

Les bases du HTML

Par Josselin Willette 

Date de publication : 24 juillet 2009

Ce cours complet sur le HTML va vous permettre d'en apprendre les bases conformément aux recommandations et normes en vigueur établies par le W3C. Il répondra à quasiment toutes les questions que vous pouvez vous poser sur ce langage.

Je tiens à remercier **Kerod** et **Auteur** qui m'ont conseillé et accompagné tout au long de la rédaction de ce tutoriel et qui ont participé à la relecture de cet article, ainsi que **dourouc05**, **koopajah** et **RideKick** pour leur relecture attentive.

Des remarques ? Des questions ? N'hésitez pas, commentez cet article : .




I - Introduction.....	4
I-A - Introduction au HTML.....	4
I-A-1 - Qu'est-ce que le HTML ?.....	4
I-A-2 - Qu'est-ce qu'une balise HTML ?.....	4
I-A-2-a - Les balises simples.....	4
I-A-2-b - Les balises doubles.....	4
I-A-2-c - Imbrication des balises.....	5
I-B - Structure d'un document HTML.....	5
I-C - Le DOCTYPE.....	5
I-D - Les commentaires.....	7
II - Le corps du document - body.....	7
II-A - Les niveaux de titres.....	7
II-B - Les paragraphes de texte.....	8
II-C - La mise en forme de texte.....	8
II-C-1 - Le texte en gras.....	8
II-C-2 - Le texte en italique.....	8
II-C-3 - Le texte souligné.....	8
II-C-4 - Le texte barré.....	8
II-C-5 - Le texte en exposant.....	9
II-C-6 - Le texte en indice.....	9
II-C-7 - Modifier la couleur du texte.....	9
II-C-8 - Modifier la police du texte.....	9
II-C-9 - Modifier la taille du texte.....	9
II-D - Les images et zones réactives.....	10
II-D-1 - Les images.....	10
II-D-2 - Les zones réactives.....	10
II-D-2-a - Les zones rectangulaires.....	11
II-D-2-b - Les zones circulaires.....	11
II-D-2-c - Les zones polygonales.....	12
II-E - Les liens hypertextes.....	12
II-E-1 - Les liens externes.....	12
II-E-2 - Les liens internes.....	12
II-E-2-a - Les ancrs.....	13
II-E-2-b - Les liens vers d'autres cadres.....	13
II-E-3 - Les liens e-mails.....	13
II-E-4 - Les protocoles.....	14
II-E-5 - Informations complémentaires.....	14
II-F - Les cadres.....	15
II-F-1 - À quoi servent-ils ?.....	15
II-F-2 - Débuter avec les cadres.....	15
II-F-3 - Les cadres locaux.....	16
II-F-4 - Les cadres aujourd'hui.....	17
II-G - Les listes.....	17
II-G-1 - Les différents types de listes.....	17
II-G-1-a - Les listes non ordonnées.....	17
II-G-1-a-i - Les différents types de puces.....	18
II-G-1-a-ii - Les puces en image.....	18
II-G-1-b - Les listes ordonnées.....	19
II-G-1-b-i - Les différents types de numérotation.....	19
II-G-1-b-ii - Modifier le début de la numérotation.....	21
II-G-1-c - Les listes de définitions.....	21
II-G-1-d - Les listes de répertoires.....	21
II-G-1-e - Les listes de menus.....	22
II-G-2 - Les listes imbriquées.....	22
II-G-3 - L'utilité des listes aujourd'hui.....	23
II-H - Les tableaux.....	23
II-H-1 - À quoi servent-ils ?.....	23
II-H-2 - Du tableau simple au tableau complexe.....	23


II-H-2-a - Les bases d'un tableau.....	23
II-H-2-b - Le titre d'un tableau.....	24
II-H-2-c - La ligne d'en-tête d'un tableau.....	25
II-H-2-d - Les différentes parties d'un tableau.....	25
II-H-2-e - La fusion de lignes et de colonnes d'un tableau.....	26
II-H-3 - L'utilité des tableaux aujourd'hui.....	27
II-I - Les formulaires.....	28
II-I-1 - À quoi servent-ils ?.....	28
II-I-2 - La balise form.....	28
II-I-3 - Les champs texte monolignes.....	28
II-I-4 - Les champs texte multilignes.....	29
II-I-5 - Les champs cachés.....	29
II-I-6 - Les champs de fichiers.....	29
II-I-7 - Les boutons radio.....	30
II-I-8 - Les cases à cocher.....	30
II-I-9 - Les boutons.....	31
II-I-9-a - Le bouton simple.....	31
II-I-9-b - Le bouton d'envoi.....	31
II-I-9-c - Le bouton image.....	31
II-I-9-d - Le bouton effacer.....	31
II-I-10 - Les listes.....	32
II-I-11 - Améliorez l'accessibilité de vos formulaires.....	33
III - L'entête du document - head.....	33
III-A - Le titre.....	33
III-B - Les scripts.....	34
III-B-1 - Les scripts externes.....	34
III-B-2 - Les scripts internes.....	34
III-C - Les styles.....	34
III-C-1 - Les styles externes.....	34
III-C-2 - Les styles internes.....	35
III-D - Les meta.....	35
III-D-1 - À quoi servent-elles ?.....	35
III-D-2 - La balise meta content-type.....	35
III-D-3 - Ajoutez des infos sur vous.....	36
III-D-4 - Ajoutez des infos sur votre site.....	37
III-D-5 - Ajoutez des infos sur la création.....	38
III-D-6 - Orientez les robots.....	38
III-D-7 - Orientez les visiteurs.....	39
III-D-8 - Plus encore.....	39
IV - Aller plus loin.....	41
IV-A - Le XHTML.....	41
IV-B - Le CSS.....	41
IV-C - Le référencement.....	41
IV-D - Et encore plus.....	41

I - Introduction

I-A - Introduction au HTML

I-A-1 - Qu'est-ce que le HTML ?

Le  **HTML** (HyperText Markup Language) est un format de présentation de données permettant de créer des  **pages web** pouvant être lues dans des  **navigateurs**.

C'est un langage de description de données, et non un  **langage de programmation**. Il est figé c'est-à-dire qu'une fois le document chargé dans le navigateur, il ne répond à aucune action de l'utilisateur sur le contenu de la page. Ce langage est pourvu d'un système de balisage qui va permettre de structurer notre document.

I-A-2 - Qu'est-ce qu'une balise HTML ?

Une balise HTML est un élément que l'on va ajouter au texte de départ pour dire au navigateur de quelle manière l'afficher. Elle n'est pas affichée telle quelle dans le navigateur, elle est interprétée par celui-ci. Elle est toujours délimitée par les signes < et >.

Exemple de balise

```
<html>
```

Une balise peut en plus comporter de zéro à plusieurs attributs. Les attributs sont des informations complémentaires qui la caractérisent. Ils se présentent sous la forme **nomattribut="valeur"**. Attention à ne pas confondre un attribut HTML et une propriété CSS !

Exemple de balise et son attribut

```
<html lang="fr">
```

Chaque balise ouverte doit être fermée, cependant il existe des exceptions. On distingue deux types de balises :

- Les balises simples.
- Les balises doubles.


I-A-2-a - Les balises simples

Ce sont des balises qui sont dites "vides", c'est-à-dire qu'elles ne vont contenir aucune autre balise HTML. Ces balises n'ont pas besoin d'être fermées.

Exemple de balise simple

```

```

 *Si ces balises n'ont pas besoin d'être fermées en HTML, ça devient faux en XHTML.*
Cf : **Les balises vides en XHTML**.

I-A-2-b - Les balises doubles

Les balises doubles sont dites ouvrantes/fermantes, c'est-à-dire qu'elles nécessitent deux balises, une ouvrante et une fermante dans lesquelles on va pouvoir mettre d'autres balises ou du texte.

La balise fermante est identique à la balise ouvrante, à la différence qu'elle contient un "/" pour indiquer à quel endroit on la ferme.

Exemple de balise double

```
<p>Ici du texte ou tout autre balise.</p>
```

I-A-2-c - Imbrication des balises

Lorsque l'on cumule l'écriture de plusieurs balises ouvrantes (forcément inévitable), il est impératif de respecter la hiérarchie des balises, c'est-à-dire que la première ouverte sera la dernière à être fermée.

Le chevauchement des balises est une grave erreur.

Exemple de balises imbriquées

```
<b><u>Ici les balises sont correctement imbriquées.</u></b>
```

Exemple de balises chevauchées

```
<b><u>Ici les balises ne sont</b> pas correctement imbriquées, c'est une erreur.</u>
```

Commentez l'article :

I-B - Structure d'un document HTML

Avant de commencer tout développement, il est important de connaître la structure d'une page HTML.

Commençons donc par voir celle-ci avec la structure minimale obligatoire d'une page qui sera notre base de travail tout au long de ce tutoriel :

Document HTML de base

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Titre de la page</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body>


  <!-- Ici votre site -->

</body>
</html>
```

La première ligne du document s'appelle le DOCTYPE. C'est une balise indispensable pour conserver la compatibilité du rendu de votre page sur les différents navigateurs modernes.

Après le DOCTYPE, vient la balise `<html>`. Elle encadre l'ensemble des autres balises.

La balise `<html>` contient deux grandes balises : `<head>`, qui va contenir tout l'en-tête de la page et `<body>` qui va contenir tout le corps du document.


 Si vous incluez des pages à l'aide d'un langage serveur (PHP, ASP, JSP...), ayez toujours en tête que le DOCTYPE, la balise `<html>` ainsi que les balises `<head>` et `<body>` sont **uniques** dans une page. Si tel n'est pas le cas dans votre code source, vous aurez de mauvaises surprises à l'affichage de votre site.


Commentez l'article :

I-C - Le DOCTYPE

À quoi sert ce fameux DOCTYPE ? Est-il vraiment utile ?

Oui, il est utile, même très utile.

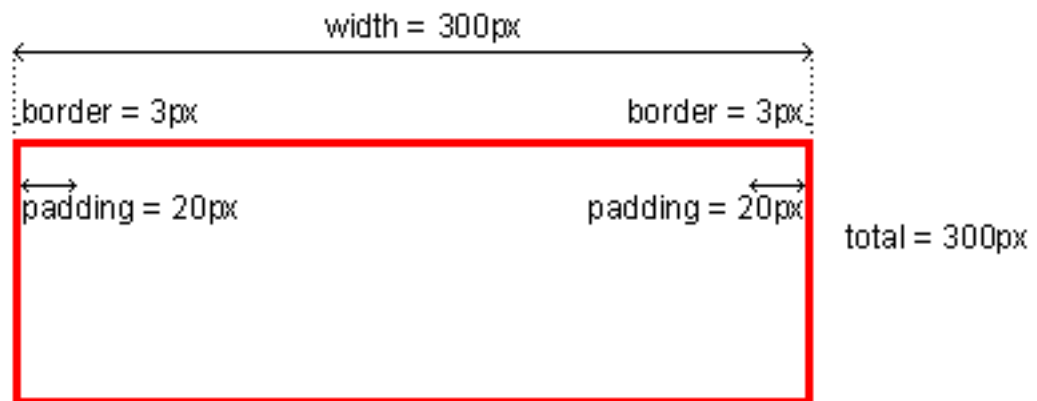
Il sert d'une part à déclarer quel "langage", "grammaire", ou norme on applique à notre page. Cette déclaration nous permet de soumettre nos pages au **validateur** du  **W3C** (World Wide Web Consortium) qui vérifie que notre code

HTML ne comporte aucune erreur de syntaxe (balises mal fermées, utilisation d'attributs inexistantes...) en fonction du DOCTYPE, donc de la  **DTD** (Document Type Definition) choisie. D'autre part, il indique au navigateur de quelle manière interpréter telle balise, telle propriété CSS. L'absence de cette balise est appelée le mode Quirks, ou non conforme, alors qu'au contraire, la présence de cette balise est appelée le mode Standard, ou conforme.

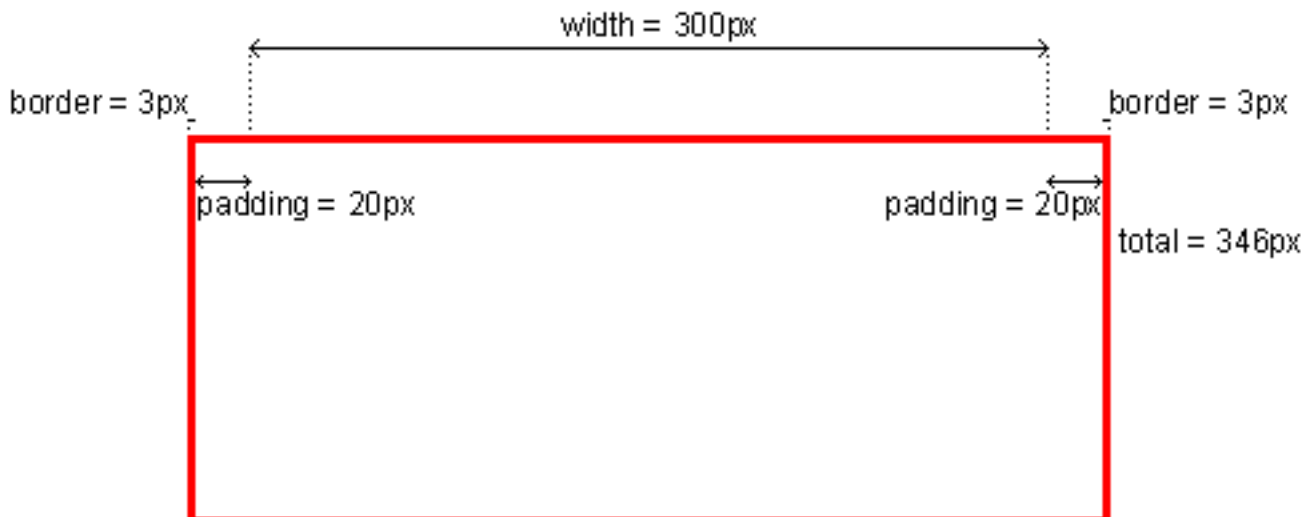
Pour vous faire une idée de son importance, en mode Quirks, le modèle de boîte Microsoft ne calcule pas de la même manière la taille d'un bloc que le **modèle W3C**, lorsque celui-ci a des marges intérieures (padding), voire même des bordures.

Une illustration vous montrera le résultat des deux modèles de boîte en mode Quirks :

Modèle Microsoft



Modèle W3C



Si vous pensez que c'est le modèle Microsoft qui réagit de la bonne manière, alors détrompez-vous. Il vous suffit de rajouter le DOCTYPE en première ligne pour vous rendre compte que ce n'est pas le navigateur qui respecte le modèle W3C qui va changer de comportement, mais bien celui qui utilise le modèle Microsoft.

Et l'URL sur la ligne du DOCTYPE, est-elle vraiment utile, elle ?

La réponse est encore une fois : oui. Vous pouvez certes faire valider vos pages sur le site du validateur du W3C sans l'URL, mais son oubli a la fâcheuse tendance à faire passer Internet Explorer en mode Quirks, ce qui anéantira tous vos efforts de compatibilité entre les différents navigateurs.

Plus d'infos sur les  **DTD** (Document Type Definition).

Vous pouvez choisir votre DOCTYPE parmi quelques-uns en fonction de votre degré de rigueur et/ou vos besoins.

Une liste non exhaustive est disponible dans la [FAQ \(X\)HTML](#).

Commentez l'article :

I-D - Les commentaires

Les commentaires permettent de laisser des informations ou des descriptions du code et sont utilisables à n'importe quel endroit de votre document. Cependant, ce ne sont pas ces commentaires-là qu'il faudra utiliser pour les langages JavaScript et CSS qui possèdent leur propre système de commentaires.

Ces commentaires ne seront pas affichés dans le navigateur mais ils resteront présents dans le code source, donc il faut à tout prix éviter de mettre des informations confidentielles dedans.

Ils sont délimités par `<!--` et `-->` et peuvent être multilignes.

Par souci de compatibilité entre les navigateurs et de mal interprétation de ceux-ci, il est fortement déconseillé d'écrire deux tirets d'union (ou plus) consécutifs.

Commentaires HTML

```
<!-- Voici un premier commentaire sur une ligne. -->

<!-- Voici un deuxième commentaire.
    Celui-ci est multilignes. -->

<!------- Cette méthode est déconseillée ! ----->
```

Commentez l'article :

II - Le corps du document - body

C'est dans la balise `<body>` que nous allons mettre tout le texte et toutes les images qui apparaîtront lorsqu'un visiteur viendra sur notre page.

II-A - Les niveaux de titres

Il existe différents niveaux de titres, ils vont de 1 à 6 et ont chacun leur importance. La balise représentant ces niveaux de titre est `<hX>` où X représente le niveau.

Exemple :

Exemple de titre

```
<h1>Titre de ma page</h1>
```

Il est important de respecter la hiérarchie des différents niveaux de titres, sachant que l'élément le plus haut est le `<h1>`. En effet, un titre `<h2>` ne se retrouvera jamais au-dessus d'un `<h1>` dans un document.

Le titre `<h1>` étant le plus important, il convient également de ne le faire apparaître qu'une seule fois dans un document.



Vous ne devez pas choisir un titre pour le rendu qu'il a dans le navigateur (différent selon les navigateurs et pouvant être modifié à l'aide de CSS), mais bien pour le niveau hiérarchique qu'il représente.

Commentez l'article :

II-B - Les paragraphes de texte

Comme vous tapez du texte dans un traitement de texte, vous segmentez votre texte en paragraphes. En HTML c'est la même chose, sauf qu'il faut préciser nous-même où commence un paragraphe et où il se termine. C'est ce à quoi sert la balise `<p>`.

Les paragraphes de texte en HTML

```
<p>Voici un premier paragraphe.</p>
```

```
<p>Voici un deuxième paragraphe.</p>
```

Commentez l'article :

II-C - La mise en forme de texte

*Les différentes techniques citées ci-dessous pour mettre en forme du texte sont **obsolètes**. Si elles fonctionnent encore en HTML 4.01, il vaut mieux s'en séparer et les remplacer par un langage fait pour ça et beaucoup plus puissant : le CSS.*



Voir :

- Les **cours sur le CSS**
- Un exemple d'**une feuille de style de base**

II-C-1 - Le texte en gras

Pour mettre le texte en gras on l'encadre de la balise `` :

Texte en gras

```
<b>Ce texte s'affichera en gras.</b>
```

II-C-2 - Le texte en italique

Pour mettre le texte en italique on l'encadre de la balise `<i>` :

Texte en italique

```
<i>Ce texte s'affichera en italique.</i>
```

II-C-3 - Le texte souligné

Pour souligner le texte on l'encadre de la balise `<u>` :

Texte souligné

```
<u>Ce texte sera souligné.</u>
```

II-C-4 - Le texte barré

Pour barrer le texte on l'encadre de la balise `<s>` :

Texte barré

```
<s>Ce texte sera barré.</s>
```


II-C-5 - Le texte en exposant

Pour placer le texte en exposant on l'encadre de la balise `<sup>` :

Texte en exposant

```
<sup>Ce texte sera en exposant.</sup>
```

II-C-6 - Le texte en indice

Pour placer le texte en indice on l'encadre de la balise `<sub>` :

Texte en indice

```
<sub>Ce texte sera en indice.</sub>
```

II-C-7 - Modifier la couleur du texte

Pour modifier la couleur du texte on utilise l'attribut `color` de la balise `` :

Texte en rouge

```
<font color="#ff0000">Ce texte sera en rouge.</font>
```

Les couleurs peuvent être écrites de deux manières :

- En hexadécimal de type RVB et précédées d'un dièse (#) ; Exemples : #ff0000 => rouge, #00ff00 => vert, #0000ff => bleu.
- Textuelles en anglais US ; Exemples : red, yellow, pink.

II-C-8 - Modifier la police du texte

Pour modifier la police du texte on utilise l'attribut `face` de la balise `` :

Texte en verdana

```
<font face="verdana">Ce texte sera en verdana.</font>
```

On a tendance à écrire des familles de polices plutôt qu'une seule police. En effet, il se peut que sur un système différent du vôtre la police choisie ne soit pas installée et il faut qu'il puisse alors afficher le texte dans une police proche de celle que vous vouliez avoir.

On sépare alors les différentes polices par une virgule dans l'ordre de sélection le cas où la précédente ne peut pas être affichée.

Texte en verdana ou sans-serif

```
<font face="verdana,sans-serif">Ce texte sera en verdana ou en sans-serif si verdana n'est pas installée.</font>
```

II-C-9 - Modifier la taille du texte

Pour modifier la taille du texte on utilise l'attribut `size` de la balise `` :

Texte en taille 5

```
<font size="5">Ce texte sera en taille 5.</font>
```

Par défaut, la valeur de l'attribut `size` vaut **"3"**.

Il y a deux manières de modifier la valeur de cet attribut : et les valeurs possibles sont les entiers de **"1"** à **"7"**.

- En absolu : Les valeurs possibles sont les entiers de "1" à "7".
- En relatif : Les valeurs possibles les entiers de "-2" à "+4".



Et il est bien évidemment possible de renseigner ces trois attributs (**color**, **face**, **size**) à la fois dans la même balise ``.

Commentez l'article :

II-D - Les images et zones réactives

II-D-1 - Les images

Pour afficher une image en HTML, on utilise la balise ``. Elle admet différents attributs, dont les plus importants sont **src** et **alt**.


L'attribut **src** permet de renseigner le chemin de l'image, qui peut être  **absolu** HTTP ou  **relatif**, très utile pour pouvoir l'afficher.

L'attribut **alt** affiche un texte alternatif à la place de l'image dans le cas où, pour une raison ou autre, celle-ci ne peut pas être chargée. Il aide de plus les moteurs de recherche à référencer l'image et les déficients visuels à savoir ce que l'image représente.

Afficher une image

```

```

 L'attribut **alt** n'est pas censé créer une infobulle lors du survol de l'image, qui est le rôle de l'attribut **title**, cependant sur Internet Explorer le **alt** affiche cette infobulle.

Les différents formats d'image supportés par cette balise sont :

- GIF
- JPG
- PNG
- BMP (à éviter)

II-D-2 - Les zones réactives

Le HTML permet de créer des zones réactives sur les images, afin de définir plusieurs liens différents sur une seule et même image, ou de ne rendre qu'une seule partie de l'image cliquable. On appelle l'ensemble de ces zones une carte, ou map.

Les balises qui permettent de faire ces zones réactives sont `<map>` et `<area>` étroitement liées avec la balise ``. En effet, on doit forcément préciser sur quelle image ces zones vont être présentes.

L'attribut de la balise `` qui va contenir le nom de la map dont elle fait référence est **usemap**. Le nom de cette map est précédé du signe dièse (#).

Image mappée

```

```

La balise qui va contenir l'ensemble de nos zones réactives pour cette image est la balise `<map>`, qui doit obligatoirement avoir l'attribut **name** renseigné et qui aura pour valeur le nom de la map, le même renseigné dans l'attribut **usemap** de notre image.

Comme vous l'aurez sans doute deviné et comme son nom l'indique, la balise `<area>` sera une zone réactive dans notre map. Il y a une balise area par zone.

Structure d'une map

```
<map name="nomDeLaMap">
  <!-- Ici nous allons mettre nos différentes zones réactives -->
  <area ...>
```

Structure d'une map

```
<area ...>
</map>
```

Les différents attributs de la balise `<area>` nécessaires à son bon fonctionnement sont :

- **coords**, qui va contenir les différentes coordonnées de notre zone.
- **href**, qui va définir l'adresse de destination du lien.
- **shape**, qui va renseigner le type de zone que l'on souhaite afficher.

On distingue trois types de zones :

- Les zones rectangulaires,
- Les zones circulaires,
- Les zones polygonales.

On peut bien évidemment avoir autant de zones que l'on veut sur une map, et on peut également combiner les différents types de zones sur une seule et même image.

Attention toutefois à ne pas faire se chevaucher les zones ! Le cas échéant, c'est la zone définie la première qui s'affichera au-dessus.

II-D-2-a - Les zones rectangulaires

Pour définir une zone rectangulaire, l'attribut **shape** doit prendre pour valeur **"rectangle"** ou **"rect"**.

Les coordonnées définies dans l'attribut **coords** se réfèrent aux abscisses et ordonnées des coins supérieur gauche et inférieur droit du rectangle que l'on souhaite tracer par rapport au coin haut gauche de l'image. Ces valeurs sont séparées par une virgule et exprimées en pixels.

Syntaxe :

Zone rectangulaire

```
<area shape="rect" coords="x1,x2,y1,y2" href="adresse/de/destination">
```

Exemple :

Zone rectangulaire

```
<area shape="rect" coords="10,50,60,60" href="adresse/de/destination">
```

Voir le résultat

II-D-2-b - Les zones circulaires

Une zone circulaire est définie par la valeur **"circle"** ou **"circ"** de l'attribut **shape**.

Les valeurs de l'attribut **coords** se réfèrent aux abscisse et ordonnée du centre du cercle ainsi que le rayon de ce cercle par rapport au coin haut gauche de l'image. Ces valeurs sont séparées par une virgule et exprimées en pixels.

Syntaxe :

Zone circulaire

```
<area shape="circ" coords="x,y,rayon" href="adresse/de/destination">
```

Exemple :

Zone circulaire

```
<area shape="circ" coords="40,40,10" href="adresse/de/destination">
```

Voir le résultat

II-D-2-c - Les zones polygonales

C'est la valeur **"polygon"** ou **"poly"** de l'attribut **shape** qui permet de définir une zone polygonale. Les valeurs de l'attribut **coords** se réfèrent aux abscisse et ordonnée de chacun des points par rapport au coin haut gauche de l'image. Ces valeurs sont séparées par une virgule et exprimées en pixels.
Syntaxe :

Zone polygonale

```
<area shape="poly" coords="x1,y1,x2,y2,x3,y3,...,xn,yn" href="adresse/de/destination">
```

Exemple :

Zone polygonale

```
<area shape="poly" coords="0,0,40,50,78,101,17,150" href="adresse/de/destination">
```

Voir le résultat

Commentez l'article :

II-E - Les liens hypertextes

Un lien hypertexte est un élément HTML permettant d'envoyer le visiteur vers une nouvelle page. On peut insérer dans cet élément toute sorte de textes, images et autres balises dites "inline" (comme les balises ``, ``, etc). Un lien sera représenté dans le code par la balise `<a>`. Tout ce qui sera dans cette balise fera office de lien. L'adresse de destination doit se trouver dans l'attribut **href**.
L'attribut **title**, quant à lui, permet de renseigner un petit descriptif sur le contenu de la page cible, et est un atout non négligeable en matière de référencement et d'accessibilité.

Exemple texte :

Exemple d'un lien texte

```
<a href="adresse/de/destination.html">Texte du lien</a>
```

Exemple image :

Exemple d'un lien image

```
<a href="adresse/de/destination.html"></a>
```

On distingue deux types de liens, les liens externes et les liens internes.

II-E-1 - Les liens externes

Un lien externe est un lien hypertexte qui renvoie vers une autre page. Cette autre page peut être une page de votre site ou celle d'un site tiers.

II-E-2 - Les liens internes

Un lien interne est un lien qui renvoie vers la même page, mais pas forcément au même endroit de la page. Attention à ne pas confondre "liens internes" avec "lien interne au site" !
Un lien interne au site est en général un lien externe à la page.

II-E-2-a - Les ancres

Les ancres permettent de naviguer à l'intérieur d'une même page. Par exemple, ce tutoriel a en haut un sommaire avec des liens qui renvoient plus bas dans la page, ce sont des ancres.

Pour faire une ancre, il faut utiliser le signe dièse (#) suivi d'un mot-clef. Ce mot-clef sera repris dans l'attribut **id** d'une balise à laquelle on fait faire une ancre :

Exemple d'une ancre interne

```
<a href="#motClef">Texte de mon ancre</a>

[...]

<h2 id="motClef">Texte vers lequel l'ancre est renvoyée</h2>
```

Si vous voulez faire une ancre vers une page externe, il vous suffit de rajouter l'adresse de destination dans la balise de lien :

Exemple d'une ancre externe

```
<a href="adresse/de/destination.html#motClef">Texte de mon ancre</a>
```

II-E-2-b - Les liens vers d'autres cadres

Si vous utilisez des cadres pour votre navigations, vous aurez certainement envie que les liens de votre menu, dans le cadre nommé "cadreMenu" par exemple, pointent vers votre cadre principal que l'on appellera "cadrePrincipal". Dans ce cas, il faut renseigner l'attribut **target** de la balise de lien :

Exemple d'un lien vers un autre cadre

```
<a href="adresse/de/destination.html" target="cadrePrincipal">Lien de mon menu</a>
```

Il existe plusieurs valeurs pour l'attribut **target** :

Valeur	Description
_self	Affichage dans le cadre courant (par défaut).
_blank	Ouvre le lien dans une nouvelle fenêtre.
_top	Affichage dans la totalité de la fenêtre courante.
_parent	Affichage dans le cadre qui enveloppe la structure courante.
NomDuCadre	Affichage dans le cadre spécifié.

Pour plus d'infos sur les cadres, reportez-vous à la section "**Les cadres**".

II-E-3 - Les liens e-mails

Vous pouvez permettre à vos visiteurs d'écrire sur votre adresse e-mail avec la syntaxe "**mailto:**" :

Exemple d'un lien vers une adresse e-mail

```
<a href="mailto:jwillette@monsite.com">Ecrivez-moi</a>
```



Ce type de liens ouvre le logiciel de messagerie par défaut de vos visiteurs.

Vous pouvez aussi renseigner un sujet par défaut de l'e-mail, ainsi qu'un corps voire même les deux :

Exemple d'un lien vers une adresse e-mail avec un sujet par défaut

```
<a href="mailto:jwillette@monsite.com?subject=Sujet du mail">Ecrivez-moi</a>
```

Exemple d'un lien vers une adresse e-mail avec un corps par défaut

```
<a href="mailto:jwillette@monsite.com?body=Corps du mail">Ecrivez-moi</a>
```

Exemple d'un lien vers une adresse e-mail avec un sujet et un corps par défaut

```
<a href="mailto:jwillette@monsite.com?subject=Sujet du mail&body=Corps du mail">Ecrivez-moi</a>
```

Vous pouvez enfin mettre plusieurs destinataires ou mettre une personne en copie du mail :

Exemple d'un lien vers plusieurs adresses e-mail

```
<a href="mailto:destinataire1@monsite.com;destinataire2@monsite.com">Ecrivez-nous</a>
```

Exemple d'un lien vers une adresse e-mail et une autre en copie

```
<a href="mailto:destinataire1@monsite.com?cc=destinataire2@monsite.com">Ecrivez-nous</a>
```

II-E-4 - Les protocoles

Un protocole est un moyen qu'ont deux ordinateurs de communiquer. Il diffère selon le type de données échangées. Il existe de nombreux protocoles disponibles par lien hypertexte, en voici une liste non exhaustive :

Protocole	Signification de l'acronyme	Description	Syntaxe utilisée
HTTP	HyperText Transfer Protocol	Permet de transférer des données sur l'Internet.	<code>...</code>
HTTPS	HyperText Transfer Protocol Secured	Permet de transférer des données sur l'Internet en permettant de vérifier l'identité du site grâce à un certificat d'authentification.	<code>...</code>
SMTP	Simple Mail Transfer Protocol	Permet de transférer des e-mails.	<code>...</code>
FTP	File Transfer Protocol	Permet de transférer des fichiers.	<code>...</code>
NNTP	Networks News Transfer Protocol	Permet de lire, distribuer, ... les news de newsgroups.	<code>...</code>
TELNET		Permet d'accéder à un ordinateur distant.	<code>...</code>
IRC	Internet Relay Chat	Permet de se connecter à un serveur de Chat.	<code>...</code>

II-E-5 - Informations complémentaires

- Pour éviter d'avoir le cadre bleu autour d'une image lorsqu'elle est dans un lien, il vous suffit de renseigner l'attribut **border** à "0" :

Image sans bordure bleue

```
<a href="adresse/de/destination.html"></a>
```

- Si vous voulez avoir une infobulle qui s'affiche sur votre lien lors du passage de la souris, vous pouvez renseigner l'attribut **title** :

Infobulle sur le lien

```
<a href="adresse/de/destination.html" title="Description de la page cible">Texte du lien</a>
```

- Vous pouvez remplacer l'adresse du lien par l'appel d'une fonction Javascript qui s'exécutera au clic du lien.

Fonction Javascript appelée dans un lien

```
<a href="javascript:maFonctionJavascript()">Texte du lien</a>
```

Commentez l'article :

II-F - Les cadres

II-F-1 - À quoi servent-ils ?

Les cadres servent à séparer la fenêtre en différents morceaux indépendants qui peuvent chacun contenir une page HTML différente, ces morceaux ayant la capacité de communiquer entre eux.

On utilise au minimum deux cadres dans une fenêtre, en général un cadre pour le menu à gauche et un autre pour le contenu du site. Parfois on rajoute un cadre en haut pour un bandeau fixe.

II-F-2 - Débuter avec les cadres

Le document HTML qui va appeler les différents cadres ne pourra servir qu'à ça. Aucun texte ou aucune image inséré dans ce document ne sera affiché dans le navigateur du visiteur. Donc dans le cas où nous avons deux cadres, un pour le menu et un pour le contenu, nous aurons trois fichiers différents. Comme les deux pages du menu et du contenu sont des pages ordinaires, nous ferons l'impasse dessus. En revanche, la page qui appelle les cadres est tout à fait différente.

Ce sont les balises `<frameset>` et `<frame>` qui vont être utilisées pour la mise en oeuvre du jeu de cadres.

Document HTML de base pour un jeu de cadres

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
  <title>Titre de la page</title>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<frameset>

<!-- Ici le contenu de la frameset principale -->

</frameset>
</html>
```

On remarque la présence de la balise `<frameset>` qui remplace d'ordinaire la balise `<body>` et le DOCTYPE approprié pour l'usage des cadres.

La balise `<frameset>` va désigner le nombre de cadres et leur taille.

Si on renseigne l'attribut `cols`, la page sera coupée verticalement, alors que si on renseigne l'attribut `rows`, la page sera coupée horizontalement. Ces attributs prennent pour valeur les tailles des différents cadres séparées par une virgule. Si une des valeurs est une étoile, ça signifie que le cadre en question prendra toute la place disponible restante dans la fenêtre. On ne peut évidemment pas utiliser ces deux attributs conjointement sur une seule et même balise `<frameset>`.

Deux cadres verticaux

```
<frameset cols="20%, *">
```

Trois cadres verticaux

```
<frameset cols="20%, *, 20%">
```

Deux cadres horizontaux

```
<frameset rows="100, *">
```

Dans cette balise `<frameset>` nous allons mettre l'ensemble de nos cadres. Le nombre de cadres correspond au nombre de valeurs renseignées dans les attributs `cols` et `rows`.

Exemple de cadres

```
<frameset cols="200, *">
  <frame>
  <frame>
</frameset>
```

C'est dans la balise `<frame>` que l'on va fixer la source de chacun des cadres, grâce à l'attribut `src`.

Un attribut important de cette balise est `name`. En effet, il va falloir dire à nos liens qui sont dans le menu, par exemple, d'aller changer la source d'un autre cadre. L'attribut `target` du lien prendra alors pour valeur le nom du cadre ciblé.

On peut aussi jouer sur les bordures avec l'attribut `frameborder` qui prend pour valeur `"0"` ou `"1"`.

De plus, par défaut, les cadres peuvent être redimensionnés. Si on veut interdire cette fonctionnalité, on renseignera l'attribut `noresize` qui prend pour seule valeur `"noresize"`.

Enfin, on peut éliminer les marges intérieures latérales avec les attributs `marginwidth` et `marginheight` renseignés à `"0"`.

Exemple de cadres

```
<frameset cols="200, *">
  <frame name="menu" src="menu.html">
  <frame name="contenu" src="page1.html">
</frameset>
```

Exemple d'un lien sur la page menu.html

```
<a href="page2.html" target="contenu">Aller à la page 2</a>
```

On peut aller encore plus loin, en imbricant différents cadres. Par exemple, on veut un bandeau en haut qui prend toute la largeur de l'écran. En-dessous, on veut deux cadres verticaux, un pour le menu et l'autre pour le contenu. Le code HTML ressemblera alors à quelque chose comme :

Exemple de cadres

```
<frameset rows="100, *">
  <frame name="bandeau" src="bandeau.html">
  <frameset cols="200, *">
    <frame name="menu" src="menu.html">
    <frame name="contenu" src="page1.html">
  </frameset>
</frameset>
```

II-F-3 - Les cadres locaux

Le cadre local ou **"iframe"** permet également d'insérer une page externe dans notre site, à la différence qu'il sera à la suite du contenu de notre propre document et non pas séparé par un troisième fichier qui fait la jointure.

L'attribut de la balise `<iframe>` qui va servir à afficher le contenu d'une autre page est `src`. On lui renseigne l'adresse complète si le site est différent du nôtre, ou le chemin relatif ou absolu dans le cas où l'on veut afficher une de nos propres pages.

On peut supprimer les bordures de l'iframe pour lui donner l'impression de mieux s'intégrer à notre document en renseignant l'attribut `frameborder` à `"0"` (`"1"` par défaut).

Aussi, les marges intérieures sont assez gênantes quand on a une page qui doit faire exactement les dimensions données à cette balise. On peut alors les supprimer avec les attributs `marginwidth` et `marginheight` renseignés à `"0"`.

Dans le cas où un navigateur ne peut pas afficher ce cadre local, il est intéressant de laisser un texte alternatif. C'est le texte que l'on écrit à l'intérieur de la balise.

Exemple de cadre local

```
<iframe frameborder="0" marginwidth="0" marginheight="0" src="http://adresse/d/une/page.html">
```


Exemple de cadre local

Désolé, votre navigateur ne supporte pas les cadres locaux
</iframe>

II-F-4 - Les cadres aujourd'hui

De nos jours, il est fortement déconseillé d'utiliser ce genre de procédé. Voici une liste non-exhaustive des problèmes encourus de part l'utilisation des frames :

- Impossibilité d'ajouter aux favoris la page désirée.
- Mauvaise gestion de l'impression par les navigateurs.
- Perte du sens véritable de la nature d'un document HTML censé être identifié par une URL précise et unique.
- Référencement des pages du site impossible dans les moteurs de recherche.
- Technologie obsolète qui est totalement abandonnée dans les futures versions du (X)HTML.

Commentez l'article :

II-G - Les listes

Elles servent, comme leur nom l'indique, à créer une liste d'items, plus précisément à les énumérer, comme dans le cas d'un sommaire par exemple.

II-G-1 - Les différents types de listes

On distingue cinq types de listes différents, chacun prévu pour un usage bien précis :

- Les listes non ordonnées, ou à puces.
- Les listes ordonnées.
- Les listes de définitions.
- Les listes de répertoires.
- Les listes de menus.

II-G-1-a - Les listes non ordonnées

Elles sont démarquées par la balise `` et chacun des items est entouré de la balise `` :

Exemple de liste non ordonnée

```
<p>Ma liste de courses :</p>
<ul>
  <li>Champignons</li>
  <li>Avocat</li>
  <li>Dentifrice</li>
  <li>Céréales</li>
</ul>
```

Résultat :

Ma liste de courses :

- Champignons
- Avocat
- Dentifrice
- Céréales

II-G-1-a-i - Les différents types de puces

On va pouvoir jouer sur le design des puces à l'aide de trois types différents. On renseigne le type des puces avec l'attribut **type** de la balise ``.

Les types sont :

- **disc** (par défaut au niveau 1 d'imbrication)
- **circle** (par défaut au niveau 2 d'imbrication)
- **square** (par défaut au niveau 3 et suivants d'imbrication)

Dans l'exemple précédent, nous avons donc vu une liste non ordonnée avec une puce de type disc. Voici à quoi ressemblent les deux autres :

Exemple de liste non ordonnée

```
<p>Ma liste de courses :</p>
<ul type="circle">
  <li>Champignons</li>
  <li>Avocat</li>
  <li>Dentifrice</li>
  <li>Céréales</li>
</ul>
```

Résultat :

Ma liste de courses :

- Champignons
- Avocat
- Dentifrice
- Céréales

Exemple de liste non ordonnée

```
<p>Ma liste de courses :</p>
<ul type="square">
  <li>Champignons</li>
  <li>Avocat</li>
  <li>Dentifrice</li>
  <li>Céréales</li>
</ul>
```

Résultat :

Ma liste de courses :

- Champignons
- Avocat
- Dentifrice
- Céréales

II-G-1-a-ii - Les puces en image

Des disques, des cercles ou des carrés, c'est bien beau, mais si on veut une puce en forme de flèche et de couleur bleue ?

Pour cela, rien de plus simple, il suffit d'utiliser une image pour remplacer la puce originale. Cependant cette méthode n'est possible qu'à l'aide de CSS et de la propriété **list-style-image** :

Exemple de liste non ordonnée

```
<p>Ma liste de courses :</p>
<ul>
  <li>Champignons</li>
  <li>Avocat</li>
  <li>Dentifrice</li>
  <li>Céréales</li>
</ul>
```

CSS

```
li
{
  list-style-image : url(puce.gif);
}
```

Résultat :

Ma liste de courses :

- ▶ Champignons
- ▶ Avocat
- ▶ Dentifrice
- ▶ Céréales

Pour plus d'informations sur les puces imagées, je vous renvoie à l'article CSS plus complet :

CSS débutants : Puces en gifs avec les CSS

II-G-1-b - Les listes ordonnées

Elles sont introduites par la balise `` et chacun des items est encadré par la balise `` :

Exemple de liste ordonnée

```
<p>Top 3 du tiercé :</p>
<ol>
  <li>Jolly</li>
  <li>Tonnerre</li>
  <li>Tornado</li>
</ol>
```

Résultat :

Top 3 du tiercé :

1. Jolly
2. Tonnerre
3. Tornado

II-G-1-b-i - Les différents types de numérotation

Il existe cinq manières différentes de numéroter les liste ordonnées. On renseigne le type de numérotation avec l'attribut **type** de la balise ``.

Les types sont :

- **1** : Numérotation par les chiffres arabes (par défaut).
- **I** : Numérotation par les chiffres romains.
- **i** : Numérotation par les chiffres romains minuscules.
- **A** : Numérotation par les lettres de l'alphabet.
- **a** : Numérotation par les lettres de l'alphabet minuscules.

Dans l'exemple précédent, nous avons donc vu une liste ordonnée avec une numérotation par les chiffres arabes. Voici à quoi ressemblent les autres :

Exemple de liste ordonnée

```
<p>Top 3 du tiercé :</p>
<ol type="I">
  <li>Jolly</li>
  <li>Tonnerre</li>
  <li>Tornado</li>
</ol>
```

Résultat :

Top 3 du tiercé :

- I. Jolly
- II. Tonnerre
- III. Tornado

Exemple de liste ordonnée

```
<p>Top 3 du tiercé :</p>
<ol type="i">
  <li>Jolly</li>
  <li>Tonnerre</li>
  <li>Tornado</li>
</ol>
```

Résultat :

Top 3 du tiercé :

- i. Jolly
- ii. Tonnerre
- iii. Tornado

Exemple de liste ordonnée

```
<p>Top 3 du tiercé :</p>
<ol type="A">
  <li>Jolly</li>
  <li>Tonnerre</li>
  <li>Tornado</li>
</ol>
```

Résultat :

Top 3 du tiercé :

- A. Jolly
- B. Tonnerre
- C. Tornado

Exemple de liste ordonnée

```
<p>Top 3 du tiercé :</p>
<ol type="a">
  <li>Jolly</li>
  <li>Tonnerre</li>
  <li>Tornado</li>
</ol>
```

Résultat :

Top 3 du tiercé :

- a. Jolly
- b. Tonnerre
- c. Tornade

II-G-1-b-ii - Modifier le début de la numérotation

Il est tout à fait possible, pour une raison ou une autre, de modifier le numéro de départ par lequel va s'incrémenter la numérotation.

Pour cela, il suffit de préciser une valeur à l'attribut **start** de la balise ``. Cette valeur doit dans tous les cas être numérique :

Exemple de liste ordonnée

```
<p>Top 3 inversé du tiercé :</p>
<ol type="I" start="8">
  <li>Tortue</li>
  <li>Escargot</li>
  <li>Limace</li>
</ol>
```

II-G-1-c - Les listes de définitions

Elles sont délimitées par la balise `<dl>` qui comprend une liste de termes à définir représentés par la balise `<dt>` qui peuvent avoir une ou plusieurs définitions elles-mêmes encadrées par la balise `<dd>` (une pour chaque définition).

Exemple de liste de définitions

```
<dl>
  <dt>Oreiller</dt>
  <dd>Pièce de literie bourrée de plumes, de duvet, de crin... destinée à soutenir la tête pendant le repos.</dd>
  <dt>Puce</dt>
  <dd>Insecte de très petite taille se déplaçant par sauts.</dd>
  <dd>Marché où l'on vend des objets d'occasion.</dd>
</dl>
```

Résultat :

Oreiller	Pièce de literie bourrée de plumes, de duvet, de crin... destinée à soutenir la tête pendant le repos.
Puce	Insecte de très petite taille se déplaçant par sauts. Marché où l'on vend des objets d'occasion.

II-G-1-d - Les listes de répertoires

Elles sont encadrées par la balise `<div>` et chacun des items est compris dans la balise ``.

Exemple de liste de répertoires

```
<div>
  <li>css</li>
  <li>fichiers</li>
  <li>images</li>
  <li>js</li>
</div>
```

Résultat :

- css
- fichiers
- images
- js

II-G-1-e - Les listes de menus

Elles sont représentées par la balise `<menu>` et chacun des items est délimité par la balise ``.

Exemple de liste de menus

```
<menu>
  <li>Accueil</li>
  <li>Page 1</li>
  <li>Page 2</li>
</menu>
```

Résultat :

- Accueil
- Page 1
- Page 2

II-G-2 - Les listes imbriquées

L'imbrication des listes permet une présentation claire et détaillée d'un menu ou d'un sommaire par exemple. On peut imbriquer les listes non ordonnées et ordonnées sans distinction et même les mélanger :

Exemple de listes imbriquées

```
<ol type="I">
  <li>La genèse</li>
  <li>La vie
    <ol type="A">
      <li>L'enfance</li>
      <li>L'adolescence</li>
      <li>Le passage à l'âge adulte
        <ul>
          <li>Le jeune adulte</li>
          <li>La crise de la quarantaine</li>
        </ul>
      </li>
      <li>La vieillesse</li>
    </ol>
  </li>
  <li>La mort</li>
</ol>
```

Résultat :

- I. La genèse
- II. La vie
 - A. L'enfance
 - B. L'adolescence
 - C. Le passage à l'âge adulte
 - Le jeune adulte
 - La crise de la quarantaine
 - D. La vieillesse
- III. La mort

II-G-3 - L'utilité des listes aujourd'hui

La principale utilisation des listes de nos jours se fait par les listes non ordonnées pour la création de divers menus. L'imbrication de ces listes non ordonnées couplée avec un soupçon de CSS permet de faire des menus à multiples niveaux d'affichage, comme les menus déroulants.

D'autre part, on peut retrouver les listes ordonnées principalement pour la rédaction de sommaires, notamment destinés aux FAQ.

Voici quelques exemples de menus qui utilisent ces listes non ordonnées : [Cours CSS : Menu et barre de navigation](#)
[Commentez l'article](#) :

II-H - Les tableaux

II-H-1 - À quoi servent-ils ?

Un tableau qu'est-ce que c'est ? Tout simplement une suite de lignes et de colonnes qui forment un ensemble de cellules.

Les tableaux servent avant tout à présenter des données tabulaires dont voici un exemple :

Elèves	Français	Mathématiques	Histoire	Sport
Laure	14	10	11	11
Philippe	8.5	12	9	15
Jean	15	14.5	10.5	12
Maurice	7.5	18	12	14

Malgré cela, l'emploi le plus fréquent des tableaux reste la mise en page de documents. En effet, l'implémentation des tableaux étant assez intuitive, les débutants préféreront en user ainsi. S'ajoute à cela les éditeurs WYSIWYG des logiciels de création de sites qui génèrent ce genre de code automatiquement.

Cependant, la mise en page par tableau devient dépréciée : Voir le paragraphe [L'utilité des tableaux aujourd'hui](#).

II-H-2 - Du tableau simple au tableau complexe

II-H-2-a - Les bases d'un tableau

Un tableau est délimité par la balise `<table>`. Le contenu d'un tableau HTML est généré horizontalement. C'est-à-dire qu'il n'est pas créé colonne par colonne mais ligne par ligne. Pour créer un tableau, nous commençons par créer des lignes grâce aux balises `<tr>`. Puis nous insérons dans ces lignes les cellules du tableau grâce aux balises `<td>`.

Le contenu (textes, images, autres balises) se trouve uniquement dans les balises `<td>` et ne doit se trouver **en aucun cas** en dehors.

Donc pour avoir un tableau d'une cellule, soit une ligne et une colonne, on obtient ce code :

Tableau d'une cellule

```
<table border="1">
  <tr>
    <td>Contenu de la cellule</td>
  </tr>
</table>
```

On obtient :

Contenu de la cellule

Et pour un tableau de deux lignes et deux colonnes :

Tableau de deux lignes et deux colonnes

```
<table border="1">
  <tr>
    <td>Cellule 1</td>
    <td>Cellule 2</td>
  </tr>
  <tr>
    <td>Cellule 3</td>
    <td>Cellule 4</td>
  </tr>
</table>
```

On obtient :

Cellu 1	Cellule 2
Cellu 3	Cellule 4

Par défaut, la bordure n'est pas visible, dans les codes précédents on a rajouté l'attribut **border** à la balise **<table>** pour la faire apparaître. On peut évidemment épaissir la bordure en augmentant la valeur.

On peut voir sur les tableaux affichés précédemment que le texte n'est pas directement collé à la bordure. Il s'agit là d'un espace interne par défaut des navigateurs. Pour le supprimer, on doit renseigner l'attribut **cellpadding** et lui mettre la valeur "0".


De la même manière, la bordure ne fait pas effectivement 1 pixel de large alors qu'on lui a renseigné la valeur 1. Il persiste un espace par défaut entre chaque cellule que l'on peut faire disparaître en affectant la valeur "0" à l'attribut **cellspacing**.

Un autre attribut nécessaire est l'attribut **summary**. Il contient du texte et fournit un résumé du tableau, utile pour les personnes malvoyantes.

Si on reprend tout ce dont on vient de parler, on aurait un code comme :

Récapitulatif des attributs de la balise table

```
<table border="1" cellpadding="0" cellspacing="0" summary="Résumé du tableau pour les personnes malvoyantes">
  <tr>
    <td>Cellule 1</td>
    <td>Cellule 2</td>
  </tr>
</table>
```

 **Attention à bien avoir le même nombre de **<td>** pour chaque ligne, sinon le navigateur risque de mal interpréter le tableau.**

II-H-2-b - Le titre d'un tableau

Pour décrire le contenu d'un tableau, par exemple expliquer dans notre tout premier exemple de tableau qu'il s'agit des notes de la classe de 3èmeB, il faut utiliser la balise **<caption>**, à placer directement après l'ouverture du tableau. Cette balise prend l'attribut **align** qui accepte comme valeurs **"top"** (par défaut), **"bottom"**, **"left"** et **"right"**, en fonction de l'endroit où l'on veut afficher cette légende.

Exemple d'un titre de tableau

```
<table border="1">
  <caption align="bottom">Notes de la classe de 3èmeB</caption>
  <tr>
    <td>Elèves</td>
    <td>Français</td>
    <td>Mathématiques</td>
    <td>Histoire</td>
    <td>Sport</td>
  </tr>
```


Exemple d'un titre de tableau

```
</tr>
<tr>
  <td>Laure</td>
  <td>14</td>
  <td>10</td>
  <td>11</td>
  <td>11</td>
</tr>
<tr>
  [...]
</tr>
</table>
```

II-H-2-c - La ligne d'en-tête d'un tableau

Pour reprendre l'exemple des notes de la classe de 3èmeB, on remarque que la première ligne peut faire office d'en-tête. En effet, elle renseigne le contenu de chacune des colonnes et n'a donc pas le même genre de contenu que tout le reste du tableau.

Pour préciser au navigateur qu'il s'agit donc de cellules spéciales, on va les modifier en remplaçant pour cette ligne seulement les balises `<td>` par des balises `<th>` :

Utilisation de la balise th

```
<table border="1">
  <caption align="bottom">Notes de la classe de 3èmeB</caption>
  <tr>
    <th>Elèves</th>
    <th>Français</th>
    <th>Mathématiques</th>
    <th>Histoire</th>
    <th>Sport</th>
  </tr>
  <tr>
    <td>Laure</td>
    <td>14</td>
    <td>10</td>
    <td>11</td>
    <td>11</td>
  </tr>
  <tr>
    [...]
  </tr>
</table>
```

II-H-2-d - Les différentes parties d'un tableau

Quand on arrive à avoir un tableau de grande taille, il devient moins évident à gérer. Pour cela, on peut regrouper les lignes par grandes parties : l'en-tête, le corps et le pied, respectivement représentées par les balises `<thead>`, `<tbody>` et `<tfoot>`.

Ces balises vont englober les lignes en fonction de l'endroit où les lignes doivent être placées. Elles sont évidemment facultatives mais aident à la maintenance du code.



Elles doivent apparaître dans l'ordre suivant : `<thead>`, `<tfoot>` et `<tbody>`.

Notre sempiternel exemple de la classe de 3èmeB contient clairement un en-tête, mais pas de pied. Qu'à cela ne tienne, on va en rajouter un, parce que la natalité ayant tant augmenté que les classes sont nombreuses et qu'arrivé en fin de tableau on ne sait plus à quoi correspondent les colonnes. Ce qui donne :

Exemple de groupage des lignes d'un tableau

```
<table border="1">
  <caption align="bottom">Notes de la classe de 3èmeB</caption>
```

Exemple de groupage des lignes d'un tableau

```

<thead>
  <tr>
    <th>Elèves</th>
    <th>Français</th>
    <th>Mathématiques</th>
    <th>Histoire</th>
    <th>Sport</th>
  </tr>
</thead>

<tfoot>
  <tr>
    <th>Elèves</th>
    <th>Français</th>
    <th>Mathématiques</th>
    <th>Histoire</th>
    <th>Sport</th>
  </tr>
</tfoot>

<tbody>
  <tr>
    <td>Laure</td>
    <td>14</td>
    <td>10</td>
    <td>11</td>
    <td>11</td>
  </tr>
  <tr>
    [...]
  </tr>
</tbody>
</table>

```

II-H-2-e - La fusion de lignes et de colonnes d'un tableau

On distingue deux types de fusion : la fusion horizontale où sont en jeu deux (ou plus) colonnes, la fusion verticale où sont en jeu deux (ou plus) lignes. Dans les deux cas, elle s'opère sur les cellules, donc sur la balise `<td>`. Elle se traduit par l'utilisation de l'attribut **colspan** dans le cas d'une fusion horizontale et par l'attribut **rowspan** dans le cas d'une fusion verticale.

Ces attributs prennent pour valeurs le nombre de cellules à fusionner.

Exemple d'utilisation de colspan

```

<table border="1">
  <tr>
    <td colspan="5">Cellule qui se répand sur 5 colonnes</td>
  </tr>
  <tr>
    <td>Colonne 1</td>
    <td>Colonne 2</td>
    <td>Colonne 3</td>
    <td>Colonne 4</td>
    <td>Colonne 5</td>
  </tr>
</table>

```

On obtient :

Cellule qui se répand sur 5 colonnes				
Colonne 1	Colonne 2	Colonne 3	Colonne 4	Colonne 5



La somme des **colspan de chaque ligne doit être identique à toutes les autres lignes.**

En ce qui concerne l'attribut **rowspan**, on aurait un code tel que :

Exemple d'utilisation de rowspan

```
<table border="1">
  <tr>
    <td rowspan="2">Cellule sur deux lignes</td>
    <td>Ligne 1 - Colonne 2</td>
    <td>Ligne 1 - Colonne 3</td>
    <td>Ligne 1 - Colonne 4</td>
  </tr>
  <tr>
    <td>Ligne 2 - Colonne 2</td>
    <td>Ligne 2 - Colonne 3</td>
    <td>Ligne 2 - Colonne 4</td>
  </tr>
</table>
```

Il est tout de suite beaucoup plus difficile de se rendre compte de son erreur, le **rowspan** est déjà moins intuitif que le **colspan**.

On pourrait aller plus loin en combinant les deux techniques sur un seul et même tableau :

Exemple d'utilisation de colspan et rowspan

```
<table border="1">
  <tr>
    <td colspan="3">Cellule 1</td>
    <td>Cellule 2</td>
  </tr>
  <tr>
    <td rowspan="3">Cellule 3</td>
    <td>Cellule 4</td>
    <td>Cellule 5</td>
    <td>Cellule 6</td>
  </tr>
  <tr>
    <td>Cellule 7</td>
    <td colspan="2" rowspan="2">Cellule 8</td>
  </tr>
  <tr>
    <td>Cellule 9</td>
  </tr>
</table>
```

On remarque que le code est devenu extrêmement compliqué et que la moindre erreur de fusion de notre part entraînerait une interprétation du tableau complètement bancal par les navigateurs.

II-H-3 - L'utilité des tableaux aujourd'hui

Grâce aux techniques de fusion vues précédemment, la tendance il y a quelques années, était de profiter de ce système pour mettre en page assez rapidement l'ensemble de son site web. Cependant, si cette méthode peut séduire, elle est fortement déconseillée.

En effet, les tableaux sont censés (et doivent) être utilisés uniquement pour l'affichage de données tabulaires.

Dans le cas d'une mise en page, préférez la solution du couple "DIV + CSS". Voir l'article : **[Une mise en page rapide et facile avec des div et du CSS](#)**.

On peut facilement voir dans notre dernier exemple où l'on combine les deux techniques de fusion, que notre code sera difficilement maintenable dans le cas d'une retouche ou de l'ajout d'une nouvelle cellule.

Voici quelques raisons pour lesquelles il ne faut pas utiliser les tableaux pour la mise en page :

- Ils alourdissent les pages HTML : beaucoup de balises pour finalement y mettre peu de contenu.
- Ils sont très difficiles à maintenir dans le cas d'une retouche.
- Ils nuisent à l'accessibilité : problème d'interprétation des navigateurs spécialisés destinés aux malvoyants par exemple.
- Ils sont mal gérés par les navigateurs à l'impression dans le cas de tableaux complexes.

J'espère avoir réussi à vous convaincre de ne pas/plus utiliser les tableaux à mauvais escient.

Commentez l'article :

II-I - Les formulaires

II-I-1 - À quoi servent-ils ?

Les formulaires servent à envoyer des données au serveur, données remplies en général par un visiteur, que ce soit pour l'inscription sur un site ou un formulaire de contact par exemple.

Les formulaires sont dotés de divers contrôles comme des champs de saisie, des boutons, des listes, etc, qui permettent au visiteur d'interagir avec la page qu'il est en train de consulter.

II-I-2 - La balise form

Tous les champs de formulaires quels qu'ils soient doivent se trouver dans cette balise `<form>`.

C'est cette balise qui va permettre de renseigner la page de destination du formulaire, à l'aide de l'attribut `action`, qui peut prendre pour valeurs une URL en absolu ou en relatif, voire même une adresse e-mail.

Elle est aussi utilisée pour préciser le mode d'envoi des données. C'est avec l'attribut `method` que l'on va le préciser. Cet attribut prend deux valeurs différentes qui sont `"get"` et `"post"`. Les valeurs envoyées en `"GET"` passent par l'URL alors qu'en `"POST"` elles sont envoyées de manière transparente, aussi, on préférera l'utilisation de la méthode d'envoi `"POST"`.

L'attribut `enctype`, quant à lui, sert à indiquer le type de données envoyées au serveur dans le cas d'utilisation de la méthode `"POST"`. La valeur généralement utilisée pour cet attribut est `"multipart/form-data"` qui sert lors de l'envoi de fichiers chargés avec un **champ de type "file"**.

Voyons maintenant à quoi pourrait ressembler notre balise `<form>` :

Exemple de balise form

```
<form action="envoi-formulaire.php" method="post">
  <!-- Contenu du formulaire -->
</form>
```

*Si vous pouvez bien entendu mettre plusieurs formulaires dans un même document, ceux-ci doivent **en aucun cas** être imbriqués.*



De plus, il est déconseillé d'utiliser des espaces dans l'attribut `name` de tous les champs utilisés dans un formulaire, puisque celui-ci sera retranscrit en caractère de soulignement `"_"` lors de la soumission du formulaire et peut donc induire en erreur lors de la récupération des données dans vos scripts.

II-I-3 - Les champs texte monolignes

On distingue deux types de champs texte monolignes : les champs de type `"texte"` et les champs de type `"mot de passe"`.

Le champ de type `"texte"` est l'élément de base d'un formulaire, il permet d'entrer du texte court sur une ligne, comme un login, un e-mail, ou autres alors que le champ de type `"mot de passe"` permet de renseigner un mot de passe, dont l'affichage sera camouflé aux yeux des autres personnes.

Le champ de texte monoligne est caractérisé par la balise `<input>` et l'attribut `type` dont la valeur sera `"text"` ou `"password"`.

On renseignera aussi l'attribut `name` qui sera le nom du champ, celui qui va nous permettre de retrouver la valeur lors de l'envoi du formulaire.

D'autres attributs sont disponibles, comme `value` qui permet d'attribuer une valeur par défaut au champ, `size` qui permet de préciser la taille du champ en nombre de caractères, `maxlength` qui sert à limiter le nombre de caractères possibles, `readonly` qui prend pour seule valeur `"readonly"` et verrouille le champ, `disabled` qui prend pour seule valeur `"disabled"` et permet de désactiver le champ.



Lorsqu'un champ est défini en "disabled", sa valeur **n'est pas** envoyée au serveur.

Voici quelques exemples de champs texte :

Exemples de champs texte

```
<input type="text" size="5" name="code_postal" maxlength="5">

<input type="text" name="nom_champ" readonly="readonly" value="Champ verrouillé">

<input type="text" name="login" value="Entrez ici votre login" size="20">

<input type="password" name="mdp">
```


II-I-4 - Les champs texte multilignes

Si vous souhaitez que vos visiteurs puissent saisir de longs textes, comme des champs de commentaires pour des articles d'un blog par exemple, il faudra se tourner vers l'utilisation de la balise `<textarea>`.

Contrairement au champ de texte monoligne, le `textarea` est une balise double et son contenu est écrit entre les balises ouvrante et fermante et non dans un attribut `value`.

Deux attributs sont nécessaires au `<textarea>` : `rows` qui contient le nombre de lignes et `cols` qui contient le nombre de colonnes.

On peut, tout comme le champ de texte monoligne, rajouter les attributs `name`, `readonly` et `disabled`.

L'attribut `maxlength` n'existe pas pour cette balise, et pour en limiter le nombre de caractères il est possible de passer par du  **JavaScript**.

Voici quelques exemples de `<textarea>` :

Exemples de champs texte multilignes

```
<textarea name="commentaire" rows="5" cols="80"></textarea>

<textarea name="sujet" rows="6" cols="100">Texte par défaut</textarea>
```

II-I-5 - Les champs cachés

Il nous arrivera souvent de vouloir envoyer des données au serveur de manière transparente pour le visiteur. On utilisera alors un champ caché, qui ne sera pas affiché dans le navigateur du visiteur mais dont les données seront envoyées. Attention, cette balise apparaît tout de même dans le code source de la page.

Pour cela, on utilise la balise `<input>` avec `"hidden"` comme valeur de l'attribut `type`.

Il est aussi nécessaire de renseigner l'attribut `name` pour retrouver ces données cachées du côté du serveur ainsi que l'attribut `value` qui va contenir ces données.

Exemple de champ caché :

Exemple de champ caché

```
<input type="hidden" name="clef" value="a182f7d8e844d956a65b18e84f">
```

II-I-6 - Les champs de fichiers

Pour permettre aux visiteurs d'envoyer des fichiers par l'intermédiaire du formulaire, que ce soit en pièce jointe d'un e-mail ou le chargement d'une image sur le serveur, on va utiliser la balise `<input>` dont l'attribut `type` est renseigné à `"file"`.

Ceci fait, un champ avec un bouton "Parcourir" sera disponible.

Mais pour que le champ soit opérationnel, il faut impérativement renseigner l'attribut `enctype` de la balise `<form>` à `"multipart/form-data"`.

Il est aussi plus qu'utile de préciser l'attribut `name` du champ "file". On peut aussi renseigner l'attribut `size` qui affecte la taille du champ dans lequel sera écrit le nom du fichier.

Exemple de champ "file" :

Exemple de champ file

```
<input type="file" name="image">
```

Il est également possible de limiter le poids du fichier en rajoutant un champ caché dans le formulaire contenant le champ "file". Ce champ caché doit avoir pour **name** la valeur "**MAX_FILE_SIZE**" et pour **value** le poids maximum en octet.

Limitation du poids du champ file

```
<input type="hidden" name="MAX_FILE_SIZE" value="20000">
```



Cette limitation ne vous affranchit pas de faire la vérification côté serveur ! En effet, le code HTML peut facilement être modifié par les visiteurs, donc cette valeur aussi.

II-I-7 - Les boutons radio

Les boutons radio sont utilisés lorsque vous voulez laisser aux visiteurs un choix et un seul parmi une liste de propositions. La balise est alors `<input>` dont l'attribut **type** est renseigné à "**radio**".

Ces boutons radio vont par groupe, c'est-à-dire qu'ils doivent avoir le même nom pour un groupe de propositions. Le nom est renseigné avec l'attribut **name**.

La valeur de l'attribut **value** va être transmise au serveur en fonction du bouton choisi. Vous pouvez aussi forcer un bouton radio d'un groupe à être coché au chargement de la page en lui mettant l'attribut **checked** dont la seule valeur est "**checked**".

Voyons maintenant des exemples de groupes de boutons radio :

Exemples de groupes de boutons radio

```
<p>
  <input type="radio" name="civilité" value="mlle"> Mademoiselle
  <input type="radio" name="civilité" value="mme"> Madame
  <input type="radio" name="civilité" value="mr"> Monsieur
</p>

<p>
  <input type="radio" name="genre" value="homme" checked="checked"> Homme
  <input type="radio" name="genre" value="femme"> Femme
  <input type="radio" name="genre" value="et"> Alien
  <input type="radio" name="genre" value="indéfini"> Indéfini
</p>
```



*Si vous donnez un nom différent aux boutons d'un groupe, le navigateur **ne saura pas qu'ils appartiennent au même groupe** (y a pas de miracle), donc plusieurs choix seront possibles et on perd toute l'utilité des boutons radio.*

II-I-8 - Les cases à cocher

Les cases à cocher sont sensiblement identiques aux boutons radio, mais permettent un choix multiple pour un groupe de propositions. Les attributs sont identiques, à la différence que l'attribut **type** prend pour valeur "**checkbox**". Un problème cependant, lorsque vous désignez un groupe de cases à cocher par le même nom, seule la dernière sera envoyée au serveur. Pour contrer cela, il suffit de rajouter des crochets ouvrant et fermant au nom du groupe. Il indiquera au serveur qu'il s'agit d'un tableau de valeurs à récupérer et non une seule valeur (comme dans le cas des boutons radio). En revanche, dans le cas d'utilisation d'une case à cocher unique (sans groupe), comme par exemple des CGU à valider, les crochets sont inutiles.

Voici un exemple de groupe de cases à cocher :

Exemple de cases à cocher

```
<input type="checkbox" name="qualite[]" value="intelligent"> Intelligent
<input type="checkbox" name="qualite[]" value="beau"> Beau
<input type="checkbox" name="qualite[]" value="serviable"> Serviable
<input type="checkbox" name="qualite[]" value="fort"> Fort
```

Exemple de cases à cocher

```
<input type="checkbox" name="qualite[]" value="généreux"> Généreux
```

II-I-9 - Les boutons

Il existe deux balises différentes pour présenter les boutons : la balise `<input>` et la balise `<button>`. La différence entre les deux est que la balise `<button>` est double, donc qu'elle a une balise ouvrante et une balise fermante alors que la balise `<input>` est simple. Il est de plus possible des balises de mise en forme dans la balise `<button>`. Il y a 4 types différents de boutons :

- Le bouton simple
- Le bouton d'envoi
- Le bouton image
- Le bouton effacer

II-I-9-a - Le bouton simple

Il est défini par l'attribut **type** des balises `<input>` et `<button>` ayant pour valeur **"button"**. Ce bouton n'est utile que lors d'appel de scripts JavaScript au moment du clic. Il n'a aucune autre fonction particulière. Exemples de boutons simples :

Exemples de boutons simples

```
<input type="button" name="addition" value="Additionner" onclick="addition();">

<button type="button" name="multiplication" onclick="multiplication();"><b>Multiplier</b></button>
```

II-I-9-b - Le bouton d'envoi

Le bouton d'envoi sert, comme son nom l'indique, à envoyer les données du formulaire au serveur. C'est celui qui va permettre de valider le formulaire. Il est caractérisé par la valeur **"submit"** de l'attribut **type** des balises `<input>` et `<button>`. Exemples de boutons d'envoi :

Exemples de boutons d'envoi

```
<input type="submit" name="envoyer" value="Valider">

<button type="submit" name="soumettre">Soumettre<br>les réponses</button>
```

II-I-9-c - Le bouton image

Le bouton image se comporte exactement comme un bouton d'envoi, c'est-à-dire qu'il sert à la validation du formulaire. Il sert à personnaliser son bouton, dans le cas où l'on ne trouve pas les boutons de base à notre goût. Il est défini par l'attribut **type** de la balise `<input>` ayant pour valeur **"image"** et l'on renseigne l'adresse de l'image avec l'attribut **src**. C'est le seul bouton qui ne peut pas être géré par la balise `<button>`. Exemple de bouton image :

Exemple de bouton image

```
<input type="image" name="envoyer" src="adresse/de/l/image">
```

II-I-9-d - Le bouton effacer

Ce bouton sert à réinitialiser toutes les valeurs du formulaire. Il est caractérisé par la valeur **"reset"** de l'attribut **type** des balises `<input>` et `<button>`. Exemples de boutons effacer :

Exemples de boutons effacer

```



<button type="reset" name="effacer">Réinitialiser</button>

```

II-I-10 - Les listes

Les listes permettent de laisser un choix aux visiteurs, au même titre que les boutons radio ou les cases à cocher, mais sont plus pratiques lorsqu'il s'agit de proposer un grand nombre de choix.

On distingue deux types de listes, les listes normales et les listes déroulantes.

Les deux sont introduites par la balise `<select>` qui comprend les différents choix sous forme de balise `<option>` dont on renseigne la valeur qui sera envoyée au serveur grâce à son attribut `value`.

On peut même suggérer une proposition par défaut en précisant l'attribut `selected` de la balise `<option>` de notre choix. Par défaut, une liste est déroulante.

On peut (doit) renseigner le nom de la liste avec l'attribut `name`.

Continuons avec un exemple :

Exemple de liste

```

<select name="couleur_yeux">
  <option value="bleus">Bleus</option>
  <option value="bruns" selected="selected">Bruns</option>
  <option value="verts">Verts</option>
  <option value="noisettes">Noisettes</option>
  <option value="gris">Gris</option>
  <option value="vairons">Vairons</option>
</select>

```

Cette liste est donc une liste déroulante avec un seul choix possible parmi les six proposés.

On peut rendre cette liste non déroulante en rajoutant l'attribut `size` qui prend pour valeur le nom de choix à afficher, par exemple `"3"`.

Ce qui donne :

Exemple de liste

```

<select name="couleur_yeux" size="3">
  <option value="bleus">Bleus</option>
  <option value="bruns">Bruns</option>
  <option value="verts" selected="selected">Verts</option>
  <option value="noisettes">Noisettes</option>
  <option value="gris">Gris</option>
  <option value="vairons">Vairons</option>
</select>

```

Ici encore, un seul choix est possible. Pour pouvoir proposer plusieurs choix, comme dans le cas de cases à cocher, il suffit de rajouter l'attribut `multiple` qui prend pour seule valeur `"multiple"`.


Voyons pour l'exemple :

Exemple de liste à choix multiple

```

<select name="matieres_preferrees" size="5" multiple="multiple">
  <option value="maths">Maths</option>
  <option value="français">Français</option>
  <option value="anglais">Anglais</option>
  <option value="histoire">Histoire</option>
  <option value="sport">Sport</option>
  <option value="dessin">Dessin</option>
  <option value="musique">Musique</option>
  <option value="physique">Physique</option>
  <option value="sciences naturelles">Sciences naturelles</option>
</select>

```


Le choix multiple ne peut pas se faire sur une liste déroulante. L'attribut `size` doit être renseigné pour que le multiple fonctionne.

II-I-11 - Améliorez l'accessibilité de vos formulaires

Pour améliorer l'accessibilité de vos formulaires aux handicapés, malvoyants, quelques balises ont été créées et il est toujours bon de les utiliser, d'autant plus qu'elles ne perturbent pas l'utilisation pour les autres visiteurs et clarifient même la compréhension des formulaires.

Il s'agit d'une part de la balise `<fieldset>` qui permet de grouper logiquement des champs de formulaires de même nature. Par exemple, pour un formulaire d'inscription vous pouvez grouper tous les champs en rapport avec les informations personnelles du visiteur, puis grouper tous les champs concernant ses loisirs, etc.

La balise qui va permettre de donner ce nom au groupage de champs est `<legend>`.

Regardons quelques exemples :

Exemples de fieldset

```
<fieldset>
  <legend>Informations personnelles</legend>

  <!-- Ici divers champs sur les informations personnelles, comme la saisie du nom, prénom et adresse --
>
</fieldset>

<fieldset>
  <legend>Loisirs</legend>

  <!-- Ici divers champs sur les loisirs du visiteur, comme la musique et les films qu'il aime etc --
>
</fieldset>
```

Il y a d'autre part, la balise `<label>` qui permet de donner un libellé à un champ de formulaire en les associant à l'aide de l'attribut `for` de cette balise qui doit être identique à l'attribut `id` du champ auquel elle est rattachée. Le texte mis dans le label est alors cliquable.

Par exemple :


Exemples de fieldset

```
<label for="nom">Votre nom</label> <input type="text" name="nom" id="nom">

Aimez-vous la viande ?
<label for="oui">Oui</label> <input type="radio" name="viande" id="oui">
<label for="non">Non</label> <input type="radio" name="viande" id="non">
```

Commentez l'article :

III - L'entête du document - head

La balise `<head>` contient de nombreux renseignements sur la page, sur l'auteur, ainsi que la plupart des scripts ( **JavaScript**, VBscript,...) et styles CSS.

III-A - Le titre

Le titre de votre page, qui se trouve entre les balises `<title>` et `</title>`, est sans doute l'une des balises les plus importantes du document.

En effet, c'est son contenu qui sera affiché en titre des résultats de la recherche dans les moteurs de recherche, donc il faut essayer au maximum d'y mettre les mots-clefs importants.

Il est très conseillé, voire fortement recommandé de mettre un titre unique pour chaque page de votre site, toujours en rapport avec le contenu de la page en question.


Ce titre est également ce qui apparaît dans la barre de titre du navigateur ainsi que dans les signets lorsque vous mettez le site en favoris.

Exemple de titre

```
<title>Tutoriel HTML - Apprenez les bases du HTML</title>
```

Commentez l'article :


III-B - Les scripts

Les scripts sont de petits programmes en  **JavaScript** ou en VBScript exécutés côté client qui augmentent l'ergonomie de vos pages lorsqu'ils sont utilisés à bon escient.

On distingue deux moyens d'inclure des scripts :

- Les scripts externes.
- Les scripts internes.


III-B-1 - Les scripts externes

Les scripts externes sont des fichiers externes à la page, avec l'extension .js par exemple pour les scripts  **JavaScript**.

On les inclut grâce à l'attribut **src** de la balise `<script>` :

Balise script

```
<script type="text/javascript" src="chemin/du/fichier/javascript.js"></script>
```


 *Dans le cas d'inclusion d'un script externe, tout le code inséré entre les balises `<script>` sera ignoré !*

III-B-2 - Les scripts internes

Les scripts internes sont des morceaux de code directement écrits entre les balises `<script>` et `</script>`. Ils sont donc insérés directement dans la page comme suit :


Balise script

```
<script type="text/javascript">
    function maFonction()
    {
        /* du code ici */
    }
</script>
```

 *Lors de la création d'une fonction, celle-ci est unique à la page, donc pour pouvoir la réutiliser il faut la réécrire dans une autre page : duplication inutile de code.*

Commentez l'article :

III-C - Les styles

Les feuilles de style en cascade ou  **CSS** (Cascading Style Sheets) s'utilisent pour mettre en page notre code HTML.

Aller plus loin avec le CSS

III-C-1 - Les styles externes

Il y a deux manières d'importer un fichier CSS externe, soit avec une balise HTML, soit avec du code CSS.

Import avec une balise HTML

```
<link rel="stylesheet" type="text/css" href="chemin/du/fichier.css">
```

Import avec du code CSS

```
<style type="text/css">
  @import url("chemin/du/fichier.css");
</style>
```

Quelle méthode doit alors être préférée ?


Au niveau "performance", aucune. Mais on va préférer la seconde méthode pour la simple et bonne raison que cette syntaxe n'est pas reconnue par les anciens navigateurs (comme Netscape 4.x) et donc ceux-ci interprétant très mal les propriétés CSS ne risqueront pas de rendre le site illisible pour le visiteur.

III-C-2 - Les styles internes

Les styles internes à la page sont à écrire entre les balises `<style>` et `</style>` :

CSS interne

```
<style type="text/css">
  body
  {
    background-color : #fff;
  }
</style>
```

 *Les styles internes sont uniques à la page, donc pour pouvoir les réutiliser il faut les réécrire dans une autre page : duplication inutile de code.*

Commentez l'article :

III-D - Les meta

III-D-1 - À quoi servent-elles ?

Les balises meta, obligatoirement placées dans la balise `<head>`, permettent de renseigner diverses informations sur la page en cours, comme son auteur, une courte description du document, des mots-clefs, la date de création, son jeu de caractère (ou encodage), etc.

On peut différencier deux grandes familles de balises meta :

- Les meta **name** : Touchent toutes les informations concernant le document, l'auteur, le site, les outils utilisés, etc.
- Les meta **http-equiv** : Concernent les meta qui communiquent avec le navigateur.

S'il existe de nombreuses balises meta différentes, la plupart sont inutiles, et de ce fait totalement facultatives.

Pour garder la validité W3C de votre page, seule la balise concernant son encodage est nécessaire.

Il y a quelques années, certaines balises étaient indispensables pour le référencement de son site Web, comme la description et les mots-clefs, cependant, **les techniques de référencement** étant en perpétuelle évolution, aujourd'hui leur intérêt est minime.

III-D-2 - La balise meta content-type

Cette meta est la plus importante et la seule obligatoire pour passer la validation W3C.

- meta **content-type** □ Précise le **type MIME** de la page ainsi que le jeu de caractères utilisé (encodage).

Exemple

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

III-D-3 - Ajoutez des infos sur vous

- **meta author** □ Renseigne le prénom et nom de l'auteur de la page courante. S'il y a plusieurs auteurs, les séparer d'une virgule.

Exemple

```
<meta name="author" content="Josselin WILLETTE">
```

- **meta publisher** □ Renseigne le prénom et nom de l'éditeur du site. S'il y a plusieurs éditeurs, les séparer d'une virgule. Dans le cas d'une société, il faut renseigner son nom.

Exemple

```
<meta name="publisher" content="Josselin WILLETTE">
```

- **meta reply-to** □ Définit l'adresse e-mail de l'auteur de la page. Attention, cette balise est scannée par les robots spammeurs !

Exemple

```
<meta name="reply-to" content="jwillette at monsite dot com">
```

- **meta contact** □ Définit l'adresse e-mail de la personne ou service à contacter. Attention, cette balise est scannée par les robots spammeurs !

Exemple

```
<meta name="contact" content="contact at masociete dot com">
```

- **meta contactname** □ Spécifie le nom de la personne à contacter.

Exemple

```
<meta name="contactname" content="Josselin WILLETTE">
```

- **meta contactorganization** □ Indique le nom de la société détentrice du site.

Exemple

```
<meta name="contactorganization" content="MaSociété SARL">
```

- **meta contactstreetaddress1** □ Indique la première ligne de l'adresse de l'auteur ou de la société détentrice du site.

Exemple

```
<meta name="contactstreetaddress1" content="15 rue de la poupée qui tousse">
```

- **meta contactstreetaddress2** □ Indique la deuxième ligne de l'adresse de l'auteur ou de la société détentrice du site.

Exemple

```
<meta name="contactstreetaddress2" content="Bâtiment A, Etage 3">
```

- **meta contactzip** □ Indique le code postal de l'auteur ou de la société détentrice du site.

Exemple

```
<meta name="contactzip" content="75000">
```

- **meta contactcity** □ Indique la ville de l'auteur ou de la société détentrice du site.

Exemple

```
<meta name="contactcity" content="MAVILLE">
```

- **meta contactstate** □ Indique le pays de l'auteur ou de la société détentrice du site.

Exemple

```
<meta name="contactstate" content="FRANCE">
```

III-D-4 - Ajoutez des infos sur votre site

- **meta description** □ Courte description du site qui sera affichée dans les résultats de la plupart des moteurs de recherche. Ne pas dépasser les 200 caractères sous peine de voir la description tronquée.

Exemple

```
<meta name="description" content="Tutoriel HTML sur les éléments fondamentaux à savoir pour bien débiter la programmation HTML">
```

- **meta keywords** □ Contient une liste de mots-clefs sur lesquels les moteurs de recherche vont se référer. Les mots-clefs sont séparés par une virgule. Avoir trop de mots-clefs peut être considéré par les moteurs comme du spam, donc ne pas dépasser les 500 caractères.

Exemple

```
<meta name="keywords" content="tutoriel html, cours html, balise meta, formulaires, listes, tableaux, cadres">
```

- **meta identifier-url** □ Renseigne l'URL d'entrée du site.

Exemple

```
<meta name="identifier-url" content="http://j-willette.developpez.com/">
```

- **meta subject** □ Expose le sujet de votre site en quelques mots.

Exemple

```
<meta name="subject" content="Site d'apprentissage du HTML">
```

- **meta distribution** □ Définit le type de réseau de publication du site.

Valeurs	Description
global	Visible sur l'Internet.
local	Visible uniquement sur un poste local.
iu	Visible sur un intranet.

Exemple

```
<meta name="distribution" content="global">
```

- **meta rating** □ Définit le type de public visé par le site.

Valeurs	Description
general	Tout public.
mature	Réservé aux adultes.
restricted	Accès restreint.
14years	Accès interdit en-dessous de 14 ans.

Exemple

```
<meta name="rating" content="general">
```

- **meta language** □ Renseigne la langue utilisée sur le site.

Exemple

```
<meta name="language" content="fr">
```

III-D-5 - Ajoutez des infos sur la création

- **meta copyright** □ Indique le ou les copyrights auxquels la page est soumise.

Exemple

```
<meta name="copyright" content="© 2007 Josselin WILLETTE">
```

- **meta generator** □ Déclare quels sont les logiciels utilisés pour faire le site. Les différents logiciels sont séparés d'une virgule.

Exemple

```
<meta name="generator" content="Notepad++, Gimp">
```

- **meta date-creation-ddmmyyyy** □ Indique la date de création de la page sous la forme JJMMAAAA.

Exemple

```
<meta name="date-creation-ddmmyyyy" content="01122007">
```

- **meta date-revision-ddmmyyyy** □ Indique la date de dernière modification de la page sous la forme JJMMAAAA.

Exemple

```
<meta name="date-revision-ddmmyyyy" content="01122007">
```

- **meta content-script-type** □ Précise le langage de scripts utilisé dans la page par son **type MIME**.

Exemple

```
<meta http-equiv="content-script-type" content="text/javascript">
```

- **meta content-style-type** □ Indique l'utilisation d'une feuille de style dans la page et renseigne son **type MIME**.

Exemple

```
<meta http-equiv="content-style-type" content="text/css">
```

III-D-6 - Orientez les robots

- **meta robots** □ Indique aux robots indexeurs si la page doit être indexée ou non. Plusieurs valeurs sont possibles, séparées par une virgule. Par défaut, la valeur est *all*.

Valeurs	Description
index	Autorise les robots à indexer la page.
noindex	Interdit aux robots d'indexer la page.
follow	Autorise les robots à suivre les liens de la page.
nofollow	Interdit aux robots de suivre les liens de la page.
all	Correspond à "index, follow".
none	Correspond à "noindex, nofollow".
noarchive	Interdit aux moteurs de recherche de proposer une version de cache de la page.

Exemple

```
<meta name="robots" content="index, nofollow, noarchive">
```

- **meta revisit-after** □ Précise aux robots le nombre de jours d'intervalle à respecter avant leur prochaine visite sur le site. Attention, beaucoup de robots à l'heure actuelle ne se réfèrent plus à cette balise.

Exemple

```
<meta name="revisit-after" content="3 days">
```

III-D-7 - Orientez les visiteurs

- **meta refresh** □ Permet de rafraichir la page toutes les x secondes où x est la valeur de *content*. Si une URL est renseignée, alors la navigateur redirigera le visiteur au bout de x secondes vers la page indiquée.

Exemple

```
<meta http-equiv="refresh" content="60">
<meta http-equiv="refresh" content="0; url=http://j-willette.developpez.com/">
```

III-D-8 - Plus encore...

- **meta window-target** □ Indique le cadre de destination pour l'ensemble des liens du site dans le cas d'utilisation de frames. Permet d'éviter de renseigner l'attribut target sur chacun des liens.

Exemple

```
<meta http-equiv="window-target" content="corps">
```

- **meta pragma** □ Interdit la mise en cache des pages avec son unique valeur : *no-cache*. Balise très peu fiable, à coupler avec une solution côté serveur.

Exemple

```
<meta http-equiv="pragma" content="no-cache">
```

- **meta expires** □ Renseigne la date d'expiration de la page, au format "JJ Mois (en anglais) AAAA". Au-delà de cette date, la page précédente est rechargée par le navigateur. La page peut ne jamais expirer en indiquant la valeur *never*.

Exemple

```
<meta http-equiv="expires" content="never">
<meta http-equiv="expires" content="21 December 2007">
```

- **meta set-cookie** □ Crée un cookie sur l'ordinateur du visiteur.

Exemple

```
<meta http-equiv="set-cookie" content="dejavenu=oui; path=/; expires=Thursday, 14-
April-08 00:00:00 GMT">
```

- meta **imagemap** ☐ Supprime la barre d'outils qui s'affiche sur les images lorsque la souris pointe dessus sur Internet Explorer. Sa seule valeur est *no*.

Exemple

```
<meta http-equiv="imagemap" content="no">
```

- meta **site-enter** ☐ Crée une transition à l'entrée du site. Ne fonctionne qu'avec Internet Explorer.
- meta **site-exit** ☐ Crée une transition à la sortie du site. Ne fonctionne qu'avec Internet Explorer.
- meta **page-enter** ☐ Crée une transition à l'entrée de la page. Ne fonctionne qu'avec Internet Explorer.
- meta **page-exit** ☐ Crée une transition à la sortie de la page. Ne fonctionne qu'avec Internet Explorer.

Valeurs	Description
blendTrans(duration=4)	Crée un fade in en entrée et un fade out en sortie.
revealTrans(duration=4,transition=0)	Crée une transition carrée entrante.
revealTrans(duration=4,transition=1)	Crée une transition carrée sortante.
revealTrans(duration=4,transition=2)	Crée une transition ronde entrante.
revealTrans(duration=4,transition=3)	Crée une transition ronde sortante.
revealTrans(duration=4,transition=4)	Crée une transition rideau vers le haut.
revealTrans(duration=4,transition=5)	Crée une transition rideau vers le bas.
revealTrans(duration=4,transition=6)	Crée une transition rideau vers la droite.
revealTrans(duration=4,transition=7)	Crée une transition rideau vers la gauche.
revealTrans(duration=4,transition=8)	Crée une transition rideau multiple vers la droite.
revealTrans(duration=4,transition=9)	Crée une transition rideau multiple vers le bas.
revealTrans(duration=4,transition=10)	Crée une transition damier vers la droite.
revealTrans(duration=4,transition=11)	Crée une transition damier vers le bas.
revealTrans(duration=4,transition=12)	Crée une transition brush.
revealTrans(duration=4,transition=13)	Crée une transition de porte qui se ferme verticalement.
revealTrans(duration=4,transition=14)	Crée une transition de porte qui s'ouvre verticalement.
revealTrans(duration=4,transition=15)	Crée une transition de porte qui se ferme horizontalement.
revealTrans(duration=4,transition=16)	Crée une transition de porte qui s'ouvre horizontalement.
revealTrans(duration=4,transition=17)	Crée une transition diagonale NE-SO.
revealTrans(duration=4,transition=18)	Crée une transition diagonale SE-NO.
revealTrans(duration=4,transition=19)	Crée une transition diagonale NO-SE.
revealTrans(duration=4,transition=20)	Crée une transition diagonale SO-NE.
revealTrans(duration=4,transition=21)	Crée une transition par lignes horizontales.
revealTrans(duration=4,transition=22)	Crée une transition par lignes verticales.
revealTrans(duration=4,transition=23)	Crée une transition aléatoire.

Exemple

```
<meta http-equiv="page-enter" content="blendTrans(duration=2)">
```



Vous pouvez modifier la valeur duration comme bon vous semble, elle renseigne le temps de transition.

Et encore plus...

Vous avez peut-être déjà remarqué dans le code source de certains sites des balises meta qui commencent par "DC.", comme par exemple :

Exemple

```
<meta name="dc.keywords" content="liste, de, mots, clefs">
```


Ces balises font partie de ce que l'on appelle le **Dublin Core**.

Pour plus d'informations, vous pouvez lire la traduction officielle du **Guide d'utilisation du Dublin Core** ainsi que la **liste de ces meta** (en).

L'idée de créer une cohérence dans la création de métadonnées partait d'une bonne intention, seulement elle n'a pas su s'imposer dans les mœurs à cause sans doute d'une trop grande timidité sur la propagation de cette idée.

De ce fait, elles ne sont pas très utilisées.

[Commentez l'article :](#)

IV - Aller plus loin

IV-A - Le XHTML

Vous pouvez apprendre le XHTML qui est une évolution orientée XML du HTML. Cela implique par exemple d'écrire les balises et attributs entièrement en minuscules, de toujours fermer les balises, même les balises simples (Ex : `
` devient `
`), mais bien évidemment de changer de DOCTYPE.

Pour plus d'informations concernant les différences entre HTML et XHTML et pour en avoir une introduction, reportez-vous **au tutoriel d'Adrien Pellegrini**.

[Commentez l'article :](#)

IV-B - Le CSS

CSS, ou "*Cascading StyleSheets*" est un langage qui sert à mettre en forme des documents (X)HTML ou XML. A ce titre, il est beaucoup plus puissant que les balises de mises en forme vues précédemment. Aussi, dès qu'il s'agit de présenter des données dans une page, que ce soit du texte ou des blocs entiers, je vous conseille fortement de l'utiliser en lieu et place des balises obsolètes comme ``.

Pour débuter dans ce langage, voyez plutôt **la section CSS de Développez.com**.

[Commentez l'article :](#)

IV-C - Le référencement

Avoir une page HTML, voire un site complet maintenant que vous avez suivi ce tutoriel, c'est bien beau, mais si on ne peut pas le trouver dans Google ou tout autre moteur de recherche, ça ne sert pas à grand chose. Aussi, pour aller plus loin dans le dur apprentissage de la création de sites, il faut passer par des techniques de référencement qui permettront à notre site d'être visible sur la toile.

Ces techniques sont en perpétuelle évolution, donc il faudra autant que faire se peut se tenir informé des nouvelles techniques pour rester dans les meilleures places.

Nous vous proposons des tutoriels complets et régulièrement révisés pour vous tenir à la page de ces techniques dans **notre section Webmarketing**.

[Commentez l'article :](#)

IV-D - Et encore plus...

Le monde du Web ne s'arrête pas au simple HTML, il existe des langages très puissants permettant de générer dynamiquement du HTML ou d'interagir directement avec le visiteur.

Pour connaître la suite, rendez-vous sur **notre section Développement Web**.

[Commentez l'article :](#)