur de la version de HTML que vous utilisez, et doit apparaître en première position dans chaque page web
  - est essentiel pour que le document web s'affiche et fonctionne correctement dans des navigateurs conformes aux normes, comme Mozilla, IE5/Mac, et IE6/Win
  - Attention, ne pas mettre d'URI en fin de balise comme en XHTML et HTML 4

HTML 5

Retour sur les éléments de structure

- <header>
  - Selon la spécification HTML5, l'élément header représente :
    - Un groupe d'aide de navigation ou d'introduction
  - Un élément header contient de façon générale les headings (les éléments h1 à h6 ou l'élément hgroup)
    - Il peut aussi contenir d'autres éléments, comme une table de matières, un formulaire de recherche, ou des logos
- <nav>
  - Permet de regrouper une liste de liens de navigation
  - Convient également pour la navigation dans le site

Exemple : structure-nav.html

```
<body>
   <header>
      <h1>Wake up sheeple!</h1>
      <p><a href="news.html">News</a> -
      <a href="blog.html">Blog</a> -
      <a href="forums.html">Forums</a></p>
      <p>Last Modified: <time>2009-04-01</time></p>
      <nav>
         <h1>Navigation</h1>
         <ul>
            <li><a href="articles.html">Index of al articles </a></li>
            <li><a href="today.html">Things sheeple need to
               wake up for today</a></li>
            <li><a href="successes.html">Sheeple we have
               managed to wake</a></li>
            </ul>
      </nav>
   </header>
   <article>
      <p>...page content would be here...</p>
   </article>
   <footer>
      <p>Copyright © 2006 The Example Company</p>
      <p><a href="about.html">About</a> -
      <a href="policy.html">Privacy Policy</a> -
      <a href="contact.html">Contact Us</a></p>
   </footer>
</body>
```

HTML 5

La structure

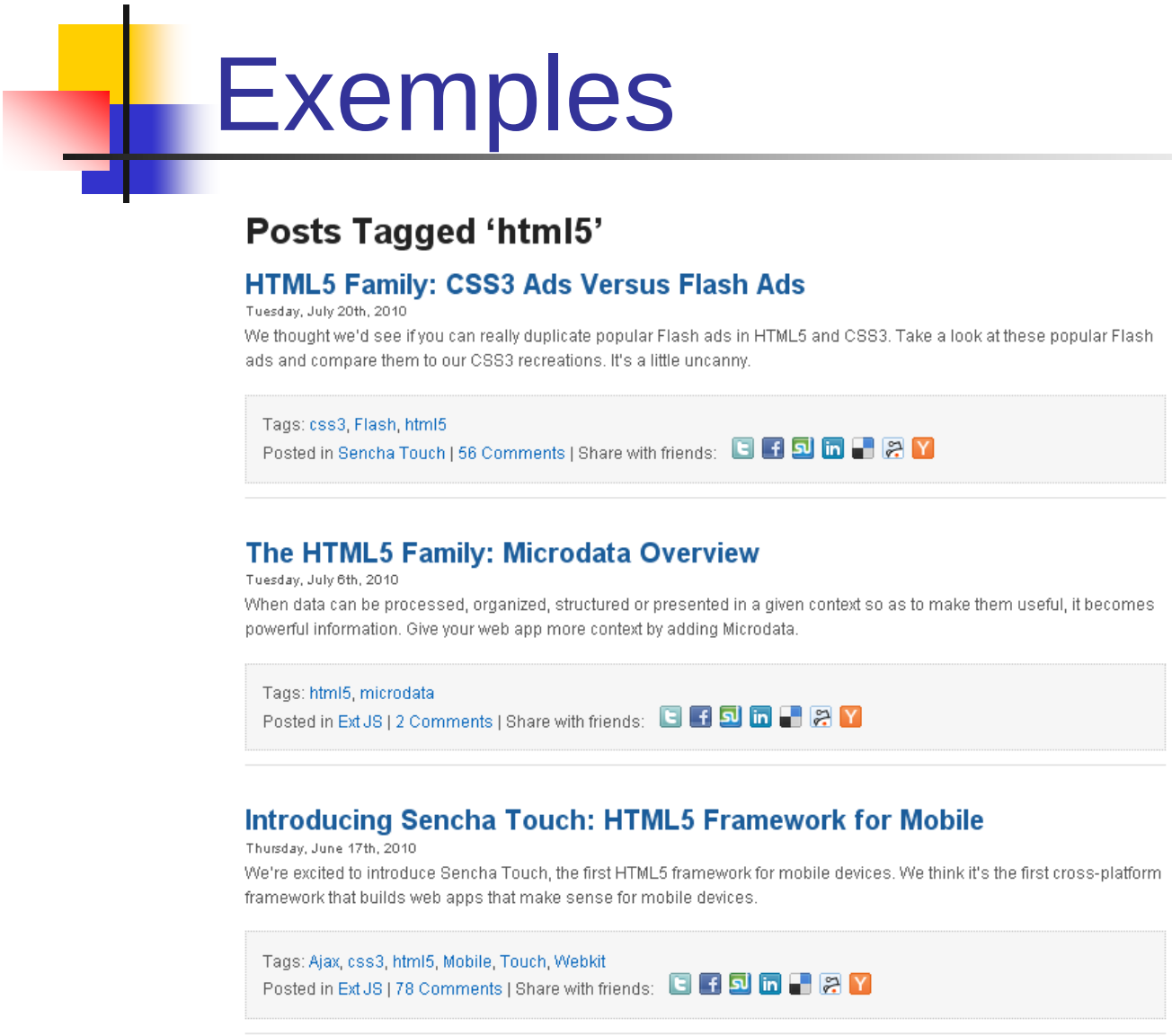
- <footer> : structure-footer.html
  - Représente le bas de la section à laquelle il s'applique
  - Un *pied* contient typiquement une information sur sa section comme son auteur, des liens vers des documents liés, les données de copyright et autres données du même type
- <aside>
  - représente une note, une astuce, un encadré, un extrait, une remarque incidente, ou quelque chose qui est juste en dehors du courant principal du récit
  - On peut utiliser <aside> dans deux contextes
    - le premier est un article
    - et le second, le site Web lui-même

HTML 5

La structure

- <section>
  - Représente une partie générique d'un document ou d'une application, comme un chapitre par exemple
- <article>
  - Représente une section indépendante d'un document, d'une page ou d'un site
  - Convient pour du contenu comme des *nouvelles* ou des articles de blog, des messages de forum ou des commentaires individuels

Exemples



Posts Tagged ‘html5’

HTML5 Family: CSS3 Ads Versus Flash Ads

Tuesday, July 20th, 2010

We thought we'd see if you can really duplicate popular Flash ads in HTML5 and CSS3. Take a look at these popular Flash ads and compare them to our CSS3 recreations. It's a little uncanny.

Tags: css3, Flash, html5
Posted in Sencha Touch | 56 Comments | Share with friends:

The HTML5 Family: Microdata Overview

Tuesday, July 6th, 2010

When data can be processed, organized, structured or presented in a given context so as to make them useful, it becomes powerful information. Give your web app more context by adding Microdata.

Tags: html5, microdata
Posted in Ext JS | 2 Comments | Share with friends:

Introducing Sencha Touch: HTML5 Framework for Mobile

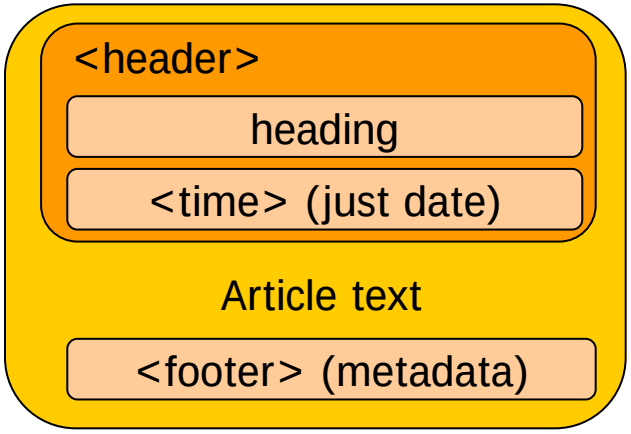
Thursday, June 17th, 2010

We're excited to introduce Sencha Touch, the first HTML5 framework for mobile devices. We think it's the first cross-platform framework that builds web apps that make sense for mobile devices.

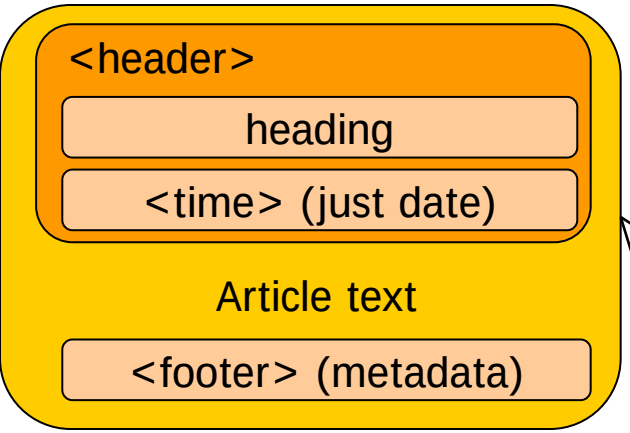
Tags: Ajax, css3, html5, Mobile, Touch, Webkit
Posted in Ext JS | 78 Comments | Share with friends:

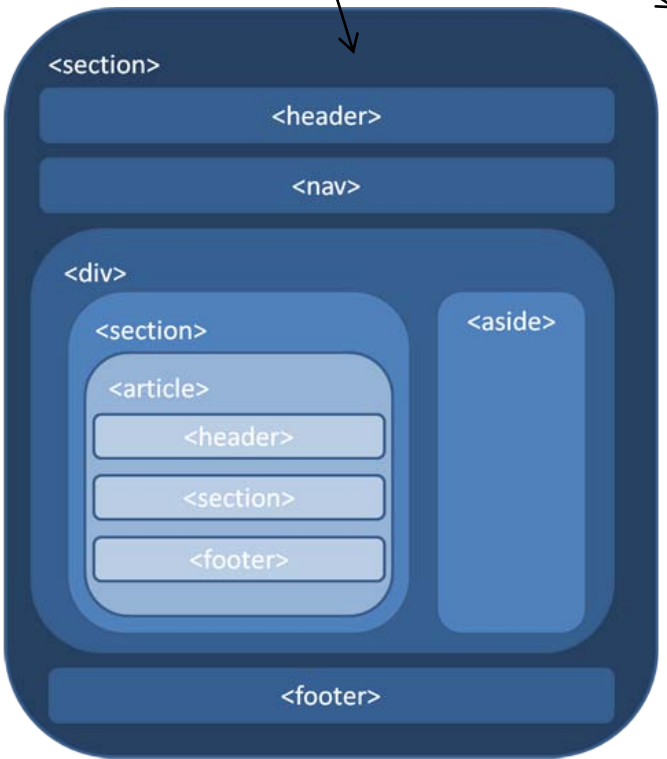
La structure d'article

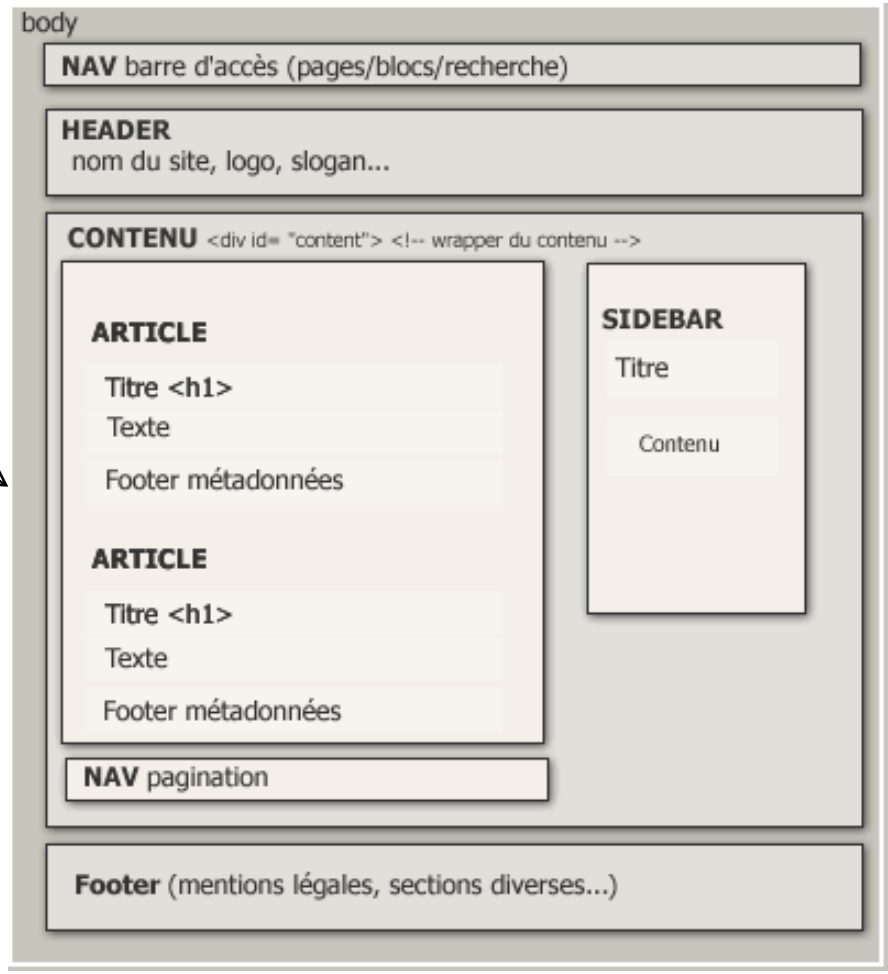
- Spécification
  - Un article doit être structuré de la même manière quelque soit le blog
  - HTML4 le représente plus ou moins bien avec des ce qui n'assure pas toujours sa visibilité : titrage, sections…
  - HTML5 harmonise sa structure et renforce sa sémantique à l'aide de 4 nouvelles balises en plus du <Hx>
    - <article>,
    - <header>,
    - <footer>,
    - <time>



Contextes d'utilisation







La structure d'article

- Ce qui change avec HTML5
  - La balise a un sens
  - De plus, ce sens peut varier en fonction des éléments parents :
  - <article> :
    - représente une section isolable du site, qui peut s'auto-suffire en dehors du contexte de la page (dans un lecteur RSS par exemple)
    - mais, enfant d'un autre <article>,

      ➔ Il représente un commentaire sur cet article

La structure d'article

- <header> :
  - n'est pas exclusivement réservé à l'en-tête du site !
  - à l'intérieur d'un <article>,
    - il sert à repérer ses **titres et métadonnées** comme la date de publication et le nom de l'auteur
- <footer> :
  - lui non plus n'est pas réservé au pied de page du site
  - placé dans un <article>, il :
    - marque les informations relatives à l'article mais non indispensables, telles que tags et catégories

La structure d'article

- <time>
  - Sert à indiquer une date : a plusieurs attributs
    - datetime indique une date à un format standardisé
    - Si on lui rajoute l'attribut pubdate, on signifie au parseur la **date de publication** de l'<article> parent
  - En d'autres termes
    - pubdate est un booléen qui indique si la date indiquée par <time> est la date de publication d'un article ou de tout le contenu du <body>

- Exemple de pubdate : time-pubdate.html

```
<article>
    <header>
        <h1>Viens à ma fête le <time datetime=2010-12-01>1 Décembre</time>
        </h1>
        <p>Publié le <time datetime=2010-06-20 pubdate>20 juin 2010</time> </p>
    </header>
    <p>Je vais organiser une fête à la discothèque... </p>
</article>
```

  - Comme vous le voyez, il y a 2 dates, pubdate permet de lever l'ambiguïté

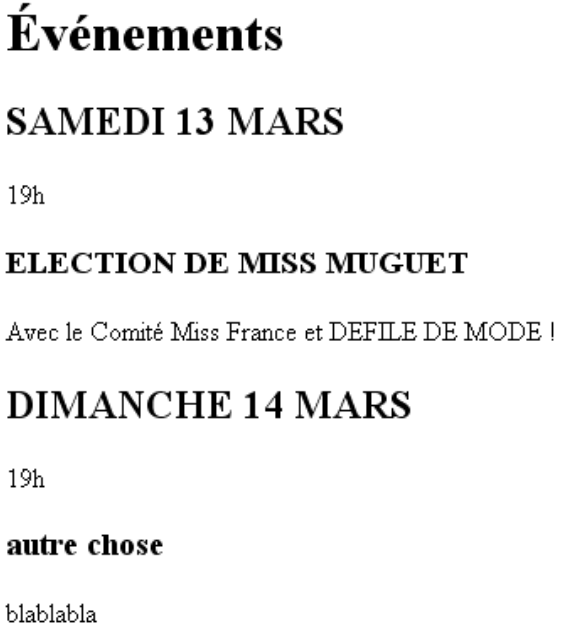
Viens à ma fête le 1 Décembre

Publié le 20 juin 2010

Je vais organiser une fête à la discothèque...

- Exemple avec des <header > sémantiquement différents : time1.html

```
<header> <h1>Événements</h1> </header>
<h2><time datetime="2010-04-13">SAMEDI 13 MARS</time></h2>
<article>
   <header>
     <time datetime="2010-04-13T00:19:00+02:00">19h
     </time>
     <h3>ELECTION DE MISS MUGUET</h3>
   </header>
   <p>Avec le Comité Miss France ….!</p>
</article>
<h2><time datetime="2010-04-13">DIMANCHE
      14 MARS</time>
</h2>
<article>
   <header>
        <time datetime="2010-04-13T00:19:00+02:00">
        <h3>autre chose</h3>
   </header>
    <p>blablabla</p>
</article>
```



Événements

SAMEDI 13 MARS

19h

ELECTION DE MISS MUGUET

Avec le Comité Miss France et DEFILE DE MODE !

DIMANCHE 14 MARS

19h

autre chose

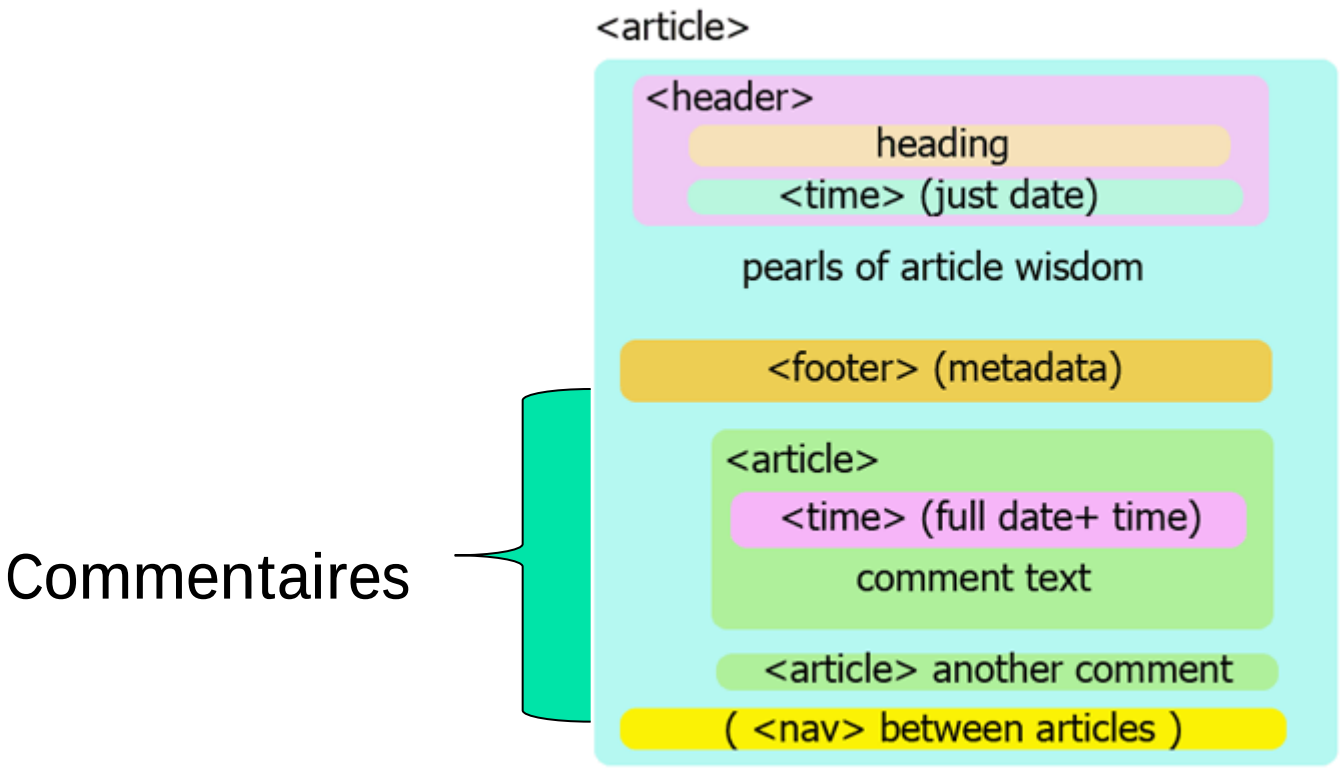
blablabla

La structure d'article

- Chaque article dans une page peut avoir ses propres <header> et <footer>
- Ces <header> et <footer> s'ajoutent à ceux principaux de la page
- On peut les cibler au niveau de la CSS par des sélecteurs adéquat :
  - body>header
  - body>footer
- et
  - article>header
  - article>footer
- La spécification autorise plusieurs footers pour un même élément

La structure d'article

- Ajout de messages et de commentaires aux articles
  - Un article peut contenir en plus du corps, des messages et des commentaires ➔ sous d'autres balises <article>



- Exemple : article-comment.html

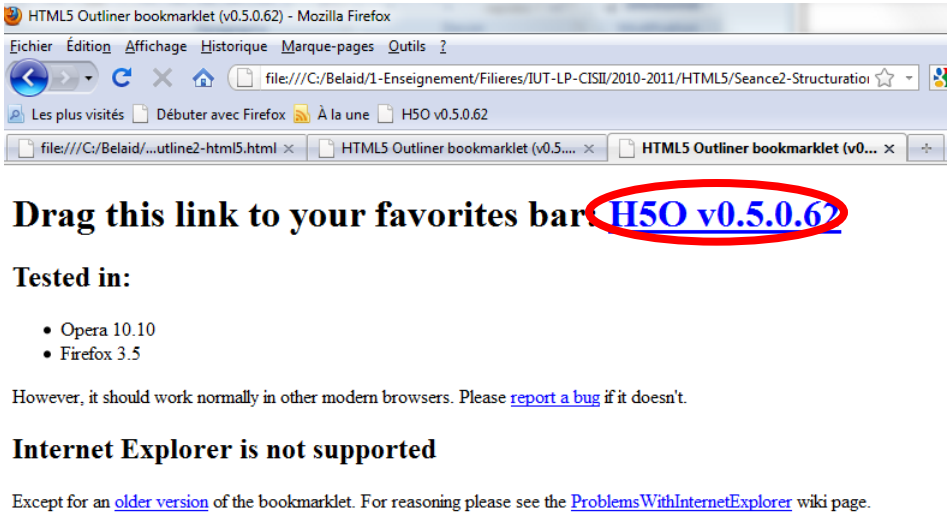
```
<article>  <!-- Début du billet-->
  <header>
    <h1>Viens à la fête le <time datetime=2010-12-01>1er décembre</time></h1>
    <p>Publié le <time datetime=2010-06-20 pubdate>20 juin 2010</time> </p>
  </header>
     <p>Je suis à la fête de ...</p> <!-- Corps de l'article -->
  <footer>Publié par Bruce...</footer>
<article> <!-- Début commentaire -->
<header>
    <p> Commentaire de : <a href="http://remysharp.com">
    Remy Sharp</a> à <time datetime="2010-05-01
     T08:45z">8.45 le 1er mai 2010</time> </p>
</header>
<p>je serai là-bas...</p>
</article> <!-- Fin du commentaire-->
<article> <!-- Début commentaire -->
<header>
    Commentaire de : <a href="http://splintered.co.uk">Patrick Lauke</a> à
    <time datetime="2010-05-02T10:45z">10.45 le 2 mai 2010</time>
</header>
<p>Pardon mate...</p>
</article> <!-- Fin du commentaire-->
</article> <!-- Fin du billet-->
```

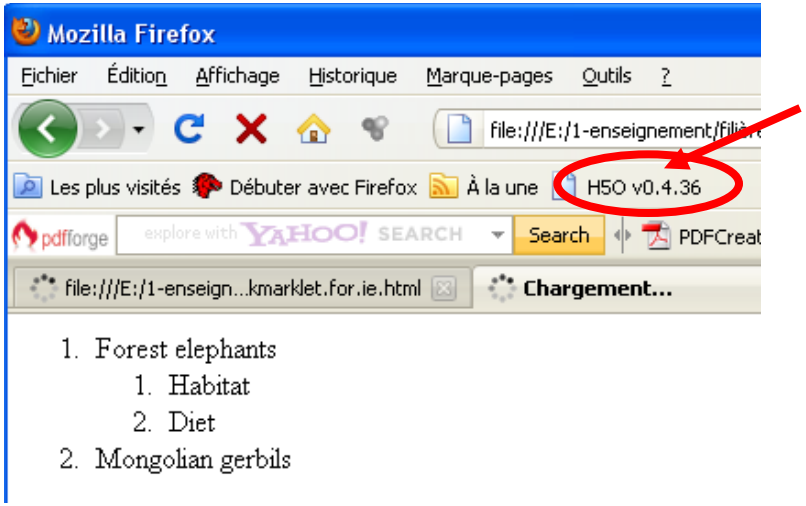
Table des matières

en vue d'étudier la visibilité du sectionnement

- Construction
  - Il est possible de construire la table des matières d'un fichier HTML
    - ➔ **La table des matières sert à renseigner sur la visibilité de la structure du contenu**
  - Obtention :
    - Téléchargez les fichiers suivants :
      - outliner.0.5.0.62.html
      - outliner.0.5.0.62.js
    - L'exécution de outliner.0.5.0.62.html permet d'avoir un lien que l'on met dans la ligne des favoris de son navigateur
      - ➔ voir page suivante
    - Ensuite, il suffit de charger le fichier html dans le navigateur, puis de cliquer sur ce lien, ce qui fournit la table des matières

Autre type de contenu :
Tables des matières





Exécution de : outliner.0.5.0.62.html

Déplacement de : H50 v0.5.0.62

Autre type de contenu :
Les sections

- HTML5 règle les problèmes
  1. L'absence de l'attribut **class** dans crée des ambiguïtés et ne permet pas de savoir ce que l'on crée
     ➔ remplacé par <section>
  2. La fusion de plusieurs éléments pour définir le contenu est difficile
     ➔ HTML5 a résolu ce point par l'apport de balises spécifiques :
     <article>,
     <section>,
     <nav>,
     <aside>

3. De plus, en HTML4, tous les titres sont pris en considération dans la TdM, il n'est pas possible de cacher un sous-titre ou un titre secondaire pour mieux révéler une section :

```
<h1>Justine</h1>
    <h2>Les Malheurs de la Vertu</h2>
```

- crée la TdM

  1. Justine

  1.1 Les Malheurs de la Vertu

➔ HTML5 introduit l'élément <hgroup> qui cache tous les éléments heading sauf le premier du rang le plus élevé :

```
<hgroup>
  <h1>Justine</h1>
    <h2>Les Malheurs de la Vertu</h2>
</hgroup>
```

- Crée la TdM

  1. Justine

4. En HTML4, la section fait partie intégrante de l'ossature du document
   - mais les documents n'ont pas une structure toujours linéaire
   - certaines sections peuvent être à part (des encadrés, etc.) renseignant sur le contenu

   ➔ HTML5 introduit l'élément <aside> pour ça

5. Enfin, en HTML4,
   - Il n'existe aucun moyen d'avoir une section contenant des informations connexes non pas au document mais à l'ensemble du site, comme les logos, les menus, la table des matières, ou les droits d'auteur et mentions légales

   ➔ À cette fin, HTML5 introduit trois éléments de sections spécifiques
   - <nav> pour les collections de liens, comme une table des matières
   - <footer>,
   - <header> pour plus d'informations concernant le site

Différence entre <section> et <article>

- Question souvent posée
  - Un article
    - est une partie indépendante qui est complète en soi
      - Par ex. un mail dans une liste de mails
      - On peut s'y référer de l'extérieur du site
      - On peut le syndiquer avec une RSS : i.e. suivre son évolution
  - Une section
    - n'est pas une partie indépendante
    - C'est soit une manière de subdiviser une page en plusieurs sujets soit une manière de sectionner un article en sections

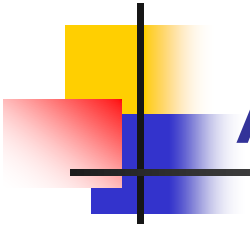
- Les titres <hi> jouent un rôle important pour définir des sections implicites

Ce code HTML définit deux sections de haut-niveau :

```
<section>
    <h1>Forest elephants</h1>
    <section>
        <h2>Introduction</h2>
        <p>In this section, we discuss the lesser known
        forest elephants
    </section>
    <section>
        <h2>Habitat</h2>
        <P>Forest elephants do not live in trees but among
        them.
    </section>
    <aside>
        <p>advertising block
    </aside>
</section>
<footer>
  <p>(c) 2010 The Example company
</footer>
```

– Cette structuration est bien rendue par la TdM :

1. Section
   1.1 Section
   1.2 Section
   1.3 Section (aside)
2. Section (footer)



Audio et vidéo

- Avantages
  - Le HTML5 apporte un lecteur vidéo standardisé intégré directement au navigateur
  - L'API est encore en développement, mais les principales fonctions sont pleinement supportées par tous les navigateurs récents
  - La balise peut être "décorée" avec du CSS
- Inconvénients
  - Toutes les spécifications n'apportent pas tous les avantages apportés par les plugins
    - Par exemple, les vidéos ne sont pas protégées contre la copie

Vidéo

- Syntaxe

```
<video src="movie.mp4"></video>
```

ou de façon plus détaillée

```
<video>
        <source src="movie.mp4" type= "video/mp4" />
</video>
```

  - La source peut être un lien externe

La Balise Audio

- Syntaxe

```
<audio src="music.mp3"></audio>
```

  - ou de manière plus précise, suivant les codecs :

```
<audio>
        <source src="music.mp3" type= "audio/mpeg" />
        <source src="guitv.wav" type="audio/x-wav"> What
          You Feel</source>
        <source src="untrue.ogg" type="audio/ogg">What You
          Feel</source>
</audio>
```

Codecs

- Voici un tableau récapitulatif des codecs supportés par les principaux navigateurs
  - Compatibilité des codecs vidéo

OGG	Chrome, Opera, Firefox 3 et 4
MP4	Chrome, IE 9, Safari
WebM	Chrome, Opera, Firefox 4

  - Compatibilité des codecs audio

MP3	Chrome, IE9, Safari
OGG	Chrome, Firefox 4, Opera
WAV	Firefox 4, Opera, Safari

Liste des attributs

- Autoplay : *ex1_autoplay.html*
  - Démarre automatiquement la vidéo au chargement de la page
  - Déconseillé car intrusif et consomme beaucoup de bande p.

```
<video autoplay >
        <source src="movie.mp4" type= "video/mp4" />
</video>
```

- Controls 

  - Barre de contrôle proposée par le navigateur
  - Accessible via le clavier
  - Composée d'un bouton 'play/pause', 'une barre de progression' et 'le contrôle du volume'

```
<video controls>
        <source src="movie.mp4" type= "video/mp4" />
</video>
```

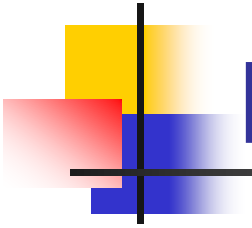
- Loop *(ex5_loop.html)*
  - Passe la vidéo en boucle (cést un booléen)

```
<video loop >
        <source src="movie.mp4" type= "video/mp4" />
</video>
```

- Src *(ex6_src.html) sur FireFox*

  - Affiche une image au cas où la vidéo ne peut pas être chargée
  - En effet, tous les navigateurs ne sont pas en mesure de lire tous les formats vidéo

```
<video src="non_supportee.jpg" >
        <source src="movie.mp4" type= "video/mp4" />
</video>
```



Personnalisation d'un Lecteur

- Fonctions de l'API
  - L'API dispose de nombreuses fonctions mises à disposition par défaut, tel que :
  - play() Démarrer la lecture
  - pause() Mettre en pause
  - canPlayType() Interroger le navigateur pour déterminer si ce type de fichier peut être joué
  - buffered() Attribut qui définit l'intervalle entre le début et la fin de la partie tampon (buffer) de votre fichier
  - (Tout ce qui peut s'appliquer aux éléments et aux attributs en général)
- Vous pouvez également créer vos propres fonctions en JavaScript

- Exemple : video_interaction1.html (Chrome)

  - Passage par JavaScript

```
<button onclick="document.getElementById('v1').play()">Play</button>
<button onclick="document.getElementById('v1').pause()">Pause</button>
<button onclick="document.getElementById('v1').volume += 0.25">Volume
  Up </button>
<button onclick="document.getElementById('v1').volume -= 0.25">Volume
  Down</button>
<button onclick="document.getElementById('v1').muted = true;">Mute
  </button>
<button onclick="document.getElementById('v1').muted = false">Unmute
  </button>

<video id="v1" src="sources_video/video.mp4" controls>
  <p>Sorry, your browser does not support the video element.</p>
</video>
```

Le style

http://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html

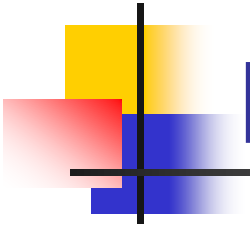
- Le style peut s'appliquer de deux manières
  - Interne

    <style>

    Selecteurs {propriétés}

    </style>
  - Externe
    - En ajoutant dans le head un lien à un fichier .css

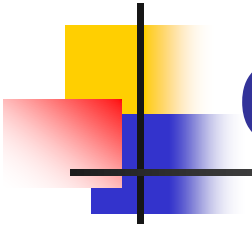
      <link rel=stylesheet href="fichier.css" />



La CSS

- Rappel des principaux sélecteurs de la CSS1 et la CSS2
  - * : n'importe que élément
  - E : tous les éléments E
  - E[toto] : tout élément E possédant un attribut toto
  - E[toto="titi"] : tout élément E dont l'attribut toto vaut titi
  - E[foo~="bar"] : tout élément E dont l'attribut toto contient une liste de valeurs séparées par des espaces, une de ces valeurs étant titi. Par exemple div[class ~="toto"] sélectionnera tous les div dont une des classes est toto
  - E[toto|="titi"] : tout élément E dont l'attribut toto est une liste de valeurs sélarées par des tirets et commença par titi. Par exemple div[lang|="fr"] ramènera les div dont l'attribut lang est "fr-FR" ou "fr-CA" ou "fr-BE", etc.
  - E.toto : tout élément E dont la classe est toto. La façon dont la classe est déterminée dépend de la grammaire utilisée. Cést l'attribut class en HTML, mais ça peut être autre chose

- E#toto : l'élément E dont l'identifiant unique est toto
- E:lang(fr) : tout élément dont la langue humaine est le français
- E:first-child : tout élément E premier dans la liste des enfants de son père
- E::first-line : première ligne de contenu de tout E. Utile par exemple en typographie pour mettre en majuscule la première ligne d'un paragraphe
- E::first-letter : la première lettre de tout élément E. Pour faire des lettrines par exemple
- E::before et E::after sont un cas un peu particulier puisqu'ils permettent généralement d'insérer du contenu depuis une CSS avant ou après une balise.
- E F : tout élément F figurant parmi les descendants de E
- E > F : tout F fils immédiat de E
- E + F : tout F immédiatement précédé d'un E
- E:link : tout élément E source d'un lien qui n'a pas encore été visité
- E:visited : tout élément E source d'un lien qui a déjà été visité
- E:focus : utile en HTML pour désigner un élément qui a le focus. Pas sûr que ça serve dans d'autres cas.
- E:active : idem, tout élément E actif
- E:hover : idem, tout élément E pendant le survol de la souris



CSS3

- Les nouveaux sélecteurs dans CSS3
  - **Gestion des espaces de noms**
  - Il n'est pas rare qu'un document XML contienne des éléments de plusieurs namespaces
    - Un fichier XUL par exemple pourra également contenir du SVG, du XHTML...
  - On peut désormais restreindre un sélecteur à un espace de nom
  - Il faut pour cela déclarer un alias pour le namespace, puis utiliser la syntaxe alias|selecteur
  - Exemple:

    ```
    @namespace xhtml url(http://www.w3.org/1999/xhtml);
    xhtml|image { border: 1px solid black }
    ```
  - affectera une bordure aux images en XHTML mais pas en XUL

La CSS3

http://designshack.developpez.com/tutoriels/css/introduction-css3/

- Nouveaux sélecteurs d'attribut
  - CSS introduit 3 nouvelles méthodes pour sélectionner des éléments en fonction de leurs attributs :
    - E[toto^="titi"] : tout élément E dont l'attribut toto commence par "titi"
    - E[toto$="titi"] : tout élément E dont l'attribut toto finit par "titi"
    - E[toto*="titi"] : tout élément E dont l'attribut toto contient "titi"

- Exemple

```
p.example{
margin:0; padding:10px; color:#000;
}
p.example[title^="ess"]{
color:#fff; background:#333;
}
```

- Appliqué sur :

```
<p class="example"> je n'ai pas d'attribut title</p>
<p class="example" title="comment"> j'ai un attribut title mais il ne commence pas par "ess"</p>
<p class="example" title="essai"> j'ai un attribut title commençant par "ess"</p>
<p class="example" title="esson"> j'ai un attribut title commençant par "ess" également</p>
```

je n'ai pas d'attribut title

j'ai un attribut title mais il ne commence pas par "ess"

j'ai un attribut title commençant par "ess"

j'ai un attribut title commençant par "ess" également

[attr$="stringValue"]

```
p.example2{
    margin:0;
    padding:10px;
    color:#000;
}
p.example2[title$="sai"]{
    color:#fff;
    background:#045FB4;
}
```

```
<p class="example2"> je n'ai pas d'attribut title</p>
<p class="example2" title="comment"> j'ai un attribut title mais il
ne finit pas par "sai"</p>
<p class="example2" title="essai"> j'ai un attribut title finissant
par "sai"</p>
<p class="example2" title="esson"> j'ai un attribut title finissant
par "sai" également</p>
```

je n'ai pas d'attribut title

j'ai un attribut title mais il ne commence pas par "ess"

j'ai un attribut title commençant par "ess"

j'ai un attribut title commençant par "ess" également

[attr="stringValue"]*

```
p.example3{
    margin:0;
    padding:10px;
    color:#000;
}
p.example3[title*="val"]{
    color:#fff;
    background:#990000;
}
```

```
<p class="example3"> je n'ai pas d'attribut title</p>
<p class="example3" title="comment"> j'ai un attribut title mais il ne
contient pas "val"</p>
<p class="example3" title="val"> j'ai un attribut title contenant au
moins "val"</p>
<p class="example3" title="evaluer"> j'ai un attribut title contenant au
moins "val" également</p>
<p class="example3" title="eval"> j'ai un attribut title contenant au
moins "val" également</p>
```

je n'ai pas d'attribut title

j'ai un attribut title mais il ne contient pas "val"

j'ai un attribut title contenant au moins "val"

j'ai un attribut title contenant au moins "val" également

j'ai un attribut title contenant au moins "val" également

Le combinateur d'adjacence directe

```
.example4 div{
    margin:0;
    padding:10px;
    color:#000;
}
.example4 div~p{
    color:#fff;
    margin:20px;
    width:200px;
    padding:5px;
    border:1px solid #333;
    background:#006644;
}
```

```
    <div class="example4">

        <div>je suis l'élément particulier div</div>
        <p> je suis un p qui suit le div (l'éléent particulier)</p>
        <p>je suis un p qui suit le div (l'élément particulier)</p>
        <span>je suis un span</p>
        <p>je suis un p qui ne suit pas le div (l'élément particulier)</p>

    </div>
```

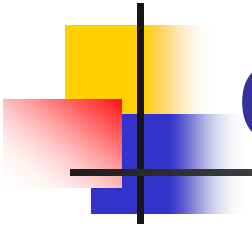
je suis l'élément particulier div

je suis un p qui suit le div (l'élement particulier)

je suis un p qui suit le div (l'élement particulier)

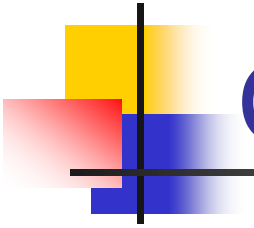
je suis un span

je suis un p qui ne suit pas le div (l'élement particulier)



CSS3

- Des pseudos-classes
  - CSS3 voit votre document comme une arborescence DOM dans laquelle vous pouvez vous balader :
    - :root qui désigne la racine d'un document
    - E:nth-child(xxx) : tous les éléments E qui sont le xxx fils de leur père
    - xxx peut être :
      - un chiffre even ou odd : tous les fils pairs ou impairs, pratique par exemple pour avoir un style appliqué alternativement aux lignes d'une table
      - an+b, avec a et b deux chiffres
      - n prendra toutes les valeurs à partir de zéro. Par exemple 2n représente tous les enfants pairs, 2n+1 tous les enfants impairs. a peut être négatif, et le sélecteur ramènera tous les éléments pour lesquels an+b est positif. tr:nth-child(-n+6) sélectionne les 6 première lignes de toutes les tables du document.



CSS3

- E:nth-last-child(xxx) reprend le même principe mais en comptant à partir de la fin
- :nth-last-child(2) sélectionnera tous les avant-derniers éléments
- :nth-last-child(-n+3) les 3 derniers fils de tout élément
- E:last-child est le pendant de E:first-child et synonyme de :nth-last-child(1)
- E:nth-of-type(xxx) et E:nth-last-of-type(xxx) sélectionnent tous les éléments qui ont xxx - 1 frères de même type quéux avant ou après eux. Par exemple img:nth-of-type(2) sélectionnera toutes les 2° images
- E:first-of-type et E:last-of-type sont des raccourcis dont je vous laisse deviner le sens
- E:only-child sélectionne tous les fils uniques
- E:only-of-type renvoie tous les éléments qui sont seuls de leur type parmi les enfants directs d'un élément
- E:empty représente tous les éléments qui n'ont pas dénfants (attention, le texte à l'intérieur d'un nœud est un nœud texte)

:nth-child(expression)

```
.exampleTable{
    width:100%;
    border:1px solid #444;
}
.exampleTable tr:nth-child(even){ /*tous les enfants aux numéros pairs*/
    background:#999999;
    text-shadow: 2px 2px 5px #111;
    color:#fff;
}
.exampleTable tr:nth-child(odd){ /*tous les enfants aux numéros impairs*/
    background:#990000;
    color:#fff;
}
.exampleTable tr:nth-child(3n){ /*tous les 3 enfants*/
    background:#045FB4;
    color:#fff;
}
.exampleTable tr:nth-child(7){ /*l'enfant numéro 7*/
    background:#006400;
    text-shadow: 2px 2px 2px #fff;
    color:#000;
}
```

```
<table class="exampleTable">
    <tr>
        <td>1ere ligne</td>
    </tr>
    <tr>
        <td>2eme ligne</td>
    </tr>
    <tr>
        <td>3eme ligne</td>
    </tr>
    <tr>
        <td>4eme ligne</td>
    </tr>
    <tr>
        <td>5ere ligne</td>
    </tr>
    <tr>
        <td>6eme ligne</td>
    </tr>
    <tr>
        <td>7eme ligne</td>
    </tr>
    <tr>
        <td>8eme ligne</td>
    </tr>
</table>
```

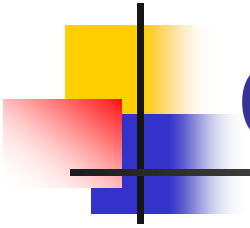
1ere ligne
2eme ligne
3eme ligne
4eme ligne
5ere ligne
6eme ligne
7eme ligne
8eme ligne

:nth-last-child(expression)

```
.exampleTable2{
    width:100%;
    border:1px solid #444;
}
.exampleTable2 tr:nth-last-child(odd){ /*tous les enfants aux numéros impairs depuis la fin*/.
    background:#990000;
    color:#fff;
}
.exampleTable2 tr:nth-last-child(-n+2){ /*les 2 derniers enfants*/
    background:#045FB4;
    color:#fff;
}
.exampleTable2 tr:nth-last-child(7){ /*l'enfant numero 7 en partant de la fin donc la 2eme ligne du tableau*/
    background:#006400;
    text-shadow: 2px 2px 2px #fff;
    color:#000;
}
```

```
<table class="exampleTable2">
    <tr>
        <td>1ere ligne</td>
    </tr>
    <tr>
        <td>2eme ligne</td>
    </tr>
    <tr>
        <td>3eme ligne</td>
    </tr>
    <tr>
        <td>4eme ligne</td>
    </tr>
    <tr>
        <td>5ere ligne</td>
    </tr>
    <tr>
        <td>6eme ligne</td>
    </tr>
    <tr>
        <td>7eme ligne</td>
    </tr>
    <tr>
        <td>8eme ligne</td>
    </tr>
</table>
```

1ere ligne
2eme ligne
3eme ligne
4eme ligne
5ere ligne
6eme ligne
7eme ligne
8eme ligne



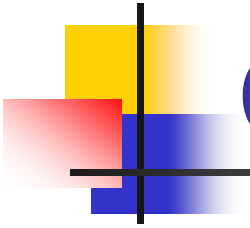
CSS3

- Transformations
  - Pour utiliser ces fonctions, il faut mettre un préfixe devant le nom de la fonction
    - -webkit pour utiliser les fonctions sous Google Chrome et Apple Safari
    - -moz pour les utiliser sous firefox
    - -o pour Opera
  - Exemple :

    -webkit-transition-property

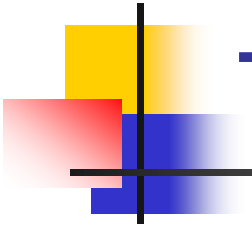
    -moz-transition-duration

    -o-transition-timing-function



CSS3

- Transitions
  - Permettre une transition douce entre l'ancienne valeur et la nouvelle valeur d'une propriété CSS lorsqu'un événement est déclenché
  - Pour définir une nouvelle transition animée, il est nécessaire de préciser au minimum :
    - La ou les propriété(s) à animer
    - La durée de l'animation



Transition

- Propriétés
  - transition-property : précise les propriétés CSS à transformer
  - transition-duration : précise la durée de la transition
  - transition-timing-fonction : précise la fonction de transition à utiliser, le modèle d'interpolation (accélération, décélération …)
  - transition-delay : précise le retard avec lequel la transition se déclenche

Transition

- Exemples

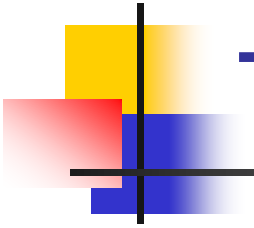
```
selecteur {
    transition-property: color, width;
  }
selecteur {
    transition-duration: 5s;
  }
selecteur {
    transition-delay: 1s;
  }
```

- Exemple
  - Animer la couleur d'un titre : transition-couleur.html

```
/* Fonctionne déjà sur webkit */
-webkit-transition-property: color;
-webkit-transition-duration: 2s;
/* Bientôt supporté par Firefox */
-moz-transition-property: color;
-moz-transition-duration: 2s; /*
/* et ceci lorsque ce sera standardisé */
transition-property: color;
transition-duration: 1s;
```

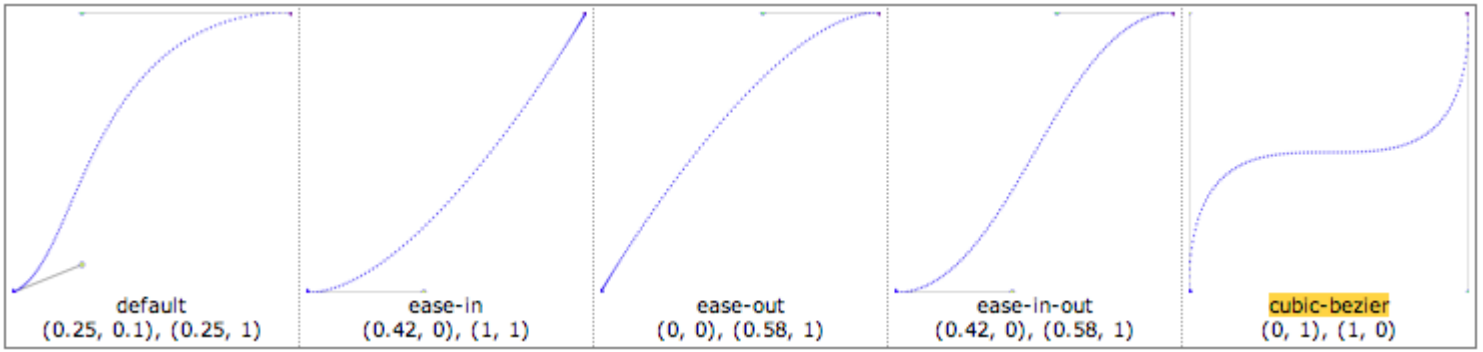
Survolez-moi ! **Survolez-moi !**

avant après



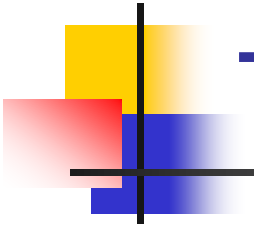
Transition

- La propriété transition-timing-function
  - Définit le mode de transition parmi :
    - **ease** : rapide sur le début et ralenti sur la fin
    - **linear** : la vitesse est constante sur toute la durée de l'animation
    - **ease-in** : lent sur le début et accélère de plus en plus vers la fin
    - **ease-out** : rapide sur le début et décélère sur la fin
    - **ease-in-out** : le départ et la fin sont lents
  - Exemple : transition-timing-function.html



ease: Par défaut, correspond à cubic-bezier(0.25, 0.1, 0.25, 1.0). Cela donne un départ et une arrivée ralentis.
linear: Une fonction linéaire, que l'on peut exprimer aussi par cubic-bezier(0.0, 0.0, 1.0, 1.0).
ease-in: cubic-bezier(0.42, 0, 1.0, 1.0). Départ ralenti.
ease-out: cubic-bezier(0, 0, 0.58, 1.0). Arrivée ralentie.
ease-in-out: cubic-bezier(0.42, 0, 0.58, 1.0). Départ et arrivée ralentis.

- Exemple : annuaire.xml



Transformation

- La propriété transform
  - permet d'appliquer des transformations sur un élément :
    - rotation, décalage, zoom, déformation, perspective
  - Les transformations présentées ci dessous peuvent avoir un point d'origine différent de *top right* ou *0 0*

```
-webkit-transform-origin: 0 0;
-moz-transform-origin: 0 0;
-o-transform-origin: 0 0;
transform-origin: 0 0;
```

Transformation

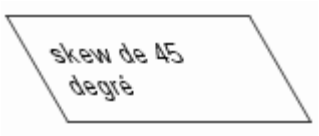
- La propriété transform
  - *Proportionalité*
    - le *scale* peut autoriser 2 valeurs ce qui ne permet plus de garder la proportionalité

```
-webkit-transform: scale(1.5, 0.75);
-moz-transform: scale(1.5, 0.75);
-o-transform: scale(1.5, 0.75);
-ms-transform: scale(1.5, 0.75);
transform: scale(1.5, 0.75);
```

Transformation

- La propriété transform
  - *Skew*
    - le *skew* permet de faire une translation horizontale d'un tag HTML

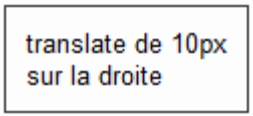
```
-webkit-transform: skew(30deg);
-moz-transform: skew(30deg);
-o-transform: skew(30deg);
-ms-transform: skew(30deg);
transform: skew(30deg);
```



Transformation

- La propriété transform
  - *Translate*
    - le *translate* permet de faire une translation d'un tag HTML

```
-webkit-transform: translate(10px, 0px);
-moz-transform: translate(10px, 0px);
-o-transform: translate(10px, 0px);
-ms-transform: translate(10px, 0px);
transform: translate(10px, 0px);
```



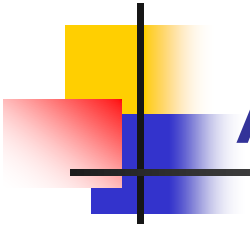
- *Les animations complexes sous webkit*
  - Les navigateurs basés webkit permettent de mettre en place des animations :

```
@-webkit-keyframes resize {
    0% {
        padding: 0;
    }
    50% {
        padding: 0 20px;
        background-color:rgba(255,0,0,0.2);
    }
    100% {
        padding: 0 100px;
        background-color:rgba(255,0,0,0.9);
    }
}

.boxTransformAnimation {
    -webkit-animation-name: resize;
    -webkit-animation-duration: 1s; (pour un animation en boucle, utilisez infinity)
    -webkit-animation-iteration-count: 4;
    -webkit-animation-direction: alternate;
    -webkit-animation-timing-function: ease-in-out;
}
```

Transformation

- Exemple : ExpandingCaptionedImages
  - Il s'agit d'agir sur une suite d'images pour leur faire subir différents types de transformations
  - En les survolant, elles bougent
  - En cliquant dessus, elles s'agrandissent



Autres fonctions

- box-shadow
  - Permet de créer une ombre portée sur n'importe quel élément de Html
  - Indication du décalage de l'ombre portée, de la force du dégradé ou encore la couleur.
  - Nécessite un filtre pour fonctionner sous IE jusqu'à la version 8
  - Les ombrages sont légèrement différents d'un navigateur à l'autre

box-shadow

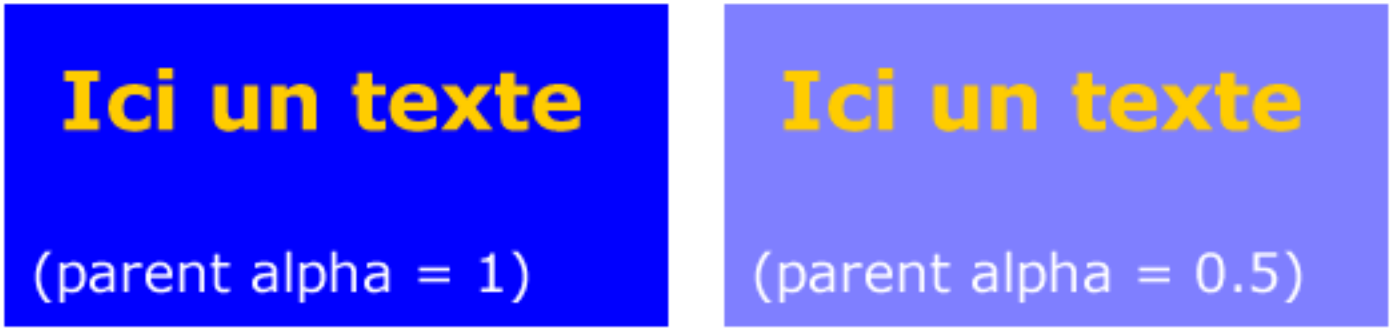
```
.ombrage{
   border: 5px solid #fff;
   -moz-box-shadow: 8px 8px 12px #aaa;
   -webkit-box-shadow: 8px 8px 12px #aaa;
   box-shadow: 8px 8px 12px #555;
}

.ombrage{
   filter:prgid:DXImageTransform.Microsoft.Shadow(color='#aaaaaa', Direction=135, Strength=12);
   zoom: 1;
}
```

RGBa

- La première valeur correspond au taux de rouge
- La seconde valeur correspond au taux de vert
- La troisième valeur correspond au taux de bleu
- La quatrième valeur correspond au taux d'opacité

```
div{
   background-color: rgba(0, 0, 255, 0.5);
}
```

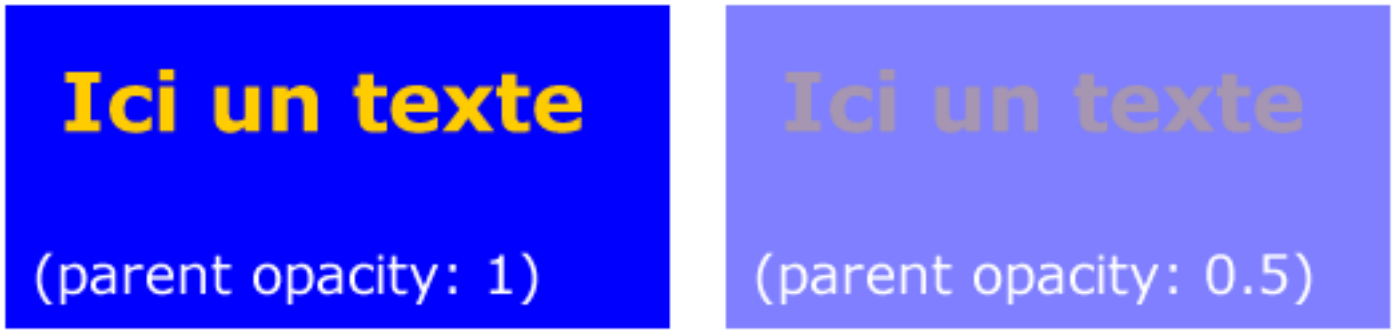


Le parent alpha correspond à l'opacité

Opacity

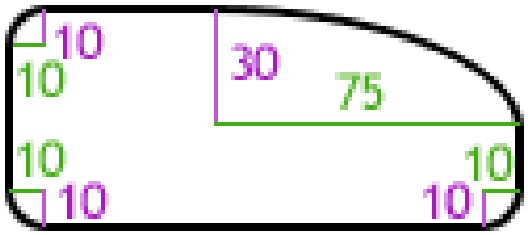
→ opacity ne gère qu'une seule valeur, l'opacité

```
div{
   opacity: 0.5;
}
```



Border-radius

```
.arrondi{
   width: 120px;
   height: 50px;
   border: 2px solid #000;
   -moz-border-radius: 10px 75px 10px 10px / 10px 30px 10px 10px;
   -webkit-border-radius: 10px 75px 10px 10px / 10px 30px 10px 10px;
   border-radius: 10px 75px 10px 10px / 10px 30px 10px 10px;
}
```



border-radius: 10px 75px 10px 10px / 10px 30px 10px 10px;

Gradient

- Pour les moteurs WebKit et Firefox

- Voici le code pour la div de dessus :

```
margin:10px auto;
width:400px;
height:80px;
background: -webkit-gradient(linear, left top, left
   bottom, from(#FF0), to(#0FF));
background: -moz-linear-gradient(top,#FF0,#0FF);
   background:blue;
```

- Exemple d'interface à essayer : http://gradients.glrzad.com/

Gradient Direction

Select the direction for the gradient.

xStart: Right | yStart: Custom 0 | xEnd: Right | yEnd: Top

Color Format

Select the format you would like your colors generated in. Available options are HEX and RGB.

Your option will be remembered for the next time you visit.

RGB

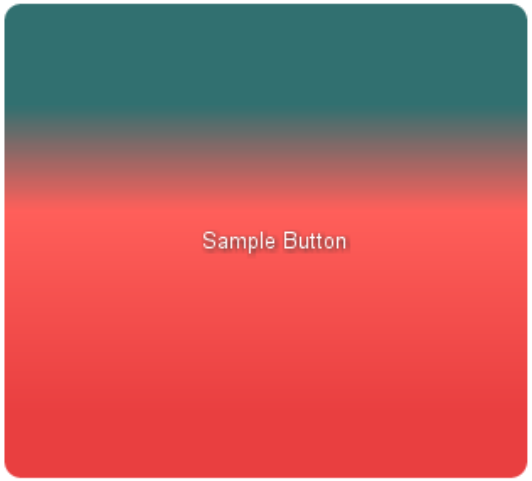
Color Swatches

Generate Random Gradient

Select colors to make up your gradient.

Gradient Sample

The sample updates live while you're adjust your gradient.

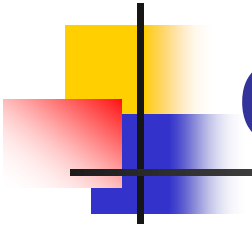


The Code

The code sample updates live while you're adjusting your gradient.

Copy the code

```
background-image: -webkit-gradient(
    linear,
    right bottom,
    right top,
    color-stop(0.14, rgb(233,63,64)),
    color-stop(0.57, rgb(255,95,90)),
    color-stop(0.79, rgb(49,112,112))
);
background-image: -moz-linear-gradient(
    center bottom,
    rgb(233,63,64) 14%,
    rgb(255,95,90) 57%,
    rgb(49,112,112) 79%
);
```



Conclusion

- Pour le moment, l'utilisation de CSS3 est encore rare car les navigateurs ne supportent toujours pas toutes les applications/fonctions de ce dernier

- A terme, ces dernières pourront être une alternative plus simple que le JavaScript et le Flash