

Ecole Centrale de Lyon

UE INF tc3

Projet d'Application Web

*Applications web,
mise en œuvre côté client*

René Chalon
Rene.Chalon@ec-lyon.fr

Daniel Muller
Daniel.Muller@ec-lyon.fr



4. Applications web, mise en œuvre côté client

Plan de la séance

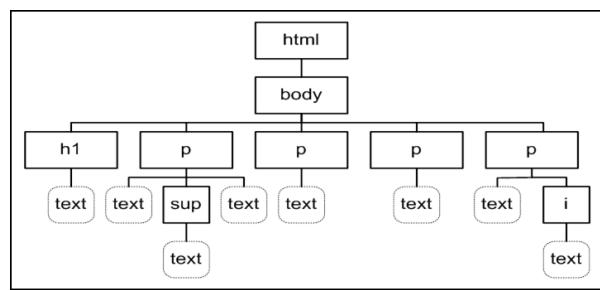
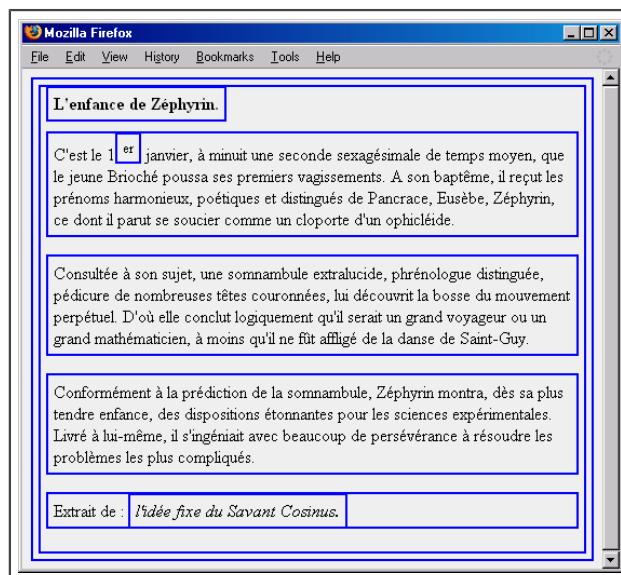
- 4.1. Introduction à HTML et CSS
 - 4.1.1. Éléments et attributs HTML
 - 4.1.2. Le DOM (*Document Object Model*)
 - 4.1.3. Propriétés CSS
 - 4.1.4. Éléments d'histoire
- 4.2. Description de l'interface (*HTML*)
 - 4.2.1. Structure d'un document
 - 4.2.2. Éléments inline
 - 4.2.3. Listes
 - 4.2.4. Tables
 - 4.2.5. Formulaires
- 4.3. Présentation (CSS)
 - 4.3.1. Règles, déclarations et sélecteurs
 - 4.3.2. Spécificité des sélecteurs
 - 4.3.3. Unités CSS et propriétés de mise en page
 - 4.3.4. Présentation des boîtes
 - 4.3.5. Présentation du texte
 - 4.3.6. Dimensions et position des boîtes
- 4.4. Interactivité (*Javascript*)
 - 4.4.1. Manipulation du DOM
 - 4.4.2. Gestion des événements
 - 4.4.3. Requêtes AJAX



4.1 Introduction à HTML et CSS

4.1.1 Éléments et attributs HTML

Un navigateur affiche des objets emboîtés appelés « éléments » que l'on peut représenter sous la forme d'un arbre :



HTML (*Hypertext Markup Language*) est un format texte qui permet de décrire une telle arborescence.

Dans le texte, le début et la fin d'un élément sont matérialisés à l'aide de « balises » (*tags*) ouvrantes et fermantes, qui doivent être correctement emboîtées :

```
<html>
  <body>
    <h1>L'enfance de Zéphyrin</h1>
    <p>C'est le 1<sup>er</sup> janvier ...</p>
    <p>Consultée à son sujet, ...</p>
    <p>Conformément à la prédiction ...</p>
    <p>Extrait de : <i>...</i></p>
  </body>
</html>
```

Attributs

HTML permet également de préciser le sens ou le fonctionnement attendu de certains éléments à l'aide d'attributs.



Les attributs sont matérialisés sous la forme d'une expression `nom="valeur"` portée par la balise ouvrante.

```
<html lang="fr">
  <body>
    <h1 class="title">L'enfance de Zéphyrin</h1>
    <p id="intro">C'est le 1er janvier ...</p>
    <p>Consultée à son sujet, ...</p>
    <p>Conformément à la prédiction ...</p>
    <p class="cite">Extrait de : <i>...</i></p>
  </body>
</html>
```

N.B. De nombreux attributs HTML permettent également d'indiquer la présentation souhaitée pour l'élément. Ces attributs sont obsolètes et leur usage n'est pas une bonne pratique.

Ainsi, HTML est réservé au contenu sémantique. Pour la présentation on préfère recourir aux feuilles de style CSS.

4.1.2 Le DOM (*Document Object Model*)

L'arborescence des éléments tels que manipulés par le navigateur peut être observée à l'aide des « outils du développeur » disponibles sur la plupart des clients (*PC/Mac/Linux - sauf tablettes et smartphones*).

(démo live)

4.1.3 Propriétés CSS

Chaque élément possède de très nombreuses propriétés (*ou attributs au sens de l'orientation objet - ils sont 220 dans Firefox au 17/11/15*) auxquelles le navigateur doit associer une valeur pour pouvoir représenter l'élément (*police de caractères, couleurs, marges, bordure, position, ombres...*).

En l'absence de précision donnée par l'auteur, chacune de ces propriétés se voit affecter une valeur par défaut qui peut être :

- caractéristique de la propriété (*par défaut, la couleur de fond d'un élément est transparente*),
 - héritée de l'élément parent (*par défaut, la couleur du texte est héritée de l'élément parent*),
 - issue des bonnes pratiques (*par défaut, les navigateurs affichent du texte noir sur fond blanc*).

Une feuille de style permet d'associer une valeur autre que la valeur par défaut aux propriétés de présentation des éléments afin d'obtenir le rendu désiré.

Feuilles de style CSS

L'usage d'une feuille de style CSS permet de modifier drastiquement l'affichage d'une page HTML :

Exemple

Avec ou sans feuille de style

Ce document peut être affiché **avec** ou **sans** feuille de style,
ou avec une feuille de style **au choix**.

Exemple de code CSS

```
BODY {  
    border: solid 2px blue;  
    padding: 5px;  
}
```

Exemple

Avec ou sans feuille de style

Ce document peut être affiché **avec** ou **sans** feuille de style,
ou avec une feuille de style **au choix**.

Exemple de code CSS

```
BODY {  
    border: solid 2px blue;  
    padding: 5px;  
}
```

© Daniel Muller

mercredi 11/11/0



Exemple

Avec ou sans feuille de style

Ce document peut être affiché avec ou sans feuille de style, ou avec une feuille de style au choix.

Exemple de code CSS

```
BODY {  
    border: solid 2px blue;  
    padding: 5px;  
}
```

© Daniel Muller mercredi 11/11/09

Exemple

Avec ou sans feuille de style

Ce document peut être affiché avec ou sans feuille de style, ou avec une feuille de style au choix.

Exemple de code CSS

```
BODY {  
    border: solid 2px blue;  
    padding: 5px;  
}
```

© Daniel Muller mercredi 11/11/09

(démonstration)

Voir aussi : [\[CSS Zengarden\]](#)

4.1.4 Éléments d'histoire

HTML

- 1992, **HTML 1.0** - Aux débuts du Web HTML est essentiellement sémantique.
- fin 1995, **HTML 2.0** est la première version formellement spécifiée. [\[HTML2\]](#)
- début 1997, **HTML 3.2** introduit de nombreux éléments et attributs de mise en forme (*i.e. non sémantiques*) et crée le mélange des genres entre forme et contenu. [\[HTML32\]](#)
- 1998, **HTML 4.0** est nettement plus complexe. Il rend obsolètes les éléments et attributs de mise en forme (*inutiles avec CSS*) ce qui produit les versions HTML strict vs. transitionnel. [\[HTML401\]](#)
- 2000, **XHTML 1.0** est une tentative d'imposer une syntaxe plus rigoureuse compatible avec XML. [\[XHTML\]](#)
- 2014, **HTML5** consacre la résistance du marché à la branche XHTML. [\[HTML5\]](#)

Pour en savoir plus : [Histoire de HTML](#)

N.B. HTML5 peut être sérialisé en « HTML » ou en « XML » (à la XHTML).

N.B.2 Cf. [Can I Use](#) pour un comparatif des implémentations HTML5.

CSS

- fin 1996, apparaît **CSS 1.0** dont les implémentations complètes sont tardives (*le premier est IE5 en 2000, rapidement suivi par les autres navigateurs*). [\[CSS1\]](#)
- 1998, **CSS 2.0** n'a jamais été complètement ni correctement implémenté. [\[CSS2\]](#)
- 2002, **CSS 2.1** sous forme d'un Working Draft. Il s'agit d'une révision (*simplifications*) de CSS 2.0 qui supprime certaines fonctionnalités et en précise d'autres. Après une histoire compliquée, CSS 2.1 est recommandé en juin 2011... Corrections en cours pour former CSS 2.2. [\[CSS2.1\]](#)
- à partir de 2011, **CSS 3.0** devient modulaire. Chaque module possède sa propre histoire et son propre niveau. [\[Instantané CSS 2015\]](#)

[\[Comprendre les Specs CSS\]](#)

Les fonctionnalités CSS3 implémentées par les navigateurs actuels sont souvent associées à HTML5 (*nouveaux sélecteurs, coins arrondis, texte ombré, boîtes ombrées, affichage multi-colonnes, polices téléchargeables...*)

Pour un comparatif des implémentations CSS au sein des navigateurs, cf. [\[Quirksmode\]](#)



4.2 Description de l'interface (HTML)

4.2.1 Structure d'un document

Syntaxe XHTML

La syntaxe d'un document XHTML est conforme aux règles générales de XML (*eXtended Markup Language*) :

- noms des éléments et attributs en minuscules,
- valeurs des attributs délimitées par des guillemets " ",
- balises fermantes obligatoires même pour les éléments vides (*i.e. sans contenu*) :

```
<br></br> ou <br/>
</img> ou 
```

- éléments correctement imbriqués,
- un seul élément racine, nommé `<html>`,
- caractères spéciaux échappés par un appel d'entité :

```
<          &lt;
&          &amp;
"          &quot;
'          &apos;
>          &gt;
```

Structure d'un document XHTML5

Un document HTML5 conforme aux spécifications, sérialisé (*écrit*) en XHTML respecte le gabarit suivant :

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">

    <head>
        <title>Titre du document</title>
        <link rel="stylesheet" href="style.css" type="text/css"/>
        <meta charset="utf-8" />
    </head>

    <body>Contenu du document (balises et texte)</body>

</html>
```

Le titre, le contenu, et la référence à la feuille de style sont spécifiques à chacun des documents.
[exemple]

Noter l'existence de deux sections consécutives et bien distinctes :

- `<head>` donne des informations au sujet du document dont certaines sont obligatoires,
- `<body>` contient le contenu du document visible par l'utilisateur.



Syntaxe HTML5

Bien que le code source paraisse plus simple, et contrairement à ce qui est communément admis, HTML est plus complexe à appréhender que XHTML.

En effet, certaines balises peuvent être omises (*certaines fermantes, d'autres ouvrantes et fermantes*). Elles sont néanmoins implicites et les éléments sont bien présents dans l'arbre (*DOM*).

```
<!DOCTYPE html>

<title>Titre du document</title>
<link rel="stylesheet" href="style.css" type="text/css">
<meta charset="utf-8">
```

Contenu du document (balises et texte)

N.B. Les balises ouvrantes et fermantes des éléments `<html>`, `<head>` et `<body>` sont absentes, ainsi que les balises fermantes de `<link>` et `<meta>`, mais ces éléments sont présents dans l'arbre, ce que l'on peut vérifier par exemple avec DOM inspector. [\[exemple\]](#)

Eléments structuraux

Avant HTML5, le découpage structurel d'un document s'effectuait avec des balises que nous qualifierons de graphico-sémantiques. En effet :

- elles donnent du sens (*titre, sous-titre, paragraphe...*),
- elles possèdent une présentation par défaut (*même sans feuille de style*).

Ces balises permettent d'asseoir la structure générale d'un document :

```
<h1>Titre de niveau 1</h1>
```

☞ Pour les sous-titres il existe également les balises de `<h2>` à `<h6>`...

```
<p>Paragraphe</p>
```

☞ La plupart du temps, la balise fermante `</p>` peut être omise.

```
<pre>Paragraphe préformatté</pre>
```

☞ Texte préformatté. La présentation par défaut respecte le nombre, la nature, et la position des espaces (*blancs, tabulations, passages à la ligne*).

```
<blockquote>Paragraphe en exergue</blockquote>
```

```
<div>Bloc non sémantique</div>
```

[\[exemple\]](#)

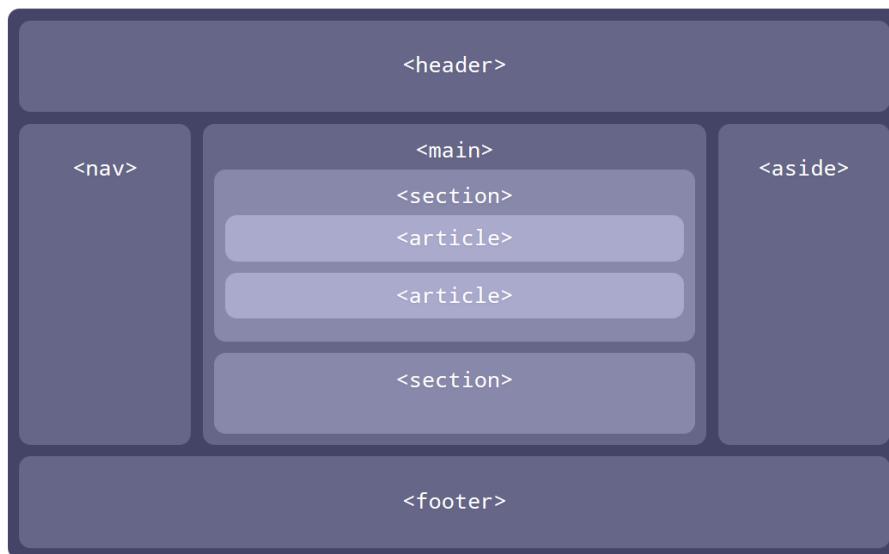


HTML5 a par ailleurs introduit de nombreux éléments structuraux purement sémantiques (*sans feuille de style leur contenu apparaît à l'écran, mais sans aucune mise en forme particulière*) :

Élément	Description
<header>	Utilisé pour regrouper les éléments formant l'entête des sections d'un document (<i>section, article, le document lui-même</i>). Correspond typiquement à l'entête d'un site qui apparaît sur toutes les pages, ou celle d'une grosse section ou d'un long article.
<footer>	Contient les informations de bas de page d'un site, d'un article ou d'une section.
<nav>	Destiné à contenir les principaux liens de navigation d'une page (<i>internes ou externes</i>).
<article>	Contenu indépendant qui pourrait être extrait du document sans perdre son sens (<i>article de blog, par exemple</i>).
<section>	Section générique permettant de regrouper différents articles, ou pour définir les sections d'un article. Comprend généralement une entête.
<aside>	Section dont le contenu est peu corrélé au reste du document, mais apportant des informations supplémentaires.
<figure> <figcaption>	Respectivement destinés à contenir une figure (<i>image</i>) et sa légende, et sa légende.
<main>	Indique le contenu principal d'un document ou d'une application. Il ne peut y avoir qu'un seul élément main au sein d'un document.

Éléments structuraux introduits par HTML5

Voici une disposition (*layout*) possible pour ces éléments. Ce n'est pas la seule : il n'y a pas de présentation par défaut pour ces éléments purement sémantiques.



Notez bien qu'il s'agit d'éléments **sémantiques** : il ne faut pas les utiliser pour être à la mode ou respecter une quelconque spécification, il faut les utiliser pour **donner du sens**.

Par conséquent, si cela donne du sens à un document :

- Une section peut contenir des articles.
- Un article peut contenir des sections.
- Un article ou une section peuvent contenir un bloc de navigation (*nav*).
- Un article ou une section peuvent contenir une entête (*header*) et un pied (*footer*).
- Un article ou une section peuvent contenir un ou plusieurs encarts (*aside*).

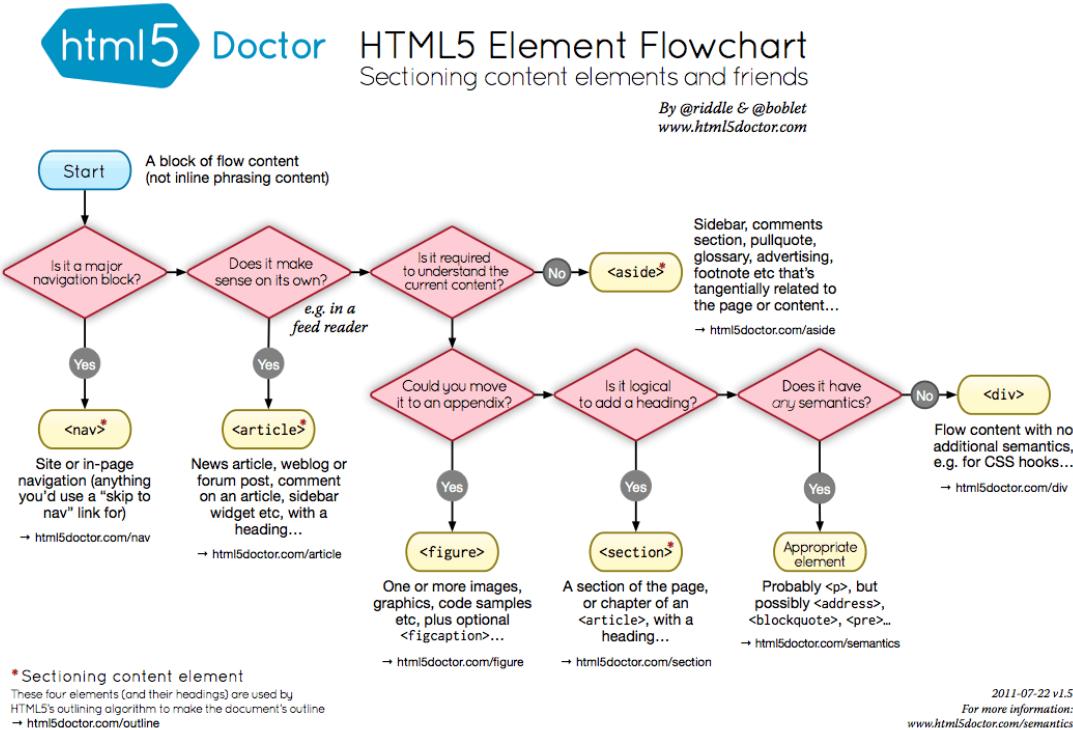
Les éléments `<section>`, `<article>`, `<nav>` et `<aside>` sont des éléments structurants (*sectioning elements*) qui définissent un découpage hiérarchique du document. [En savoir plus]



Les bonnes pratiques imposent que :

- chacun de ces éléments comporte un titre `<h1>` à `<h6>`,
- le niveau des titres corresponde au niveau d'emboîtement des éléments structurants.

Comment décider quel est le meilleur élément bloc à utiliser ?



© 2015 HTML5 Doctor, [CC NC-BY 2.0 (<http://creativecommons.org/licenses/by-nc/2.0/uk/>)]

4.2.2 Éléments inline

Tous les éléments vus jusqu'ici sont des **éléments bloc** (*block-level elements*).

La mise en page par défaut d'un élément bloc consiste pour le navigateur à :

- lui affecter une boîte rectangulaire,
- affecter à la boîte la largeur maximale autorisée par le contexte (*largeur de la fenêtre ou du bloc contenant*),
- empiler la boîte verticalement dans la page ou le bloc contenant.

N.B. D'autres éléments bloc comme les listes, `<table>` ou `<form>` seront vus plus loin.

Il existe une autre catégorie d'éléments appelée **inline** qui sont affichés en leur affectant une boîte dont la largeur correspond à celle de leur contenant, qui sera mise en page dans le flot du texte.

Ces éléments (*inline elements*) permettent typiquement de préciser le sens d'une petite portion de texte ou de modifier ses caractéristiques typographiques (*gras, italiques...*), de définir un lien, ou d'insérer une image.

[Spécification des éléments inline]



Panorama de quelques éléments inline

Permet de mettre un mot ou une partie de phrase en exergue.

```
le <em>chocolat</em> est bon pour la santé  
le chocolat est <em>bon</em> pour la santé  
le chocolat est bon pour la <em>santé</em>
```

le *chocolat* est bon pour la santé
le chocolat est *bon* pour la santé
le chocolat est bon pour la *santé*

☞ La position exacte de cet élément peut changer le sens de la phrase...

Indique un point important, sérieux, ou urgent.

```
Le chocolat est <strong>bon pour la santé</strong>
```

Le chocolat est **bon pour la santé**

☞ Cet élément ne modifie pas le sens de la phrase.

<small>

Est utilisé pour une note, une restriction, un avertissement, un copyright...

```
Pour avoir une bonne note il faut lire le cours <small>(et les références)</small>.
```

Pour avoir une bonne note il faut lire le cours (et les références).

☞ Doit être réservé à de courts extraits...

<cite>

Indique l'auteur, le titre ou l'URL d'une oeuvre, d'un ouvrage, d'un article...

```
<cite>Tim Berners-Lee</cite> est co-auteur de la <cite>RFC 1866</cite>.
```

Tim Berners-Lee est co-auteur de la RFC 1866.

☞ Ne doit pas être utilisé pour une *citation*, qui est un *extrait* d'une œuvre.

<q>

Délimite une citation provenant d'une autre source.

```
<q>Toute technologie suffisamment avancée est indiscernable de la magie</q>  
<cite>Arthur C. Clarke</cite>
```

« *Toute technologie suffisamment avancée est indiscernable de la magie* » Arthur C. Clarke

☞ Les guillemets autour de la citation sont censés être ajoutés par le navigateur. Dans la pratique on aura recours à CSS pour cela.

<dfn>

Indique un terme en cours de définition.

```
<dfn>HTML</dfn> est le format de données conçu pour représenter les pages web.  
<cite>(Wikipédia)</cite>.
```

HTML est le format de données conçu pour représenter les pages web. (*Wikipédia*).

☞ Le bloc parent de <dfn> doit donner la définition du terme considéré.



<abbr>

Utilisé pour signaler une abréviation ou un acronyme.

```
<abbr title="eXtended Markup Language">XML</abbr> est un méta-langage.
```

XML est un méta-langage.

- ☞ Il est d'usage que <abbr> possède un attribut `title` donnant la signification du contenu, qui sera affichée lors du survol de l'élément par le curseur.

<code>

Représente un fragment de code informatique (*balise HTML, nom de fichier, variable*).

```
<code>&lt;code&gt;</code> est une balise HTML.
```

<code> est une balise HTML.

- ☞ Lorsqu'un élément <pre> contient du code, ce dernier peut être délimité par <code>.

<var>

Sert à identifier un nom de variable.

```
<code><var>x</var> = 33</code>
```

x = 33

<sup>

Délimite un texte mis en exposant.

```
a <var>x</var><sup>2</sup> + b <var>x</var> + c = 0
```

$a x^2 + b x + c = 0$

<sub>

Délimite un texte mis en indice.

```
a <var>x<sub>0</sub></var><sup>2</sup> + b <var>x<sub>0</sub></var> + c = 0
```

$a x_0^2 + b x_0 + c = 0$

<i>

Délimite une partie de texte destinée à apparaître dans un rendu différent.

```
L'araignée de la page de garde est une <i>Agelenidae labyrinthica</i>.
```

L'araignée de la page de garde est une *Agelenidae labyrinthica*.

- ☞ Cet élément est traditionnellement mis en italiques. Il est toutefois possible avec CSS d'obtenir n'importe quel rendu désiré, y compris sans italiques...

Délimite une partie de texte destinée à apparaître dans un rendu différent, sans pour autant modifier son degré d'importance.

```
Les <b>auditeurs du fond</b> ont l'air ailleurs...
```

Les **auditeurs du fond** ont l'air ailleurs...

- ☞ Le rendu traditionnel de cet élément est effectué avec des caractères gras. Il est toutefois possible avec CSS d'obtenir n'importe quel rendu désiré.



Délimiteur générique.

```
Une page web n'est pas exempte de <span class="draggable">magie</span>.
```

Une page web n'est pas exempte de magie.

- ☞ Cet élément n'a pas de rendu par défaut particulier. Son seul intérêt est de pouvoir porter des attributs tels que `class` ou `id`.

Représente un passage à la ligne.

```
<address>  
136, av. Guy de Collongue<br/>  
69131 Ecully  
</address>
```

136, av. Guy de Collongue
69131 Ecully

- ☞ `
` n'a pas de contenu (*c'est un élément vide*). Il doit être utilisé uniquement lorsque le saut de ligne fait sémantiquement partie du texte. Pas pour obtenir visuellement un changement de paragraphe.

<a>

Définit un lien hypertexte.

```
<a href="http://www.w3.org">The World Wide Web Consortium</a>
```

The World Wide Web Consortium

- ☞ L'attribut `href` donne l'URL de la ressource liée. Le contenu identifie la zone active.

Permet d'insérer une image.

```
<figure>  
  
<figcaption>fig.1 Sir Tim Berners-Lee à la conférence WWW2012 à Lyon</figcaption>  
</figure>
```

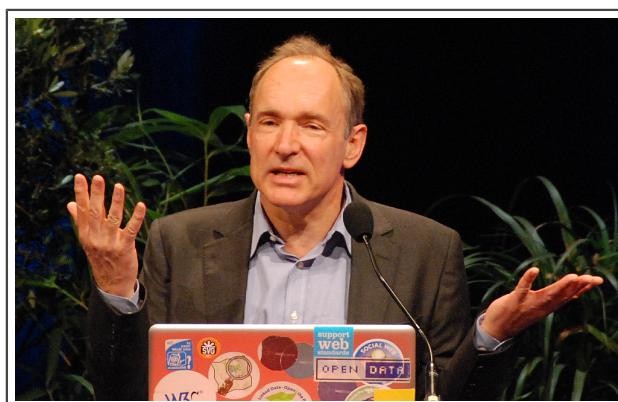


fig.1 Sir Tim Berners-Lee à la conférence WWW2012 à Lyon
Photo © D. Muller (Own work)

- ☞ L'attribut `src` donne l'URL de l'image. L'attribut `alt` est obligatoire. Il sert à donner une alternative textuelle à l'image pour les clients non visuels. Si une telle alternative n'est pas pertinente (*image purement décorative*) l'attribut doit être présent mais vide.



4.2.3 Listes

Liste à puces (*unordered list*).

```
<ul>
<li>primo ...
<li>secundo ...
<li>tertio
</ul>
```

- primo ...
- secundo ...
- tertio

☞ Les items de liste `` peuvent contenir du texte, des éléments inline, et des éléments bloc y compris une liste imbriquée.

CSS permet de jouer sur le style de la puce, y compris en fournissant une image propriétaire.

Liste ordonnée (*numérotée*).

```
<ol>
<li>primo ...
<li>secundo ...
<li>tertio
</ol>
```

1. primo ...
2. secundo ...
3. tertio

☞ Les items de liste `` peuvent contenir du texte, des éléments inline, et des éléments bloc y compris une liste imbriquée.

CSS permet de jouer sur le style de la numérotation (*chiffres arabes, romains majuscules ou minuscules, lettres majuscules ou minuscules, alphabet grec, numérotation arménienne, géorgienne...*).

Certains navigateurs supportent d'autres styles de numérotation non recommandés par les spécifications (*hébreux, katakana, bengali, ethiopien, thai...*). [\[Styles de liste Mozilla\]](#)

<dl>

Liste descriptive.

```
<dl>
<dt>hier</dt>
<dd>la veille d'aujourd'hui</dd>
<dt>aujourd'hui</dt>
<dd>le lendemain d'hier, et la veille de demain</dd>
</dl>
```

hier

la veille d'aujourd'hui

aujourd'hui

le lendemain d'hier, et la veille de demain

☞ Les items de liste `<dt>` et `<dd>` peuvent contenir du texte, des éléments inline, et des éléments bloc y compris une liste imbriquée.

[\[Listes imbriquées\]](#)



4.2.4 Tables

<table>

Permet de représenter de l'information sous forme d'une table (*ou tableau*).

```
<table>
  <caption>jeu du morpion</caption>
  <tr>
    <td>X</td> <td> </td> <td>O</td>
  </tr>
  <tr>
    <td> </td> <td>X</td> <td> </td>
  </tr>
  <tr>
    <td>O</td> <td> </td> <td>X</td>
  </tr>
</table>
```

X		O
	X	
O		X

jeu du morpion

☞ Le modèle de table HTML consiste à empiler des lignes <tr> constituées de cellules <td>.

Par défaut les bordures des lignes et colonnes sont invisibles, elles sont visibles ici grâce à CSS.

Bien qu'il ne s'agisse pas d'une pratique recommandée, les tables servent parfois à disposer les blocs sur la page (*layout*).

Pour des tables complexes il est possible de disposer des cellules sur plusieurs lignes et/ou colonnes :

Table 334.1 - Rapport du courant à vide au courant en charge pour des transformateurs de divers types et diverses puissances.

Catégories (Transformateurs triphasés)	kVA	I _v : I _{ch} en %			
		25 p : s		50 p : s	
		5 000 V	30 000 V	5 000 V	30 000 V
I. - Transformateurs à huile et refroidissement naturel (pertes normales)	1	32		28	
	5	25		21,5	
	10	22	35	18,5	28
	20	18,3	25	15	20,5
	50	13,7	17,2	10,8	14
	100	11,6	14,0	9,3	11
	≥ 250	9,5	11,5	7,6	9,3
II. - Transformateurs à huile et circulation d'eau	300		12,5		10,7
	1000		8,8		7,4
	≥ 2000		≤ 8,0		6,5
III. - Transformateurs à sec	1	28		22	
	10	18		14,3	
	50	8,5		6,8	
	100	7,5		5,8	

D'après :

H. Delalande - Essais des machines électriques - p 334
Librairie polytechnique Ch. Béranger - 1923

☞ Pour des raisons d'accessibilité aux clients non visuels, il est déconseillé d'utiliser ce type de cellules pour le layout.



Tabelle 100.1 - Expansionsgeschwindigkeit für einzelne Sterngruppen

Physischer Typ	Zahl der Sterne	$2 \bar{v}_R [\text{km/sec}]$
1. Überriesen der Typen B o I-B 3 I	11	8
2. Überriesen der Typen B 8 I-B 9 I	6	8
3. Überriesen der Typen A o I-A 5 I	8	17
4. Übrige Sterne vom Typ B	16	15
5. O-Sterne vom Typ B	4	36
6. Überriesen der Typen M o I-M 4 I	7	6

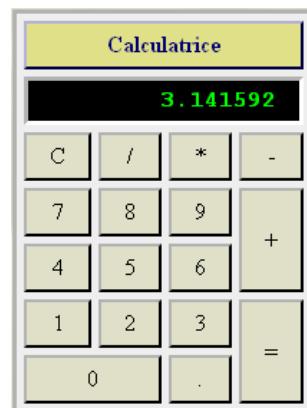
D'après :

V.A. Ambarzumjan - Über die Sternassoziation Perseus I
Armenische Akademie der Wissenschaften, Observatorium Bjurakan - 1958

Comme pour l'ensemble des éléments HTML, la présentation des tables est très largement dépendante des informations de mise en forme qui l'accompagnent.

Ces informations sont exprimées à l'aide du langage CSS qui fait l'objet du chapitre à venir.

```
<table>
<tr><th colspan="4">Calculatrice</th></tr>
<tr><th colspan="4">3.141592</th></tr>
<tr><td>C</td><td>/</td><td>*</td><td>-</td></tr>
<tr><td>7</td><td>8</td><td>9</td><td rowspan="2">+</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td rowspan="2">=</td></tr>
<tr><td colspan="2">0</td><td>.</td></tr>
</table>
```



[Exemples de tables en ligne]

4.2.5 Formulaires

<form>

<form> est un élément bloc qui définit le périmètre d'un formulaire. Un certain nombre d'attributs permettent de préciser son fonctionnement.

```
<form action="/service" method="POST">
<fieldset>
<legend>Coordonnées utilisateur</legend>
<label>Nom: <input name="Nom"></label><br>
<label>Prénom: <input name="Prenom"></label><br>
<input type="submit" value="Envoyer">
</fieldset>
</form>
```



Coordonnées utilisateur

Nom:

Prénom:

- ☞ L'attribut `action` donne l'URL de la ressource requise pour traiter les informations du formulaire.
L'attribut `method` peut valoir GET ou POST.
Il est nécessaire de recourir à CSS pour obtenir une mise en forme ergonomique des champs et labels de formulaires...

Champs de formulaire

N.B. Les éléments d'IHM communément appelés « *champs de formulaires* » (*inputs en anglais*) seront présentés ici dans le contexte des formulaires. Ils peuvent toutefois être également utilisés pour **contrôler une Webapp** et n'auront pas, dans ce cas, de parent `<form>`.

Le principe de fonctionnement des champs de formulaire est d'associer *une valeur* entrée par l'utilisateur, à *une clé* (*correspondant au nom du champ*) spécifiée par le code source. Lors de la soumission du formulaire, c'est cette association clé = valeur qui sera transmise au serveur pour traitement.

`<input type="text">`

Les champs les plus simples sont spécifiés à l'aide de l'élément `<input>`. Ce dernier propose de nombreuses déclinaisons en fonction de la valeur de son attribut `type`. Voici la plus simple :

```
Nom : <input type="text" name="nom">
Prénom : <input type="text" name="prenom">
```

Nom : Prénom :

- ☞ Ceci est aussi la fonctionnalité par défaut de l'élément `<input>` si l'attribut `type` est absent.

Attributs communs aux éléments de formulaires

Les *widgets* d'interaction que l'on peut placer dans un formulaire partagent les attributs suivants :

Attribut	Description
<code>autofocus</code>	Attribut minimisé. Le champ est automatiquement activé au chargement de la page. Réservé à un champ par page.
<code>disabled</code>	Attribut minimisé. Le champ est inactivé.
<code>name</code>	Nom de la clé à laquelle sera associée la valeur entrée par l'utilisateur lors de la transmission au serveur
<code>value</code>	Valeur initiale du champ au chargement de la page

Attributs communs aux champs de formulaire

En HTML5 les **attributs minimisés** peuvent ne pas prendre de valeur. Leur seule présence suffit :

```
<input type="text" name="nom" disabled>
```

- ☞ Attention, si le document est sérialisé en **XHTML5** une valeur est nécessaire :

```
<input type="text" name="nom" disabled="disabled">
```



Mise en page d'un formulaire

<form> est un élément bloc. A ce titre il peut contenir n'importe quel autre élément (*bloc ou inline*) propre à présenter correctement le contenu du formulaire. Techniquelement, il peut contenir des paragraphes, des listes, des tables... [Comment structurer un formulaire HTML]

<fieldset> et <legend>

L'élément <fieldset> permet de regrouper visuellement des champs de formulaire sémantiquement liés. <legend> donne un nom au groupe et indique à l'utilisateur la raison du regroupement.

```
<fieldset>
  <legend>Candidat</legend>
  <label>Nom: <input name="Nom_Candidat"></label><br>
  <label>Prénom: <input name="Prenom_Candidat"></label><br>
</fieldset>
<fieldset>
  <legend>Conjoint</legend>
  <label>Nom: <input name="Nom_Conjoint"></label><br>
  <label>Prénom: <input name="Prenom_Conjoint"></label><br>
</fieldset>
```

Candidat	Conjoint
Nom: <input type="text"/>	Nom: <input type="text"/>
Prénom: <input type="text"/>	Prénom: <input type="text"/>
Email: <input type="text"/>	Email: <input type="text"/>

<label>

Cet élément sert à indiquer le label associé à un champ de formulaire. Les exemples donnés jusqu'à présent créent une association implicite en incluant le champ à l'intérieur de l'élément <label> :

```
<label>Nom: <input name="Nom"></label>
```

On peut également expliciter l'association entre le label et le champ à l'aide de l'attribut `for` du label, qui doit renvoyer vers la valeur de l'attribut `id` du champ

```
<label for="Nom">Nom: </label><input name="Nom" id="Nom">
```

☞ Le label et le nom ne sont pas nécessairement immédiatement consécutifs comme dans cet exemple.

Dans certains cas (*sérialisation XHMLT5*) il est impératif que la valeur des attributs `id` et `name` soit la même. Pour cette raison il vaut mieux éviter de prendre des habitudes différentes...

L'un des intérêts d'indiquer explicitement le label réside dans le fait que l'utilisateur peut activer le champ en cliquant sur le label, ce qui est tout particulièrement utile dans le cas des cases à cocher (cf. *infra*).

Diversité des champs de formulaire

<input type="password">

Comme un champ texte, mais n'affiche pas la valeur saisie.

```
<label>Entrer votre mot de passe : <input type="password" name="pwd"></label>
```

Entrer votre mot de passe :



<input type="email">

Champ texte. L'intérêt réside dans le contrôle par le navigateur de la valeur saisie.

```
<label>Adresse : <input type="email" name="pwd"></label>
```

Adresse :

<input type="url">

Là encore, le navigateur contrôle automatiquement la validité de la saisie.

```
<label>Votre blog : <input type="url" name="blogsite"/></label>
```

Votre blog :

<input type="checkbox">

Case à cocher.

```
Vos préférences ?<br>
<label>Crème glacée : <input type="checkbox" name="vaille" value="oui"></label>
<label>Sauce chocolat : <input type="checkbox" name="chocolat" value="oui"></label>
<label>Topping amandes : <input type="checkbox" name="amandes" value="oui"></label>
```

Vos préférences ?

Crème glacée : Sauce chocolat : Topping amandes :

- ☞ Noter la présence de l'attribut obligatoire `value` qui permet de donner une valeur au champ lorsque la case est cochée. Si la case n'est pas cochée, la clé (*ou variable*) correspondante ne sera simplement pas transmise au serveur.

<input type="radio">

Case à cocher pour choix exclusifs.

```
Aimez-vous la choucroute ?
<label>Oui : <input type="radio" name="choucroute" value="oui"></label>
<label>Non : <input type="radio" name="choucroute" value="non"></label>
```

Aimez-vous la choucroute ? Oui : Non :

- ☞ Noter là aussi la présence de l'attribut `value`. L'appairage des cases, dont une seule sera cochée, s'effectue sur la base de l'attribut `name` dont la valeur est identique pour toutes les cases concernées.

<input type="color">

Propose un choix de couleur.

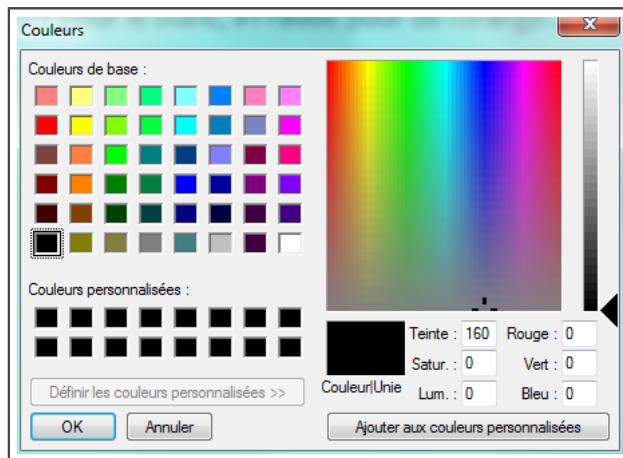
```
<label>Votre couleur préférée : <input type="color" name="couleur"></label>
```

Votre couleur préférée :

- ☞ La forme et l'ergonomie générale de ce widget dépendent largement du navigateur et du système sous-jacent, mais une chose est sûre : il permet de choisir une couleur !



Lorsque l'utilisateur sélectionne le champ, une fenêtre popup apparaît :



Popup de choix de couleur proposé par Firefox sous Windows.

- ☞ La valeur est transmise au serveur en utilisant la notation hexadécimale à 6 digits préfixée par le caractère #, classique en HTML.

Avec deux digits par couleur, dans l'ordre rouge, vert, bleu, on aura #000000 pour le noir, #ffffffff pour le blanc, #ff8000 pour de l'orange...

<input type="number">

Permet le choix d'un nombre, entier ou réel.

```
<label>Votre âge :<input type="number" name="age" min="18" max="99" step="1"></label>
```

Votre âge : 99

- ☞ Noter les attributs min, max et step permettant de caractériser la liste des valeurs autorisées.
Pour la sélection d'un réel il suffit de donner un pas fractionnaire.

<input type="range">

Permet le choix d'un nombre « à la louche ».

```
<label for="age">Vous vous sentez plutôt :</label>  
Jeune <input type="range" name="age" id="age" min="18" max="99" step="1"> Vieux
```

Vous vous sentez plutôt : Jeune ————— Vieux

- ☞ Noter là encore les attributs min, max et step qui permettent de caractériser la liste des valeurs autorisées.
Comme pour le widget de choix de couleur, la présentation varie beaucoup entre navigateurs et d'un système à l'autre.

<input type="hidden">

Champ invisible pour l'utilisateur, mais dont la valeur est transmise au serveur.

```
<form action="http://localhost:8080/service" method="POST">  
<input type="hidden" name="userid" value="catwoman">  
<input type="submit"/>  
</form>
```



☞ Ce type de champ est en général utilisé pour transmettre au serveur un jeton (*token, cookie*) permettant de gérer une session dans un contexte d'enchaînement de formulaires.

<input type="file">

Utilisé pour sélectionner un fichier à télécharger vers le serveur.

```
<form action="http://localhost/service" method="POST" enctype="multipart/form-data">
  <input type="file" name="fichier">
  <input type="submit">
</form>
```

No file selected.

☞ Lorsque ce champ est présent au sein d'un formulaire il faut **impérativement** que l'élément <form> porte les attributs :

- `method="POST"`
 - `enctype="multipart/form-data"`
- pour que le fichier soit téléchargé (*uploadé*) vers le serveur.

<input type="submit">

Matérialisé par un bouton permettant de soumettre le formulaire au serveur.

```
<input type="submit" name="soumission" value="choix 1"/>
<input type="submit" name="soumission" value="choix 2"/>
```

☞ L'attribut `value` permet de modifier le libellé du bouton.

En l'absence d'attribut `name` aucune information relative au bouton n'est passée au serveur. Si le champ est nommé, son nom et sa valeur sont transmises (*utile pour un formulaire avec plusieurs boutons de soumission*).

<input type="reset">

Bouton permettant de réinitialiser le contenu du formulaire.

```
<input type="reset" value="Réinitialiser">
<input type="submit" value="Envoyer">
```

Autres champs de formulaire

<button>

Bouton générique qui peut contenir du code html arbitraire, et soumet le formulaire par défaut.

```
<button name="button" value="smilie">
  Bouton Gai<br>
</button>
<button name="button" value="frownie">
  Bouton Triste<br>
</button>
```



☞ Accepte un attribut `type` qui peut prendre les valeurs "submit" (*valeur par défaut*), "reset" ou "button". Dans ce dernier cas le bouton ne fait rien (*pas de comportement par défaut*) sauf à l'équiper de code Javascript.

Permet de créer des boutons personnalisés avec n'importe quel contenu, et notamment du texte sur plusieurs lignes, des images, des caractères gras, des italiques...

<textarea>

Champ texte acceptant de la prose sur plusieurs lignes.

```
<label for="prose">Lâchez-vous : </label>
<textarea id="prose" name="prose">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ultrices cursus
semper. In vel gravida nisi. Sed ac tortor at mauris convallis ullamcorper ac
in sapien. Morbi eu metus auctor, accumsan eros non, efficitur massa. Aenean
rutm, risus eu accumsan egestas, ligula elit sollicitudin odio, sit amet
tincidunt eros ante nec tellus.
</textarea>
<input type="submit" value="Envoyer">
```

Lâchez-vous :

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras ultrices cursus semper. In vel gravida nisi. Sed ac tortor at mauris convallis ullamcorper ac in sapien. Morbi eu metus auctor, accumsan eros non, efficitur massa. Aenean rutm, risus eu accumsan egestas, ligula elit sollicitudin odio, sit amet tincidunt eros ante nec tellus.

☞ Accepte les attributs `rows` et `cols` pour fixer les dimensions initiales de l'affichage. Noter la poignée dans le coin inférieur droit permettant de redimensionner la zone.

Les attributs `minlength` et `maxlength` permettent d'imposer des contraintes sur le nombre de caractères entrés par l'utilisateur.

<select>

Affiche un widget proposant un choix parmi une liste d'items.

```
<select name="parfum">
  <option>Vanille</option>
  <option>Café</option>
  <option>Chocolat</option>
  <option>Framboise</option>
  <option>Fraise</option>
  <option value="Rhum">Rhum-Raisin</option>
</select>
```

Faites votre choix :

☞ Accepte un attribut `size` permettant de fixer la hauteur de la fenêtre de choix (*nombre de lignes*). Apprécier au passage comment le navigateur adapte l'interface graphique à la dimension disponible :

The image shows four separate dropdown menus side-by-side, each with a different size attribute set. The first menu has size="1" and contains the option 'Vanille'. The second menu has size="2" and contains 'Vanille' and 'Café'. The third menu has size="3" and contains 'Vanille', 'Café', and 'Chocolat'. The fourth menu has size="4" and contains 'Vanille', 'Café', 'Chocolat', and 'Framboise', with 'Fraise' and 'Rhum-Raisin' visible below the menu area.

Pour les gourmands, utiliser l'attribut `multiple`.

```
<select name="parfum" size="4" multiple> ... </select>
```

Faites votre choix :

The image shows a single dropdown menu with the size attribute set to 4. It contains the options 'Vanille', 'Café', 'Chocolat', and 'Framboise'. The menu is currently open, displaying these options vertically.



<option>

Sert à lister les items de <select>.

```
<option>Fraise</option>
<option value="Rhum">Rhum-Raisin</option>
```

☞ L'attribut `value` indique la valeur transmise au serveur lors de la soumission du formulaire. En l'absence de cet attribut, le navigateur utilise le contenu de l'élément.

<optgroup>

Sert à grouper les items de <select> pour un rendu visuel plus agréable dans le cas de listes longues.

```
<optgroup label="topping">
  <option value="topping_Amandes">Amandes</option>
  <option value="topping_Cacahuetes">Cacahuètes</option>
  <option value="topping_Chocolat">Pépites de chocolat</option>
</optgroup>
```

Faites votre choix :

4.3 Présentation (CSS)

4.3.1 Règles, sélecteurs et déclarations

Une *feuille de style* CSS est constituée d'une série ordonnée de **règles**.

Une *règle* CSS applique une liste de **déclarations** à un élément (*ou ensemble d'éléments*) désignés par un **sélecteur** :

```
h1, em, b {
  font-weight: bold;
  letter-spacing: 0.1em;
}
```

Une *déclaration* associe une **valeur** à une **propriété** :

```
font-weight: bold;
```

CSS et HTML

Une feuille de style CSS peut être attachée à un document HTML de plusieurs manières :

- Fichier indépendant, réutilisable depuis plusieurs documents HTML :

```
<link rel="stylesheet" href="style.css" type="text/css">
```



- Feuille de style locale au document HTML :

```
<style type="text/css">
p {
    color: red;
}
</style>
```

- Déclaration spécifique à un élément :

```
<p style="color:maroon;">
    Lorem ipsum...
</p>
```

N.B. les éléments `<link>` et `<style>` prennent place dans l'entête `<head>` du document HTML.
[Document HTML avec CSS]

Sélecteurs simples

Soit un document HTML :

```
<h1 id="special">Exemple</h1>
<p class="ok">texte <b>spécial</b></p>
<div>texte banal</div>
```

Auquel on applique une feuille de style avec les règles suivantes :

- sélecteur par type d'élément :

```
h1, div, p { color: red; }
```

- par valeur de l'attribut `class` :

```
.ok { color: green; }
```

- par valeur de l'identifiant `id` :

```
#special { color: blue; }
```

Exemple
texte spécial
texte banal

[demo sélecteurs simples]

☞ cf. démo, on constate que « certaines règles semblent "prioritaires" par rapport à d'autres »...

Sélecteurs contextuels

Soit maintenant le document :

```
<h1>Exemple</h1>
<p>texte <b>gras</b>, <i>très <b>gras</b></i></p>
<p>texte banal</p>
```

Auquel on applique une feuille de style avec les règles suivantes :

- descendants d'un parent (*éventuellement lointain*) :

```
p b { color: red; }
```



- enfants directs d'un parent :

```
p < b { color: green; }
```

- suivant dans l'ordre du document :

```
h1 ~ p { color: lightslategrey; }
```

- suivant immédiat dans l'ordre du document :

```
h1 + p { color: blue; }
```

Exemple

texte **gras**, très **gras**
texte banal

[demo sélecteurs contextuels]

☞ En intervertissant certaines règles on obtient un résultat différent. On constate que « *dans certains cas l'ordre des règles est signifiant* »...

Sélecteurs par attribut

D'autres sélecteurs permettent de désigner des éléments en fonction de la présence ou de la valeur d'un attribut :

- existence d'un attribut

```
<span title="popup">info-bulle</span> lors du survol d'un
<span title="Objet HTML">élément</span>

span[title] {
    box-shadow: 0 0 7px green;
    background-color: white;
    border-radius: 3px;
    padding: 0 0.1em;
    margin: 0 0.5em;
}
```

L'attribut "title" permet faire apparaître une **info-bulle** lors du survol d'un **élément**.

- valeur d'un attribut

```
<span title="sujet">Raymond</span> <span title="verbe">aime</span>
<span title="article">le</span> <span title="complément">chocolat</span>

span[title] { border-bottom: 2px solid grey; }
span[title=sujet] { border-bottom: 2px solid green; }
span[title=verbe] { border-bottom: 2px solid red; }
span[title=complément] { border-bottom: 2px solid blue; }
```

Raymond aime le chocolat.



Il est également possible de capturer des éléments en fonction de la valeur **partielle** d'un attribut :

- item dans une liste de valeurs séparées par des espaces

```
<span class="special police">Special Police</span>

span[class~=special] { color: navy; }
.police { font-family: Haettenschweiler; letter-spacing: 0.2em; }
```

Special Police

☞ L'attribut `class` accepte plusieurs valeurs séparées par des espaces. Dans ce cas, l'opérateur `~` donne le même résultat que le point `.` mais fonctionne aussi avec d'autres attributs que `class`...

- présence d'une sous-chaîne

```
L'<a href="http://www.ec-lyon.fr">Ecole Centrale de Lyon</a> est membre de
l'<a href="http://www.universite-lyon.fr">Université de Lyon</a>.
```

```
a[href*=ec-lyon] { color: black; font-weight: bold; }
```

L'Ecole Centrale de Lyon est membre de l'Université de Lyon.

- terme en début de chaîne

```
Voir l' <a href="https://fr.wikipedia.org/wiki/Https">article</a> de
<a href="http://fr.wikipedia.org/">Wikipédia</a> sur https.
```

```
a[href^=https] { color: green; }
```

Voir l' article de Wikipédia sur https.

☞ Utilisé comme ici avec l'attribut `href`, cet opérateur permet d'instrumenter facilement une page pour faire clairement apparaître à l'utilisateur les liens sécurisés ou les liens sortant d'un site.

- terme en fin de chaîne

```
Allez voir <a href="http://www.ec-lyon.fr/index.html">le site</a>
de l'<a href="http://www.ec-lyon.fr/">Ecole Centrale de Lyon</a>.
```

```
a[href$=".html"] { font-weight: bold; color: darkviolet; text-decoration: none; }
```

Allez voir le site de l'Ecole Centrale de Lyon.

☞ Permet par exemple de styler les liens en fonction du type de la ressource, ou les images en fonction de leur type (`gif`, `png` ou `jpeg`).

N.B. Ces sélecteurs peuvent être combinés comme dans :

```
a[href^=https] [href$=".html"] { font-weight: bold; color: green; }
```

Pseudo-classes

Les pseudo-classes permettent de désigner des éléments à partir d'informations qui ne concernent pas leur place dans l'arborescence, ou qui ne pourraient pas être simplement exprimées avec d'autres sélecteurs.

Pseudo-classes dynamiques

Certaines pseudo-classes sont dynamiques. Un élément acquiert ou perd une pseudo-classe dynamique en fonction des interactions avec l'utilisateur.



Pseudo-classe	Élément(s) concerné(s)
:link	liens hypertexte non encore visités
:visited	liens hypertexte déjà visités
:hover	élément survolé par le curseur
:active	élément en cours de clic (<i>entre le moment où le bouton de souris est enfoncé et celui où il est relâché</i>)
:focus	élément qui reçoit les entrées clavier (<i>champ de formulaire, bouton, hyperlien...</i>)

Pseudo-classes dynamiques

Pseudo-classes pour éléments interactifs

Les pseudo-classes suivantes s'appliquent aux éléments interactifs de l'interface :

Pseudo-classe	Élément(s) concerné(s)
:enabled	éléments d'interaction non désactivés
:disabled	éléments d'interaction désactivés
:checked	boutons radio ou cases à cocher qui sont cochés

Pseudo-classes pour éléments interactifs

Pseudo-classe de négation

La pseudo-classe de négation prend un sélecteur en paramètre pour inverser son sens. Il est interdit d'imbriquer deux négations :

```
button:not(:disabled) /* tous les buttons sauf ceux qui sont désactivés */
p:not(.highlighted)   /* tous les p sauf ceux de la classe highlighted */
*:not(#main)          /* tous les éléments sauf celui avec l'id main */
```

Pseudo-classes structurelles

Ces pseudo-classes sont attribuées en fonction de la position des éléments concernés dans l'arborescence, mais aucun sélecteur existant ne pourrait correctement les désigner. [Spécification des pseudo-classes]

Pseudo-classe	Élément(s) concerné(s)
:root	élément racine de l'arborescence, c'est-à-dire <code>html</code> dans notre cas.
:nth-child()	désigne des éléments en fonction de leur position au sein de leur fratrie. <code>nth-child(even)</code> et <code>nth-child(odd)</code> sont des cas possibles. D'autres paramètres plus complexes peuvent être employés.
:nth-last-child()	désigne des éléments en fonction de leur position au sein de leur fratrie en partant de la fin.
:nth-of-type()	désigne des éléments en fonction de leur position parmi la liste des éléments de même type au sein du document.
:nth-last-of-type()	désigne des éléments en fonction de leur position parmi la liste des éléments de même type au sein du document, en partant de la fin.
:first-child	synonyme de <code>nth-child(1)</code> .
:last-child	synonyme de <code>nth-last-child(1)</code> .
:first-of-type	synonyme de <code>nth-of-type(1)</code> .
:last-of-type	synonyme de <code>nth-last-of-type(1)</code> .
:only-child	synonyme de <code>:first-child:last-child</code> .
:only-of-type	synonyme de <code>:first-of-type:last-of-type</code> .
:empty	élément sans contenu (<i>ni texte, ni sous-éléments</i>).

Pseudo-classes structurelles



Pseudo-éléments

Les pseudo-éléments sont des éléments abstraits qui n'apparaissent pas tels quels dans le document. Ils permettent d'accéder à la première ligne ou au premier caractère d'un bloc pour leur appliquer des règles particulières. Ils permettent également de générer du contenu depuis la feuille de style.

La syntaxe des pseudo-éléments ressemble à celle des pseudo-classes, à cela près que leur nom est précédé d'un double caractère « *double-point* » comme dans `::first-line`.

- ☞ En CSS niveau 1 et 2 les pseudo-éléments étaient précédés d'un seul caractère « `:` » ce qui explique que cette syntaxe soit toujours reconnue par les navigateurs. Il est toutefois recommandé d'utiliser la nouvelle syntaxe CSS3.

`::first-line`

Désigne la première ligne d'un paragraphe, ou plus généralement d'un bloc ou d'une cellule de table.

```
<pre>
#fichier example.py
for n in range(5):
    print('hello')
</pre>

pre::first-line {
    background-color: yellow;
}
```

Résultat :

```
#fichier example.py
for n in range(5):
    print('hello')
```

- ☞ `p::first-line` se comporte comme un élément *inline* avec certaines restrictions sur les propriétés CSS qu'il est possible de modifier. Seules sont modifiables les propriétés de présentation du texte (*couleurs, police, taille et espacement des caractères...*).

`::first-letter`

Désigne la première lettre d'un paragraphe, ou plus généralement d'un bloc ou d'une cellule de table.

```
<p>Lorem ipsum dolor sit amet... </p>
<p>Cras dui elit, venenatis at est ut ... </p>

p::first-letter {
    font-size: 233%;
    float: left;
    padding-right: 0.1em;
    margin-top: 0.22em;
    color: navy;
    text-shadow: 1px 1px 3px navy;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed viverra velit sit amet dignissim feugiat. Nullam condimentum enim eu elementum cursus. Etiam rutrum tellus ac eros placerat, aliquam dignissim leo hendrerit.

Cras dui elit, venenatis at est ut, fringilla lobortis nisl. Integer ullamcorper pellentesque risus eget facilisis. Interdum et malesuada fames ac ante ipsum primis in faucibus. In hac habitasse platea dictumst.



::before

Permet d'insérer du contenu **avant** les éléments concernés par le sélecteur.

```
<label for="Nom"></label><input id="Nom" name="Nom">
<label for="Prénom"></label><input id="Prénom" name="Prénom">
<label for="Email"></label><input id="Email" name="Email">

label[for]::before {
  content: attr(for) " : ";
  padding-right: 0.5em;
  color: navy;
}
```

Nom : Prénom : Email :

⚠️ Attention ! ::before s'applique uniquement aux éléments non vides, ce qui exclut entre autres <input>,
, ... (*liste non exhaustive*).

Ceci s'explique par le fait que le contenu généré est inséré avant le contenu réel de l'élément, ce qui est impossible dans le cas des éléments vides.

⚠️ On peut noter que le contenu inséré peut être stylé comme n'importe quel élément.

N.B. La propriété `content` permet d'insérer du texte, la valeur d'attributs, des images, ou la valeur de compteurs CSS. Pour plus d'information voir les spécifications de CSS 2.1 sur le contenu généré à la section concernant [\[la propriété content\]](#).

::after

Utilisé pour insérer du contenu **après** les éléments concernés par le sélecteur.

L'exemple suivant permet d'annoter les liens sortants avec une étoile rouge :

```
L'Ecole Centrale de Lyon' est une Grande Ecole.

a[href^=http]::after {
  content: "*";
  position: relative;
  font-size: 80%;
  top: -0.5em;
  color: red;
}
```

[L'Ecole Centrale de Lyon*](#) est une Grande Ecole.

⚠️ Attention ! Comme ::before, et pour les mêmes raisons, ::after s'applique uniquement aux éléments non vides.

N.B. Dans ce document ::before et ::after sont utilisés conjointement pour ajouter automatiquement des crochets autour de certains hyperliens :

```
a.small-ref::before { content: "["; }
a.small-ref::after { content: "]"; }
```

4.3.2 Spécificité des sélecteurs

Rappel : les premiers exemples ont parfois montré de la part de CSS un comportement qui interpelle :

- certaines règles semblent "prioritaires" par rapport à d'autres,
- dans certains cas l'ordre des règles est signifiant...



Voyons comment déterminer de manière univoque quelle sera la valeur de chacune des propriétés d'un élément qui voit potentiellement de nombreuses règles qui s'appliquent à lui.

L'un des mécanismes les plus importants, trop souvent méconnu, consiste à trier les règles CSS en fonction de la spécificité du sélecteur et à retenir celles ayant le sélecteur le plus spécifique. (... certaines règles semblent prioritaires)

☞ Exemple de sélecteurs par ordre de spécificité décroissante : sélection par id, par classe, par type d'élément... (cf. infra pour plus de détails)

Lorsqu'il reste en lice plusieurs règles avec des sélecteurs de spécificité équivalente, et en dernier recours, est retenue la dernière règle dans l'ordre d'apparition dans la feuille de style. (... parfois, l'ordre est signifiant)

Calcul de la spécificité des sélecteurs

La spécificité est uniquement basée sur la forme du sélecteur. Elle est calculée de la façon suivante. Soient :

- a=1 si la règle provient d'un attribut html style, sinon a=0,
- b le nombre de sélecteurs id (introduits par le caractère « # »),
- c le nombre de sélecteurs par classe, par attribut et pseudo-classes,
- d le nombre d'éléments et pseudo-éléments.

La spécificité est obtenue en concaténant les quatre nombres a.b.c.d dans une base appropriée :

```
*          /* spécificité = 0.0.0.0 */
h1         /* spécificité = 0.0.0.1 */
ul li      /* spécificité = 0.0.0.2 */
p::first-line /* spécificité = 0.0.0.2 (pseudo-élément) */
.quote     /* spécificité = 0.0.1.0 */
em.quote   /* spécificité = 0.0.1.1 */
a::visited /* spécificité = 0.0.1.1 (pseudo-classe) */
#signature /* spécificité = 0.1.0.0 */
.spe #ex1  /* spécificité = 0.1.1.0 */
code em.red/* spécificité = 0.0.1.2 */
.table th  /* spécificité = 0.0.1.1 */
style=""   /* spécificité = 1.0.0.0 */
```

Savoir former les sélecteurs appropriés pour désigner la cible des règles CSS de mise en forme est une compétence incontournable pour qui veut concevoir des interfaces web, malheureusement trop souvent mal maîtrisée (*en particulier en ce qui concerne la spécificité*).

Voici quelques références pour approfondir ou voir le contenu du cours sous un angle différent :

- [30 sélecteurs CSS que vous devez connaître](#)
- [Les sélecteurs CSS2.1](#)
- [Les sélecteurs CSS3](#)
- [Les sélecteurs CSS4](#)

Une fois ces concepts maîtrisés, le reste n'est plus qu'une question de vocabulaire (*ou presque*) dans la mesure où il faut connaître à la fois le nom des propriétés et les valeurs que l'on peut leur assigner.

Quelques-unes de ces propriétés seront présentées dans la suite, mais sans être aucunement exhaustif : le web est plein d'excellentes ressources à ce sujet.

(Pour avoir des informations fiables, et surtout à jour, essayez tout de même de privilégier les organismes de référence comme le W3C ou les éditeurs de navigateurs comme Mozilla).



4.3.3 Unités CSS et propriétés de mise en page

Un certain nombre de propriétés CSS admettent pour valeur une **longueur**. (*cf. hauteur ou largeur d'une marge, d'un bloc, d'une image, d'une cellule de tableau...*).

Une longueur est toujours représentée avec une valeur **munie d'une unité**. En fonction de la propriété considérée, la valeur attendue peut être un nombre entier ou réel, positif ou négatif. La seule valeur autorisée à ne pas porter d'unité est la valeur 0 (zéro).

Longueurs relatives

Les unités de longueur **relatives** expriment une longueur par rapport à une autre. Les documents qui utilisent ce type d'unités s'adaptent en général plus facilement à divers dispositifs d'affichage.

symbole	unité relative à
em	la taille de la police de caractères de l'élément
ex	hauteur du caractère 'x' de la police de l'élément
ch	largeur du caractère '0' de la police de l'élément
rem	taille de la police de l'élément racine
vw	1% de la largeur de la fenêtre
vh	1% de la hauteur de la fenêtre
vmin	1% de la plus petite des dimensions de la fenêtre
vmax	1% de la plus grande des dimensions de la fenêtre

unités de longueur relatives

Longueurs absolues

Les unités de longueur **absolues** peuvent être converties entre elles. Sauf exception elles sont à réserver aux cas où l'on maîtrise parfaitement l'environnement de rendu.

A noter : en CSS3 la taille d'un pixel a été arbitrairement définie comme correspondant à une résolution de 96dpi (*dots per inch*). Par conséquent, sur un dispositif dont la résolution est différente, soit 1 pixel CSS ne correspond pas à un pixel physique, soit les unités de longueur ne correspondent pas à leur équivalent physique...

symbole	nom	valeur
cm	centimètre	$1\text{cm} = 96\text{px} / 2.54$
mm	millimètre	$1\text{mm} = 0.1\text{cm}$
q	$\frac{1}{4}$ millimètre	$1\text{q} = 0.25\text{mm} = 0.025\text{cm}$
in	inch	$1\text{in} = 2.54\text{cm} = 96\text{px}$
pc	pica	$1\text{pc} = 1\text{in} / 6$
pt	point	$1\text{pt} = 1\text{in} / 72$
px	pixel	$1\text{px} = 1\text{in} / 96$

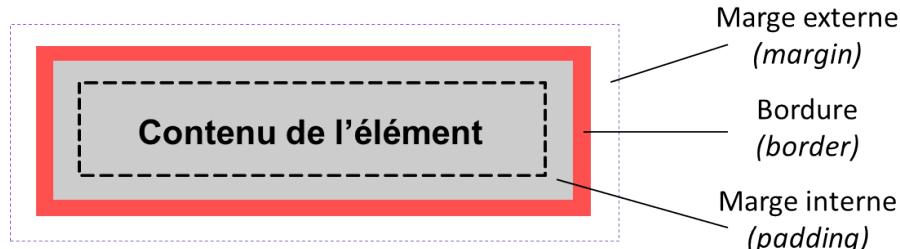
unités de longueur absolues

[Unités de longueur CSS3]



Modèle de mise en page

Le modèle de mise en page CSS voit chaque élément comme une boîte rectangulaire, avec une marge externe, une bordure, et une marge interne.



La marge externe est transparente et laisse entrevoir la couleur ou l'image de fond de l'élément parent.

La bordure possède ses propres caractéristiques (*couleur, largeur, et style du trait*).

La marge interne est également transparente, et laisse voir la couleur ou l'image de fond de l'élément lui-même.

Valeurs acceptées par les propriétés des boîtes

propriété	par défaut	valeurs possibles
margin	0	%, longueur éventuellement < 0 ou "auto"
padding	0	%, dimension >= 0
border-width	medium	%, dimension >= 0, "thin", "medium" ou "thick"
border-style	none	"none", "dotted", "dashed", "solid", "double", "groove", "ridge", "inset", "outset"
border-color		couleur ou "transparent"

valeurs des propriétés des boîtes

☞ De nombreuses propriétés CSS admettent comme ici des valeurs exprimées en %. La référence dépend du contexte. Il s'agit ici de la largeur de l'élément parent.

La dimension des marges externe et interne, la largeur, la couleur et le style de la bordure peuvent se spécifier indépendamment pour chacun des côtés.

```
span.demo {  
    border-top-style: solid;    border-top-color: grey;    padding-top: 5px;  
    border-bottom-style: solid; border-bottom-color: grey; padding-bottom: 7px;  
}
```

span.demo

☞ La déduction du nom de l'ensemble des propriétés pour chacun des côtés à partir de cet exemple est laissée à la sagacité du lecteur...

Il existe une syntaxe raccourcie pour chacune de ces propriétés, qui permet de modifier simultanément les 4 côtés avec une seule déclaration. Pour cela, ces raccourcis admettent de 1 à 4 valeurs dans l'ordre : *haut, droite, bas, gauche*.



S'il manque 1 donnée, la gauche prendra la même valeur que la droite. S'il en manque deux, la valeur pour le bas sera identique à celle du haut, s'il en manque 3 les 4 côtés prendront la valeur spécifiée.

```
span.demo {  
    border-style: solid;  
    padding: 5px 0.5em 7px;  
    border-color: grey transparent;  
    border-width: medium;  
}
```

span.demo

Remarque importante : en l'absence de bordure ou de marge interne entr'elles, les marges haute et basse de deux blocs adjacents ou imbriqués, ou les marges haute et basse d'un élément vide s'interpénètrent de manière à ce que la marge résultante soit égale à la plus grande des marges spécifiées (*collapsing margins*). [Interpénétration des marges]

Modification du type d'affichage des éléments

Nous avons vu qu'il existe essentiellement deux types d'éléments en HTML : les **éléments bloc** et les **éléments inline** qui ne sont pas mis en page de la même manière.

CSS permet de modifier le type d'affichage par défaut d'un élément à l'aide de la propriété `display`:

valeur	interprétation
none	L'élément n'est pas mis en page. Il est absent de l'affichage.
block	L'élément est mis en page comme un élément bloc.
inline	L'élément est mis en page comme un élément inline.
inline-block	Le contenu de l'élément est mis en page à l'intérieur d'un bloc, dont la largeur s'adapte à son contenu. Le bloc est ensuite mis en page comme un élément inline au sein de son parent.

valeurs possibles pour la propriété display

N.B. La propriété `display` accepte de nombreuses autres valeurs qui ne seront pas détaillées ici.
[la propriété `display`]

4.3.4 Présentation des boîtes

CSS permet de modifier la couleur du fond, du texte et de la bordure des éléments :

```
span.demo { padding: 3px 0.5em; border-width: 3px 0; border-style: solid none;  
background-color: rgba(0, 255, 0, 0.3);  
border-color: lime;  
color: #000;  
}
```

Ce bloc contient un élément span.demo « caché » dans le texte...



Différentes syntaxes sont possibles pour exprimer une couleur : [couleurs CSS3]

expression	exemples
un nom (X11, SVG) [démonstration couleurs X11]	red, pink, orange, cornflowerblue
le modèle #RRGGBB à trois composantes hexadécimales	#008000
une syntaxe #RGB raccourcie	#08F → #0088FF
la notation rgb à composantes décimales ou %	rgb(0, 127, 255)
la notation rgb avec opacité	rgba(0, 127, 255, 0.5)
la notation hsl [démonstration couleurs hsl]	hsl(45, 100%, 50%)

comment exprimer une couleur

Ombres

CSS3 a introduit la possibilité de spécifier des ombres pour les boîtes des éléments.

Pour une ombre extérieure, il faut indiquer dans l'ordre :

- le décalage horizontal et vertical de l'ombre par rapport à la boîte,
- le « *rayon de floutage* » (*optionnel - plus il est grand, plus c'est flou*),
- la différence de taille entre la zone floutée et la boîte (*optionnel*),
- la couleur de l'ombre (*optionnelle mais la valeur par défaut dépend du navigateur*).



☞ Il est intéressant de noter (cf. dernier exemple) qu'une boîte peut posséder simultanément plusieurs ombres différentes...

Ombre intérieure

Il est également possible de spécifier une ombre **intérieure** à la boîte, en préfixant la valeur associée à la propriété `box-shadow` par le mot-clé `inset` :



et de combiner des ombres externe et interne pour une même boîte :

`box-shadow: inset 3px 3px 0 3px orange, 5px 5px 7px teal`

Tout savoir sur [l'ombre des boîtes](#).
Apprendre à créer divers [effets d'ombrage](#).



Angles arrondis

border-radius: 10px; 1.33em;

border-radius: 20px / 10px;

border-radius: 50% / 50%;

border-radius: 2em 0 / 100% 0;

Cette propriété peut bien sûr être combinée avec les autres propriétés de présentation des boîtes :

border-radius: 0 30px 30px 0;
border-left: 5px solid maroon;
box-shadow: 5px 5px 7px darkcyan;

☞ La propriété `border-radius` est un raccourci. Les propriétés individuelles sont :

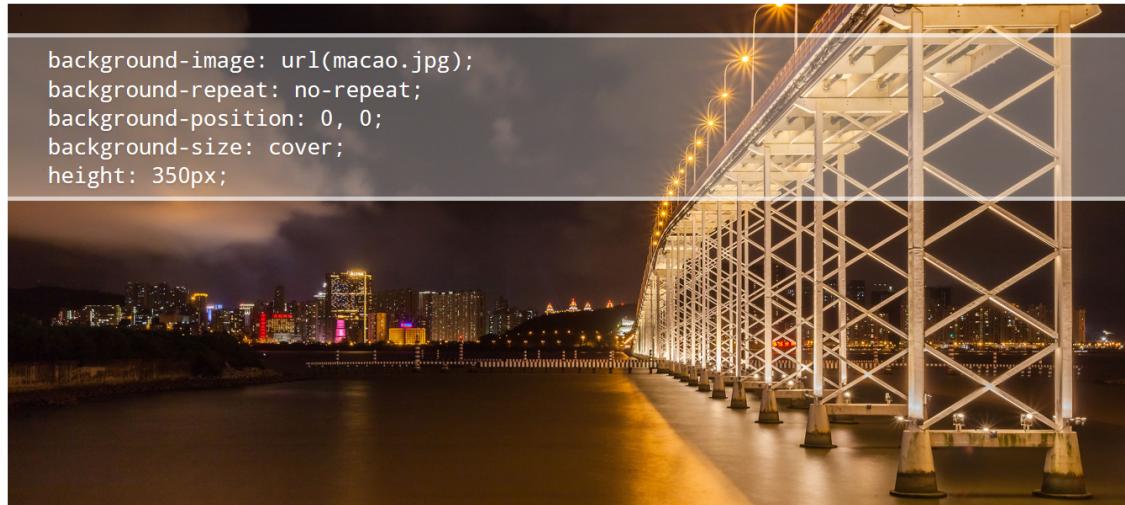
- `border-top-left-radius`,
- `border-top-right-radius`,
- `border-bottom-right-radius`,
- `border-bottom-left-radius`.

Elles admettent chacune une ou deux valeurs (*dimension ou %*).

Image de fond

Une boîte peut non seulement comporter un fond uniformément coloré, mais aussi une image de fond qu'il est possible de redimensionner, de repositionner, et de répéter ou non.

background-image: url(macao.jpg);
background-repeat: no-repeat;
background-position: 0, 0;
background-size: cover;
height: 350px;



Aller au fond des choses avec les propriétés du fond.

Photo Diego Delso [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>)], via Wikimedia Commons

N.B. Dans certains cas, et notamment comme ici lorsqu'on veut qu'une boîte soit plus grande que son contenu, il peut être utile d'imposer ses dimensions avec les propriétés `width` et/ou `height` (*cf. infra*).



Gradients

CSS3 possède également des propriétés pour remplir le fond d'un élément avec une couleur qui varie linéairement entre plusieurs valeurs d'un endroit à l'autre de l'élément.

```
background: linear-gradient(0deg, silver, black);
```

Le premier paramètre correspond à l'angle (*noter l'unité*), et la séquence peut comporter plusieurs couleurs :

```
background: linear-gradient(90deg,
    rgba(255, 255, 255, 0),
    rgba(255, 255, 255, 0.6),
    rgba(255, 255, 255, 0)
);
```

Le premier paramètre possède une variante :

```
background: linear-gradient(to bottom, silver, white, silver);
```

Vous trouverez facilement sur le web de très nombreuses ressources à ce sujet (à *peine effleuré ici* - cf. *répétition du gradient, gradient radial*).

[\[W3C gradients\]](#) [\[Moz linear-gradient\]](#) [\[CSS3 patterns Gallery\]](#) [\[CSS3 gradients\]](#) [\[3D Gradient Box\]](#)

4.3.5 Présentation du texte

Police de caractères

Depuis toujours, CSS permet d'indiquer la police de caractères désirée pour le rendu d'un élément. Faut-il encore que cette police soit effectivement disponible sur le poste client. C'est pourquoi la propriété `font-family` permet de spécifier une liste de polices par ordre de priorité décroissante :

```
font-family: Calibri, Arial, Helvetica, sans-serif;
```

En tant que créations de l'esprit, les polices de caractère sont en général protégées par des droits de recopie, ce qui a longtemps freiné la possibilité de télécharger automatiquement la police désirée sur le poste client.

Depuis peu, cette difficulté a été levée par l'apparition de polices open source.
[\[Open Font Library\]](#) [\[Google Fonts\]](#) [\[Polices libres sur Github\]](#)

Voici comment déclarer une police téléchargeable :

```
@font-face {
    font-family: 'Droid Sans Mono';
    font-style: normal;
    font-weight: 400;
    src: url(fonts/DroidSansMono.woff) format('woff');
}
```

[\[CSS3 Font Module\]](#)



Caractéristiques de la police

La taille, la graisse et le style de la police désirée pour le texte d'un élément sont indiqués de la manière suivante :

```
font-size: 120%;  
font-weight: bold;  
font-style: italic;
```

La taille de la police pourra être fixée à l'aide d'une unité absolue (*cm, mm, in, pt, pc*) ou relative (*px, %, em ou ex*). Par défaut la graisse et le style valent `normal`.

Caractéristiques du texte

[CSS 2.1 Text]

text-indent – Indentation de première ligne, valeur par défaut 0

```
text-indent: 3em
```

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus quam velit, egestas a tincidunt ut, accumsan ut sem. Morbi eu porttitor diam. Cras eu aliquet orci, eu sodales nunc. Donec at eros odio. Maecenas vestibulum vestibulum posuere.

```
text-indent: -2em
```

 Ut nec lectus mauris. Aliquam erat volutpat. Mauris ornare tristique hendrerit. Proin sed turpis ac neque dictum elementum. Etiam lectus quam, maximus et blandit ac, venenatis eu orci. Nulla facilisis mattis mattis.

text-align – *left, right, center, justify*.

```
text-align: justify
```

 Nullam consequat quam at justo pellentesque, vel pretium lacus pharetra. Vivamus laoreet elit lectus, eu varius sem blandit at. Morbi facilisis, orci quis venenatis elementum, justo mi tristique massa, sed tempus massa dolor pretium leo.

```
text-align: center
```

 Nunc rhoncus tincidunt viverra. Donec tristique vulputate quam sit amet vestibulum. Quisque sapien est, euismod ac gravida a, tincidunt vitae ante. Fusce nisl ipsum, sollicitudin sed lectus quis, sollicitudin sollicitudin sem.

text-decoration – *none, underline, overline, line-through*.

```
text-decoration: none;
```

 Lorem ipsum dolor sit amet

```
text-decoration: underline;
```

 Lorem ipsum dolor sit amet

```
text-decoration: overline;
```

 Lorem ipsum dolor sit amet

```
text-decoration: line-through;
```

 Lorem ipsum dolor sit amet

```
text-decoration: overline underline;
```

 Lorem ipsum dolor sit amet



letter-spacing – Espace entre les lettres, valeur par défaut `normal`.

```
letter-spacing: 0.1em;
```

Aliquam erat volutpat. Pellentesque eu libero tincidunt, lobortis felis vel, pretium leo. Integer convallis venenatis metus, a molestie risus convallis a.

word-spacing – Espace entre les mots, valeur par défaut `normal`.

```
letter-spacing: 0.2em; word-spacing: 0.5em;
```

Aliquam erat volutpat. Pellentesque eu libero tincidunt, lobortis felis vel, pretium leo. Integer convallis venenatis metus, a molestie risus convallis a.

text-transform – `capitalize`, `uppercase`, `lowercase`, `none`.

```
text-transform: none;
```

lorem ipsum dolor sit amet

```
text-transform: capitalize;
```

LOREM IPSUM DOLOR SIT AMET

```
text-transform: uppercase;
```

LOREM IPSUM DOLOR SIT AMET

```
text-transform: lowercase;
```

lorem ipsum dolor sit amet

white-space – Gère le traitement des espaces.

valeur	traitement du texte par le client
<code>none</code>	Valeur par défaut. Normalise et compacte le texte. Passe à la ligne où c'est nécessaire.
<code>pre</code>	Respecte tous les espaces présents dans le code source y compris les sauts de ligne. Ne passe pas automatiquement à la ligne.
<code>nowrap</code>	Normalise et compacte le texte. Ne passe pas à la ligne.
<code>pre-wrap</code>	Respecte tous les espaces présents dans le code source y compris les sauts de ligne. Passe automatiquement à la ligne si nécessaire.
<code>pre-line</code>	Normalise et compacte les espaces, sauf les sauts de ligne qui sont respectés. Insère des passages à la ligne si nécessaire.

text-shadow – Ombre du texte.

La syntaxe est similaire à celle de `box-shadow` à ceci près que la notion d'ombre interne n'existe pas pour le texte.

```
text-shadow: 1px 1px 0 white;
```

☞ Notamment utilisé pour obtenir des effets de relief sur des boutons (*texte gravé, cf. supra*).

[CSS3 Font Module]



4.3.6 Dimensions et position des boîtes

Les propriétés CSS suivantes permettent de redimensionner les boîtes, sauf celles des éléments texte *inline* : [Dimensionnement des boîtes]

propriété	valeurs possibles	fonctionnement
width	auto, dimension, %	fixe la largeur du contenu de la boîte (<i>à l'intérieur du padding</i>)
min-width	0, dimension, %	fixe la largeur minimale du contenu de la boîte
max-width	none, dimension, %	fixe la largeur maximale du contenu de la boîte
height	auto, dimension, %	fixe la hauteur du contenu de la boîte (<i>à l'intérieur du padding</i>)
min-height	0, dimension, %	fixe la hauteur minimale du contenu de la boîte
max-height	none, dimension, %	fixe la hauteur maximale du contenu de la boîte

propriétés de redimensionnement des boîtes

☞ Lorsque les dimensions sont exprimées en % elles se réfèrent à la dimension correspondante de l'élément parent.

```
width: 50%;  
margin: 1.33em auto;
```

☞ Lorsqu'un élément est moins large que la place disponible, la valeur auto pour les deux marges latérales a pour effet de centrer l'élément.

box-sizing

A priori, les propriétés de redimensionnement spécifient la taille de la **zone de contenu**, c'est à dire située à l'intérieur des marges internes (*padding*). La propriété `box-sizing` permet de modifier ce point : [specs box-sizing]

valeur	fonctionnement
content-box	Comportement par défaut
padding-box	Les dimensions préconisées par les propriétés de redimensionnement incluent la marge intérieure.
border-box	Les dimensions préconisées par les propriétés de redimensionnement incluent la marge intérieure et la bordure.

valeurs de la propriété box-sizing

overflow

Parfois le contenu d'une boîte peut déborder, notamment lorsqu'on interdit le passage automatique à la ligne avec la propriété `white-space`. La propriété `overflow` indique au client comment régir dans ce cas :

valeur	fonctionnement
visible	Le contenu est affiché même s'il dépasse de la boîte.
hidden	Le contenu qui dépasse est simplement rogné par le bord de la boîte.
scroll	Le contenu qui dépasse est rogné. Le client affiche des barres de défilement, qu'elles soient nécessaires ou non.
auto	Le contenu qui dépasse est rogné. Le client affiche des barres de défilement fonction du besoin.

valeurs de la propriété overflow

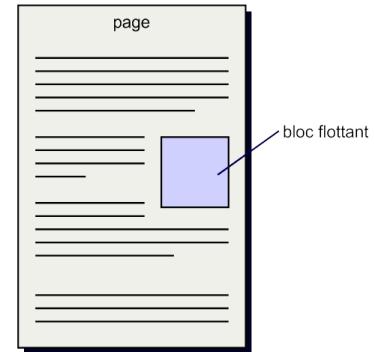


Éléments flottants

Il est possible de sortir un bloc du processus de mise en page standard, pour le rendre flottant.

Il est alors aligné sur l'un des bords de la page, et automatiquement contourné par le reste du texte.

```
<p id="illustration">paragraphe flottant</p>  
  
#illustration {  
    float: right;  
    margin-left: 1em;  
}
```



- ☞ La propriété `clear` permet d'éviter qu'un bloc ne se place à côté d'un élément flottant. Le cas échéant, il ira alors se positionner dessous :

```
p.special { clear: left; }
```

Les valeurs possibles sont : `left`, `right`, et `both`.

Positionnement relatif

CSS permet de déplacer une boîte par rapport à sa position normale dans le flot du texte.

```
position: relative;  
left: 40%; /* on pourrait aussi utiliser les */  
top: -3em; /* propriétés bottom et right... */
```

Le déplacement des éléments peut les entraîner à se chevaucher les uns les autres, ce qui peut être un effet recherché ou pas...

N.B. On aurait pu obtenir le même résultat avec une marge supérieure négative, ce qui aurait entraîné tous les blocs suivants vers le haut, ce qui n'est pas le cas ici car l'espace réservé pour l'élément avant son déplacement est conservé tel quel.

- ☞ Ce mécanisme est par exemple très utile pour mettre des éléments inline en indice ou en exposant, dans le cas où les éléments HTML `<sub>` et `<sup>` ne conviendraient pas.

Positionnement absolu

Il est également possible de complètement retirer un bloc du processus de mise en page pour le positionner où bon nous semble.

Cette possibilité a un coût : il de notre responsabilité de « faire de l'espace » à l'endroit où devra se trouver le bloc déplacé, pour éviter tout chevauchement malencontreux.

```
position: absolute;  
left: 50%;  
top: 200px;
```

- ☞ En cas de positionnement absolu, la référence des coordonnées correspond au **coin haut gauche du parent positionné le plus proche**.

Un élément est dit « *positionné* » lorsque sa propriété `position` a une valeur différente de `static` (*valeur par défaut*).



N.B. On utilise très souvent la position `relative` sans déplacer un élément, afin que celui-ci puisse servir de référence de coordonnées pour ses enfants positionnés.

Attention à ne pas tomber dans le travers de tout vouloir positionner manuellement, ce type de design est très difficile à rendre fluide...

4.4 Interactivité (*introduction à Javascript*)

JavaScript permet la programmation impérative, structurée avec une syntaxe héritée du langage C (*comme Java, Perl, PHP et d'autres...*).

La syntaxe des opérateurs (*arithmétiques, logiques, binaires, comparaison, incrément/décrément, affectation, ternaire ?:*) et des instructions (*if, else, for, while, switch, break, continue, ...*) est relativement classique, même si elle diffère par moments de celle de Python.

Differences notables :

- En C, et en Java la portée des variables est limitée par le bloc {} englobant, en JavaScript les variables locales sont limitées à la **fonction** au sein de laquelle elles ont été définies.
- L'opérateur « + » est également utilisé pour la concaténation de chaînes de caractères.
- L'orientation objet de Javascript est particulière (*héritage par prototypes, pas de classes*).
- Les fonctions sont des citoyens de première classe (*first class citizen*). Une fonction est un objet, qui peut être affecté à une variable, passé en paramètre à une fonction, ou renvoyé par une fonction.

[Wikipedia Javascript] [specs ECMAScript]

4.4.1 Manipulation du DOM (*Document Object Model*)

L'interfaçage avec le navigateur s'effectue essentiellement via le DOM (*Document Object Model*).

Le DOM est une API (*Application Programming interface*) ayant donné lieu à plusieurs recommandations du W3C (*World Wide Web Consortium*).

[DOM2 Core] [DOM2 HTML]

Il permet notamment :

- d'accéder aux éléments HTML contenus dans la page, en lecture et en écriture,
- d'accéder au style (CSS) des éléments contenus dans la page, en lecture et en écriture,
- de programmer des gestionnaires d'événement pour réagir aux interactions de l'utilisateur...

DOM - Récupération d'éléments de la page

`document.getElementById(id)`

Renvoie l'élément possédant l'attribut *id* dont la valeur est spécifiée.

```
<div id="toto">Je suis un paragraphe qui sera pourpre</div>

<script>
  var element_a_colorer = document.getElementById('toto');

  element_a_colorer.style.backgroundColor = 'purple';
  element_a_colorer.style.color = 'white';
</script>
```

Je suis un paragraphe qui sera pourpre



`document.getElementsByClassName(name)`

Renvoie les éléments possédant un attribut *class* avec la valeur spécifiée.

```
<div class="tata">Je suis un paragraphe qui sera bleu</div>
<div>Je suis un paragraphe avec <span class="tata">un mot</span> qui sera bleu</div>

<script>
    var elements_a_colorer = document.getElementsByClassName('tata');

    for ( var n = 0; n < elements_a_colorer.length; n++ ) {
        var element = elements_a_colorer[n];
        element.style.backgroundColor = '#77C';
        element.style.color = 'white';
    }
</script>
```

Je suis un paragraphe qui sera bleu

Je suis un paragraphe avec un mot qui sera bleu

`document.getElementsByTagName(name)`

Renvoie tous les éléments correspondant au nom de balise spécifié.

```
<div id="titi">
    Je suis un paragraphe avec des <span>mots</span>
    qui seront <span>verts</span>
</div>

<script>
    var paragraphe = document.getElementById('titi');

    var elements_a_colorer = paragraphe.getElementsByTagName('SPAN');
    for ( var n = 0; n < elements_a_colorer.length; n++ ) {
        var element = elements_a_colorer[n];
        element.style.backgroundColor = '#4C4';
        element.style.color = 'white';
    }
</script>
```

Je suis un paragraphe avec des mots qui seront verts

`document.querySelector(sel)`

Renvoie le premier élément répondant au sélecteur CSS spécifié.

`document.querySelectorAll(sel)`

Renvoie l'ensemble des éléments répondant au sélecteur CSS spécifié.

```
<div id="tutu">
    Je suis un paragraphe avec des <span>mots</span>
    qui seront <span>roses</span>
</div>

<script>
    var elements_a_colorer = document.querySelectorAll('#tutu span');
    for ( var n = 0; n < elements_a_colorer.length; n++ ) {
        var element = elements_a_colorer[n];
        element.style.backgroundColor = 'pink';
        element.style.color = 'white';
    }
</script>
```

Je suis un paragraphe avec des mots qui seront roses

Ces deux méthodes sont relativement récentes (02/2013). [Selectors API]



DOM - Création d'éléments dans la page

```
document.createElement(type)
```

Crée un objet Javascript correspondant à un élément du type spécifié (*nom de la balise*), et le renvoie.

```
node.appendChild(element)
```

Insère l'élément spécifié dans l'arbre des éléments du navigateur, après le dernier enfant du noeud *node*.

```
<div id="riri">Ce texte sera suivi par une ligne horizontale.</div>

<script>
    var paragraphe = document.getElementById('riri');

    var ligne = document.createElement('HR');
    paragraphe.appendChild(ligne);
</script>
```

Ce texte sera suivi par une ligne horizontale.

```
document.createTextNode(texte)
```

Crée un objet Javascript "noeud texte" et le renvoie.

```
<div id="fifi" style="border: 1px solid #CCC; padding: 5px 10px;">
    Texte initial du paragraphe.<br>
</div>

<script>
    var paragraphe = document.getElementById('fifi');

    var texte = document.createTextNode('Ce contenu a été ajouté par programme.');
    paragraphe.appendChild(texte);
</script>
```

Texte initial du paragraphe.
Ce contenu a été ajouté par programme.

```
node.insertBefore(nouveau,reference)
```

Insère un nouveau noeud dans l'arbre des éléments du navigateur. Ce noeud sera un enfant du noeud *node*, placé avant le noeud de référence spécifié.

```
<div id="loulou" style="border: 1px solid #CCC; padding: 5px 10px;">
    <span>Phrase 1.</span><span>Phrase 2.</span><span>Phrase 3.</span>
</div>

<script>
    var paragraphe = document.getElementById('loulou');
    var phrases = paragraphe.getElementsByTagName('SPAN');

    for ( var n = 1; n < phrases.length; n++ ) {
        var ligne = document.createElement('HR');
        paragraphe.insertBefore(ligne,phrases[n]);
    }
</script>
```

Phrase 1.
Phrase 2.
Phrase 3.



DOM - Autres propriétés et méthodes des noeuds

`node.parentNode`

Correspond au noeud parent du noeud `node`.

`node.childNodes`

Correspond à la liste des noeuds enfants du noeud `node`.

`node.firstChild`

Correspond au premier enfant du noeud `node`.

`node.lastChild`

Correspond au dernier enfant du noeud `node`.

`node.previousSibling`

Correspond au frère précédent du noeud `node`.

`node.nextSibling`

Correspond au frère suivant du noeud `node`.

`node.nodeValue`

Correspond à la valeur textuelle d'un noeud texte.

`node.getAttribute(attr)`

Renvoie la valeur de l'attribut spécifié du noeud `node`.

`node.setAttribute(attr,value)`

Modifie la valeur de l'attribut spécifié du noeud `node`.

Exemple de navigation au sein du DOM

Soient les cellules de la table suivante à colorer de manière appropriée :

```
<table id="hsl">
<tr><td> </td><td colspan=5>saturation</td></tr>
<tr><td>lum</td><td>100%</td><td>75%</td><td>50%</td><td>25%</td><td>0%</td></tr>
<tr><td>100%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>87.50%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>75%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>62.5%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>50%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>37.5%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>25%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>12.5%</td><td> </td><td> </td><td> </td><td> </td></tr>
<tr><td>0%</td><td> </td><td> </td><td> </td><td> </td></tr>
</table>
```

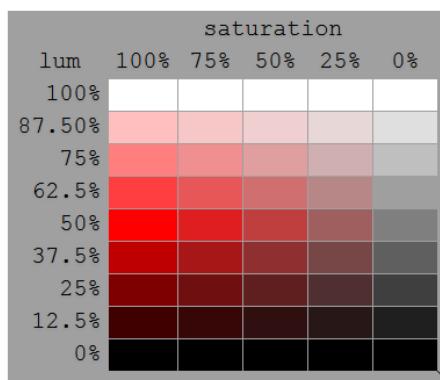
saturation					
lum	100%	75%	50%	25%	0%
100%					
87.50%					
75%					
62.5%					
50%					
37.5%					
25%					
12.5%					
0%					



Navigation au sein du DOM

Exemple de solution :

```
<script>                                // calcul couleur des cellules
    var table = document.getElementById('hsl');      // table concernée
    var rows = table.getElementsByTagName('TR');        // lignes de la table
    for ( var n = 2; n < rows.length; n++ ) {          // boucle sur les lignes
        tr = rows[n];
        var lum = tr.firstChild.firstChild.nodeValue; // lum = texte 1ère cellule
        var satNode = rows[1].firstChild.nextSibling; // sat = 2ème cellule 2ème ligne
        for ( var td = tr.childNodes[1]; td; td = td.nextSibling ) {
            var couleur = 'hsl(0,'+satNode.firstChild.nodeValue+', '+lum+')';
            td.style.backgroundColor = couleur;           // couleur de la cellule
            td.setAttribute('title',couleur);             // info-bulle visible au survol
            satNode = satNode.nextSibling;                // sat = cellule suivante ligne 2
        }
    }
</script>
```



Style d'un élément

Ainsi qu'aperçu précédemment, le style CSS d'un élément est modifiable via la propriété `style` qui représente la valeur de l'attribut éponyme de cet élément.

Les propriétés de l'objet `style` sont nommées d'après les propriétés CSS en remplaçant les éventuels tirets par une majuscule (*i.e. la propriété CSS `background-color` devient par exemple `style.backgroundColor`*).

```
<p id="dynstyle">Le style de ce paragraphe a été modifié par Javascript</p>

<script>
    var paragraphe = document.getElementById('dynstyle');
    var style = paragraphe.style;
    style.backgroundColor = '#cc8030';
    style.color = 'white';
    style.padding = '5px 10px';
    style.width = '680px';
    style.border = '3px outset #cc8030';
</script>
```

Le style de ce paragraphe a été modifié par Javascript



En lecture, l'objet `style` permet d'accéder à la valeur des propriétés CSS d'un élément ayant été modifiées par programme, ou via un attribut `style=`

```
<script>
  var style = document.getElementById('dynstyle').style;

  var str = 'background-color : '+style.backgroundColor+'\n'+
    'color : '+style.color+'\n'+
    'padding : '+style.padding+'\n'+
    'width : '+style.width+'\n'+
    'border : '+style.border+'\n';

  document.getElementById('showstyle').appendChild(document.createTextNode(str));
</script>
```

Voici, récupérées grâce au programme ci-dessus, les valeurs des propriétés modifiées du paragraphe page précédente :

```
background-color : rgb(204, 128, 48)
color : white
padding : 5px 10px
width : 680px
border : 3px outset rgb(204, 128, 48)
```

Autres propriétés des éléments HTML

Plus généralement, les objets Javascript correspondant à des éléments HTML possèdent chacun toute une série de propriétés, une pour chacun des attributs autorisés pour l'élément.

```
<p id="otherprops" class="smaller> Hello </p>

<script>
  var p = document.getElementById('otherprops');
  p.align = 'center';
  p.title = "Le contenu de ce paragraphe a été centré";

  var str = 'p.id = '+p.id+'\n'+p.class = '+p.className;

  document.getElementById('showprops').appendChild(document.createTextNode(str));
</script>
```

```
p.id = otherprops
p.class = smaller
```



4.4.2 Gestion des événements

Gestionnaires d'événements HTML4

En HTML 4.0 les gestionnaires d'événements sont spécifiés sous la forme d'attributs. De ce fait il ne peut y avoir qu'un seul gestionnaire par événement :

```
<button onclick="modifier_paragraphe()">Hello</button>

<script>
function modifier_paragraphe() {
    var p = document.getElementById('p1');
    p.style.backgroundColor = 'tomato';
    p.style.color = 'white';
    p.style.padding = '5px 10px';
    p.replaceChild(document.createTextNode('Paragraphe modifié !'), p.firstChild);
}
</script>

<p id="p1">Ce paragraphe sera modifié en actionnant le bouton.</p>
```

Ce paragraphe sera modifié en actionnant le bouton.

Paragraphe modifié !

Gestionnaires d'événements HTML4 via Javascript

Comme pour les autres attributs, les gestionnaires d'événements HTML 4.0 peuvent être accédés directement sous forme d'une propriété de l'élément concerné :

```
<button id="button2"> OK </button>

<script>
document.getElementById('button2').onclick = function() {
    var p = document.getElementById('p2');
    p.style.backgroundColor = 'darkslateblue';
    p.style.color = 'white';
    p.style.padding = '5px 10px';
    p.replaceChild(document.createTextNode('Paragraphe modifié !'), p.firstChild);
}
</script>

<p id="p2">Ce paragraphe sera modifié en actionnant le bouton.</p>
```

Ce paragraphe sera modifié en actionnant le bouton.

Paragraphe modifié !



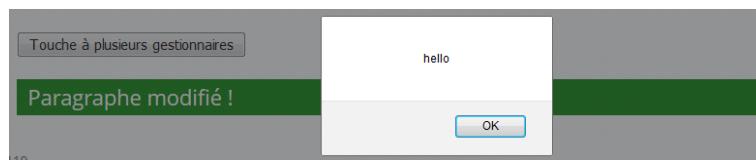
Gestionnaires d'événements : modèle du DOM

Le modèle de gestion des événements du DOM permet l'enregistrement de plusieurs gestionnaires pour le même événement d'un élément donné : [DOM2 Events]

```
<button id="multibutton"> Touche à plusieurs gestionnaires </button>

<script>
  function modifier_para() {
    var p = document.getElementById('p3');
    p.style.backgroundColor = 'green';
    p.style.color = 'white';
    p.style.padding = '5px 10px';
    p.replaceChild(document.createTextNode('Paragraphe modifié !'), p.firstChild);
  }
  var button = document.getElementById('multibutton');
  button.addEventListener('click', modifier_para );
  button.addEventListener('click', function(){ alert('hello'); } );
</script>

<p id="p3">Ce paragraphe sera modifié en actionnant le bouton.</p>
```



HTML Events

Les événements suivants sont regroupés sous l'appellation "HTML Events" :

événement	éléments source	déclenchement
load	window, frameset, object	au chargement de la page
unload	body, frameset	au passage à une autre page
select	input, textarea	sur sélection de texte
change	input, select, textarea	si contenu modifié
submit	form	lors de la soumission
reset	form	sur réinitialisation
focus	label,input, select, textarea, button	sur prise du focus
blur	label,input, select, textarea, button	sur perte du focus
resize	window	lors d'un modification de taille
scroll	window	sur déroulement de la fenêtre

HTML events

```
var sc = 0;
window.onscroll = function() {
  var pre = document.getElementById('scrolled');
  pre.replaceChild(document.createTextNode('scrolled '+ (sc++)), pre.firstChild);
}

scrolled 37
```



Mouse Events

Les événements de la catégorie Mouse Events sont :

événement	éléments source	déclenchement
click	tous éléments	lors d'un clic de souris (down-up-click)
mousedown	tous éléments	au début d'un clic de souris
mouseup	tous éléments	à la fin d'un clic de souris
mouseover	tous éléments	lors d'un survol du curseur
mousemove	tous éléments	lors d'un déplacement du curseur
mouseout	tous éléments	lorsque le curseur quitte l'élément

Mouse Events

```
var p = document.getElementById('mouse-style');
p.addEventListener('mouseover', function() {
    this.style.backgroundColor = '#cc8030';
    this.style.color = 'white';
    this.style.borderColor = '3px outset #cc8030';
});
```

Le style de ce paragraphe est modifié en fonction de la position du curseur

Le style de ce paragraphe est modifié en fonction de la position du curseur

Le style de ce paragraphe est modifié en fonction de la position du curseur

Question : quels sont les autres gestionnaires d'événement nécessaires pour obtenir un comportement correct lors d'un clic (bouton enfoncé) et que le bouton retrouve son aspect initial lorsque le curseur le quitte.

Détails de l'événement

À l'occurrence d'un événement, le gestionnaire est appelé avec un paramètre correspondant à un objet du type MouseEvent. Cet objet possède entre autres les propriétés suivantes :

```
long      detail;          // détail - cf. cas par cas
long      screenX;        // abscisse du clic (coords écran)
long      screenY;        // ordonnée du clic (coords écran)
long      clientX;        // abscisse du clic
long      clientY;        // ordonnée du clic
boolean   ctrlKey;        // touche 'ctrl' actionnée
boolean   shiftKey;       // touche 'shift' actionnée
boolean   altKey;         // touche 'alt' actionnée
boolean   metaKey;        // touche 'meta' actionnée
unsigned short button;    // bouton souris actionné (0,1,2)
EventTarget currentTarget; // élément porteur du gestionnaire
EventTarget relatedTarget; // élément - cf. cas pas cas
```

```
var mp = document.getElementById('multicolore');
mp.parentNode.addEventListener('mousemove', function(event) {
    var h = 180 + Math.atan2(event.clientY-450,event.clientX-550)*180/Math.PI;
    mp.style.backgroundColor = mp.style.borderColor = 'hsl('+h+',100%,50%)';
});
```

La couleur de ce paragraphe varie en fonction de la position du curseur...



4.4.3 AJAX

Ajax est un acronyme pour *Asynchronous JavaScript and XML*.

Avec Ajax, une application Web peut transmettre et récupérer de l'information vers et depuis le serveur, de manière asynchrone (*i.e. en tâche de fond*) sans interférer avec l'affichage et les interactions ayant lieu sur la page existante.

Cette opération s'effectue à l'aide d'un objet JavaScript nommé XMLHttpRequest.

☞ En dépit de son nom, l'usage d'`XML` n'est pas obligatoire (*on utilise souvent JSON, parfois HTML, ou même un autre format...*) et la requête n'est pas forcément asynchrone... [Utiliser XMLHttpRequest] [W3C XMLHttpRequest]

```
<div id="display_ajax"></div>

<script type="text/javascript" id="js_ajax">
    var dropzone = document.getElementById("display_ajax");
    var r = new XMLHttpRequest();
    r.onload = function() {
        dropzone.innerHTML = this.responseText;
    }
    r.open('GET', '/coucou/Raymond/Deubaze', true);
    r.send();
</script>
```

Ce programme s'utilise avec le serveur décrit lors du cours précédent.

Le contenu de la boîte ci-dessous provient d'un appel Ajax au serveur développé à l'occasion du cours n°2.

Bonjour Raymond Deubaze

Appel paramétré

Ce programme peut être facilement adapté pour effectuer un appel Ajax paramétré, qui prend ses informations dans des champs de formulaires :

```
var dropzone = document.getElementById("display_ajax")
, prenom = document.getElementById("prenom")
, nom = document.getElementById("nom")
;
document.getElementById('send').addEventListener('click', function() {
    var r = new XMLHttpRequest();
    r.onload = function() {
        dropzone.innerHTML = this.responseText;
    }
    r.open('GET', '/coucou/' + prenom.value + '/' + nom.value, true);
    r.send();
});
```

Prénom :

Nom :

Envoyer la requête

Bonjour Justin Poedau



Appel asynchrone

On peut également appeler le serveur lors de la frappe des nom et prénom :

```
var dropzone = document.getElementById("display_ajax")
, prenom = document.getElementById("prenom")
, nom = document.getElementById("nom")
;
prenom.addEventListener('keyup', send);
nom.addEventListener('keyup', send);

function send(event) {
    var r = new XMLHttpRequest();
    r.onload = function() {
        dropzone.innerHTML = this.responseText;
    }
    r.open('GET', '/coucou/' + prenom.value + '/' + nom.value, true);
    r.send();
}
```

☞ Noter `keyup` (*non vu précédemment*), l'événement utilisé pour réagir à toute frappe de touche dans l'un des champs de formulaire. [\[keyup event\]](#) [\[W3C UI Events\]](#)



Références et liens

[CSS Zengarden], p.5

D. SHEA. *CSS Zen Garden - The Beauty of CSS Design*. .
En ligne, disponible sur <http://www.csszengarden.com>
[consulté le 17 nov 2015]

[HTML2], p.5

T. BERNERS-LEE, D. CONNOLLY. *RFC 1866 - Hypertext Markup Language - 2.0*. IETF (Internet Engineering Task Force) Network Working Group, nov. 1995, 77 p.
En ligne, disponible sur <https://tools.ietf.org/html/rfc1866>
[consulté le 17 nov. 2015]

[HTML32], p.5

D. RAGGET. *HTML 3.2 Reference Specification*. W3C Recommendation, janv. 1997, ~45 p.
En ligne, disponible sur <http://www.w3.org/TR/REC-html32>
[consulté le 17 nov. 2015]

[HTML401], p.5

D. RAGGET, A. LE HORS, I. JACOBS (ÉD.). *HTML 4.01 Specification*. W3C Recommendation, déc. 1999, 389 p.
En ligne, disponible sur <http://www.w3.org/TR/html401/>
[consulté le 17 nov. 2015]

[XHTML], p.5

S. PEMBERTON ET AL.. *XHTML. 1.0 - A Reformulation of HTML 4 in XML 1.0*. W3C Recommendation, janv. 2000, rév. août 2002, 32 p.
En ligne, disponible sur <http://www.w3.org/TR/xhtml1/>
[consulté le 17 nov. 2015]

[HTML5], p.5

I. HICKSON, R. BERJON, ET AL.. *HTML5 - A vocabulary and associated APIs for HTML and XHTML*. W3C Recommendation, 28 oct. 2014, ~552 p.
En ligne, disponible sur <http://www.w3.org/TR/html5/>
[consulté le 17 nov. 2015]

[Histoire de HTML], p.5

I. HICKSON, R. BERJON, ET AL.. *HTML5 - A vocabulary and associated APIs for HTML and XHTML*. W3C Recommendation, 28 oct. 2014, § 1.4 History.
En ligne, disponible sur <http://www.w3.org/TR/html5/introduction.html#history-0>
[consulté le 22 nov. 2015]

[Can I Use], p.5

A. DEVERIA. *Can I use ... ?*. .
En ligne, disponible sur <http://caniuse.com>
[consulté le 17 nov. 2015]

[CSS1], p.5

H.W. LIE, B. BOS. *Cascading Style Sheets, level 1*. W3C Recommendation, 17 déc. 1996.
En ligne, disponible sur <http://www.w3.org/TR/CSS1/>
[consulté le 17 nov. 2015]

[CSS2], p.5

B. BOS, H.W. LIE, C. LILLEY, I. JACOBS. *Cascading Style Sheets, level 2 - CSS2 Specification*. W3C Recommendation, 12 mai 1999.
En ligne, disponible sur <http://www.w3.org/TR/1998/REC-CSS2-19980512/>
[consulté le 17 nov. 2015]



[CSS2.1], p.5

B. BOS, T. ÇELIK, I. HICKSON, H.W. LIE. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation, 7 juin 2011.
En ligne, disponible sur <http://www.w3.org/TR/CSS2/>
[consulté le 17 nov. 2015]

[Instantané CSS 2015], p.5

T. ATKINS JR., E.J. ETEMAD, F. RIVOAL. *CSS Snapshot 2015*. W3C Working Group Note, 13 oct. 2015.
En ligne, disponible sur <http://www.w3.org/Style/CSS/read.en.html>
[consulté le 18 nov. 2015]

[Comprendre les Specs CSS], p.5

E.J. ETEMAD. *Understanding the CSS Specifications*. W3C, 7 août 2015.
En ligne, disponible sur <http://www.w3.org/Style/CSS/read.en.html>
[consulté le 18 nov. 2015]

[Quirksmode], p.5

P.P. KOCH. *QuirksMode.org - CSS*. .
En ligne, disponible sur <http://www.quirksmode.org/css/index.html>
[consulté le 17 nov. 2015]

[exemple], p.6

D. MULLER. *Structure d'un document XHTML5*. 20 nov. 2015, 1 p.
En ligne, disponible sur <http://dmolinarius.github.io/demofiles/inf-tc3/cours3/xhtml-structure.xhtml>
[consulté le 20 nov. 2015]

[exemple], p.7

D. MULLER. *Structure d'un document HTML5*. 20 nov. 2015, 1 p.
En ligne, disponible sur <http://dmolinarius.github.io/demofiles/inf-tc3/cours3/html-structure.html>
[consulté le 20 nov. 2015]

[exemple], p.7

D. MULLER. *Éléments structuraux HTML4*. 20 nov. 2015, 1 p.
En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/html_bloc-1.html
[consulté le 20 nov. 2015]

[En savoir plus], p.8

C. EISENBRAN ET AL.. *Sections and Outlines of an HTML5 Document*. Mozilla Developer Network, 1 Oct. 2015.
En ligne, disponible sur https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Sections_and_Outlines_of_an_HTML5_document
[consulté le 21 nov. 2015]

[Spécification des éléments inline], p.9

I. HICKSON, R. BERJON, ET AL.. *HTML5 - A vocabulary and associated APIs for HTML and XHTML*. W3C Recommendation, 28 oct. 2014, § 4.5 Text-level semantics.
En ligne, disponible sur <http://www.w3.org/TR/html5/text-level-semantics.html>
[consulté le 22 nov. 2015]

[Styles de liste Mozilla], p.13

F. SCHOLZ ET AL.. *list-style-type*. Mozilla Developer Network, 13 Nov. 2015, § Extensions Mozilla.
En ligne, disponible sur https://developer.mozilla.org/fr/docs/Web/CSS/list-style-type#Extensions_Mozilla
[consulté le 23 nov. 2015]



[Listes imbriquées], p.13

D. MULLER. *Exemple de listes imbriquées*. mars 2003, 1 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/listes_imbriquees.html
[consulté le 23 nov. 2015]

[Exemples de tables en ligne], p.15

D. MULLER. *Exemples de tables*. déc. 2012, 4 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/html_table-1.html
[consulté le 23 nov. 2015]

[Comment structurer un formulaire HTML], p.17

S. BUCK ET AL.. *How to structure an HTML form*. Mozilla Developer Network, 29 Oct. 2015.

En ligne, disponible sur https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms/How_to_structre_an_HTML_form
[consulté le 23 nov. 2015]

[Document HTML avec CSS], p.23

D. MULLER. *CSS et HTML*. jan. 2014, 1 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/css_link.html
[consulté le 25 nov. 2015]

[demo sélecteurs simples], p.23

D. MULLER. *Sélecteurs simples*. jan. 2014, 1 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/css_selecteurs-simples.html
[consulté le 25 nov. 2015]

[demo sélecteurs contextuels], p.24

D. MULLER. *Sélecteurs contextuels*. jan. 2014, 1 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/css_selecteurs-contextuels.html
[consulté le 25 nov. 2015]

[Spécification des pseudo-classes], p.26

T. ÇELIK, E.J. ETEMAD, D. GLAZMAN, I. HICKSON. *Selectors Level 3*. W3C Recommendation, 29 sept. 2011, § 6.6 Pseudo-classes.

En ligne, disponible sur <http://www.w3.org/TR/css3-selectors/#pseudo-classes>
[consulté le 27 nov. 2015]

[la propriété content], p.28

B. BOS, T. ÇELIK, I. HICKSON, H.W. LIE. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation, 7 juin 2011, § 12 Generated content, automatic numbering, and lists.

En ligne, disponible sur <http://www.w3.org/TR/CSS2/generate.html#content>
[consulté le 27 nov. 2015]

[30 sélecteurs CSS que vous devez connaître], p.29

J. WAY. *The 30 CSS selectors you must memorize*. Envato Pty Ltd., 9 juin 2011.

En ligne, disponible sur <http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048>
[consulté le 28 nov. 2015]

[Les sélecteurs CSS2.1], p.29

B. BOS, T. ÇELIK, I. HICKSON, H.W. LIE. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation, 7 juin 2011, § 5 Selectors.

En ligne, disponible sur <http://www.w3.org/TR/CSS2/selector.html>
[consulté le 28 nov. 2015]



[Les sélecteurs CSS3], p.29

T. ÇELIK, E.J. ETEMAD, D. GLAZMAN, I. HICKSON. *Selectors Level 3*. W3C Recommendation, 29 sept. 2011.

En ligne, disponible sur <http://www.w3.org/TR/css3-selectors/>
[consulté le 28 nov. 2015]

[Les sélecteurs CSS4], p.29

E.J. ETEMAD, T. ATKINS JR. (ÉD). *Selectors Level 4 - Editor's Draft*. W3C, 13 sept. 2015.

En ligne, disponible sur <https://drafts.csswg.org/selectors-4/>
[consulté le 28 nov. 2015]

[Unités de longueur CSS3], p.30

T. ATKINS (ÉD). *CSS Values and Units Module Level 3*. W3C Candidate Recommendation, 11 juin 2015, § 5. Distance Units: the <length> type.

En ligne, disponible sur <http://www.w3.org/TR/css-values/#lengths>
[consulté le 28 nov. 2015]

[Interpénétration des marges], p.32

B. BOS, T. ÇELIK, I. HICKSON, H.W. LIE. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation, 7 juin 2011, § 8.3.1 Collapsing margins.

En ligne, disponible sur <http://www.w3.org/TR/CSS2/box.html#collapsing-margins>
[consulté le 29 nov. 2015]

[la propriété display], p.32

R. BOULANGER ET AL.. *display*. Mozilla Developer Network, 17 nov. 2015.

En ligne, disponible sur <https://developer.mozilla.org/fr/docs/Web/CSS/display>
[consulté le 29 nov. 2015]

[couleurs CSS3], p.33

T. ÇELIK, C. LILLEY, L.D. BARON (ÉD). *CSS Color Module Level 3*. W3C Recommendation, 7 juin 2011.

En ligne, disponible sur <http://www.w3.org/TR/css3-color/>
[consulté le 29 nov. 2015]

[démo couleurs X11], p.33

D. MULLER. *Couleurs X11*. jan. 2014, 1 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/css_color-X11.html
[consulté le 29 nov. 2015]

[démo couleurs hsl], p.33

D. MULLER. *Couleurs hsl*. jan. 2014, 1 p.

En ligne, disponible sur http://dmolinarius.github.io/demofiles/inf-tc3/cours3/css_color-hsl.html
[consulté le 29 nov. 2015]

[l'ombre des boîtes], p.33

B. BOS, E.J. ETEMAD, B. KEMPER (ÉD). *CSS Backgrounds and Borders Module Level 3*. W3C Candidate Recommendation, 9 sept. 2014, § 7.1. Drop Shadows: the 'box-shadow' property.

En ligne, disponible sur <http://www.w3.org/TR/css3-background/#the-box-shadow>
[consulté le 29 nov. 2015]

[effets d'ombrage], p.33

PAULUND. *CSS Box Shadows Effects*. 5 oct. 2015.

En ligne, disponible sur <http://www.paulund.co.uk/creating-different-css3-box-shadows-effects>
[consulté le 29 nov. 2015]



[les propriétés du fond], p.34

B. BOS, E.J. ETEMAD, B. KEMPER (ÉD). *CSS Backgrounds and Borders Module Level 3*. W3C Candidate Recommendation, 9 sept. 2014, § 3. Backgrounds.
En ligne, disponible sur <http://www.w3.org/TR/css3-background/#backgrounds>
[consulté le 29 nov. 2015]

[W3C gradients], p.35

E.J. ETEMAD, T. ATKINS JR. (ÉD). *CSS Image Values and Replaced Content Module Level 3*. W3C Candidate Recommendation, 17 avril 2012, § 4. Gradients.
En ligne, disponible sur <http://www.w3.org/TR/css3-images/#gradients>
[consulté le 30 nov. 2015]

[Moz linear-gradient], p.35

NAZCANGE ET AL.. *linear-gradient*. Mozilla Developer Network, 13 oct. 2015.
En ligne, disponible sur <https://developer.mozilla.org/fr/docs/Web/CSS/linear-gradient>
[consulté le 30 nov. 2015]

[CSS3 patterns Gallery], p.35

L. VEROU. *CSS3 Patterns Gallery*. mars 2013.
En ligne, disponible sur <http://lea.verou.me/css3patterns/>
[consulté le 30 nov. 2015]

[CSS3 gradients], p.35

L. VEROU. *CSS3 gradient from #000 to #fff on 45 minutes*. mai 2011.
En ligne, disponible sur <http://lea.verou.me/css3-gradients/>
[consulté le 30 nov. 2015]

[3D Gradient Box], p.35

DAN LUTON. *CSS3 Experiments #1: 3D Gradient Box*. Toolbox Digital Services, 6 mars 2011.
En ligne, disponible sur <http://toolboxdigital.com/2011/03/css3-3d-gradient-box/>
[consulté le 30 nov. 2015]

[Open Font Library], p.35

C. ADAMS. *Open Font Library*. Fabricatorz, mai 2011.
En ligne, disponible sur <https://fontlibrary.org/>
[consulté le 30 nov. 2015]

[Google Fonts], p.35

GOOGLE (ÉD). *Google Fonts*.
En ligne, disponible sur <https://www.google.com/fonts>
[consulté le 30 nov. 2015]

[Polices libres sur Github], p.35

GITHUB INC. (ÉD). *Fon*t. .
En ligne, disponible sur <https://github.com/showcases/fonts>
[consulté le 30 nov. 2015]

[CSS3 Font Module], p.35

J. DAGGET (ÉD). *CSS Fonts Module Level 3*. W3C Candidate Recommendation, 3 oct. 2013.
En ligne, disponible sur <http://www.w3.org/TR/css3-fonts/#font-face-rule>
[consulté le 30 nov. 2015]

[CSS 2.1 Text], p.36

B. BOS, T. ÇELIK, I. HICKSON, H.W. LIE. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation, 7 juin 2011, § 16 Text.
En ligne, disponible sur <http://www.w3.org/TR/CSS2/text.html>
[consulté le 30 nov. 2015]



[CSS3 Font Module], p.37

E.J. ETEMAD, K. ISHII (ÉD). *CSS Text Decoration Module Level 3*. W3C Candidate Recommendation, 1 août 2013, § 4. Text Shadows: the 'text-shadow' property.
En ligne, disponible sur <http://www.w3.org/TR/css-text-decor-3/#text-shadow-property>
[consulté le 30 nov. 2015]

[Dimensionnement des boîtes], p.38

B. BOS, T. ÇELIK, I. HICKSON, H.W. LIE. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C Recommendation, 7 juin 2011, § 10 Visual formatting model details.
En ligne, disponible sur <http://www.w3.org/TR/CSS2/visudet.html>
[consulté le 1 déc. 2015]

[specs box-sizing], p.38

T. ÇELIK, F. RIVOAL. *CSS Basic User Interface Module Level 3 (CSS3 UI)*. W3C Candidate Recommendation, 7 juillet 2015, § 3.1. 'box-sizing' property.
En ligne, disponible sur <http://www.w3.org/TR/css3-ui/#box-sizing>
[consulté le 1 déc. 2015]

[Wikipedia Javascript], p.40

WIKIPÉDIA (ÉD). *JavaScript*. 1 déc. 2015.
En ligne, disponible sur <http://en.wikipedia.org/wiki/JavaScript>
[consulté le 1 déc. 2015]

[specs ECMAScript], p.40

ECMA INTERNATIONAL (ÉD). *Standard ECMA-262, ECMAScript Language Specification*. juin 2011.
En ligne, disponible sur <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>
[consulté le 1 déc. 2015]

[DOM2 Core], p.40

A. LE HORS, PH. LE HÉGARET, L. WOOD ET AL. (ÉD). *Document Object Model (DOM) Level 2 Core Specification*. W3C Recommendation, 13 nov. 2000.
En ligne, disponible sur <http://www.w3.org/TR/DOM-Level-2-Core/>
[consulté le 1 déc. 2015]

[DOM2 HTML], p.40

J. STENBACK, PH. LE HÉGARET, A. LE HORS (ÉD). *Document Object Model (DOM) Level 2 HTML Specification*. W3C Recommendation, 9 jan. 2003.
En ligne, disponible sur <http://www.w3.org/TR/DOM-Level-2-HTML/>
[consulté le 1 déc. 2015]

[Selectors API], p.41

A. VAN KESTEREN, L. HUNT (ÉD). *Selectors API Level 1*. W3C Recommendation, 21 fév. 2013.
En ligne, disponible sur <http://www.w3.org/TR/selectors-api/>
[consulté le 1 déc. 2015]

[DOM2 Events], p.47

T. PIXLEY (ÉD). *Document Object Model (DOM) Level 2 Events Specification*. W3C Recommendation, 13 nov. 2000.
En ligne, disponible sur <http://www.w3.org/TR/DOM-Level-2-Events/>
[consulté le 1 déc. 2015]

[Utiliser XMLHttpRequest], p.49

D. BRUANT ET AL.. *Using XMLHttpRequest*. Mozilla Developer Network, 15 sept. 2015.
En ligne, disponible sur https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
[consulté le 1 déc. 2015]



[W3C XMLHttpRequest], p.49

A. VAN KESTEREN, J. AUBOURG, J. SONG, H.R.M. STEEN (ÉD). *XMLHttpRequest Level 1*. W3C Working Draft, 30 jan. 2014.

En ligne, disponible sur <http://www.w3.org/TR/XMLHttpRequest/>
[consulté le 1 déc. 2015]

[keyup event], p.50

J.-Y. PERRIER ET AL.. *keyup*. Mozilla Developer Network, 19 juin 2014.

En ligne, disponible sur <https://developer.mozilla.org/en-US/docs/Web/Events/keyup>
[consulté le 1 déc. 2015]

[W3C UI Events], p.50

G. KACMARCIK, T. LEITHEAD (ÉD). *UI Events (formerly DOM Level 3 Events)*. W3C Working Draft, 28 avril 2015.

En ligne, disponible sur <http://www.w3.org/TR/uievents/>
[consulté le 1 déc. 2015]