

# Problèmes avec HTML4.01

- Une multitudes de versions :
  - HTML4, HTML4,01, XHTML2,0, XHTML1,0, etc..
  - La version strict, transationnal, frameset, etc.
- Une multitude de variante de la balise **Meta** en fonction du jeu de caractère
- Complexité de la définition du **Doctype**
- HTML5 est là!
  - Première contribution: simplification du DOCTYPE et META pour garantir une compatibilité exhaustive avec tout type de navigateur quelque soit sa version !

# Simplification Doctype en HTML5

- Le **doctype** est simplifié :

**Syntaxe** : `<!DOCTYPE html>`

- Il n'est pas sensible à la casse (on peut écrire `<!doctype html>` par exemple).
- Avant le Doctype pour un document xhtml 1.0

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0
    Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
2 <html>
3 <head>
4   <title>Head First Lounge</title>
5   <meta http-equiv="content-type" content="text/html;
  charset=UTF-8">
6 </head>
7 <body>
8   <h1>Welcome to Head First Lounge</h1>
9   <p>
10     
11   </p>
12   <p>
13     Join us any evening for refreshing <a
  href="elixirs.html">elixirs</a>,
14     conversation and maybe a game or two of Tap
  Tap Revolution.
15     Wireless access is always provided; BYOWS (Bring
  Your Own
  Web Server).
16   </p>

```

```

1 <!doctype html>
2 <html>
3 <head>
4   <title>Head First Lounge</title>
5   <meta charset="utf-8">
6   <link rel="stylesheet" href="lounge.css">
7   <script src="lounge.js"></script>
8 </head>
9 <body>
10   <h1>Welcome to Head First Lounge</h1>
11   <p>
12     
13   </p>
14   <p>
15     Join us any evening for refreshing <a href="elixirs.html">elixirs</a>,
16     conversation and maybe a game or two of Tap Tap Revolution.
17     Wireless access is always provided; BYOWS (Bring Your Own Web Server).
18   </p>
19 </body>
20 </html>

```

# Apports de HTML5

- Amélioration des formulaires

# Nouveautés de HTML5

- Balises
  - Nouveaux éléments et attributs pour les formulaires
  - Video/audio natifs (sans le recours à un Plugin externe)
  - Canvas drawing widget : outil de dessin
- Nouveaux éléments sémantiques
- API Javascripts
  - Web storage
  - Offline application
  - Web workers
  - Geolocalisation
  - Web socket

# Les formulaires HTML (5)

## Nouveaux éléments et Attributs

HTML5 a enrichi les formulaire d'un nouveau ensemble de **balises**, de **types** et d'attributs pour ne plus avoir besoin de la validation de formulaire coté client par des scripts (de type JavaScript par exemple).

Nouveaux éléments :

**<datalist>**

**<output>**

# HTML5 : nouveaux éléments de formulaire

- La balise **<datalist>** : Elle définit une liste d'options à utiliser avec l'élément **<input>**, pour définir les valeurs permises pour cette zone de saisie.

– *Le <datalist> et ses options ne seront pas*

*officiellement*

```
<!DOCTYPE HTML><html><body>  
My car : <input list="cars"  
>
```

```
<datalist id="cars">
```

```
  <option
```

```
value="Fiesta"/>
```

```
  <option value="Ford"/>
```

```
  <option
```

```
value="Focus"/>
```

lier cet élément avec le

My car :

  
Fiesta  
Ford  
Focus

- **<output>** : est utilisé pour afficher le résultat d'un calcul effectué par un script JS par exemple.
- **Exemple:**

```
<form
oninput="result.value=parseInt(a.value)+
parseInt(b.value)">
<input type="number" name="b" value="50" />
+
<input type="number" name="a" value="10" />
=
<output name="result">60</output>
```



# Output

```
2
3 <form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
4     <input type="number" name="b" value="50" /> +
5     <input type="number" name="a" value="10" /> =
6     <output name="result">60</output>
7 </form>
8 |
```

+  = 60

# Nouvelles valeurs de l'attribut Type

- Input type=
  - radio
  - *range: HTML5*
  - reset
  - *search: HTML5*
  - submit
  - *tel: HTML5*
  - text
  - time
  - *url: HTML5*
  - *week: HTML5*
- button
- Checkbox
- *color: HTML5*
- *date: HTML5*
- *datetime-local: HTML5*
- *email: HTML5*
- file
- hidden
- image
- *month: HTML5*
- *number: HTML5.*

# Nouveaux attributs de input

- autocomplete
  - autofocus
  - form
  - formaction
  - formenctype
  - formmethod
  - formnovalidate
  - formtarget
  - height and width
  - list
  - min and max
  - multiple
  - pattern (regexp)
  - placeholder
  - required
  - step
- <http://html5doctor.com/demos/forms/forms-example.html>

# Les nouveaux attributs

## HTML5 new types

Username (required):

Step number

Hint

Three letter country code(Country code with regular expression) :

Select images:

 選択されていません

Quantity (between 1 and 5)

Enter a date after 2000-01-01:



Submit as normal

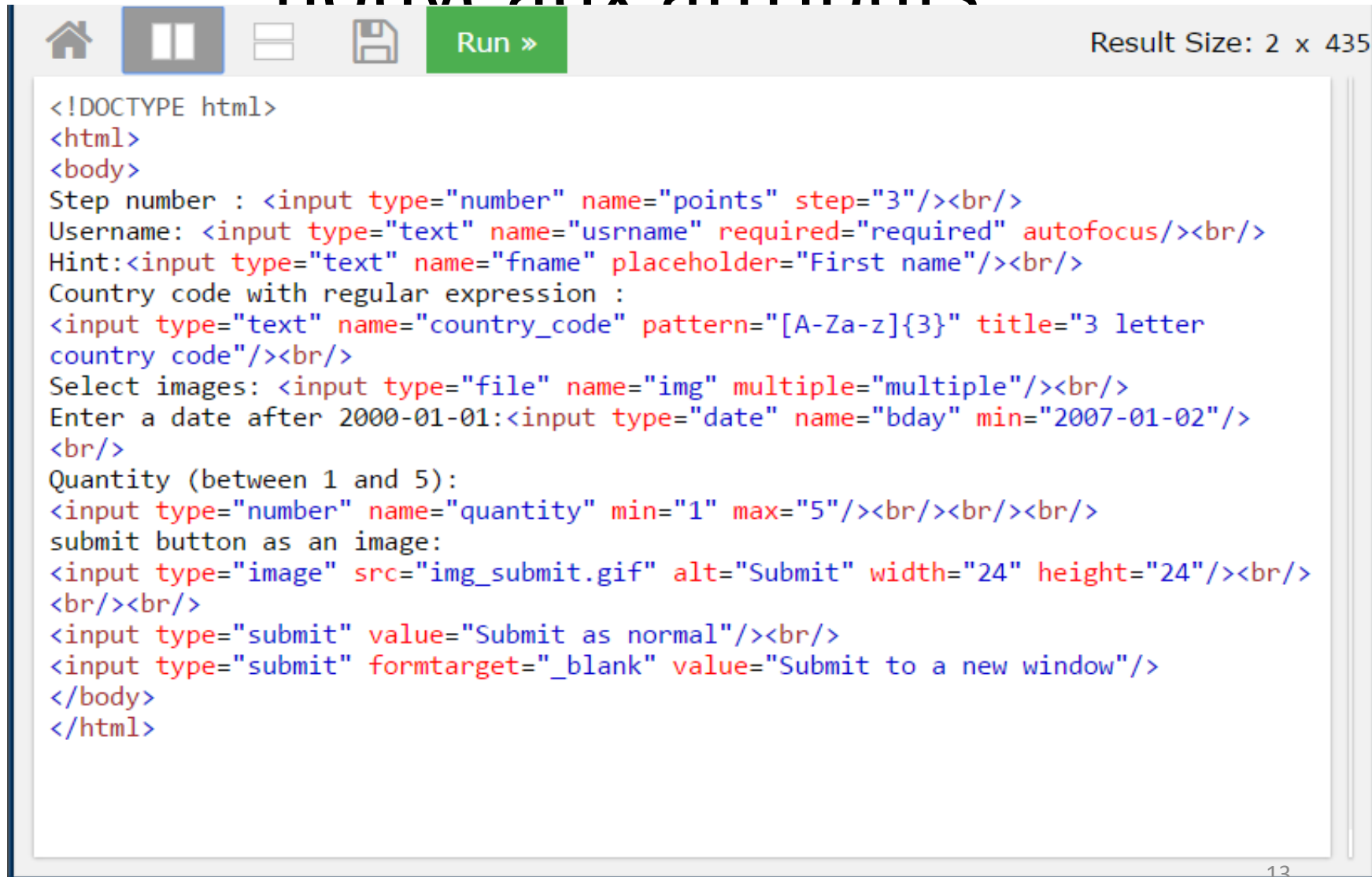
Submit to a new window

<< < 2012年(平成24年) 10月 > >>

日	月	火	水	木	金	土
30	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	30	31	1	2	3
4	5	6	7	8	9	10

今日 クリア

# Nouveaux types de <input> et nouveaux attributs



The screenshot shows a web browser window with a toolbar at the top containing icons for home, back, forward, and a green 'Run »' button. The address bar shows 'Result Size: 2 x 435'. The main content area displays the following HTML code:

```
<!DOCTYPE html>
<html>
<body>
Step number : <input type="number" name="points" step="3"/><br/>
Username: <input type="text" name="usrname" required="required" autofocus/><br/>
Hint:<input type="text" name="fname" placeholder="First name"/><br/>
Country code with regular expression :
<input type="text" name="country_code" pattern="[A-Za-z]{3}" title="3 letter
country code"/><br/>
Select images: <input type="file" name="img" multiple="multiple"/><br/>
Enter a date after 2000-01-01:<input type="date" name="bday" min="2007-01-02"/>
<br/>
Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5"/><br/><br/><br/>
submit button as an image:
<input type="image" src="img_submit.gif" alt="Submit" width="24" height="24"/><br/>
<br/><br/>
<input type="submit" value="Submit as normal"/><br/>
<input type="submit" formtarget="_blank" value="Submit to a new window"/>
</body>
</html>
```

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>HTML5 Tags</title>
  </head>
  <body>
    <form action="index.php" method="post" name="f1" id="f1">
      E-mail: <input type="email" name="user_email" /><br/>
      Range : <input type="range" name="points" min="1"
max="10" /><br/>
      Points: <input type="number" name="points" min="1"
max="10" /><br/>
      Homepage: <input type="url" name="user_url" /><br/>
      Date: <input type="date" name="user date" /><br/>
      Date and time: <input type="datetime-local" name="user_date" /><br/>
    </form>
  </body>
</html>

```

HTML5 Tags

127.0.0.1:8000/my%20first%20web%20site/HTML5.html

E-mail:

Range :

Points:

Homepage:

Date:

Date and time:

# Apports de HTML5

- Insertion des objets multimédia

# **Rappel**

## **objets MultiMedia en HTML4.01**



# Types d'objets

- Par objet, on entend tout type de fichier situé hors d'un fichier HTML et devant y être incorporé:
  - Un fichier de données ( un tableau Excel)
  - Un dessin AUTO CAD,
  - Un fichier de musique Midi,
  - Une animation Flash,
  - Une source en transit (streaming) pour la transmission radio,etc.
  - Mais il peut s'agir aussi d'un fichier exécutable par le navigateur Web, à savoir d'un programme. Ex:

# Types d'objets

- Ce repère ne peut certes pas résoudre le problème de l'affichage d'un fichier quelconque chez l'utilisateur mais il propose tout au moins une syntaxe uniforme et contribue de ce fait à la simplification de HTML.
- On spécifie de quelle sorte d'objet il s'agit avec l'attribut **type** en indiquant le type mime, et la source du document avec l'attribut **data** en indiquant son URL.
- Vous devez toujours noter les mentions de largeur

(**width** `<object data="..." type="..." width="..." height="...">`  
**Syntaxe** `</object>`

# Contenu de la balise <object>

- Pour des fichiers de données référencés avec **data=**, notez en plus l'attribut **type=**, c'est le **type Mime** du fichier.
- **Mime** pour **M**ultipurpose **I**nternet **M**ail **E**xtensions.
  - Conçu à l'origine pour les systèmes de messagerie (E-mails) comportant divers type de fichiers attachés; Il fallait trouver une convention à l'intérieur du fichier pour en différencier les différentes parties (par exemple le texte du courriel et le fichier ZIP joint).
  - Après, il s'est avéré utiles pour différents types de<sup>19</sup>

# Examples

## Examples :

```
<object data="data/test.html"  
type="text/html" width="300"  
height="200">  
</object>
```

```
<object data="data/test.pdf"  
type="application/pdf" width="300"  
height="200">  
alt : <a href="data/test.pdf">test.pdf</a>  
</object>
```

# Autres exemples

- Vous avez parfois besoin de spécifier des paramètres relatifs au document par

```
<object type="audio/x-wav"  
data="data/test.wav" width="200"  
height="20">  
  <param name="src" value="data/test.wav">  
  <param name="autoplay" value="false">  
  <param name="autoStart" value="0">  
  alt : <a href="data/test.wav">test.wav</a>  
</object>
```

# **Objets multimedia en HTML5.1**

# Les buts de HTML5

- Minimiser le recours à des technologies/plugins tel que **Adobe Flash, JavaFX, and Microsoft Silverlight** pour **insérer les objets multimedia**
- Concrétiser la vision **Rich internet Application (RIA)** sans avoir recours à des technologies tierces ou à la partie serveur :
  - Des application aussi riches en fonctionnalités que les « desktop application » qui tournent sur toute versions de navigateurs et qui ne requiert pas l'installation d'un plugin et ne fait appelle à la

# RIA ?

- A rich Internet application (RIA) is a Web application designed to deliver the same features and functions normally associated with desktop applications. RIAs generally split the processing across the Internet/network divide by locating the user interface and related activity and capability on the client side, and the data manipulation and operation on the application server side.
- An RIA normally runs inside a Web browser and usually does not require software



# Les nouvelles balises multimedia HTML5

- `<video>`
- `<audio>`
- `<figure>`
- `<canvas>`
- `<svg>`
- `<embed>`
- `<source>`
- `<track>`

# <video>

- L'élément **<video>** permet d'insérer des vidéos dans une page Web de façon standard SANS avoir recours à un plugin tel que ADOBE Flash ou Apple QuickTIME etc.
- Défini les attributs : **preload**, **autoplay**, **loop**, **src**, **controls**, pour indiquer comment la vidéo doit être lu:
- Les formats supportés:

```
<video src="videofile.webm" autoplay poster="posterimage.jpg">
Sorry, your browser doesn't support embedded videos, but don't
worry, you can <a href="videofile.webm">download it</a> and
watch it with your favorite video player! </video>
```

# Webm?

- WebM video plays directly in your web browser using HTML5.
- No plug-ins are needed, but you must install a modern web browser that supports HTML5 and WebM.

# Les attributs de l'élément <video>

- L'attribut **controls** est un attribut booléen qui indique si l'auteur souhaite que cette interface utilisateur soit présente ou non par défaut.
- L'attribut facultatif **poster** peut être utilisé pour spécifier une image qui sera affichée à la place de la vidéo avant qu'elle ne commence.

```
<video src="video.webm" controls="controls"  
poster="poster.jpg" width="320" height="240">  
  <a href="video.ogv">Télécharger le film</a>  
</video>
```

# Example : <video>


Edit and Click Me >>

```
<!DOCTYPE HTML>
<html>
<body>

<video src="movie.ogg" width="320" height="240"
controls="controls">
Your browser does not support the video tag.
</video>

</body>
</html>
```

Your Result:



# <audio>

- Il est aussi simple d'intégrer de l'audio à une page en utilisant l'élément **<audio>**.
- L'élément **audio** n'a pas d'attributs **width**, **height** et **poster**.

```
<audio src="/music/lostmojo.wav">
<p>If you are reading this, it is because your
browser does not support the audio element.</p>
</audio>
```

# <source>

- L'élément **<source>** s'utilise avec les éléments <audio> et <video> pour indiquer des sources alternatives d'un fichier audio/video et entre lesquels le navigateur choisira en fonction des types/codecs qu'il supporte.
- Lorsque l'élément **source** est utilisé, l'attribut **src** doit être omis de l'élément parent vidéo/audio.
- ```
<audio>  
    <source src="/music/good.wma" type="audio/x-ms-wma">  
    <source src="/music/good.mp3" type="audio/mpeg">  
    <p>If you are reading this, it is because  
your browser does not support the HTML 'audio'  
element.</p>  
</audio>
```

# Exemple <audio> :

**<audio>**

**<!--**

Deux formats disponibles par ordre de priorité:

**-->**

**<source** src="trappeur.ogg" type="audio/ogg">

**<source** src="trappeur.aac" type="audio/aac">

**<!--**

Contenu alternatif si élément audio ou formats non

supportés dans le navigateur: -->

**<a** href="trappeur.ogg">

Télécharger **<cite>**Avant j'étais trappeur</cite>

**</a>**

de David TMX (format Ogg Vorbis)

**</audio>**



# Canvas 2D

- **<Canvas>** : représente une zone de dessin pouvant afficher des graphiques
  - Elle permet une interactivité avec les utilisateurs voir même créer des jeux, choses qui nécessitaient auparavant l'utilisation d'autres technologies comme les applets ou les animations Flash etc.
  - l'API (faisant partie de la spécification de HTML5) qui lui est associée met à disposition du programmeur de nombreuses méthodes accessibles en JavaScript pour la création de forme et d'effets

# Canvas 2D

- On commence par créer un <canvas> et on lui associe un **id**

Edit and Click Me »

Your Result:

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #c3c3c3;">
```

Your browser does not support the HTML5 canvas tag.

```
</canvas>
<script>
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
</script>
</body>
</html>
```



# Exemple avec Canvas

- <http://html5demos.com/video-canvas>

# <figure>

- L'élément **<figure>** peut être utilisé pour regrouper des éléments tels que des images ou des vidéos avec leur légende **<figcaption>**.

- Exemples :

```
<figure>

  <figcaption>Un petit chat mignon tout plein</figcaption>
</figure>
```

```
<p><a href="#1">Figure 1</a> provides the JavaScript code
for creating an alert box:</p>
<figure id="1">
<figcaption>Figure 1. JavaScript Alert Box.</figcaption>
<pre><code>alert('Hello!');</code></pre>
</figure>
```

# Canvas vs SVG

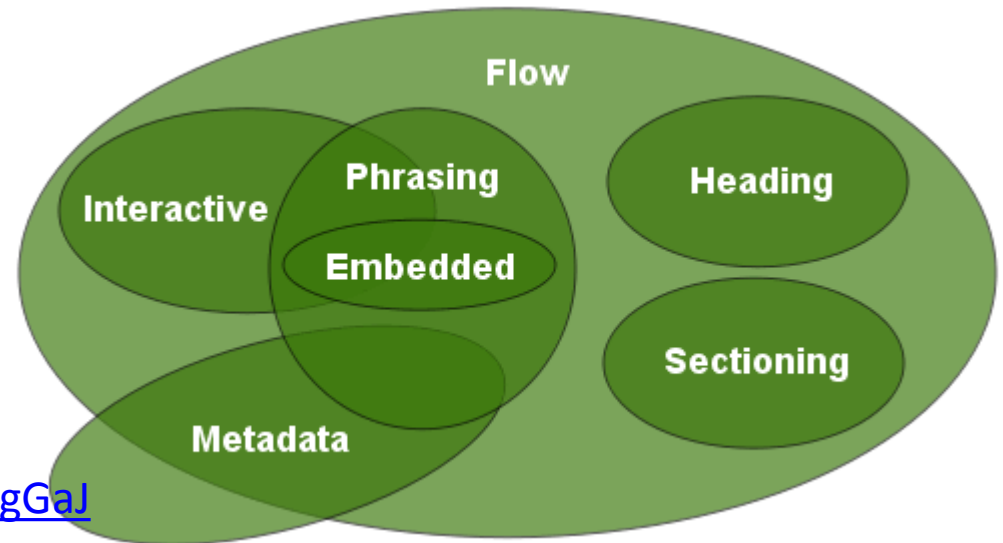
- SVG c'est quoi ??
  - SVG(scalable vector graphic) : permet de dessiner des graphiques vectorielles
  - Des graphiques qui s'adaptent à la taille de la fenêtre et qui ne sont pas affectés par le changement de résolution
  - Défini en XML imbriqué dans du HTML

```
<!DOCTYPE html>
<html lang="en">
<head>...</head>
<body>
<svg xmlns="http://www.w3.org/2000/svg">
<circle id="myCircle" cx="50" cy="50" r="100" fill="blue" />
</svg>
</body></html>
```

- Nouveaux éléments sémantiques

# La nouveauté dans HTML5

- Une grande nouveauté annoncée au niveau structurel des éléments est signalée dans l'introduction : **The new content model**
- Un nouveau schéma : les éléments HTML sont à présent uniquement regroupés en catégories sachant que les éléments peuvent appartenir à plusieurs catégories
  - Metadata content
  - Flow content
  - Sectioning content
  - Heading content



<http://www.youtube.com/watch?v=YFuzqgGajPQ>

# Avant ce modèle comment été structuré HTML4.01 ?

- **Block content**

- Les éléments qui occuperaient une ligne dans un document HTML
- Exp : `<p>`, `<div>`, ...

- **Inline content**

- Les éléments qui se trouveraient dans des éléments de bloc
- Exp : `<a>`, `<span>`, ...
- Ce modèle permettait seulement de structurer le document. Aucun sens n'est octroyé aux différentes parties.



# HTML5 content model

<http://www.youtube.com/watch?v=YFuzqgGajPQ>

- **Embedded content** : any content that imports other resources into the document. Exp. : `<object>`, `<video>`, `<canvas>`, `<embed>`, etc.
- **Interactive content** : any content specifically intended for user interaction.
  - Exp; : `<details>`, `<object>`, `<video>`, etc.
- **Heading content** : defines the header of a section, which can either be explicitly marked up with sectioning elements or implied by the heading content itself. Exp: `<h1>` à `<h6>`.
- **Phrasing content**: is the text of the document as well as elements used markup the text within paragraph level structures (inline content from the HTML 4 specification).
- **Sectioning content** is content that defines the scope of headings and footers. Using these elements will create a new section within the document.
  - Exp: `<article>`, `<aside>`, `<nav>`, `<section>`, etc.
- **Flow content** : contains the majority of elements in HTML5. Think of these elements as elements that would be included in the normal flow of the document.
- **Metadata content** is defined as being content that sets up the presentation or behavior of the rest of the content. You'll primarily find these elements in the head of the document. Exp: `<link>`, `<meta>`, `<script>`, `<title>`, etc.

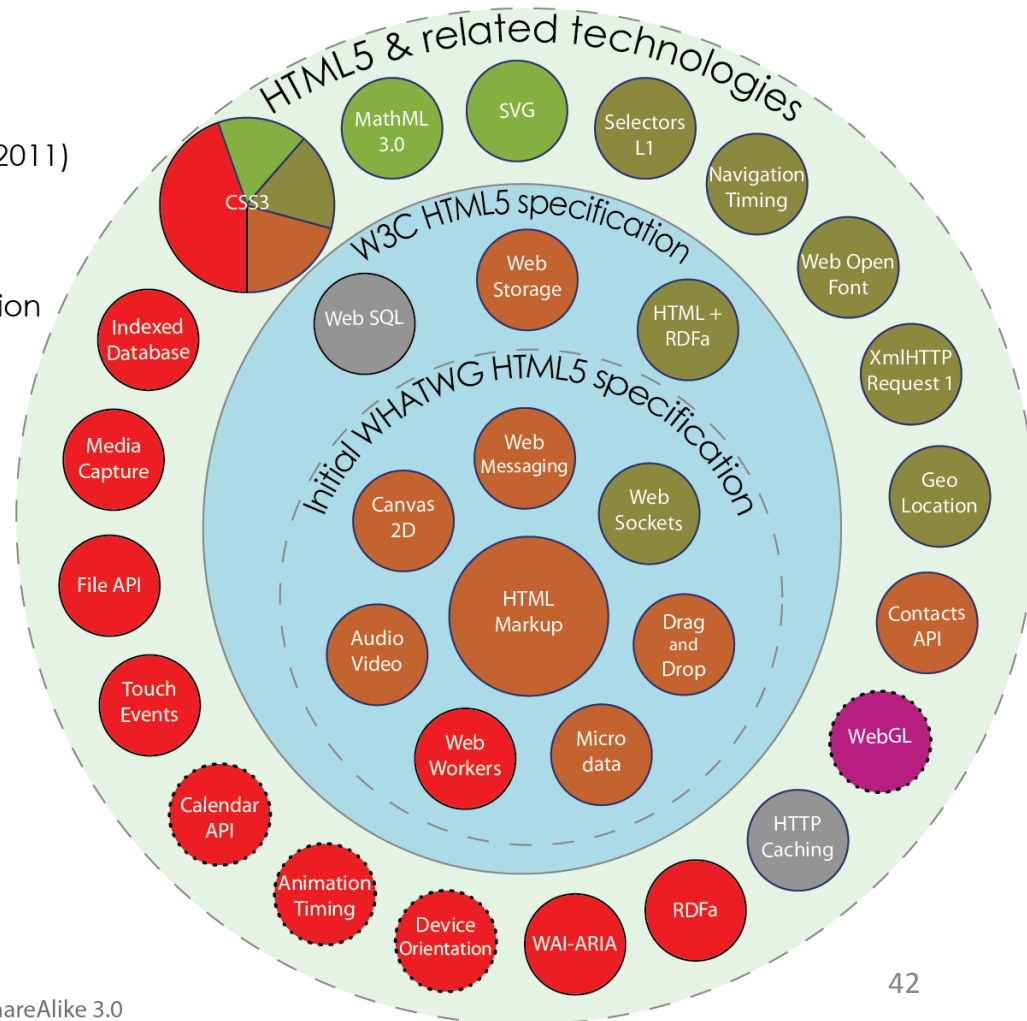
# Comment HTML5 va achever ses promesses ?

- En se basant sur HTML5 APIs + un ensemble de technologies connexes:

## HTML5

Taxonomy & Status (December 2011)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated W3C APIs



# Nouvelles règles

- Dans HTML5, il n'est pas systématiquement nécessaire de fermer tous les éléments. Ainsi, les éléments `<p>`, `<dd>`, `<dt>`, `<li>`, etc. n'ont plus besoin de balise fermante pour être valides.
- **Seule la version XHTML 5 obligera à fermer ces éléments !!!**
- Certains éléments ne nécessiteront ni balise fermante ni balise ouvrante, c'est le cas de `<html>`, `<head>`, `<body>`, `<thead>`, `<tfoot>` et

# HTML5

- Le HTML5 introduit une série complète de nouveaux éléments qui font qu'il est beaucoup plus simple de structurer les pages.
  - La plupart des pages HTML4.01 contiennent une diversité de structures identiques, comme les en-têtes, les bas de page, les colonnes les barres de navigations etc.
  - Aujourd'hui, il est relativement fréquent de les baliser par des éléments ***div***, leur attribuant à

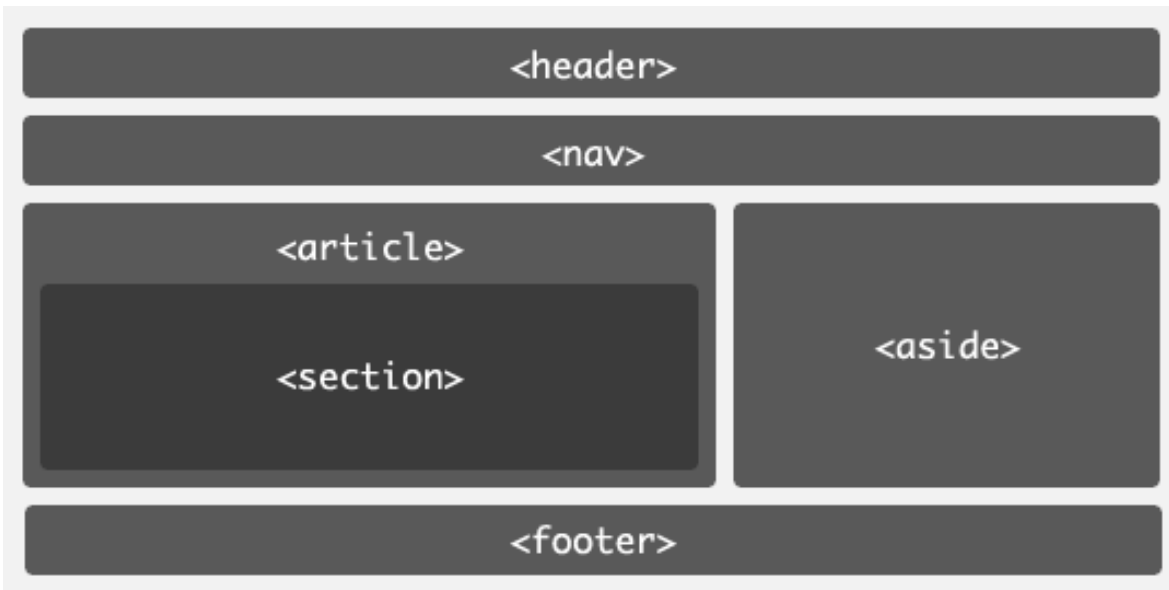


# Nouvelle structuration

- Les éléments `div` ont été remplacés par de nouveaux éléments:

***header***, ***nav***, ***section***, ***article***, ***aside*** et ***footer***.

- Le code de ce document pourrait ressembler à :

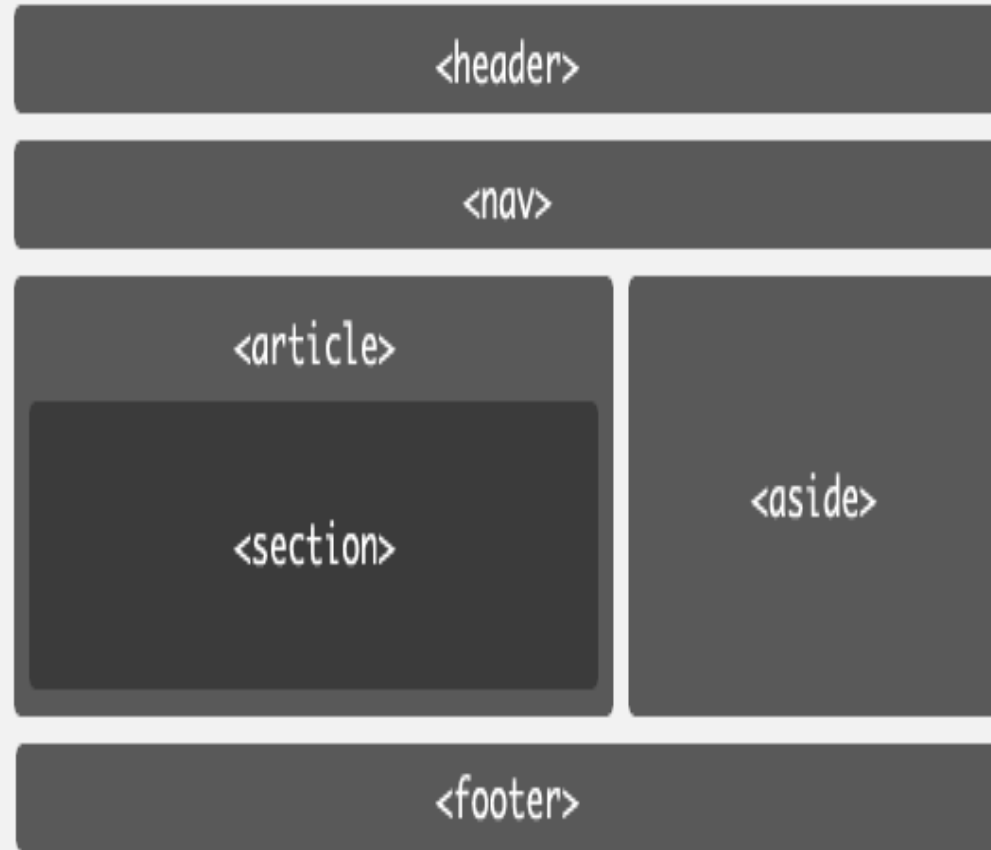


```
<!DOCTYPE html>
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <section>...</section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

```

<!doctype html>
<html>
<head>
    <title>Page title</title>
</head>
<body>
    <header>
        <h1>Page title</h1>
    </header>
    <nav>
        <!-- Navigation -->
    </nav>
    <section id="intro">
        <!-- Introduction -->
    </section>
    <section>
        <!-- Main content area -->
    </section>
    <aside>
        <!-- Sidebar -->
    </aside>
    <footer>
        <!-- Footer -->
    </footer>
</body></html>

```



# L'élément *header*

- L'élément **<header>** représente l'en-tête du document. Les en-têtes peuvent contenir plus qu'un simple titre de la page/un logo, un formulaire de recherche, un titre.
- Il peut être utilisé pour introduire la page comme pour introduire une autre section plus tard dans

**Exemple :**

```
<header>  
    <h1>A Preview of HTML 5</h1>  
</header>  
<p>The rest of my home page...</p>  
...
```



# <footer> et <nav>

- L'élément **<footer>** 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 de copyright et autres données du même type.

– **Exemple**

- L'élément

```
<nav>
  <a href="default.php">Home</a>
  <a href="tag_meter.php">Previous</a>
  <a href="tag_noscript.php">Next</a>
</nav>
```

liens de navigation. Il convient à la fois pour la navigation dans le site ou une table des

# <aside>

- L'élément **<aside>** est typiquement utile pour baliser des barres latérales.
- Il est destiné au contenu indirectement lié à l'article lui-même, il représente ce qui l'entoure comme par exemple une barre latérale d'archives.

```
<aside>
<h1>Archives</h1>
<ul><li><a href="/archives/12/05
/">
    Mai 2012</a></li>
<li><a href="/archives /12/0
6/">
    Juin 2012</a></li>
<li><a href="/archives /12/0
7/">
    Juillet 2012</a></li></ul>
</aside>
```

```
<!DOCTYPE HTML>
<body>
<p>Me and my family visited
The Epcot center this
summer.</p>
<aside>
<h4>Epcot Center</h4>
The Epcot Center is a theme
park in Disney World,
Florida.
</aside>
</body></html>
```

# <article>

- L'élément **<article>** représente un texte comme par exemple un article de journal, de blog ou de forum.

```
<article>
<p><a href="http://www.alsacreations.com/">
XHTML est mort, vive HTML5 !</a><br />
Sous ce titre quelque peu provocateur (et
faux) se cache une réalité officielle depuis
hier soir : le W3C vient d'annoncer que ses
travaux sur HTML5 se termineront en 2014.</p>
</article>
```

# L'élément <section>

- L'élément **<section>** permet de définir les différentes sections d'un document comme par exemple les chapitres, les en-tête et pied-de-page, ou toute autre section dans un document.
- Il peut être combiné avec les éléments h1, h2, h3, h4, h5, et h6 pour une meilleure définition de la structure du document.

## **Exemple :**

```
<article>
<header><h1>Welcome</h1></header>
<section>
<h4>What We Do</h4>
<p>We protect sharks...</p>
</section>
</article>
```

# Pourquoi de telles balises?

- L'introduction des éléments **Header, nav, aside, footer, article** et **section** permet de remplacer l'utilisation de l'élément `div`
- **Les nouvelles balises ont un sens (sémantique)!!!**
- **Avantage: Améliorer la recherche dans les pages et mieux évaluer le contenu d'une page.**
- **Concrétiser la vision Web 3.0**

- Nouvelles API JavaScript

# Local storage

- Les serveurs utilisent le mécanisme de session/cookies pour reconnaître les utilisateurs revenant à leurs sites
  - Les informations sont stockées en tant que cookies sur la machine de l'utilisateur
- Problèmes avec les cookies
  - Leurs tailles est limité (4k max/cookie)
  - Manque de fiabilité : elles peuvent être effacées par l'utilisateur
  - Doivent être ré-envoyer au serveur à chaque accès pour que le serveur reconnait l'utilisateur accédant à des pages différentes
- Plusieurs propositions de chez Adobe (Flash6),  
Microsoft (IE 7), Google (Chrome), Mozilla (Firefox)

# Local storage ...

```
<!DOCTYPE html>
<html>
<body>

<div id="result"></div><br/>
<div id="resultmod"></div>
<script>
    localStorage.lecture="Web";
    document.getElementById("result").innerHTML="Lecture: " +
localStorage.lecture;
    localStorage.setItem('lecture','Web 3.0');
var  x= localStorage.getItem('lecture');
    document.getElementById("resultmod").innerHTML="Lecture
modified : " + x;
</script>
</body></html>
```



# Session Storage

- sessionStorage vs. localStorage ??
- L'objet `sessionStorage` a le même but que l'objet `localStorage`, à une exception prêt:
  - `sessionStorage` garde les données le long d'une session du navigateur seulement,
  - `localStorage` garde les données indéfiniment jusqu'à ce que l'utilisateur les effacent en appelant la fonction `clear()`.

# Web Workers

- Il arrive souvent que le navigateur se plante car un certain script tourne en arrière plan
- Les Web workers ont été proposé pour permettre que des scripts JS tournent en arrière plan sans bloquer le navigateur pendant que vous puissiez naviguer dans la fenêtre en toute liberté
- <http://html5demos.com/worker>
- [http://www.w3schools.com/html/tryit.asp?filename=tryhtml5\\_webworker](http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_webworker)

# Offline web applications

- HTML5 application cache permet d'accéder à une application Web même en mode Offline
- Ceci aura pour avantage :
  - Navigation optimisée vu que seulement le contenu qui a été sujet à des MAJ sera chargé à partir du serveur
  - Navigation plus rapide vu que les ressources stockées en local sont chargées plus rapidement
  - Navigation en mode Offline.

# Geolocation

- HTML5 dispose d'une API qui permet la géolocalisation du navigateur du client moyennant une fonction JS **getLocation()** qui renvoie la position actuelle (longitude+latitude)
- [http://www.w3schools.com/html/tryit.asp?filename=tryhtml5\\_geolocation](http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_geolocation)
- <http://html5demos.com/geo>
- L'API de géolocalisation du W3C n'est pas

# Enfin

- <http://caniuse.com/#cats=HTML5>
  - Ce site indique les éléments HTML5 supportées par les différentes version des navigateurs à ce jour
- <http://html5demos.com/>
  - Un ensemble de DEMO des nouvelles API de HTML5
- <http://refcardz.dzone.com/refcardz/html5-new-standards-web-interactivity>
  - Un résumé des différentes API et leurs fonctions( pdf)

# Test de HTML5

- Pour voir si votre navigateur supporte HTML5 et savoir quelles élément/attributs sont supportés :

<http://html5test.com/>

- Pour une description détaillée des balises de HTML5:

<http://www.w3schools.com/html5/default.asp>

# HTML5 support via code

- Pour voir si la navigateur qui ouvre la page supporte un certaine balise/attribut, il y a :
  - **Moderniz** : an open source, MIT-licensed JavaScript library
  - Pour lui faire appelle, il suffit de rajouter l'appel du script suivant à vos pages

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script
src="modernizr.min.js"></script
>
</head>
<body>
...</body></html>
```

# HTML5 support via code...

- Lorsqu'elle s'exécute, cette librairie va créer un objet global appelé **modernizr** au quel est associé un ensemble de propriétés avec des valeurs booléennes.
- Si votre navigateur supporte le canvas API,

```
Modernizr.canvas() {  
  // let's draw some shapes!  
} else {  
  // no native canvas support  
  available :(  
  }
```



# HTML5 support via code...

- Tester la fonction Javascript qui permet de manipuler l'élément html5 en question comme :

```
function supports_canvas() {  
    return  
    !!document.createElement('canvas').getContext;  
}
```

```
function supports_video() {  
    return  
    !!document.createElement('video').canPlayType;  
}
```

# Les nouvelles balises de structuration de texte

- HTML 5 a introduit 20 nouvelles <balises>.

Tag	Description
<article>	For external content, like text from a news-article, blog, forum, or any other content from an external source
<aside>	For content aside from the content it is placed in. The aside content should be related to the surrounding content
<command>	A button, or a radiobutton, or a checkbox
<details>	For describing details about a document, or parts of a document
<summary>	A caption, or summary, inside the details element
<figure>	For grouping a section of stand-alone content, could be a video
<figcaption>	The caption of the figure section
<footer>	For a footer of a document or section, could include the name of the author, the date of the document, contact information, or copyright information
<header>	For an introduction of a document or section, could include navigation

# Les nouvelles balises de structuration de texte (cont.)

Tag	Description
<code>&lt;hgroup&gt;</code>	For a section of headings, using <code>&lt;h1&gt;</code> to <code>&lt;h6&gt;</code> , where the largest is the main heading of the section, and the others are sub-headings
<code>&lt;mark&gt;</code>	For text that should be highlighted
<code>&lt;meter&gt;</code>	For a measurement, used only if the maximum and minimum values are known
<code>&lt;nav&gt;</code>	For a section of navigation
<code>&lt;progress&gt;</code>	The state of a work in progress
<code>&lt;ruby&gt;</code>	For ruby annotation (Chinese notes or characters)
<code>&lt;rt&gt;</code>	For explanation of the ruby annotation
<code>&lt;rp&gt;</code>	What to show browsers that do not support the ruby element
<code>&lt;section&gt;</code>	For a section in a document. Such as chapters, headers, footers, or any other sections of the document
<code>&lt;time&gt;</code>	For defining a time or a date, or both
<code>&lt;wbr&gt;</code>	Word break. For defining a line-break opportunity.

# Obsolete and deprecated elements

- Ces balises figuraient dans HTML4.01 mais plus dans HTML5
  - `<acronym>`
  - `<applet>`
  - `<basefont>`
  - `<big>`
  - `<center>`
  - `<dir>`
  - `<font>`
  - `<frame>`
  - `<frameset>`
  - `<noframes>`
  - `<strike>`
  - `<tt>`

# Promesses de HTML5

- Usage extensif de Javascript (et ses différentes librairies intégré dans HTML5) et de AJAX pour se rapprocher des desktop applications en terme de fonctionnalités.
- Ajout de nouvelles balises ayant une sémantique
  - Indexation beaucoup plus facile des sites par les moteurs de recherche (Google).
  - Recherche plus efficace.
- En HTML4.01, tout est de type string, des

# Promesses de HTML5

- Parfaite séparation entre le contenu et la forme
  - En séparant plusieurs éléments comme : `<big>`, `<font>`, `<strike>`, `<u>`, `<center>`, ...
- Haut niveau d'interopérabilité
  - Même comportement peu importe le navigateur utilisé (avant l'API DOM différait d'un navigateur à un autre)
- Accès universel au document de divers périphériques, de diverses plateformes