# Le langage HTML



---

# Sections

# Introduction

HTML est un langage de description de document utilisé sur Internet pour faire des pages Web. Son sigle signifie *« HyperText Markup Language »* en anglais, littéralement « langage de marquage hypertexte ». Le balisage HTML est incorporé dans le texte du document et est interprété par un navigateur Web.

Le XHTML est quant à lui une évolution du HTML ; le sigle signifie *Extensible Hypertext Markup Language*. Nous verrons plus loin la différence entre les deux, mais la plupart des informations sont valables pour les deux langages. Le terme (X)HTML signifie donc « HTML ou XHTML ».

La formulation du langage HTML est fort simple, en effet des balises permettent d'appliquer différents formatages. Elles sont délimitées par les deux symboles "supérieur à" et "inférieur à". Dans le cas des balises en paires, chaque balise ouverte doit être fermée,

Finalement, ce langage a abouti à une nouvelle version : le HTML5. Cette version possède des ajouts pour les médias (audio, vidéo, application interactives avec CSS3/JavaScript). C'est de cette version que ce livre traitera.

## Pourquoi apprendre le HTML5 ?

Un document HTML5 est avant tout un document texte (c'est-à-dire lisible par un humain), qui contient une certaine syntaxe afin de mettre en forme ou de décrire ce document. Son nom de fichier a généralement le suffixe `.html`

(réduit à `.htm` sur les systèmes d'exploitation ne supportant pas plus de 3 caractères de suffixe).

L'immense majorité des internautes réalisent leur page Web à l'aide d'un logiciel avec une interface graphique, en utilisant la souris et en ayant un rendu immédiat ; il en existe des gratuits. Ce logiciel génère du code HTML5.

Pourquoi alors vouloir taper soi-même du code ?

Chacun aura sa réponse à cette question. Cela peut être par curiosité, pour comprendre comment marche le Web ; ou bien encore pour « nettoyer » le code HTML généré par le logiciel, le rendre plus léger.

### Description de document ou mise en forme ?

Il faut bien comprendre la différence entre « description de document » et « mise en forme ». « Décrire » un document, c'est indiquer la « fonction » de telle ou telle partie du texte : citation, passage important, titre, paragraphe… *La mise en forme, elle, est du ressort du navigateur* : c'est lui qui décide comment sera mis en forme un paragraphe (en général avec un espace vertical avant et après), une citation (en général avec une marge plus importante), un passage important (en général en italique ou en gras), un titre (en général en plus grand avec un espace vertical avant et après)…

Vous pouvez donner des indications au navigateur en utilisant une *feuille de style* (voir *Le langage CSS* mais il est préférable de lire *Le langage HTML* avant).

Bien sûr, pour des raisons esthétiques — tout à fait louables — certains voudront faire du « placement au millimètre ». C'est tout à fait possible, mais plus vous voulez faire des choses compliquées, plus le code devient compliqué… Et surtout, le risque est de faire « n'importe quoi ». Il est par exemple simple de créer des tableaux imbriqués pour placer le texte comme on veut, mais avez-vous pensé aux mal-voyants ? S'ils utilisent un logiciel de lecture vocale, celui-ci lit le contenu des cellules linéairement, de gauche à droite et de haut en bas ; est-ce bien ce que vous désirez ? Voir à ce sujet l'article de Wikipédia *Accessibilité du Web*.

Dans un premier temps, le recours au HTML5 revient à renoncer à la mise en forme pour la déléguer au navigateur. Ceci peut être frustrant, mais vous gagnez en simplicité et en universalité. La mise en forme viendra dans un deuxième temps, avec le CSS — Rome ne s'est pas faite en un jour…

Notons que le HTML ayant été créé avant le CSS, il contient de nombreux éléments de mise en forme. On trouvera donc de nombreuses références (ouvrages, pages en lignes) et de nombreux exemples (pages Web) utilisant ces balises. Nous vous invitons à ne pas suivre ces « mauvais exemples ».

## Balises et compagnie

Pour faire une page HTML (c'est à dire une page Web) vous n'avez *pas* besoin d'un logiciel spécial, il suffit d'utiliser un éditeur de texte standard (comme le Bloc-note de Windows ou TextEdit sur Mac) et d'enregistrer vos pages avec un nom finissant par `.html`. Une page Web est en fait un simple fichier texte contenant du code HTML qui est ensuite interprété par le navigateur.

---

#### Note

> Dans le protocole HTTP, ce n'est pas l'extension du fichier qui définit qu'un fichier est du HTML, mais l'en-tête (voir plus loin)… en théorie. Si un fichier HTML devrait pouvoir avoir n'importe quelle extension, dans la pratique, le navigateur se base souvent sur l'extension du fichier et non pas sur l'en-tête.

---

Un document HTML est constitué de texte normal — la plupart du temps c'est ce que vous voulez afficher à l'écran — et de HTML à proprement parler sous formes de **balises** (ou *tags* en anglais). Les balises servent à dire des choses au navigateur comme « ça c'est un titre », « ça c'est en emphase », « là je veux une image », « là je veux un lien » etc. Affichez ensuite la page dans votre navigateur. Une balise est facilement reconnaissable car elle est toujours entre `<` et `>`.

```
Voici un texte en <em>emphase</em>.
```

donne :

Voici un texte en *emphase*.

Dans cet exemple, seul le mot « emphase » sera en *emphase* car il est borné par les balises `<em>` et `</em>`.[1]

On voit déjà qu'il y a deux types de balises. Les balises qui vont par deux, pour dire des choses comme « là, commence le texte en emphase » et « là, termine le texte en emphase », et les balises qui sont toutes seules, comme pour le « là je veux une image ». Observez qu'une balise de fin s'écrit exactement comme la balise de début mais avec une barre de fraction « `/` » (*slash* en anglais) avant son nom (`em`). On appelle tout ce qui est situé entre une balise de début et de fin un *élément*. On a donc ici un élément « em » contenant le texte « emphase ».

Une balise de début peut également contenir un ou plusieurs *attributs*, qui sont des paramètres permettant de préciser certaines caractéristiques de l'élément. Le nom de la balise dit de faire quelque chose, et un attribut donne des précisions sur comment le faire. Par exemple, la balise pour faire un lien est « `a` » (pour *anchor*, « ancre »). Mais si on fait juste `<a>mon super lien</a>`, le navigateur ne saura pas où doit mener ce lien. Pour ça il y a l'attribut « `href` » (pour *hypertext reference*, « référence hypertexte ») :

```
<a href="http://fr.wikipedia.org/">Wikipédia</a> est une encyclopédie libre.
```

donne :

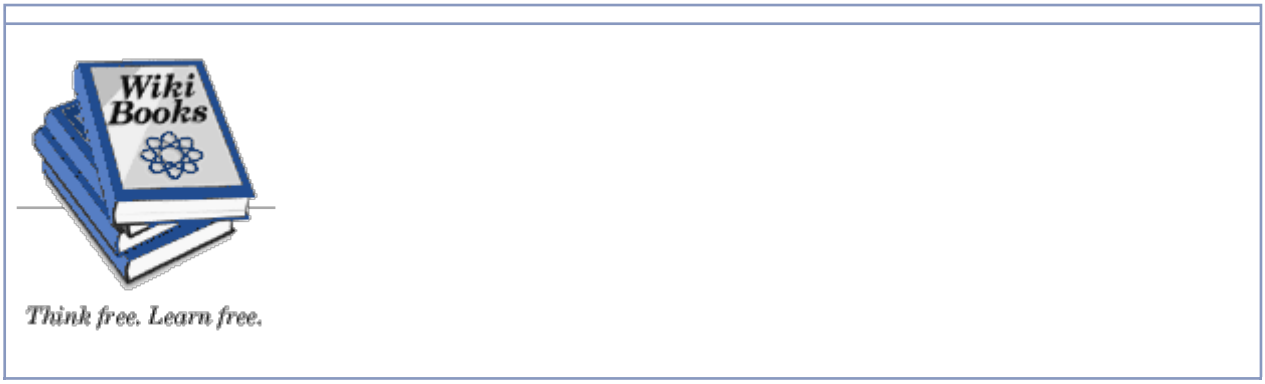
Wikipédia est une encyclopédie libre.

L'attribut « `href` » indique la destination du lien. Comme vous pouvez le voir dans l'exemple, les attributs sont placés dans la balise de début (il ne faut pas les remettre dans la balise de fin), après le nom de l'élément. Le contenu d'un attribut est toujours délimité par deux guillemets « `"` » ou deux apostrophes « `'` », précédés du signe égal « `=` ». Vous pouvez bien sûr mettre plusieurs attributs en les séparant par des espaces. Dans certains cas, le navigateur comprendra ce que vous voulez même si vous ne mettez pas les guillemets, mais il vaut mieux prendre l'habitude d'en mettre partout.

Un exemple de balise qui va toute seule maintenant :

```
<img src="../images/wiki-textbook.png" />
```

donne :




Si vous avez tout suivi, vous devriez reconnaître qu'on a la balise « `img` » (qui assez logiquement veut dire *image*) et l'attribut « `src` » (pour *source*) qui a pour valeur « ../images/wiki-textbook.png ». Et tout ça nous donne une jolie balise. Ce code affiche en fait l'image située à l'adresse « http://fr.wikibooks.org/images/wiki-textbook.png »[2], soit le logo de Wikibooks

---

**Remarque**

La barre de fraction finale « `/` » dans une balise isolée se met en XHTML, mais *pas* en HTML ; l'exemple ci-dessus en HTML donne `<img src="http://fr.wikibooks.org/images/wiki-textbook.png">`. Nous reviendrons la-dessus un peu plus bas.

---

Dernier point important sur les balises, il faut penser à *bien les imbriquer* : une balise ouvrante doit **toujours** être placée **avant** une balise fermante.

Il est interdit de « croiser les balises » c'est-à-dire qu'il n'est pas permis de fermer une balise alors qu'une autre, ouverte après elle, n'est pas encore fermée. Un exemple de balises correctement agencées (ne vous inquiétez pas si vous ne comprenez pas les balises elle seront expliquées plus tard) :

```
Un texte qui <em>parle de <strong>choses</strong> intéressantes</em>
```

donne :

Un texte qui *parle de **choses** intéressantes*

Un exemple de mauvaise imbrication des balises :

~~`Un texte qui <em>parle de <strong>choses</em> intéressantes</strong>`~~

Le code correct pour entrelacer les styles de texte serait :

```
Un texte qui <em>parle de <strong>choses</strong></em>
<strong>intéressantes</strong>
```

ce qui donne :

Un texte qui *parle de* ***choses*** **intéressantes**

- Exercice 1

# Alors, HTML/XHTML ou HTML5 ?

## Késako

Nous avons évoqué rapidement tout à l'heure le XHTML et le HTML5. Quelle est la différence entre le XHTML et le HTML5, vous demandez vous ?

Conçu initialement comme un langage simplifié par rapport au SGML, le HTML doit une part de son succès à sa tolérance syntaxique, qui en facilite à première vue l'usage : ainsi, par exemple, toutes les balises ne sont pas nécessairement fermées, l'écriture du code est indifférente à la casse, les valeurs d'attributs peuvent dans certains cas ne pas être entourées de guillemets, … Cependant, cette facilité apparente a son revers : le code HTML ne se prête pas aux traitements automatisés qui sont en revanche l'un des atouts des formats XML.

D'autre part, HTML ne peut être étendu et gagner en nouvelles fonctionnalités qu'au prix du développement et de l'intégration de nouvelles spécifications. À l'inverse, le XML est par nature un méta-format permettant de créer à volonté de nouveaux éléments.

Afin de tirer notamment parti de ces deux atouts du XML, tout en conservant la compatibilité avec le HTML, le W3C a défini un format XHTML1.0, qui reformule HTML selon des règles d'écritures plus strictes conformes au XML. Il n'y a en revanche aucune différence entre HTML4.01 et XHTML1.0 quant aux balises disponibles.

## Lequel choisir

Vous pouvez à peu près choisir la version du (X)HTML que vous voulez. Le tout est de s'y tenir. Chaque version a ses règles et ses balises. Cependant, si vous voulez être dans l'air du temps je suppose que votre préférence se portera sur le XHTML 1.0. Si vous voulez faire du HTML tout court par contre, ne cherchez pas l'originalité et faites du 4.01.

## Les variantes

Comme si ce n'était pas déjà assez compliqué, les principales versions du HTML viennent aussi avec des *variantes*. C'est le cas pour HTML 4.01 et XHTML 1.0. Là encore, une fois une variante choisie, il faudra s'y tenir. Les variantes sont au nombre de 3 : `Transitional`, `Strict` et `Frameset`. La version *strict* est celle recommandée par le W3C. La version *transitional* permet l'utilisation de balises classées comme *deprecated* (désapprouvées) qui existent parce qu'elles ont existé avant mais que le W3C aimerait bien enterrer sous trois couches de bétons et oublier complètement. La version *frameset* elle est un peu spéciale. Voyez la partie sur les cadres.

### Principales différences entre le HTML et le XHTML

On en a déjà vu une, et on ré-insistera dessus de toute façon en temps voulu, mais voilà pour la route les différences importantes entre HTML et XHTML :

- En XHTML, une barre de fraction « / » est obligatoire **à la fin** des balises isolées, comme `<br />`, `<hr />`. Il ne faut pas en mettre en HTML. Pour la compatibilité, il est conseillé de mettre un espace avant la barre de fraction.
- En XHTML, il faut toujours mettre les valeurs des attributs entre guillemets : `<input type="text" />`. En HTML on peut parfois s'en passer.
- En XHTML, un attribut doit obligatoirement avoir une valeur : `<input type="text" readonly="readonly" />`, alors qu'en HTML, ce n'est pas requis : `<input type="text" readonly >`
- En HTML, on peut aussi bien écrire `<html>` que `<HTML>` ou `<hTmL>`. En XHTML par contre tout doit être en minuscules. C'est `<html>` et pas autre chose. On dit que XHTML est sensible à la casse.
- En XHTML, si une balise est ouverte, il faut la refermer. Pas de `<em>` sans `</em>`. Ceci paraît évident, mais en HTML on peut parfois s'en passer.

En règle générale, on dit que XHTML est un dialecte XML, et on peut se référer à l'ouvrage *Structure d'un document XML* pour obtenir la liste des règles et bonnes pratiques à mettre en œuvre.

### S'en tenir à une version du (X)HTML

Comme nous l'avons déjà répété plusieurs fois, quand on choisit une version et une variante, on s'y tient. En fait soyons clair : ne pas respecter la version que vous avez choisie ne fera pas planter votre navigateur… En fait dans certains cas, ça ne se verra même pas. Pourquoi le faire alors ?

La guerre des navigateurs, dans sa première phase, allait vers la différenciation. Chaque navigateur s'autorisait à inventer des balises incompatibles avec le concurrent (stratégie de la savonnette)[3]. Cette période est révolue, et la guerre des navigateurs, dans la phase actuelle, s'appuie sur la convergence vers les standards du W3C. Ceci ne se fait pas sans casse, et certaines pages s'affichent bizarrement, ou pas du tout, si le dialecte n'est pas bien suivi.

Avoir un site valide (c'est comme ça qu'on appelle le fait de respecter la version choisie) contribue à construire le Web du futur, que l'on espère plus accessible pour tous. Le respect des normes permet d'assurer qu'un document sera bien interprété quel que soit le système d'exploitation, le navigateur (pensez notamment aux déficients visuels), les paramètres régionaux…

Comment être sûr que votre site est valide ? Et bien en le faisant passer par le validateur bien sûr ! Par exemple :

- le validateur du W3C (http://validator.w3.org/) ;
- celui du WDG (http://www.htmlhelp.com/tools/validator/index.html.fr).

# Historique

Le HTML est issu du projet SGML (*Standard Generalized Markup Language*, initié en 1979 par IBM et publié comme norme en 1986, la norme ISO 8879:1986). Le HTML date du début des années 1990. Les premières spécifications indépendante d'un éditeur de navigateur, le RFC 1866, est publié en 1995 ; c'est le HTML version 2.0. La dernière version du HTML (la version 4.01) date de fin 1999, et la première version du XHTML (la version 1.0)

date de début 2000.

La première version du CSS (le CSS1), permettant les feuilles de style, date de 1996, mais c'est seulement en 2000 qu'un navigateur a intégré totalement ses spécifications. Auparavant, les navigateurs ont développé chacun leurs balises de mise en forme, souvent indépendamment du W3C.

*Voir l'article de Wikipédia Hypertext markup language > Historique.*

### Validation d'une page web

Les pages web doivent assumer l'encodage choisi et respecter quelques règles de base, ainsi que celles imposées par la grammaire choisie et spécifiée (la DTD). Afin de vérifier les pages web que vous allez produire, il conviendra d'utiliser le validateur proposé par le W3C, disponible à cette adresse : http://validator.w3.org/

## Notes

1. En général, le texte en emphase est mis en italique, mais ceci peut varier ; on peut par exemple définir que la mise en emphase se ferait en utilisant une couleur différente.
2. le sens des deux points « `..` » sera expliqué plus loin, il suffit de comprendre qu'ils seront remplacés par « `http://fr.wikibooks.org/` »
3. Les éditeurs de logiciel (Microsoft Internet Explorer et Netscape navigator pour ne pas les nommer) créaient des balises spécifiques interprétées par eux seuls afin de concurrencer les autres et de montrer que leur navigateur avait plus de possibilité, donnant ce que l'on a appelé affectueusement la « soupe de balises »

## Liens externes

- Spécification d'HTML 4.01 (http://www.w3.org/TR/html401/)
  - Traduction française de la spécification HTML 4.01 (http://www.la-grange.net/w3c/html4.01/cover.html)
- Page wikipédia sur XHTML
- Spécification d'XHTML (http://www.w3.org/TR/xhtml1/)
  - Traduction française de la spécification XHTML 1.0 (http://www.la-grange.net/w3c/xhtml1/)
  - Traduction française de la spécification XHTML 1.1 (http://www.la-grange.net/w3c/xhtml11/)
- HTML, CDD et Dom (http://www.usenet-fr.net/fur/www/faq-HtmlCssDom.html), FAQ du forum usenet news:fr.comp.infosystemes.www.auteurs
- Martius web : cours XHTML/CSS (http://www.martiusweb.net/tutoriaux,01_00.html)
- Le site du zéro (http://www.siteduzero.com)
  - Les cours de HTML5 et CSS (http://www.siteduzero.com/tuto-3-6-0-apprenez-a-creer-votre-site-web.html)

# Structure de base d'un document HTML

Les documents HTML doivent tous avoir une structure minimale. C'est-à-dire des balises qui sont toujours présentes et au milieu desquelles vous allez ajouter votre propre contenu. Ce chapitre présente cette structure en donnant quelques explications sur les mots-clefs principaux (aussi appelés balises).

Voici un exemple de page minimale :

```
<!DOCTYPE html>

<html>

  <head>
    <title>Titre affiché dans la barre de titre du navigateur</title>
  </head>

  <body>
    <!-- C'est ici que vous mettrez votre contenu  -->
  </body>

</html>
```

## Les balises

Comme vous le savez, le HTML utilise ce que l'on nomme des balises afin de structurer les informations et pour transformer votre code source en document correct affiché à l'écran. Connaître toutes les balises par cœur n'est heureusement pas nécessaire mais il faut en avoir bien compris le principe pour pratiquer le HTML. On a de plus tendance à utiliser un nombre de balises plus restreint qu'il n'y paraît.

Pour rappel une balise est composée du nom de l'élément entouré des symboles « < » et « > ». Par exemple `<html>` est une balise.

Les balises viennent en général par deux.

- Une ouvrante dont la syntaxe est décrite juste au dessus.
- Une fermante qui s'écrit comme l'ouvrante sauf que l'on fait précéder le mot clef par le caractère « / ».

Certaines balises sont dites vides, c'est-à-dire qu'elles ne contiennent pas d'autres éléments. Ces balises ne possèdent donc pas de balises de fermeture.

Lorsqu'une balise n'est pas vide, vous pouvez mettre différentes choses entre la balise d'ouverture et la balise de fermeture comme du texte ou d'autres balises.

Il est interdit de « croiser les balises » c'est-à-dire qu'il n'est pas permis de fermer une balise alors qu'une autre, ouverte après elle, n'est pas encore fermée. Il faut toujours faire attention à bien les emboîter.

Un grand nombre de balises prennent des attributs. Ils servent à paramétrer finement la sémantique d'une balise (par exemple en indiquant la cible d'un lien). Après le mot-clef, il suffit de mettre le nom de l'attribut suivi du symbole = et

d'une valeur à donner à l'attribut, placée entre guillemets.

Ainsi dans la balise `<a href="http://fr.wikipedia.org">Lien vers wikipedia</a>` :

- le mot-clef est « a »
- elle contient un attribut nommé « href » qui a pour valeur « http://fr.wikipedia.org ».

Les attributs sont toujours de la forme `nom_de_l'attribut="valeur"`.

Chaque balise peut posséder plusieurs attributs différents. Bien sûr, comme pour les balises, il est inutile de connaître par cœur tous les attributs et toutes les valeurs.

Si ce qui précède n'est pas clair, retournez lire l'introduction.

# La page minimale

Voici une page minimale dont nous allons expliquer tous les éléments :

```
<!DOCTYPE html>

<html>

  <head>
    <title>Titre affiché dans la barre de titre du navigateur</title>
  </head>

  <body>
    <!-- C'est ici que vous mettrez votre contenu  -->
  </body>

</html>
```

## La définition de type de document

Vous le savez, on en a parlé en long en large et travers dans l'introduction, il existe plusieurs versions du HTML et avec ça plusieurs variantes. On a aussi dit qu'une fois la version (et le cas échéant la variante) choisie, il fallait s'y tenir. Mais pour savoir si vous respectez bien les règles d'une certaines version, il faut dire la version que vous utilisez ! C'est à ça que sert la première ligne. Elle paraît barbare mais ne vous inquiétez pas, vous n'aurez pas à l'apprendre par cœur (franchement je me demande si quelqu'un la connaît). Cette première balise donc est la *déclaration de type document* (appelée couramment *doctype*). Dans le cas présent, c'est du HTML5.

Vous avez ci-dessous une liste des Doctype les plus utilisés que vous pouvez directement copier / coller.

### Importance du DOCTYPE

Sans un DOCTYPE, vous ne pourrez pas faire passer votre page par un validateur.

Vous avez sûrement remarqué qu'il n'y a pas de / final. En effet, le DOCTYPE n'est pas une balise, en fait, mais a un statut bien particulier.

#### Les trois variantes du HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
```

#### Les trois variantes du XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

#### Le XHTML 1.1 (qui n'a pas de variantes)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

#### Le HTML 5

```
<!DOCTYPE html>
```

## Les Balises

Nous retrouvons ici notre exemple du haut de page, pour en expliquer les principales balises.

```
<!DOCTYPE html>
```

```
<html>

  <head>
    <title>Titre affiché dans la barre de titre du navigateur</title>
  </head>

  <body>
    <!-- C'est ici que vous mettrez votre contenu  -->
  </body>

</html>
```

### La balise <html>

Là, on entre dans le vif du sujet, les balises.

Un document HTML est entièrement compris dans une balise `html`. Même si le navigateur s'y attend, vous êtes poli et vous lui dites que vous commencez votre document HTML, puis que vous le terminez. Ainsi la balise `<html>` sera toujours la toute première après le doctype, et la balise `</html>` la toute dernière.

À l'intérieur on trouve deux parties principales : un en-tête et un corps, placé respectivement dans les balises `head` et `body`. L'en-tête est constitué de déclarations générales concernant la page HTML, destinées au navigateur, aux moteurs de recherche etc. Le corps contient le document lui-même : ce qui sera affiché par le navigateur dans la fenêtre de rendu. Cette partie ne contient aucun élément obligatoire.

### La balise <head>

La balise `<head>` délimite l'en-tête de la page dont on vient de parler. On y trouve des informations qui ne seront pas affichées directement dans la zone de rendu du navigateur. Par exemple le titre de la page, le lien vers la feuille de style, une description et des mots clés, etc... L'en-tête des documents HTML est l'objet du chapitre *L'en-tête*.

### La balise <title>

L'en-tête contient un élément obligatoire : `title` qui indique le titre de la page. C'est le titre qui s'affiche ensuite en haut de la fenêtre du navigateur.

Essayez de mettre un titre pertinent et différent pour chaque page, qui permette d'identifier le site et la page en elle-même. Par exemple, "Sommaire" est un très mauvais choix. Quand votre page se retrouvera dans les favoris de quelqu'un, cette personne sera incapable de savoir de quelle page il s'agit rien qu'en regardant le nom. Préférez des choses du style "Accueil - www.ladressedemonsite.com".

### La balise <body>

Tout le corps de notre document est dans la partie `body`. Elle comprend donc le texte, les liens, la référence des images et tout ce qu'un auteur peut vouloir mettre dans un document HTML.

### Les commentaires

Les commentaires sont du texte écrit dans le code HTML qui n'est pas visible dans le rendu de la page. Les commentaires jouent habituellement le rôle de notes pour expliquer ce qui a été fait dans la page, ou bien tout simplement pour indiquer des modifications à faire ultérieurement. Ils sont biens sûr facultatifs, mais ils peuvent vous être utiles !

Un commentaire commence par les caractères `<!--` et se termine par les caractères `-->`.

Pratiquement n'importe quelle chaîne de caractères peut être placée à l'intérieur d'un commentaire : du texte, des balises, mais pas une suite de deux traits d'union adjacents (--).

# Conclusion

Dans cette partie vous avez appris vos premières vraies balises. Elles sont importantes car elles doivent toujours être présentes (en dehors des balises de commentaires, qui sont uniquement là à titre de... commentaires).

# Bien commencer le HTML

Les navigateurs sont plutôt résistants aux erreurs. Ainsi, si vous faites une faute, la seule conséquence sera en général que vous n'obtiendrez pas le résultat attendu mais le texte s'affichera quand même. Dans le pire des cas, votre contenu ne s'affichera pas, mais vous n'aurez pas de « plantage » comme cela arrive avec la programmation.

Du fait de cette tolérance, certains créateurs, ou même logiciels générant du HTML font volontairement des fautes, par exemple ouvrir une balise et ne pas la fermer, par flemmardise. Cela peut sembler sans importance puisque la page s'affichera quand même.

Toutefois,

- des erreurs consécutives peuvent engender un effet boule de neige et il devient alors difficile de trouver d'où vient le vrai problème.
- cela empêche la traduction automatique vers d'autre langages, par exemple la conversion du HTML en LaTeX ou en syntaxe wiki.
- il n'est pas garanti que la page s'affiche correctement sur tous les navigateurs.

Il faut en fait accorder la même rigueur au HTML qu'aux autres langages de programmation, et pour cela utiliser de bons outils et prendre de bonnes habitudes.

Et prendre du recul lorsque l'on s'inspire de pages déjà existantes : la plupart des navigateurs permettent d'afficher le code source de la page, mais celui-ci peut être peu rigoureux.

## Un bon exemple vaut mieux qu'un long discours

Nous allons créer notre première page à partir de la page minimale.

Pour cela, ouvrez votre éditeur de texte préféré — nous parlons bien d'*éditeur* de texte et pas de logiciel de *traitement* de texte (comme Microsoft Word). Sous Microsoft Windows, le Bloc-Note (Notepad) fait très bien l'affaire. Prenez le texte ci-dessous, copiez-le, et collez-le dans la page vide (ou bien tapez-le).

### Exemple

```
<!DOCTYPE html>

<html>

  <head>
    <title>Premier essai</title>
  </head>

  <body>
    Bonjour le monde.
  </body>

</html>
```

Puis, utilisez la fonction **Fichier | Enregistrer sous**, et enregistrez le sous le nom `bonjour.html` (avec Notepad, il

faut choisir « **Tous les fichiers** » dans le menu déroulant **Type de fichier**). Si vous double-cliquez dessus depuis l'explorateur de disque (Explorateur Windows, Finder…), cela ouvrira normalement votre navigateur Internet par défaut, et affiche :

> « Premier essai » dans la barre de titre ;
> « Bonjour le monde. » dans la fenêtre principale.

Réessayez maintenant avec le texte suivant :

```
<!DOCTYPE html> <html> <head> <title> Premier essai </title> </head> <body> Bonjour
le monde. </body> </html>
```

(le même texte mais sur une seule ligne, sans mise en forme) — une fois le fichier sauvé, il suffit d'appuyer sur le bouton « rafraîchissement/recharger » du navigateur pour voir la différence. On remarque qu'il n'y en a aucune.

Aucune différence en ce qui concerne le rendu, mais le code source est lui bien moins lisible. Donc moins facilement modifiable, augmentable, corrigible. Il convient donc de prendre des « bonnes habitudes de programmation ».

# Bonnes habitudes de programmation

Voici quelques conseils

- aérer son code :
  - les retours à la ligne n'ont pas d'incidence sur le rendu (si ce n'est un espace) ; n'hésitez donc pas à revenir à la ligne régulièrement afin de structurer votre code ;
  - si vous mettez plusieurs espaces, cela est interprété comme un seul espace, on peut donc jouer sur la « mise en forme » du code pour se repérer, et par exemple mettre un ou plusieurs espace en début de ligne (indentation) ; en général, quand un texte est entre une balise d'ouverture et une balise de fermeture, on décale les différentes lignes de 2 ou 3 espaces par rapport à ce qui précède ;
- mettez des commentaires pour pouvoir vous repérer ;
- utilisez un éditeur de texte avec
  - gestion des indentations : il suffit d'appuyer sur la touche de tabulation pour créer un décalage, et le décalage est appliqué automatiquement aux lignes suivantes ;
  - « coloration syntaxique » : les caractères spéciaux et balises sont reconnus et mis en couleur, ce qui facilite la lecture du code.

Voici par exemple un commentaire permettant de se repérer facilement (l'exemple suivant se trouverait au sein d'un code, donc notamment après les balises `<!DOCTYPE html> <html> … <body>` et avant le `</body> </html>` final).

```
…
<!--
********************
* première partie *
********************
-->
```

```
<h1> Première partie </h1>
…
```

# Le mauvais exemple

Prenez votre traitement de texte favoris. Créez un document vide, et tapez simplement « `Bonjour le monde.` », puis enregistrez le fichier sous la forme d'un fichier HTML : menu **Fichier | Enregistrer sous**, et choisissez l'option **Page Web (*.htm, *.html)** dans le menu déroulant **Type de fichier**. Appelez ce fichier `bonjour1.html`.

Ouvrez maintenant ce fichier depuis l'éditeur de texte. Vous voyez que le fichier contient un nombre beaucoup plus important de lignes. Certaines de ces lignes peuvent contenir des informations personnelles, que vous aurez rentrées lorsque vous avez installé votre système d'exploitation, et que vous ne désirez peut-être pas voir figurer sur Internet… Et selon le logiciel (et sa version), vous aurez du code plus ou moins « propre » : dans certains cas, pour un texte long, il redéfinit à chaque paragraphe la police utilisée… Essayez de sauvegarder ainsi au format HTML un texte que vous avez déjà tapé auparavant et constatez les dégâts. Vous remarquez aussi que vous n'avez pas pu définir le titre s'affichant dans la barre de titre.

S'il est simple de générer du code HTML, simple dans le sens « en peu d'opérations et sans connaissance particulières » (« en un clic »), il faut se méfier du résultat, même si le rendu est correct.

# Avec quoi écrire un document HTML ?

Comme indiqué plus haut, il existe des éditeurs HTML plus développés, allant de l'amélioration de la présentation du code (exemple : les balises sont distinguées du texte par une couleur spécifique) à l'éditeur WYSIWYG (*« What You See Is What You Get »*, littéralement « ce que vous voyez est ce que vous obtenez », c'est-à-dire que vous voyez directement le résultat apporté par les modifications que vous entreprenez).

Vous devrez tout de même garder en tête la notion (récente) d'encodage des caractères, et faire la différence entre les principaux types (utf-8, ISO-8859-1…), et la nécessité pour votre éditeur de texte de reconnaître et respecter cet encodage, sous peine de voir afficher de drôles de caractères à la place des accents… Voir à ce sujet *Caractères spéciaux et entités*.

Voici une sélection (à compléter) d'éditeurs libres de qualité :

- Notepad++ Coloration syntaxique paramétrable, ouverture simultanée de plusieurs sources, support d'une quarantaine de langages, reconnaissance de l'encodage, macro, plugiciels...
- Emacs Coloration syntaxique, affichage de double fenêtre, barre de navigation, un catalogue complet d'extension, de nombreux raccourci clavier
- Bluefish
- JEdit
- Quanta Plus
- Atom Coloration syntaxique
- Brackets
- Sublime Text 2
- NVU et Kompozer Ces derniers seraient plutôt des éditeurs WYSIWYG mais permettent d'éditer directement la source d'une page.

Pour choisir, le mieux est de tester. Quelques éléments à prendre en compte pour faire un choix :

- Le logiciel permet-il la coloration syntaxique ? Quels langages sont supportés (PHP, CSS, HTML, JavaScript ?) ?
- Peut-on ouvrir plusieurs fichiers dans différents onglets ?
- Peut-on visionner simplement le résultat ? (par exemple avec une touche *voir cette page dans le navigateur*)
- Est-ce que les encodages de caractères sont bien gérés ?
- Y a-t-il une auto-complétion ? (quand vous écrivez une balise : le logiciel écrit directement la balise fermante)
- L'indentation est elle facilement modifiable ? Notamment, est-ce que le logiciel comporte une fonction permettant de déplacer tout un bloc de ligne vers la gauche ou vers la droite ?

# L'en-tête

L'**en-tête** est la partie du fichier HTML comprise entre les balises `<head>…</head>`. Cette partie est située juste après le *doctype* et la balise d'ouverture `<html>`.

L'en-tête contient des informations sur la page elle-même (« méta-données ») : titre, auteur, description du contenu de la page, mots-clefs, où feuille(s) de style à utiliser…

La structure de l'en-tête est décrite dans le RFC 2616[1].

---

**Note**

Nous utilisons ici le HTML5. Pour le XHTML, rajoutez simplement les barres de fraction à la fin des balises isolées, par exemple

| XHTML | HTML5 |
|---|---|
| `<meta name="`*propriété*`" content="`*attributs*`" />` | `<meta name="`*propriété*`" content="`*attributs*`">` |

---

## Principaux éléments

**Titre**

Le titre est compris entre les balises `<title>…</title>`.
Habituellement, le titre de la page est affiché dans la barre de titre du navigateur (tout en haut), et lorsque le navigateur gère les onglets, dans l'étiquette des onglets.
**Cet élément est obligatoire.**

**Encodage**

Si le fichier contient des caractères répondant à la norme ISO-8859-1 « Latin-1 », on mettra la balise

```
<meta charset="utf-8">
```

Cette information est très fortement recommandée.

**Auteur**

Le nom de l'auteur est indiqué avec la balise suivante :

```
<meta name = "author"
   content = "nom de l'auteur" >
```

**Description de la page**

La description du contenu de la page est indiquée avec la balise

suivante :

```
<meta name = "description"
   content = "phrase de description" >
```

### Mots-clefs

Les mots-clefs servent à référencer la page. Cependant, de nombreux moteurs de recherche n'ont plus recours aux mots-clefs car des auteurs utilisaient des mots-clefs sans rapport avec le contenu afin d'augmenter la fréquentation de leur page (cas notamment de nombreux sites pornographiques). Les mots-clefs sont indiqués avec la balise suivante :

```
<meta name = "keywords"
   content = "liste de mots-clefs" >
```

### Robots

Le robot sert à gérer le référencement de la page. Il prend comme argument *follow* (permet au robot de suivre les liens de la page), *index* (permet au robot d'indexer la page), les arguments *nofollow* et *noindex* sont les contraires (ne pas suivre et ne pas indexer). Les deux derniers arguments sont *all* et *none* qui, comme leur noms l'indique, active ou désactive les deux fonctions. Les instructions pour robots sont indiquées avec la balise suivante :

```
<meta name = "robots"
    content = "all|(no)follow|(no)index|none" >
```

### Feuille de style

Lorsque la page utilise une feuille de style située dans un autre fichier, on utilise la balise

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

On peut aussi écrire la feuille de style (code CSS) directement dans l'en-tête, entre les balises `<style type="text/css">…</style>`.

> **Attention !**
>
> L'attribut "rel" n'est accepté que dans les balises "a", "area", et "link", et avec des valeurs prédéfinies[2].

# Balise <meta>

La balise <meta> a été utilisée plusieurs fois ci-dessus. Vous avez remarqué que sa syntaxe générale était :

```
<meta name="propriété" content="attributs" >
```

En plus d'indiquer des méta-informations sur la page, elle permet de faire un rafraîchissement automatique de la page, et d'utiliser des fonctionnalités spécifiques à certains navigateurs (exemple : des effets visuels de transitions entre les pages).

# Entités

## Définition

Le langage HTML utilise un jeu d'**entités** pour incorporer des *caractères spécifiques* dans le document. Plus simplement, vous tapez une séquence précise de caractères (oui c'est ça, un mot magique), et miraculeusement il se transforme en un caractère (accentué, spécial, de ponctuation,…) quand vous affichez la page dans un navigateur Web. Ces entités ont toutes la même préfixe : une *esperluette* « & ». La fin d'une entité est marquée par le caractère *point-virgule* « ; ».

Il est possible de recourir à deux types d'entités :

- les entités de type numérique composées d'un nombre précédé du caractère *croisillon* # (souvent appelé à tort « dièse »[3]) entre l'*esperluette* et le *point-virgule* ;
- les entités de type caractère composées d'une chaîne de caractères entre l'*esperluette* et le *point-virgule*.

Ainsi il est possible d'écrire le signe euro (€) de deux manières :

- `&#8364;` qui en est l'entité numérique décimale (index dans la table des caractères Unicode) ;
- `&euro;` qui en est l'entité caractère.

On peut aussi taper l'entité numérique en hexadécimal, en mettant un « x » entre le croisillon et le nombre. Par exemple, `&#196;` est la même chose que `&#xC4;`, c'est-à-dire « Ä ».

## Utilité

Les entités permettent d'afficher des caractères qui ne sont pas accessibles depuis le clavier. C'est notamment utile lorsque l'on utilise des symboles mathématiques, ou que l'on veut écrire des mots d'une langue étrangère par rapport au clavier utilisé, ou pour certaines capitales accentuées (comme « É »).

Cela permet également d'utiliser un caractère qui se trouve en dehors du jeu de caractères déclaré en début de fichier (avec `<meta http-equiv="Content-Type" content="text/html; charset=…" >`, voir le chapitre *L'en-tête*).

Notons que l'entité numérique fait référence à l'adresse du caractère dans la table Unicode[4], tandis que l'entité caractère est redirigé vers la table selon la définition du type de document (DTD). Ainsi, il est plus simple de retenir l'entité caractère (mnémotechnique), mais l'entité numérique assure la compatibilité quelle que soit l'évolution des DTD.

## Liste des entités

Quelques exemples d'entités

| Caractère spécial | Entité caractère | Entité numérique | Mnémotechnique |
|---|---|---|---|
| **Lettres spéciales (diacritiques, ligaturées, non romaines)** | | | |
| É | É | É | E (accent) aigu *(acute)* |
| Ò | Ò | Ò | O (accent) grave |
| Â | Â | Â | A (accent) *circ*onflexe |
| Ã | Ã | Ã | A tilde |
| Ä | Ä | Ä | A *umlaut* (inflexion allemande, marquée par un tréma) |
| Å | Å | Å | *ring* = anneau |
| Æ | Æ | Æ | A et E ligaturés (E dans l'A) |
| æ | æ | æ | a et e ligaturés (e dans l'a) |
| Œ | Œ | Œ | O et E ligaturés (E dans l'O) |
| œ | œ | œ | o et e ligaturés (e dans l'o) |
| Ç | Ç | Ç | C cédille |
| Γ | Γ | Ɖ | gamma capitale (lettre grecque) |
| γ | γ | γ | gamma minuscule (lettre grecque) |
| **Caractères typographiques** | | | |
| — | — | — | *M dash* (tiret de la largeur d'un M, tiret long, tiret d'incise, tiret cadratin) |
| – | – | – | *N dash* (tiret de la largeur d'un N, tiret moyen, tiret d'intervalle, tiret demi-cadratin) |
| · | · | · | *middle dot* (point centré, virgule géorgienne) |
| *espace insécable*[5] |   |   | *non breakable space* |
| *espace fine*[6] |   |   | *thin space* |
| *espace fine insécable* | |   | *non breakable thin space* |
| « | « | « | *left angle quote* (guillemet angulaire gauche, ouvrant) |
| » | » | » | *right angle quote* (guillemet angulaire droit, fermant) |

| | | | |
|---|---|---|---|
| • | &bull; | &#8226; | *bullet* (puce) |
| … | &hellip; | &#8230; | *horizontal ellipse* (points de suspension) |
| **Symboles commerciaux et mathématiques** | | | |
| ™ | &trade; | &#8482; | *trade mark* (marque commerciale) |
| ® | &reg; | &#174; | *registred* (marque déposée) |
| © | &copy; | &#169; | *copyright* (tous droits réservés) |
| € | &euro; | &#8364; | |
| ² | &sup2; | &#178; | *superior 2* (2 en exposant, carré) |
| ½ | &frac12; | &#189; | fraction 1/2 |
| ± | &plusmn; | &#177; | *plus minus* (plus ou moins) |
| ← | &larr; | &#8592; | *left arrow* ; voir aussi → (*right* →), ↑ (*up* ↑), ↓ (*down* ↓), ↔ (*horizontal* ↔), ↵ (*carriage return arrow*, « retour chariot » ↵) |
| ⇒ | &rArr; | &#8658; | *right big arrow* ; voir aussi ⇒ (*right* ⇒), ⇑ (*up* ⇑), ⇓ (*down* ⇓), ⇔ (*horizontal* ⇔) |
| × | &times; | &#215; | *times* = fois |
| · | &sdot; | &#8901; | *scalar multiplication dot* (signe de multiplication scalaire) |
| ÷ | &divide; | &#247; | *divide* = diviser |
| √ | &radic; | &#8730; | radical (racine carrée) |
| ‰ | &permil; | &#8240; | *per* (« pour » en latin) mille |
| ∝ | &prop; | &#8733; | proportionnel |
| ≃ | | &#8771; | |
| ¦ | &brvbar; | &#166; | *broken vertical bar* (barre verticale interrompue, tube) |

---

**Aller plus loin**

On peut trouver une liste complète, en français, des entités utilisées par le langage HTML sur alexandre.alapetite.fr/ (http://alexandre.alapetite.fr/doc-alex/alx_special.html).

---

**Note**

Les éditeurs de texte n'ont pas toujours accepté les caractères accentués (non-ASCII pur). Pour avoir les caractères accentués, il fallait donc avoir recours à des entités (par exemple `&eacute;` pour « é »).

Depuis 2011, avec la généralisation de l'utilisation de l'UTF-8 comme encodage des caractères d'une page Web (réalisée en HTML5 entre autres), le recours aux entités HTML tend à devenir obsolète.
Une alternative consiste à afficher une page HTML contenant les caractères spéciaux (par exemple contenant des entités, numériques ou caractères) puis de faire un copier-coller de la fenêtre du navigateur dans votre code HTML.

---

### Exemple

```
<em>Dungeons &amp; Dragons</em>&trade;
est le premier jeu de rôle de l'histoire ;
c'est aussi le plus joué.
```

donne

*Dungeons & Dragons*™ est le premier jeu de rôle de l'histoire ; c'est aussi le plus joué.

# Applications

## Espaces

On voit que le HTML propose plusieurs espaces : espace justifiante (espace « classique »), espace insécable et espace fine. On n'en utilise qu'une seule à la fois.

L'espace insécable permet de ne pas séparer deux objets, par exemple :

- les blocs de trois chiffres d'un nombre long ;
- un nombre de son unité ;
- un titre (D$^{r}$, M., M$^{gr}$, …) ou une particule (de) du nom qui le suit ;
- une ponctuation double (;:!?) du mot qui le précède.

L'espace fine permet d'éloigner légèrement deux caractères pour faciliter la lisibilité, par exemple après de l'italique : comparer « (*l*) » et « (*l* ) ». On peut aussi l'utiliser pour séparer les groupes de trois chiffres par une espace fine :

1 360 000 (espace normale)
1 360 000 (espace fine)

Le problème de ces deux applications est qu'une fin de ligne peut séparer les deux objets (bien que le caractère ` ` soit considéré comme insécable par certains navigateurs, il devrait être sécable). Pour une espace fine insécable, on utilisera ` `.

Si l'on veut avoir une espace plus grande, comme pour une tabulation, on pourra utiliser :

- un tableau sans bordure (cf. *Tableaux*), permettant d'aligner plusieurs blocs de texte ;

- du CSS pour insérer une grande espace.

Par exemple, le code CSS pour mettre une espace de 5 cadratins (5 lettres « M ») à gauche serait

```
<span style="margin-left:5em">texte</span>
```

## Afficher du code HTML

Imaginez que vous vouliez afficher du code HTML sur votre page. Vous voulez par exemple afficher le texte «  ». Seulement voilà, le navigateur va croire que c'est une balise, et ne rien afficher mais mettre le texte en italique. La solution est de de ne pas utiliser les caractères < et > mais les entités correspondantes : respectivement `<` (de l'anglais *less than*, « inférieur à ») et `>` (de l'anglais *greater than*, « plus grand que »). C'est donc `<em>` qu'il faut taper.

À noter que pour afficher « C'est par là -> », même si le navigateur va en général faire ce qu'on a envie qu'il fasse à savoir afficher le symbole « > » (ou « < » le cas échéant) tel quel, il est de bon ton de remplacer quand même « > » et « < » par leurs entités, sous peine de se faire remonter les bretelles par le validateur.

De la même façon, si vous voulez afficher le texte « € », eh bien il faudra ruser pour que le navigateur ne croit pas qu'on veut afficher le caractère €. On remplacera donc le caractère « & » par l'entité correspondante : `&`.
Et pour afficher le texte "&" on fait comment ? Je sens que vous avez deviné : `&amp;`.

Comme pour < et >, si vous voulez afficher le texte « &salut; » (qui n'existe pas comme entité hein, on a juste envie de mettre une esperluette et un point virgule où ça nous chante), « Hey, &salut ça va ? », ou « Machin & Compagnie », il est recommandé de remplacer « & » par `&`.

Notez que les deux *ruses de sioux* exposées précédemment sont utilisées sur cette page, vous n'avez qu'à regarder le code source !

Il y a aussi d'autres utilisations des entités. Par exemple elle vous permettent d'utiliser des caractères que vous ne pouvez pas taper au clavier. Par exemple, je ne sais pas, tout le monde ne sait pas taper ∉ (le signe mathématique pour dire « n'appartient pas à ») alors qu'avec un petit `&notin;` , le tour est joué.

# Notes

1. *(anglais))* Request for comments nº 2616 (https://tools.ietf.org/html/rfc2616).
2. https://www.alsacreations.com/article/lire/1400-attribut-rel-relations.html
3. mais le caractère dièse est différent : « ♯ », obtenu par ♯
4. Pour en savoir plus sur Unicode, vous pouvez consulter le livre : À la découverte d'Unicode.
5. en typographie française, on met une espace insécable avant les deux-points « : », point-virgule « ; », points d'exclamation et d'interrogation « ! ? », et à l'intérieur des guillemets et des tirets d'incise ; cela évite d'avoir ces caractères séparés du mot qui les précède — ou les suit — par un retour de ligne
6. en typographie, le mot « espace » est féminin

# Titres et paragraphes

Un document HTML doit être structuré correctement afin d'être bien compris par les lecteurs humains ainsi que les programmes informatiques (par exemple les robots d'indexation des moteurs de recherche). « structuré correctement » signifie divisé en titres, sous-titres et paragraphes.

Cette partie présente la façon de spécifier le découpage du document. Vous remarquerez sûrement que les exemples ont un affichage qui ressemble au web d'il y a dix ans, mais l'important est de structurer correctement vos documents. L'habillage de ceux-ci est expliqué dans le livre sur le CSS (et l'intégration dans la code HTML dans son chapitre *Interface HTML*). On ne dirait pas comme ça mais c'est un point très important : la séparation entre contenu pour le HTML et aspect visuel pour les CSS est un élément capital pour un site moderne.

## Les paragraphes

### La balise `p`

La balise `p` sert à créer un paragraphe. Habituellement, il est isolé par défaut du reste du texte par un petit espace vertical avant et après (typographie anglaise[1]), mais vous pouvez changer ça avec les fameux CSS. De plus, le texte va automatiquement à la ligne à la fin d'un paragraphe.

### Exemple

```
<body>

   <p>Voici un paragraphe.</p>
   <p>En voici un deuxième.</p>

</body>
```

ce qui donne

Voici un paragraphe.

En voici un deuxième.

La balise `p` indique bien un paragraphe (nature de la portion de texte) et non pas un saut de ligne (forme). On ne doit pas mettre de paragraphe vide. Voir ci-dessous pour avoir un blanc vertical plus grand.

### Retours à la ligne

Comme on l'a vu précédemment, les retours de ligne sont interprétés comme une espace. Pour placer un retour de ligne au sein d'un paragraphe, on utilise la balise `<br>`[2] *(break)*.

L'utilisation de plusieurs balises `<br>` successives est à proscrire ; il s'agit d'un retour de ligne et pas d'un saut de ligne. Rappelez-vous que le HTML concerne la description de contenu et pas la mise en forme. Pour définir une espace vertical entre deux paragraphe, il faut avoir recours au CSS ; par exemple, pour un paragraphe ponctuel, on pourra utiliser

```
<p style="margin-top:20px">
…
</p>
```

pour avoir un espace de 20 pixels (cet espacement ne change pas si l'on modifie la taille de la police), ou bien

```
<p style="margin-top:2em">
…
</p>
```

pour avoir un espace de 2 fois la largeur du caractère M (cet espacement est proportionnel à la taille de la police).

### Ligne de séparation

Il est possible de mettre une ligne de séparation entre deux paragraphes, avec la balise `<hr />`[3] *(horizontal ruler)*.

## Les titres : balises h

Les titres et le sous titres sont indiqués par la balise `h1` à `h6` (*heading* — six niveaux de titres sont possibles).

Le niveau 1 est le niveau le plus haut dans la hiérarchie du document, suivi du niveau 2, etc.

### Exemple

```
<body>

  <h1>Un titre de niveau 1 (un gros titre)</h1>
    <p>Un paragraphe.</p>

    <h2>Un titre de niveau 2 (un sous titre)</h2>
      <p>Un paragraphe.</p>

      <h3>Un titre de niveau 3 (un sous-sous titre)</h3>
        <p>Etc.</p>

</body>
```

Ce qui donne quelque chose comme

### Un titre de niveau 1 (un gros titre)

Un paragraphe.

**Un titre de niveau 2 (un sous titre)**

Un paragraphe.

**Un titre de niveau 3 (un sous-sous titre)**

Etc.

Il convient de respecter l'ordre hiérarchique des titres : un titre de niveau 2 sera toujours situé sous un titre de niveau 1, un titre de niveau 3 sous un titre de niveau 2…

---

**Attention**

Il ne faut pas choisir un titre pour son rendu (taille du texte affiché) mais pour son importance dans la hiérarchie. Ainsi, on ne mettra pas un titre de niveau 2 lorsqu'il s'agit du titre principal. La taille, couleur etc. des titres sera ensuite modifiable en… CSS, vous commencez à comprendre.

---

# Notes

1. en typographie française, les paragraphes ne sont pas séparés par des espaces verticaux, mais la première ligne est décalée vers la droite (retrait de paragraphe ou « alinéa rentrant »)
2. En XHTML, cela sera `<br />`
3. même remarque que précédemment, en HTML « simple », cela sera `<hr>`

# Style de texte

Le **langage HTML/Style de texte** indique au navigateur qu'il a affaire à tel ou tel type de texte pour changer la mise en forme.

Comme pour les titres, la manière dont le texte est mis en forme dépend du réglage du navigateur, ou de la feuille de style (voir *Le langage CSS*).

Plutôt que de « style de texte », il vaudrait mieux parler d'« éléments de texte » (« éléments de phrase » et « de citation »), puisque les balises décrivent des portions de texte.

Styles de texte les plus courants

| Style | Balise | Mnémotechnique | Rendu par défaut |
|---|---|---|---|
| mise en emphase | `<em>...</em>` | emphase | italique |
| mise en emphase forte | `<strong>... </strong>` | « fort » | gras |
| citation d'une référence | `<cite>...</cite>` | ... | italiques |
| citation courte dans le texte | `<q>...</q>` | *quote* (citation) | entre guillemets |
| citation à part | `<blockquote>... </blockquote>` | « bloc de citation » | marge à gauche plus grande |
| code source | `<code>...</code>` | ... | police à chasse fixe avec empattement (type Courier) |
| texte préformaté (par exemple pour aligner avec des espaces, ou faire des dessins ASCII) | `<pre>...</pre>` | *preformed* (préformatté) | police à chasse fixe, en général avec empattement (type Courier) |

La tendance est de préférer utiliser les balises[1] :

- `<i>...</i>` par `em`.
- `<b>...</b>` par `strong`.

En effet, cela revient à définir la description de texte plutôt que faire sa mise en forme, tâche plutôt dévolue au CSS.

Dans la pratique, si cela ne change pas grand chose pour un rendu graphique (sauf si la feuille de style ou le rendu du navigateur en décide autrement), et cela favorise l'accessibilité : s'il est possible pour un lecteur d'interpréter une mise en emphase (par exemple en changeant le ton de la voix), il est en revanche impossible de rendre une variation de mise en forme.

## Exemple

```
<p>Selon Boileau :</p>
<blockquote>
   Vingt fois sur le métier
   remettez votre ouvrage
</blockquote>
<p>bref, <em>ne vous découragez pas !</em>
   <strong>Soyez patient !</strong></p>
```

ce qui donne

Selon Boileau :

> Vingt fois sur le métier remettez votre ouvrage

bref, *ne vous découragez pas !* **Soyez patient !**

Voici quelques autres styles de texte :

Autres styles de texte

| Style | Balise | Mnémotechnique | Rendu par défaut |
|---|---|---|---|
| saisie clavier | `<kbd>...</kbd>` | *keyboard* (clavier) | *idem* `<code>` |
| abréviation | `<abbr>...</abbr>` | *abbreviation* | souligné par un trait pointillé |
| acronyme | `<acronym>... </acronym>` | ... | *idem* `abbr` |

**Note**

En français, un acronyme est lexicalisé, c'est-à-dire prononcé comme un mot (comme laser, ovni), alors que dans un sigle, les lettres sont prononcées séparément (SNCF). En anglais, langue de référence du HTML, la définition est plus floue : certains distinguent les termes *initialism* (sigle) et *acronym* (sigle lexicalisé, même sens qu'en français), alors que pour d'autres *acronym* désigne un sigle qu'il soit lexicalisé ou non (voir l'article du Wikipédia anglophone *Acronym*). Dans la pratique, il n'y a pas de différence dans le traitement de `<abbr>` et de `<acronym>`, cette dernière disparaît d'ailleurs du XHTML 2.

Même lorsqu'ils ne modifient pas la mise en forme, les balises `abbr` et `acronym` sont utiles pour les analyseurs syntaxiques *(parsers)* et les correcteurs d'orthographe. Ils sont surtout utiles avec le paramètre `title` (titre), qui permet d'expliciter l'abréviation ; en général, le titre s'affiche dans une info-bulle.

**Exemple**

```
Les pages Web sont écrites en <acronym title="Hypertext Markup
Language">HTML</acronym>.
```

donne

Les pages Web sont écrites en HTML

# Paramètres

Pour les balises de citation (`<q>` et `<blockquote>`), on peut préciser la source, avec le paramètre `cite` indiquant son URL.

### Exemple

```
Selon le site Alsacréations,
<q cite="http://css.alsacreations.com/Bases-et-indispensables/Quelle-est-la-
difference-entre-un-div-et-un-calque">
  cet abus de langage
  est malheureusement demeuré très ancré
  et induit de nombreux amalgames.
</q>
```

# Balises de mise en forme

Avant le CSS, la modification de la police était déjà une préoccupation, le W3C a donc créé des balises permettant ces modifications — donc de la mise en forme — que l'on peut toujours trouver dans du code ou des ouvrages. Les balises sont donc données à titre d'information, mais elles devraient être proscrites au profit du CSS.

Autres styles de texte

| Style | Balise | Mnémotechnique |
|---|---|---|
| affichage écran (*idem* code source) | `<tt>...</tt>` | *teletype* (terminal) |
| italique | `<i>...</i>` | *italic* |
| gras | `<b>...</b>` | *bold* (gras) |
| grandes lettres | `<big>...</big>` | (grand) |
| petites lettres | `<small>...</small>` | (petit) |
| texte barré | `<strike>...</strike>` ou `<s>...</s>` | (barré) |
| texte souligné | `<u>...</u>` | *underligned* (souligné) |

On peut également modifier :

- la taille de la police : `<font size="taille">...</font>`, ou *taille* est un nombre absolu en unité arbitraire (la taille normale est 3), ou un nombre relatif (+1 pour la taille courante augmentée de 1, -2 pour la taille courante diminuée de 2) ;
- la nature de la police : `<font face="nom de la police">...</font>`.

On peut combiner les deux, par exemple `<font size="12" face="Times New Roman">...</font>`

Ces balises sont maintenant déconseillées. On peut utiliser `em` et `strong` pour l'italique et le gras, et le CSS pour le reste :

- lettres plus grandes de 10 % : `<span style="font-size:1.1em">…</span>`
- lettres plus petites de 10 % : `<span style="font-size:0.9em">…</span>`
- souligner : `<span style="text-decoration:underline">…</span>`
- barrer : `<span style="text-decoration:linethrough">…</span>`
- type de police : `<span style="font-family:"Times New Roman" ">…</span>`

mais il vaut mieux définir des styles dans un fichier à part et faire appel à ces styles (ce qui simplifie la maintenance des fichiers), par exemple, mettre dans le fichier CSS :

```
.grand {font-size:1.1em}
.petit {font-size:0.9em}
.souligner {text-decoration:underline}
.barrer {text-decoration:linethrough}
```

ce qui s'exploite dans le fichier HTML de la manière suivante :

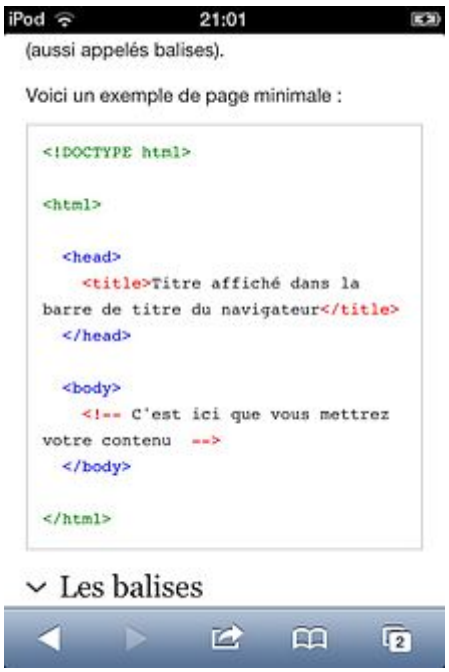
```
<p>
  Du texte en <span class="grand">grandes</span>
  ou <span class="petit">petites</span> lettres,
  <span class="souligner">souligné</span>
  ou <span class="barrer">barré</span>.
</p>
```

*Pour plus de détails voir :* ***Le langage CSS/Texte***.

# Références

1. https://www.alsacreations.com/article/lire/552-strong-b-em-i-quelle-balise-utiliser-et-pourquoi.html

# Placement des objets



Les bases du codages en HTML

Le placement des objets relève de la mise en forme. Certes, on a expliqué en introduction que le HTML ne concernait pas la mise en forme mais seulement la description du contenu ; par exemple, la balise `<blockquote>…</blockquote>` a pour effet de décaler le texte vers la droite, mais ce n'est qu'une conséquence de la feuille de style appliquée au bloc de citation.

Nous allons faire une exception à cette règle pour placement des objets. En effet, avant le CSS, le placement des objets était déjà une préoccupation, le W3C a donc créé des balises permettant ce placement — donc de la mise en forme —, on peut donc toujours trouver de telles balise dans du code ou des ouvrages. Les balises sont donc données à titre d'information, mais elles devraient être proscrites au profit du CSS.

## Ancien HTML

Par défaut, les objets — texte, images… — sont aligné à gauche, à moins que le réglage du navigateur ou la feuille de style en décide autrement.

- Il est possible de centrer les objets avec les balises `<center>…</center>`.
- Pour aligner à droite, il faut utiliser la balise `<div align="right">…</div>`.
- Le texte, quant à lui, peut être justifié avec la balise `<div align="justify">…</div>`.

On peut aussi indiquer l'alignement à l'intérieur de la balise de description, par exemple :

- pour centrer un paragraphe, on peut utiliser `<p align="center">…</p>` ;
- pour centrer une image, on peut utiliser `<img src="…" … align="center" />`.

## Méthode moderne : avec du CSS

Ceci est maintenant obsolète avec le CSS. Il est maintenant recommandé de ne pas utiliser les balises ci-dessus et d'avoir recours au CSS, par exemple :

- pour du texte : `<div style="text-align:`*valeur*`">…</div>`,
  où *valeur* est `left` (gauche), `right` (droite), `center` (centré) ou `justify` (justifié);
- pour une image :
  - pour aligner à droite : `<div style="text-align: right;padding-right: 10px">…</div>` (ce qui laisse une marge de 10 pixels à droite) ;
  - pour centrer : `<div style="text-align: center">…</div>`.

### Exemple

```
<div style="text-align:right">
   Bla bla bla bla bla bla bla
</div>
<div style="text-align: center">
   <img src="http://fr.wikibooks.org/images/wiki-textbook.png" width="129" />
</div>
```

donne

Bla bla bla bla bla bla bla

# Liens

## Introduction

Un lien (également appelé « lien hypertexte », d'où le nom d'HTML) est un mot ou groupe de mots permettant, après avoir cliqué dessus, d'ouvrir un fichier. Si ce fichier est une page HTML ou un fichier texte (`.txt` ou autre), il s'affichera dans le navigateur à la place ou en plus de la page actuelle. S'il s'agit d'un autre type de fichiers, le navigateur demandera généralement de sélectionner une application pour ouvrir le fichier ou d'enregistrer le fichier. Il est possible que l'utilisateur ait installé une extension *(plug-in)* pour pouvoir lire certains types de fichiers.

*Note* : certains formats sont devenus si courants que la plupart des navigateurs les supportent sans que l'on ne doive installer d'extension. C'est le cas *notamment* des formats d'images les plus courants (`.bmp`, `.gif`, `.jpg`, `.png`, `.tif`, etc). Cela dépend évidemment du navigateur utilisé et surtout de sa version. Les plus récents supportent généralement aussi XML.

*Note* : certains navigateurs sont livrés avec les extensions gérant les formats les plus courants, comme Java ou Flash.

*Note* : l'installation de certains logiciels entraîne le support de certains types de fichiers sur certains navigateurs. C'est le cas notamment d'Adobe Reader 6.0.

## Balise

Pour créer un lien en html, on utilise les balises `<a>` et `</a>` (de l'anglais *anchor* signifiant « ancre ») pour encadrer le contenu du lien (qui peut être du texte, une image, etc). La balise `<a>` dispose de trois attributs possibles :

- `href` : permet d'indiquer l'adresse du lien ;
- `name` : permet de définir une ancre (voir Liens vers une ancre) ;
- `target` : permet de définir la cible du lien (voir Comment ouvrir un lien dans une nouvelle fenêtre ?).

Exemples :

```
<a href="http://fr.wikibooks.org">Lien vers Wikilivres</a>
<a name="ancre_1">Définition d'une ancre</a>
<a href="http://fr.wikipedia.org" target="_blank">Lien ouvrant Wikipédia dans une
nouvelle fenêtre</a>
```

## Les URL

Une URL (*uniform resource locator*), indique au navigateur l'adresse où aller chercher une ressource ; par ressource, on entend autre page Web, mais aussi autre type de fichier : image, son…

Si le fichier à aller chercher se trouve dans le même répertoire (dossier) que la page Web que l'on écrit, on se contente de donner le nom du fichier. On a ici deux types d'adresses : les adresses relatives et les adresses absolues.

## Liens absolus

Un lien absolu indique l'adresse complète du fichier lié (de `http://` à l'extension ou à l'extension du domaine). Il est généralement utilisé pour afficher une page d'un autre site.

Exemple :

```
<a href="http://fr.wikipedia.org">Une encyclopédie vraiment géniale</a>
```

L'adresse absolue peut être utilisée lorsque l'objet se trouve sur son propre site, à condition que l'on soit sûr qu'il ne bougera pas ; par exemple, toutes les images sont mises dans un répertoire `/images` qui ne sera jamais déplacé.

Dans ce cas, on peut omettre le protocole et le nom de domaine, en donnant une adresse commençant par un slash :

```
<img src="/images/exemple.jpg"> Une belle image.
```

L'utilisation de l'adresse absolue est obligatoire lorsque l'on fait un lien vers un objet situé sur un autre site Web, ou lorsqu'un autre protocole que HTTP est utilisé :

```
<a href="ftp://ftp.download.dom/mysoftware/">Télécharger le logiciel par FTP</a>.
```

## Liens relatifs

Un lien relatif indique l'adresse du fichier **par rapport** à la page actuelle. Il est vivement conseillé de l'utiliser le plus souvent dans son site web, car cela permet de changer d'hébergeur sans devoir rééditer chaque lien.

Si le fichier se trouve dans le même répertoire que la page courante, il suffit d'indiquer le nom du fichier (avec son extension). Si le fichier se trouve dans un sous-répertoire du répertoire contenant la page courante, il suffit d'indiquer le nom du répertoire, suivi d'une barre oblique « `/` » et du nom du fichier.

Exemples :

```
<a href="exemple.html">Lien vers le fichier « exemple.html » se trouvant dans le même
répertoire que la
page actuelle</a>

<a href="repertoire_exemple/exemple.html">Lien vers le fichier « exemple.html » se
trouvant dans le
répertoire « repertoire_exemple » qui se trouve dans le même répertoire que la page
actuelle</a>
```

Si le fichier se trouve dans le répertoire parent du répertoire actuel, ou dans un sous-répertoire du répertoire parent, il faut « remonter » à l'aide de deux points « `..` ».

Exemple :

```
<a href="../exemple.html">Lien vers le fichier « exemple.html » se trouvant dans le
répertoire parent
du répertoire dans lequel se trouve la page actuelle</a>
```

L'utilisation de l'adresse relative est intéressante si l'objet visé est situé dans une branche de l'arborescence qui restera toujours solidaire. Si l'on déplace la branche (donc la page Web courante *et* l'objet), l'adresse relative reste valable.

Par contre, si la page Web change de répertoire et pas l'objet cible, l'adresse relative devient erronée.

# Liens vers une ancre

Une ancre permet d'afficher une page web à partir d'un certain point (très utile dans les longues pages web). Ainsi, ce lien affiche le chapitre « Liens » du livre « HTML » à partir de la section « Liens vers une ancre ».

Pour créer une ancre, il faut utiliser la balise `<a>` avec l'attribut `name`, comme ceci (Il n'est pas nécessaire d'inclure du texte entre les deux balises) :

```
<a name="ancre_1"></a>
```

Lorsque l'ancre est vide, elle ne doit cependant pas être écrite sous la forme `<a name="section_5" />`, formellement valide en XML, mais déconseillée par les normes XHTML et XML pour des raisons de compatibilité avec différents navigateurs.

Pour créer un lien vers une ancre, il suffit de rajouter un croisillon (#) suivi du nom de l'ancre après l'adresse (absolue ou relative) de la page. Si l'ancre se trouve sur la page actuelle, il ne faut pas noter l'adresse de la page.

Exemples :

```
<a href="exemple.html#ancre_1">Lien vers l'ancre n°1 de la page "exemple.html"</a>
<a href="#ancre_1">Lien vers l'ancre n°1 de la page actuelle</a>
```

Il faut cependant faire attention car les ancres sont soumis à la balise base. Si dans l'entête de la page se trouve une référence à la base :

```
<base href="www.monsite.com" />
```

Un lien tel que :

```
<a href="#ancre_1">Lien vers l'ancre n°1 de la page actuelle</a>
```

Ne redirigera pas vers l'ancre #ancre_1 de la page actuelle mais vers celle de la page définie dans la base. (Donc ici vers www.monsite.com/#ancre_1)

# Comment ouvrir un lien dans une nouvelle fenêtre ?

L'attribut `target` permet de définir la cible du lien, c'est-à-dire l'endroit où sera ouvert le fichier du lien. Pour ouvrir le fichier dans une nouvelle fenêtre, assigner la valeur `_blank` à l'attribut `target`.

Exemple :

```
<a href="http://fr.wikibooks.org" target="_blank">Lien ouvrant Wikilivres dans une
nouvelle fenêtre</a>
```

*Note* : Les valeurs que peut prendre l'attribut target seront détaillées dans le chapitre des cadres.

*Remarque* : target="_blank" n'est valide que dans les DTD HTML et XHTML1.0 *frameset* et *transitional*, mais ne peut être utilisé dans une DTD *strict*. Le besoin éventuel d'ouvrir des liens dans une nouvelle fenêtre du navigateur doit donc conduire à opter pour une DTD *transitional*.

# Autres types de liens

Nous avons vu que l'attribut `href` peut contenir autre chose que l'adresse d'une page Web. S'il contient un nom de fichier autre que HTML, cela télécharge le fichier, ou éventuellement cela ouvre le fichier avec un programme dédié (par exemple Adobe Acrobat Reader pour un fichier PDF).

Mais l'attribut `href` peut faire référence à des ressources disponibles par un autre protocole que le HTTP :

- transfert de fichier (protocole FTP) : par exemple `href="ftp://ftp.gnu.org/"`.
- forum usenet (protocole NNTP) : par exemple `href="news:fr.comp.infosystemes.www.auteurs"`.
- adresse de courriel (protocole SMTP) : par exemple `href="mailto:jeveuxduspam@monfournisseur.fr"`.
- adresse de messagerie instantanée (protocole XMPP/Jabber) : `href="xmpp:jeveuxduspim@monserveurjabber.fr"`.

# Images

Une image est insérée avec la balise `img` en spécifiant avec l'attribut `src` *(source)* le chemin de l'image à inclure (sous la forme d'une adresse réticulaire, URL) et avec l'attribut `alt` l'éventuel texte de remplacement de l'image :

### HTML5/HTML

```
<img src="mon_image.png" alt="texte si l'image n'est pas affichée">
```

### XHTML

```
<img src="mon_image.png" alt="texte si l'image n'est pas affichée" />
```

Par la suite, nous considérerons du HTML. Pour du XHTML, ajoutez simplement la barre de fraction avant la fermeture de la balise.

La balise `img` peut avoir divers attributs facultatifs. Seuls les attributs `src` et `alt` sont obligatoires.

- `align` : alignement de l'image (peut prendre les valeurs `top`, `bottom`, `middle`, `left` ou `right`) ;
- `alt` : texte de remplacement (affiché si le navigateur ne peut afficher l'image) ; cet attribut est obligatoire, il permet notamment à un malvoyant de comprendre l'image qu'il ne peut voir (son logiciel lit le texte à voix haute ou le transforme en braille) ;
- `title` : titre de l'image (s'affiche en info bulle dans le navigateur) ;
- `border` : largeur de la bordure de l'image (valeur exprimée en pixels) ;
- `height` : hauteur de l'image (si aucune valeur spécifiée, l'image garde sa hauteur normale) ;
- `width` : largeur de l'image (si aucune valeur spécifiée, l'image garde sa largeur normale).

## L'attribut `src`

L'attribut `src` contient donc l'adresse à laquelle on va chercher l'image (voir le chapitre *Liens*). On peut utiliser une adresse absolue ou relative.

Il faut cependant proscrire l'inclusion des objets (images, sons, vidéo) situés sur d'autres sites, ce que l'on appelle des « liens à chaud » *(hot links)*. En effet, cela génère du trafic sur le serveur cible au lieu de son propre serveur, ce que le serveur cible peut considérer comme du « vol de bande passante ». Par ailleurs, le fait que l'on utilise l'objet original n'implique pas un respect du droit d'auteur : certes on n'a pas fait de copie de l'objet, mais on utilise l'objet, et cela doit se faire avec l'accord de l'auteur et/ou de ses ayants droit.

Donc, si par exemple notre page web est `http://www.monsite.com/mapage.html` et que l'on veut inclure l'image `http://www.autresite.com/images/img1.png`, les deux solutions acceptables sont :

- demander l'autorisation au toilemestre du site `http://www.autresite.com/`, importer

l'image sur son propre site et l'exploiter en local, en affichant l'origine de l'image et les conditions d'utilisation imposées par le site d'origine ;

par exemple, si l'on place l'image en `http://www.monsite.com/image/img1.png`, on peut écrire

```
<img src="/image/img1.png" … > <br>
Reproduit avec l'aimable autorisation de
<a href="http://www.autresite.com/">Autresite.com</a>,
tous droits réservés
```

- ne pas inclure l'image mais faire le lien vers cette image, ou mieux vers la page Web contenant cette image ; par exemple

```
Voir l'illustration sur l'article dédié
du site <a href="http://www.autresite.com/lapage.html">Autresite.com</a>
```

# L'attribut `alt`

Il est nécessaire de donner, pour chaque image, un texte de remplacement pour les navigateurs ne supportant pas l'affichage d'images ou pour des raisons d'accessibilité des pages web (cf normes W3C). Le texte de remplacement est spécifié par l'attribut `alt` :

```
<img src="bienvenue.png" alt="Bienvenue !" >
```

Lorsqu'une image ne véhicule pas d'information, ce texte doit être vide (n'oubliez pas que le `alt` est lu ou rendu en braille pour un malvoyant), l'attribut est alors présent sous la forme :

```
<img src="image_decorative.png" alt="" >
```

---

**Note**

Microsoft Internet Explorer affiche le texte alternatif dans une info-bulle ; ce comportement est erroné puisque c'est normalement le titre (voir ci-dessous) qui est affiché en info-bulle.

---

# L'attribut `title`

Il est possible d'attribuer un titre à chaque image en plus de l'information alternative *(alt)*. L'attribut `title` doit contenir une information *optionnelle* sur l'image, ou reproduire l'attribut `alt`: Les navigateurs affichent cette information dans une info bulle :

```
<img src="renard.png" alt="Renard en promenade" title="Renard en promenade" >
```

**Remarque**

Les attributs `alt` et `title` peuvent être cumulés.

# Les attributs `height` et `width`

Si l'on omet ces attributs, l'image s'affiche en grandeur réelle (pixel pour pixel).

Si l'on indique une des dimensions (la largeur ou la hauteur), l'autre dimension est calculée pour que l'image s'affiche avec ses proportions originales.

Le fait d'indiquer les deux dimensions peut faciliter l'affichage : les images sont chargées après le texte, le navigateur peut ainsi réserver la place nécessaire, sinon, il doit décaler le texte au fur et à mesure que les images sont chargées. Il peut donc être intéressant d'indiquer les dimensions même si l'on utilise les dimensions originales de l'image. Cependant, l'auteur doit bien s'assurer que les proportions (rapport largeur/hauteur) sont respectées, sous peine que l'image s'affiche déformée.

# Image contenant des liens

Une image elle-même peut être à l'intérieur d'une balise `<a>…</a>` : on peut alors cliquer sur l'image pour accéder à la ressource désignée. Par exemple :

```
<a href="http://fr.wikibooks.org">
   <img src="https://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-
noslogan.svg" alt="Wikibooks" />
</a>
```

*Pour plus de détails voir : **Le langage HTML/Liens**.*

L'image peut également être découpée en zones cliquables menant vers des ressources différentes. Pour cela :

- la balise `<img>` doit contenir un attribut `usemap` dont la valeur rst un nom, le nom de la carte définissant les zones ;
- la carte est définie par une balise `<map>…<map>` qui contient des zones ; chaque zone est définie par une balise `<area />`.

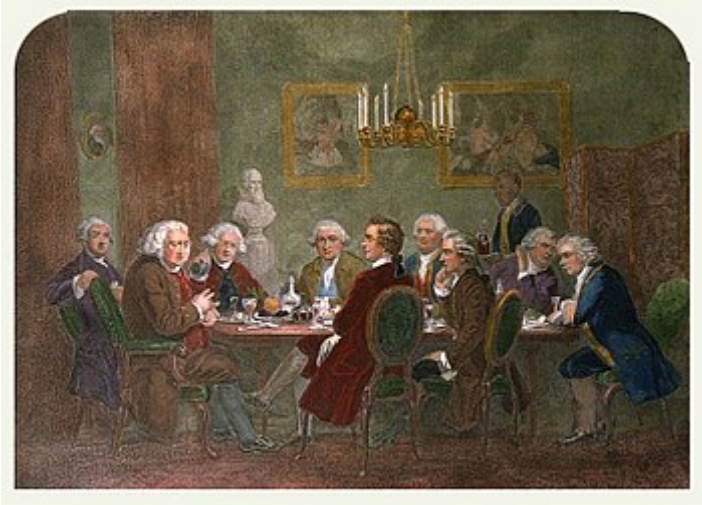
Par exemple :

```
<img src="image.png" alt="Carte du site" usemap="#nomdelacarte" />
<map name="nomdelacarte">
  <area shape="rect" coords="9,372,66,397" href="http://fr.wikipedia.org/"
alt="Wikipédia" title="Wikipédia" >
</map>
```

Les attributs de la balise `<area />` sont :

- `shape` : définit la forme de la zone, ses valeurs peuvent être `"rect"` pour un rectangle, `"poly"` pour un polygone et `"circle"` pour un disque ;

- `coords` : définit les coordonnées décrivant la zone, séparées par des virgules :
  - pour un rectangle, il s'agit des coordonnées des coins en bas à gauche et en haut à droite, sous la forme "`x1, y1, x2, y2`",
  - pour un polygone, il s'agit de la liste des coordonnées des points, sous la forme "`x1, y1, x2, y2, …, xn, yn`",
  - pour un disquee, il s'agit des coordonnées du centre et du rayon, sous la forme "`xc, yc, r`" ;
- les attributs classiques d'un lien : essentiellement `href` et `alt`



Exemple d'image découpée en zones menant vers différentes ressources. Les personnes représentées sont des The Club ; cliquer sur une personne pour accéder à l'article correspondant.

# Image SVG

Le SVG étant du XML, comme le XHTML, il est possible de définir une image SVG directement à l'intérieur du code HTML. Par exemple :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>
  <head>
    <title> Rectangle noir </title>
  </head>

  <body>
    <p>Voici un rectangle noir :</p>
    <svg width="3cm" height="2cm" version="1.1"
         xmlns="http://www.w3.org/2000/svg">
      <rect
        x="0.5cm" y="0.5cm" width="2cm" height="1cm"
      />
    </svg>
  </body>
</html>
```

*Pour plus de détails voir :* ***Découvrir le SVG***.

# Listes

Les listes permettent d'ordonner une énumération. Même si on les rencontre rarement, elles ont cet avantage qu'elles ont été crées spécialement pour faciliter une opération qu'effectue souvent tout créateur de page web et qui consiste à dresser une liste d'éléments.

## Types de liste

Il existe trois types de listes :

- une liste sans ordre précis (non ordonnée),
- une liste tenant compte de l'ordre (ordonnée),
- une liste de termes et descriptions (définitions).

### Listes non numérotées

Les listes non numérotées sont introduites par la balise `ul` (pour *unordered list*, « liste non ordonnée ») et chaque élément de la liste par `li` (*list item*, « élément de liste ») :

```
<ul>
    <li>premier élément de la liste ;</li>
    <li>deuxième élément ;</li>
    <li>etc.</li>
</ul>
```

Les balises fermantes pour les éléments `li` ne sont pas obligatoires. Ce code donnera quelque chose comme :

- premier élément de la liste ;
- deuxième élément ;
- etc.

### Listes numérotées

Les listes numérotées fonctionnent sur le même principe que les listes non numérotées. La seule différence est qu'on utilise la balise `ol` (*ordered list*, « liste ordonée ») au lieu de la balise `ul`. Les éléments restent encadrés par les balises `li`.

Exemple :

```
<ol>
    <li>premier élément de la liste ;</li>
    <li>deuxième élément ;</li>
    <li>etc.</li>
</ol>
```

Ce code donnera :

1. premier élément de la liste ;
2. deuxième élément ;
3. etc.

En HTML transitionnel, l'attribut `start` permet de définir la valeur initiale de la numérotation.

Exemple :

```
<ol start="5">
    <li>premier élément de la liste ;</li>
    <li>deuxième élément ;</li>
    <li>etc.</li>
</ol>
```

Ce code donnera :

5. premier élément de la liste ;
6. deuxième élément ;
7. etc.

## Listes de descriptions

Contrairement aux autres types de listes, les listes de description, `dl` *(definition list)*, n'utilisent pas la balise `li` pour les éléments de la liste, mais les sépare en deux parties : le terme, introduit par `dt` *(definition term)* et sa description introduite par `dd` *(definition description)*.

Par exemple :

```
<dl>
    <dt>C</dt>      <dd>Un langage efficace</dd>
    <dt>Java</dt>   <dd>Un langage portable</dd>
    <dt>Pascal</dt> <dd>Un langage pédagogique</dd>
</dl>
```

Ce code donnera :

**C**
: Un langage efficace

**Java**
: Un langage portable

**Pascal**
: Un langage pédagogique

# Propriété des listes

En CSS, les listes sont hautement paramétrable, elles ont même des propriétés valables uniquement pour elles. D'une manière générale on peut se pencher sur :

- les puces (petit élément devant chaque élément de la liste) ; ces puces pourront changer pour devenir un caractère prédéfini, personnalisable ou encore une image ;
- le type de numérotation : latin, arabe, grec, ...

*Voir Le langage CSS/Listes.*

# Tableaux

Avant toute chose, il est nécessaire de rappeler que l'utilisation de tableaux doit se limiter au classement et à la présentation des données. Ce n'est donc pas un outil de mise en page même s'il est souvent employé, par erreur, à cet effet. Car par sa rigueur voire sa rigidité, il empêche et complexifie, à l'usage, la mise en page d'un site internet.
En revanche, pour la présentation de données statistiques par exemple, il ne possède aucune concurrence.

## Balises de bases

La déclaration d'un tableau se fait grâce à la balise `table` ;

- la balise `tr` *(table row)* permet de définir une ligne, une rangée dans un tableau ; cette balise encadre les balises `td` de cellules ;
- la balise `td` *(table data)* permet de définir une cellule de donnée dans une ligne ; elle sera contenue obligatoirement dans la balise `tr` ;
- la balise `th` *(table header)* permet de définir des cellules comme étant des entêtes de lignes ou de colonnes ; généralement, le texte sera mis en emphase forte suivant les styles par défaut du navigateur ou de l'utilisateur ;
- la balise `caption` (« légende ») permet de donner un titre au tableau.

### Exemple élémentaire

```
<table border="1" >

   <caption>Table simple</caption>

   <tr>             <!-- ligne 1 -->
      <th> a1 </th> <!-- case 1 -->
      <th> a2 </th> <!-- case 2 -->
   </tr>

   <tr>             <!-- ligne 2 -->
      <td> b1 </td> <!-- case 1 -->
      <td> b2 </td> <!-- case 2 -->
   </tr>

</table>
```

donne

Table simple

| a1 | a2 |
|---|---|
| b1 | b2 |

## Attributs de la balise `table`

Comme toutes les balises, la balise `table` peut avoir des attributs. On peut bien sûr utiliser les attributs habituels (`id`, `class`, `lang`, `xml:lang`, `title`…).

### Attributs de mise en forme

Parmi les attributs spécifiques, nous avons des attributs de mise en forme :

- `border` : permet de définir si la table a des traits (bordure de cases) ou pas, et quelle est leur épaisseur en pixels :
  `border="0"` indique que la table n'a pas de traits, `border="1"` indique qu'elle a des traits de un pixel de large…
- `width` : permet d'indiquer la largeur totale du tableau, en pixels (nombre entier seul) pourcentage de la page totale (par exemple `"30%"`) ;
- `rules` : si l'on a une bordure (attribut `border` ayant une valeur non nulle), on peut choisir quels sont les traits intérieurs qui sont tracés :
  - `all` (valeur par défaut si `border` est non nul) : tous les traits sont tracés ;
  - `rows` : les lignes sont séparées par des traits, mais pas les colonnes ;
  - `cols` : les colonnes sont séparées par des traits, mais pas les lignes ;
  - `none` : aucun trait intérieur n'est tracé ;
- `frames` ou `frame` (sans s) : si l'on a une bordure (attribut `border` ayant une valeur non nulle), on peut choisir quels sont les traits du cadre du tableau qui sont tracés :
  - `border` (valeur par défaut si `border` est non nul) : tous les traits sont tracés ;
  - `hsides` : seuls les traits horizontaux sont tracés (haut et bas) ; `above` : seul le trait en haut est tracé ; `below` : seul le trait en bas est tracé ;
  - `vsides` : seuls les traits verticaux sont tracés (gauche et droite) ; `lhs` : seul le trait à gauche est tracé *(left hand side)* ; `rhs` : seul le trait à droite est tracé *(right hand side)* ;
- `void` : aucun trait de contour n'est tracé.

## Exemple

```
<table
  lang="en"
  title="Statistics for 2000"
  border="1"
  rules="all"
  frames="border"
  width="30%">

  <caption>Statistics for 2000</caption>
  …
</table>
```

Notons que ces attributs de mise en forme peuvent être remplacés par du CSS.

### Attribut d'accessibilité

Le plus grand problème des tableaux est l'accessibilité aux malvoyants. L'attribut `summary` permet de fournir une explication longue du tableau ; son contenu ne sera pas affiché par les navigateurs graphiques, mais interprété par les navigateurs en Braille ou les lecteurs vocaux.

### Attributs des balises `tr`, `th` et `td`

- `align` : indique l'alignement horizontal : `left` (aligné à gauche), `center` (centré), `right` (aligné à droite) ou `justify` (justifié) ;
- `valign` : indique l'alignement vertical : `top` (aligné en haut), `middle` (centré) ou `bottom` (aligné en bas).

## Balises de groupement

Les balises de groupement permettent comme leur nom l'indique de grouper des cellules entre elles suivant leur nature. Il est possible de créer des groupes de lignes ou de colonnes. La définition correcte de ces groupes permet entre autres une définition plus aisée des styles, comme par exemple séparer les groupes par des traits.

### Séparation des groupes

Si l'on a défini une bordure (attribut `border` ayant une valeur non nulle), on peut choisir de ne pas séparer les cellules du tableau par des traits, mais uniquement les groupes de lignes ou de colonnes. Pour cela, on indique l'attribut `rules="groups"` dans la balise d'ouverture `table` :

```
<table border="1" rules="groups">
  …
</table>
```

### Lignes

La balise `thead` *(table header)* permet de grouper les lignes d'entête. Elle est généralement utilisée avec la balise `th`.

La balise `tbody` *(table body)* permet de grouper les lignes du « corps », c'est-à-dire de la partie principale du tableau.

La balise `tfoot` *(table footer)* permet, à l'instar des deux précédentes, de grouper les lignes du pied de tableau. Elle est par exemple utilisée lorsque, dans le cas d'un tableau très long, les titres sont reproduits à la fin dans le but d'obtenir une meilleur lisibilité.

### Exemple

```
<table border="1" frame="void" rules="all">

  <thead>
     <tr>
        <th> a1 </th>
```

```
      <th> a2 </th>
   </tr>
</thead>

<tfoot>
   <tr>
      <td> a1 </td>
      <td> a2 </td>
   </tr>
</tfoot>

<tbody>
   <tr>
      <td> b1 </td>
      <td> b2 </td>
   </tr>
   <tr>
      <td> c1 </td>
      <td> c2 </td>
   </tr>
</tbody>

</table>
```

ce qui donne

| **a1** | **a2** |
|---|---|
| b1 | b2 |
| c1 | c2 |
| **a1** | **a2** |

## Colonnes

Les balises `col` et `colgroup` permettent de définir un groupe de colonnes dans le but d'appliquer un style ou une taille précise à une ou plusieurs colonnes.

Le principal attribut de `colgroup` est `span`, qui indique le nombre de colonnes faisant partie du groupe. S'il est absent, la valeur par défaut est `"1"`.

On peut utiliser tous les attributs spécifiques à `td`, ils s'appliqueront alors à toutes les colonnes du groupe.

### Exemple

```
<table border="1" rules="groups">

 <colgroup span="2" width="40" align="left" />
 <colgroup span="2" width="10" align="center" />

 <tr>
   <th> a </th> <th> b </th> <th> c </th> <th> d </th>
 </tr>
```

```
  <tr>
    <td> 1 </td> <td> 2 </td> <td> 3 </td> <td> 4 </td>
  </tr>

</table>
```

donne

| a | b | c | d |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

Au lieu de l'attribut `span`, on peut utiliser des éléments `col` pour désigner chaque colonne ; cela permet d'individualiser les attributs pour chaque colonne.

### Exemple

```
<table border="1" rules="groups">

  <colgroup align="left">
    <col width="20" />
    <col width="40" />
  </colgroup>
  <colgroup span="2" align="center" width="10" />

  <tr>
    <th> a </th> <th> b </th> <th> c </th> <th> d </th>
  </tr>
  <tr>
    <td> 1 </td> <td> 2 </td> <td> 3 </td> <td> 4 </td>
  </tr>

</table>
```

donne

| a | b | c | d |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

La balise `col` peut contenir un attribut `span`, c'est-à-dire qu'elle peut désigner plusieurs colonnes. Les mêmes attributs s'appliquent alors à ces colonnes, mais cela ne constitue pas un groupe au sens du HTML ; notamment, l'attribut `rules="groups"` de la balise `table` n'agit qu'entre les éléments `colgroup` et pas entre les éléments `col`.

## Fusionner des cellules

Il est possible de fusionner des cellules entre elles en utilisant des attributs de la balise `td`.

La fusion de plusieurs cellules d'une même ligne, c'est-à-dire de plusieurs colonnes, se fait grâce à l'option `colspan` et la fusion de plusieurs cellules appartenant à une même colonne, c'est-à-dire de plusieurs lignes, avec l'option `rowspan`. Ces deux attributs prennent la valeur correspondant aux nombres de cellules que l'on souhaite fusionner.

### Exemple

pour fusionner trois cellules d'une colonne :

```
<table>
    <tr>
        <td colspan="3">triple cellule</td>
    </tr>
    <tr>
        <td>cellule simple</td>
        <td>deuxième cellule simple</td>
        <td>troisième cellule simple</td>
    </tr>
</table>
```

Et voilà le résultat :

| triple cellule | | |
|---|---|---|
| cellule simple | deuxième cellule simple | troisième cellule simple |

---

### Note

Il ne faut pas confondre `<td colspan=…>` avec `<col span=…>`. Dans le premier cas, on a une fusion, c'est-à-dire que l'on a une seule et unique cellule ; dans le deuxième cas, on a simplement plusieurs colonnes distinctes qui partagent des attributs communs.

---

La fusion des cellules d'une colonne est un peu plus complexe, en raison de la construction linéaire des tableaux. En effet, lorsqu'une cellule occupe plusieurs lignes, elle est définie dans la première ligne et pas dans les lignes suivantes ; celles-ci contiennents donc un élément `td` de moins.

### Exemple

```
<table>
    <tr>
<!-- La première cellule des trois prochaines lignes seront fusionnées avec la
suivante -->
        <td rowspan="3">cellule 1</td>
        <td>cellule simple</td>
        <td>cellule simple</td>
        <td>cellule simple</td>
    </tr>
    <tr>
<!-- La première cellule de cette ligne étant fusionnée avec la précédente,
```

```
elle ne doit pas être définie -->
        <td>cellule simple</td>
        <td>cellule simple</td>
        <td>cellule simple</td>
    </tr>
    <tr>
<!-- La première cellule de cette ligne étant fusionnée avec la précédente,
elle ne doit pas être définie -->
        <td>cellule simple</td>
        <td>cellule simple</td>
        <td>cellule simple</td>
    </tr>
</table>
```

ce qui donne

| | | | |
|---|---|---|---|
| cellule 1 | cellule simple | cellule simple | cellule simple |
| | cellule simple | cellule simple | cellule simple |
| | cellule simple | cellule simple | cellule simple |

# Accessibilité aux malvoyants

Nous avons déjà vu ci-dessus comment mettre une description de la table à l'attention des malvoyants.

L'autre problème est celui de la lecture de la table. En effet, la table est lue linéairement, ligne par ligne et colonne par colonne. Ainsi, il est difficile de faire le lien entre le contenu d'une cellule et les en-têtes qui s'y rattachent (nom de la ligne et de la colonne), alors que c'est ce lien qui distingue la table d'une simple liste.

Pour contourner ce problème, on met un identifiant à chaque en-tête avec l'attribut `id`, et pour chaque cellule on indique l'identifiant des en-têtes qui s'y rattachent avec l'attribut `headers`.

## Exemple

```
<table
  border="1"
  summary="La table indique, pour chaque mois de l'année,
    la longueur de ce mois
    en jours et en heures" >

  <caption> Longueur des mois </caption>

  <tr>
     <td> </td>
     <th id="col2"> Janvier </th>
     <th id="col3"> Février </th>
     <th id="col4"> Mars </th>
  </tr>

  <tr>
     <th> Durée en jours </th>
```

```
    <td headers="col2"> 31 </td>
    <th headers="col3"> 28 ou 29 </td>
    <th headers="col4"> 31 </td>
  </tr>

  <tr>
    <th> Durée en heures </th>
    <td headers="col2"> 744 </td>
    <td headers="col3"> 672 ou 696 </td>
    <td headers="col4"> 744 </td>
  </tr>

</table>
```

Notons que l'en-tête peut aussi bien se trouver sur la première colonne de la ligne, mais cela ne pose en général pas de problème : puisque les données suivent l'en-tête, on associe facilement l'en-tête aux données, l'utilisation de `id` et `headers` est superflue.

S'il n'y a pas d'ambiguïté, et notamment pas de fusion de cellule, on peut utiliser l'attribut `scope` dans la balise.

Dans le cas d'un en-tête de colonne, on utilisera `scope="col"` et pour un en-tête de ligne, on utilisera `scope="row"` dans la balise `td` de la première cellule.

Si la personne veut lire directement le contenu d'une cellule, alors la ligne n'est plus lue en entier, on perd l'information de l'en-tête de ligne. Il peut donc être intéressant d'utiliser `scope="col"` et `scope="row"` conjointement.

`th`, on ne définit alors pas d'identifiant :

```
<table
  border="1"
  summary="La table indique, pour chaque mois de l'année,
    la longueur de ce mois
    en jours et en heures" >

  <caption> Longueur des mois </caption>

  <tr>
    <td> </td>
    <th scope="col"> Janvier </th>
    <th scope="col"> Février </th>
    <th scope="col"> Mars </th>
  </tr>

  <tr>
    <th scope="row" abbr="jours"> Durée en jours </th>
    <td> 31 </td>
    <td> 28 ou 29 </td>
    <td> 31 </td>
  </tr>

  <tr>
    <th scope="row" abbr="heures"> Durée en heures </th>
    <td> 744 </td>
```

```
    <td> 672 ou 696 </td>
    <td> 744 </td>
  </tr>

</table>
```

L'attribut `abbr` utilisé ci-dessus permet de raccourcir le texte lu par la suite.

# Voir aussi

- (anglais) Tables in HTML documents (http://www.w3.org/TR/html401/struct/tables.html), spécifications du W3C pour le HTML 4.01

# Formulaires

Un formulaire fournit un espace sur la page HTML où l'utilisateur peut entrer des données ; données qui pourront être envoyées au serveur pour être éventuellement traitées. Un formulaire est composé d'un ou plusieurs éléments d'entrée englobés par la balise `<form>`. Par exemple :

```
<form method="post">
    <p>Entrez votre nom : <input type="text" name="nom"></p>
    <p>Entrez votre prénom : <input type="text" name="prenom"></p>
</form>
```

**Attention** : la manière dont les données envoyées au serveur seront traitées ne dépendent pas du code HTML, qui peut juste indiquer le type de communication HTTP souhaité : *get* ou *post*. Pour traiter les données du formulaire, il faut élaborer une interface capable de les traiter sur le serveur, la plus commune étant un script PHP. Le rôle de l'HTML est d'indiquer la forme que doit avoir la requête HTTP qui sera envoyée par l'agent utilisateur au serveur et comment l'interface doit se comporter du côté client.

## La balise `<form>`

`<form>` permet de regrouper plusieurs entrées sous un seul nom, ce qui permettra de les traiter ensemble. Exemple :

```
<form name="formulaire_1" method="post" action="traitement.php">
    <p>Entrez votre nom : <input type="text" name="nom"></p>
    <input type="submit" value="Envoyer 1" />
</form>
<form name="formulaire_2" method="post" action="traitement.php">
    </p>Entrez votre prénom : <input type="text" name="prenom"></p>
    <input type="submit" value="Envoyer 2">
</form>
```

Ici, si vous cliquez sur « Envoyer 2 », les données saisies dans « nom » ne seront pas récupérées car le bouton « Envoyer 2 » ne porte que sur le formulaire dans lequel il se trouve (en l'occurrence « formulaire_2 »).

La balise `<form>` sert également à spécifier via l'attribut `action` l'URI du programme destiné à traiter les données validées, le protocole utilisé lors de la transaction HTTP, soit *get*, soit *post* via l'attribut `method` et enfin via l'attribut `accept-charset` l'encodage de caractères qui doit être accepté par le serveur pour manipuler ce formulaire.

L'attribut `enctype` permet de spécifier l'encodage pour un envoi de type POST. En particulier la valeur `enctype="multipart/form-data"` permet l'envoi de fichiers au serveur, en utilisant un champ de type fichier (`<input type="file">`). Sans l'utilisation de cet encodage, un champ fichier n'envoie que le nom du fichier au serveur, sans son contenu.

## La balise `<input>`

`<input>` est la balise la plus utilisée dans les formulaires. Elle permet par exemple de demander à l'utilisateur de la page des informations textuelles (par exemple son nom), un choix entre plusieurs options ou même de sélectionner un

fichier à envoyer.

### L'attribut `name`

Cet attribut sert à retrouver un objet (ici la balise `<input>`) afin de l'exploiter en JavaScript. Il est également utilisé lors de l'envoi du formulaire vers un serveur afin d'extraire les données saisies par l'utilisateur.

### L'attribut `id`

Comme l'attribut `name`, `id` sert à nommer un objet dans une page web. Cependant, contrairement à l'attribut `name`, tous les `id` doivent être uniques dans toute la page. Ils servent également à la balise <label> que nous présenterons plus loin.

### L'attribut `type`

Le rôle de la balise est déterminée par l'attribut `type`. Voici une liste des possibilités qui s'offrent à vous :

**text**
: C'est un champ de saisie de texte classique.

**password**
: Ce type permet d'entrer un mot de passe. Le texte saisi est remplacé par des étoiles (le caractère « * »). Attention, avoir un champ caché à l'utilisateur ne signifie pas que personne pourra le lire car le mot de passe est transmis « en clair » sur le réseau.

**checkbox**
: C'est une simple case à cocher permettant à l'utilisateur de valider ou non une option. Elle est cochée si son attribut "checked" existe.

**radio**
: C'est une case d'option. Elle n'est jamais utilisée seule car, en groupe, elle permet à l'utilisateur de choisir une option parmi plusieurs. Le groupe est identifié par l'attribut `name`.

**file**
: Ce type permet à l'utilisateur de choisir un fichier afin qu'il soit envoyé vers un serveur. Afin que le contenu du fichier soit envoyé au lieu du chemin seulement, le formulaire doit spécifier un encodage différent de celui par défaut, en utilisant l'attribut `enctype` et utiliser la méthode POST :

```
<form enctype="multipart/form-data" method="POST">
```

**submit**
: C'est un simple bouton permettant d'envoyer le formulaire.

**reset**
: C'est un bouton permettant de remettre le formulaire dans son état initial.

**image**

C'est un bouton contenant une image permettant d'envoyer le formulaire. L'URL de l'image est spécifiée par l'attribut `src`.

**hidden**
: C'est un champ caché qui permet à l'auteur du formulaire de faire passer des informations au serveur qui n'ont pas à être modifiées par l'utilisateur. Attention, même si l'utilisateur ne peux pas voir directement ces champs, leur valeur est aisément modifiable. Il ne faut donc JAMAIS faire confiance aux données venant de ces champs lors du traitement du formulaire.

**name**
: cet attribut permet d'insérer une variable dans le texte.

Si l'attribut `type` est omis, le type `text` est utilisé par défaut.

### L'attribut `value`

Cet attribut permet de spécifier la valeur par défaut d'un champ.

Exemple :

```
<form name="formulaire" method="post">
  <p>Un champ texte : <input type="text" name="nom" id="nom"> </p>
  <p>Un mot de passe : <input type="password" name="pass" id="pass"> </p>
  <p><input type="checkbox" name="choix_simple" id="choix_simple"> Une case à cocher
</p>
  <p><input type="radio" name="choix_multiple" value="choix1"> Premier choix </p>
  <p><input type="radio" name="choix_multiple" value="choix2"> Deuxième choix </p>
  <p><input type="radio" name="choix_multiple" value="choix3"> Troisième choix </p>
  <p>Un fichier à mettre sur le serveur :<input type="file" name="fichier"
id="fichier"> </p>
  <p>Un champ que l'utilisateur ne pourra pas modifier ni même voir : <input
type="hidden" name="champ_cache" id="champ_cache" value="coucou"> </p>
  <p><input type="submit" /><input type="reset" /></p>
</form>
```

### L'attribut `checked`

Les champs à sélectionner ayant cet attribut (qui s'utilise sans valeur) le sont par défaut.

### L'attribut `disabled`

Les champs ayant cet attribut (qui s'utilise sans valeur) ne sont jamais envoyé lors de la soummission[1].

## La balise `<label>`

L'habitude courante lors du développement d'un formulaire web consiste à placer le texte associé au champs d'entrée à côté de ces champs sans préciser explicitement qu'ils sont associés. Ce n'est pas un gros problème pour l'utilisateur moyen car en visualisant la page, il associe directement ces deux informations et remplit aisément le formulaire. Par

contre, pour un utilisateur ayant un handicap, cette association peut ne pas être aussi facile.

Il est donc recommandé d'utiliser un label c'est à dire la balise `<label>` permettant d'associer une légende à un champ d'entrée. De plus, un clic sur un label donne le focus au champ correspondant. Ainsi, tous les utilisateurs pourront utiliser votre formulaire sans problème.

### L'attribut `for`

Cet attribut sert à spécifier le nom du champ d'entrée dont le label est la légende.

### Exemple

```
<form name="formulaire" method="post">
  <p><label for="nom">Un champ texte : </label> <input type="text" name="nom"
id="nom"> </p>
  <p><input type="submit"></p>
</form>
```

## La balise `<fieldset>`

Cette balise sert à regrouper plusieurs champs qui ont un rapport entre eux comme par exemple un groupe de cases d'options. Le but de cette balise est le même que celui de `<label>` car elle permet d'indiquer explicitement la fonction d'un groupe de champs.

### La balise `<legend>`

On doit placer cette balise dans un bloc de `fieldset`. Elle permet de donner une légende au groupe de champs.

Exemple :

```
<form id="" action="" method="post">
 <fieldset>
  <legend>Voici une liste de cases à cocher</legend>
  <p>
    <input type="radio" name="choix_multiple" id="choix_multiple_1" value="choix1">
    <label for="choix_multiple_1">Premier choix </label>
  </p>
  <p>
    <input type="radio" name="choix_multiple" id="choix_multiple_2" value="choix2">
    <label for="choix_multiple_2">Deuxième choix </label>
  </p>
  <p>
    <input type="radio" name="choix_multiple" id="choix_multiple_3" value="choix3">
    <label for="choix_multiple_3">Troisième choix </label>
  </p>
 </fieldset>
</form>
```

# La balise <textarea>

Génère des zones de texte multilignes.

```
<TEXTAREA rows="nombre de lignes -1" name="commentaires">
```

Par exemple, pour créer une zone de saisie de six lignes :

```
<TEXTAREA rows="5" name="suggestions">
  Entrer ici vos suggestions
</TEXTAREA>
```

Pour faire en sorte que le texte affiché s'efface lorsque l'utilisateur remplit la zone :

```
<TEXTAREA rows="5" name="suggestions" placeholder="Entrer ici vos suggestions">
</TEXTAREA>
```

# La balise <select>

Cette balise affiche un menu déroulant cliquable. Exemple pour choisir une des trois propositions :

```
   <select name="Selection">
       <option value="1">Choix 1</option>
       <option value="2">Choix 2 SELECTED</option>
       <option value="3">Choix 3</option>
   </select>
```

Pour passer ce menu en lecture seule, ajoute l'attribut : `disabled="true"` (on ne peut alors plus voir les options non sélectionnées).

## multiple

L'attribut "multiple" d'une balise "select" indique que l'utilisateur peut effectuer un choix multiple :

```
   <select name="Selection2" multiple="multiple">
       <option value="1">Choix 1</option>
       <option value="2">Choix 2</option>
       <option value="3">Choix 3</option>
   </select>
```

# En savoir plus

- traduction en français de la spécification sur les formulaires (http://www.la-grange.net/w3c/html4.01/interact/forms.html)
- (anglais) version originale (http://www.w3.org/TR/html4/interact/forms.html)

# Cadres

Un cadre (*frame* en anglais) est une section de page web contenant elle-même une autre page web. La technique des cadres permet de faire évoluer plusieurs pages web simultanément. Par exemple, un cadre peut contenir le menu d'un site, et un autre son contenu (ce qui évite de devoir insérer un menu sur chaque page du site).

Le principal inconvénient des cadres est qu'ils brisent la sémantique des données, en particulier la gestion de l'historique des pages devient complexe. Pour cette raison, leur utilisation n'est plus encouragée.

## Définition d'un jeu de cadres : la balise `frameset`

### Les attributs `cols` et `rows`

Ces attributs servent à déterminer la disposition et les dimensions des cadres : `cols` pour les colonnes, `rows` pour les lignes (ou, plus précisément, une séparation verticale ou horizontale).

#### Imbrication de cadres

Comme vous l'aurez remarqué, la balise `frameset` ne peut diviser qu'à l'horizontale **ou** à la verticale. Pour combiner les deux, il est possible d'imbriquer les balises `frameset`. Ce résultat s'obtient en définissant un jeu de cadres (toujours avec la balise `frameset` donc) dans un cadre.

### La balise `noframe`

Certains navigateurs très anciens ne savent pas comment interpréter les cadres. La balise `noframe` permet d'indiquer à ces navigateurs comment produire la page web d'une manière alternative. Cette balise est ignorée par les navigateurs sachant interpréter les cadres.

## Définition d'un cadre : la balise `frame`

### L'attribut `src`

L'attribut `src` contient l'adresse (relative ou absolue) de la page à afficher dans le cadre.

### L'attribut `name`

L'attribut `name` permet de donner un nom à un cadre. Ceci permet par exemple d'identifier précisément quel cadre mettre à jour lorsque l'on suit un lien hypertexte.

### L'attribut `longdesc`

L'attribut `longdesc` s'adresse particulièrement aux non-voyants qui utilisent une interface vocale pour "lire" les pages web. Certains logiciels dédiés ont parfois des difficultés à rendre les contenus des cadres dans un ordre

pertinent. L'attribut permet d'effectuer un lien vers une description longue des cadres et de leur intérêt. Il s'agit généralement d'une page web séparée contenant un texte explicatif.

### Les marges : les attributs `marginwidth`, `marginheight` et `frameborder`

L'attribut `marginheight` accepte une valeur (pixels ou pourcentage) correspondant à une marge par rapport aux bords verticaux (supérieur et inférieur) du cadre. L'attribut `marginwidth` correspond aux marges horizontales (droite et gauche). L'attribut `frameborder` accepte les valeurs 1 ou 0 (ou bien TRUE et FALSE) et détermine si le cadre est délimité par une bordure ou non.

### Les bordures : les attributs `border`, `bordercolor` et `frameborder`

### Les attributs `noresize` et `scrolling`

`noresize` détermine si l'internaute peut ou non changer la taille du cadre.

`scrolling` détermine la présence de barres de défilement (toujours, automatique ou jamais).

## Cadres uniques : la balise `iframe`

Iframes ou fenêtres intégrées dans une page Web

L'intérêt est d'ajouter dans une page propre, une autre page. Cette méthode est utilisée entre autre pour afficher des bannières de publicité.

### Les attributs d'`iframe`

Exemple de code pour la balise Iframe :

```
<iframe name="lesteph" src="http://www.lesteph.ch/lesteph/tutoriaux/tuto-
2003/webmaster/tutoriaux/html/html-frames.html" width="468" height="60"
marginwidth="0" marginheight="0" hspace="0" vspace="0" frameborder="0" scrolling="no">
</iframe>
```

## Les valeurs de l'attribut `target`

`target` est un attribut de la balise `a`, la balise servant à inclure des liens hypertexte. Il détermine la *cible* (signification de "target" en anglais) du lien, c'est-à-dire où ce lien doit être ouvert, en se basant sur l'attribut `name` des cadres présents.

`target` peut prendre différentes valeurs :

**`nom_du_cadre`**
: ouvre le lien dans le cadre dont l'attribut `name` a la valeur

"nom_du_cadre".

**_self**
: le lien s'ouvre dans le même cadre.

**_parent**
: le lien s'ouvre dans le cadre parent du cadre de la page. C'est à dire à la place de la page contenant le `<frameset>` qui détermine le cadre de la page ou se trouve le lien.

**_top**
: le lien s'ouvre dans le fenêtre courante (annule donc tous les cadres).

**_blank**
: ouvre le lien dans une nouvelle fenêtre.

*Note* : Certains navigateurs (Mozilla Firefox, Konqueror, Opera, ...) proposent une navigation par onglets, permettant d'afficher plusieurs pages dans une seule fenêtre. Bien que ce type de navigation soit de plus en plus courant et de plus en plus utilisé, il n'existe pas de valeurs à attribuer à `target` permettant d'ouvrir un lien dans un nouvel onglet (ceci est notamment dû au fait que ce type de navigation n'était pas encore très courante quand HTML 4 est sorti - les onglets ne sont donc pas un standard, mais bien une fonctionnalité supplémentaire proposée par certains navigateurs). Cependant, certains navigateurs, éventuellement munis d'une extension, peuvent proposer d'ouvrir tous les liens ayant **_blank** comme valeur de l'attribut `target` dans un nouvel onglet, à voir dans les options de votre navigateur.

# Multimédia

Dans une page, nous pouvons insérer différents contenus autres que du textes et des images. Il peut s'agir de sons, de vidéos, d'animations (Adobe Flash ou SMIL) ou d'appliquettes *(applets)*.

---

**Note**

Consultez la section *L'attribut `src`* du chapitre *Image* concernant l'utilisation du contenu d'un autre site.

---

## Balise `object`

Pour intégrer un objet, nous utilisons la balise `<object>`. Comme toutes les balises, on peut lui définir les attributs `id`, `class`, `title`, `style`, `dir`, `lang` et `xml:lang`. En outre, cette balise dispose d'attributs spécifiques :

**`data`**
: localisation de l'objet à insérer (adresse réticulaire), c'est l'équivalent du `src` de l'élément `<img>` ;

**`type`**
: le type MIME du contenu ;

**`width`**
: largeur de l'affichage ;

**`height`**
: hauteur de l'affichage ;

**`standby`**
: est l'équivalent du `alt` pour `<img>`. Ce texte est affiché pendant le chargement.

On peut, à l'intérieur de la balise `object`, donner du code qui s'affichera si le contenu ne peut pas être affiché. On peut également donner quelques paramètres à l'objet à l'aide de la balise `<param />` qui associe à chaque paramètre une valeur booléenne. Voici les paramètres disponibles :

**`loop`**
: indique si le média doit être jouer en boucle ou seulement une fois ;

**`controller`**
: indique si il faut faire apparaître des boutons de contrôles ;

**`autoplay`**
: indique si il faut lancer la vidéo au chargement de la page ; **attention** : à éviter à tout prix si la page contient plusieurs médias.

### Insertion d'une vidéo

```
<object data="video.mpeg" title="une vidéo d'exemple" type="video/mpeg" standby="Vidéo
```

```
d'exemple">
  <param name="loop" value="false" />
  <param name="autoplay" value="false" />
  <param name="controller" value="true" />
  Impossible de lire la vidéo : <a href="video.mpeg">Téléchargez-la</a>
</object>
```

### Insertion d'une animation Flash

```
<object data="flash.swf" title="animation d'exemple" type="application/x-shockwave-
flash" standby="animation d'exemple">
  Impossible de lire l'animation Flash
</object>
```

### Insertion d'une appliquette Java

## Obsolète : balise `embed`

Avant la balise `object`, on utilisait la balise `embed`. Cette balise avait été créée par Netscape. Elle ne fait pas partie de la norme W3C et donc ne devrait pas figurer sur des sites ; cependant, les navigateurs l'interprètent en général correctement, on peut donc encore la trouver sur des sites. De ce fait, il peut être utile d'en connaître la syntaxe pour convertir de l'ancien code.

### Exemple d'utilisation simple

```
<embed src="foobar.wav" />
```

où *foobar*.wav est le nom du fichier son.

Les attributs de cette balise sont :

**`src`**
: l'emplacement (URL) de l'objet

**`alt`**
: texte de remplacement à afficher

**`height`**
: hauteur de l'affichage

**`width`**
: largeur de l'affichage

**`name`**
: nom de l'objet

# Internationalisation

L'Internet est un réseau mondial *(world wide web)*. Le rendu du contenu peut donc varier selon la configuration du navigateur. Il peut être intéressant d'indiquer la langue utilisée, ce qui permettra au navigateur d'appliquer les paramètres régionaux relatifs à cette langue (s'il le peut).

L'identification des langues est définie dans le document RFC 3066 *Tags for the Identification of Languages*.

## Comment indique-t-on la langue utilisée ?

La langue utilisée dans la majeure partie du document est un paramètre de la balise `<html>` d'ouverture :

```
<html lang="…">
```

Si on change de langue au cours du document, il faut l'indiquer dans la balise qui encadre le texte concerné. Par exemple

```
…
<html lang="fr" >

<head>
   …
</head>

<body>

<p>
   Le texte est en français par défaut.
   Mais comme on dit en italien :
   <q lang="it">Tradutore, traditore</q>.
</p>

</body>

</html>
```

En XHTML, il faut indiquer en outre la langue selon la norme XML, avec l'attribut : `xml:lang="…"`. L'exemple ci-dessus devient donc :

```
…
<html lang="fr" xml:lang="fr" >

<head>
   …
</head>

<body>

<p>
```

```
  Le texte est en français par défaut.
  Mais comme on dit en italien :
  <q lang="it" xml:lang="it">Tradutore, traditore</q>.
</p>

</body>

</html>
```

Si le texte en langue étrangère n'est pas inclus dans une balise particulière, on peut simplement utiliser la balise `<span>…</span>` (voir le chapitre *Zones de mise en forme*).

---

**Note**

L'en-tête peut contenir une indication de la « langue de l'audience » avec la balise `<meta http-equiv="content-language" content="langue">`, en utilisant le même codage que ci-dessus (par exemple `<meta http-equiv="content-language" content="fr">` pour le français). Cette balise sert pour le référencement par des moteurs de recherche, mais est indépendant de l'indication de la langue d'un élément de texte.

---

# Identificateur de la langue

De manière générale, la langue est identifiée par deux lettre selon la norme ISO 639-2 (dite « alpha-3 »), par exemple :

- allemand : `de`
- anglais : `en`
- arabe : `ar`
- chinois : `zh`
- espagnol : `es`
- français : `fr`
- grec moderne : `el`
- hébreu : `he`
- italien : `it`
- japonais : `ja`
- néerlandais : `nl`
- portugais : `pt`
- russe : `ru`

Certaines langues sont identifiées par un code de trois lettres, par exemple :

- ancien anglais (entre 450 et 1100) : `ang`
- moyen anglais (1100–1500) : `enm`
- berbère : `ber`
- créole/pidgin issu de l'anglais : `cpe`
- créole issu du français : `cpf`
- créole issu du portuguais : `cpp`

- égyptien antique : `egy`
- ancien français (842–1400) : `fro`
- français moyen (1400–1600) : `frm`
- grec ancien (jusqu'en 1453) : `grc`
- kabyle : `kab`
- massaï : `mas`

Lorsqu'un code à 3 lettres et un code à 2 lettres sont simultanément disponibles pour une même langue, seul le code à deux lettres doit être utilisé dans l'attribut `lang`.

On peut indiquer une sous-catégorie de langue. Par exemple :

- anglais : `en`
  - anglais du Royaume Uni : `en-GB`
    - anglais du Royaume Uni, selon l'orthographe du dictionnaire d'Oxford *(Oxford English Dictionary)* : `en-GB-oed`
  - anglais étatsunien : `en-US`
  - anglais cockney : `en-cockney`
- espagnol : `es`
  - espagnol d'Espagne : `es-ES`
  - espagnol de l'Amérique latine : `es-419`

Les codes régionaux ne doivent cependant être utilisés que lorsqu'ils sont explicitement nécessaires. Par exemple, on n'écrira pas `es-ES` sauf s'il est nécessaire de différencier dans la même page des textes en espagnol courant d'autres passages en `es-419`.

# Direction de l'écriture

On peut préciser si l'écriture se fait de gauche à droite (*left-to-right*, LTR) ou de droite à gauche (*right-to-left*, RTL), avec le paramètre `dir`, par exemple

```
<p lang="he" dir="rtl">עברית</p>
```

La valeur par défaut est `ltr` ; si une langue s'écrit de gauche à droite, il est inutile de le préciser.

# Exemple de feuille de style différenciant la langue

L'exemple suivant est une feuille de style (écrite en CSS) dans laquelle la mise en page d'un paragraphe dépend de la langue.

```
/* ********** Français ********** */
   [lang|="fr"] > * { quotes: '\00ab\00a0' '\00a0\00bb'; } /* guillemets de citation
*/
   p:lang(fr) {
      text-align: justify; /* texte justifié */
      text-indent: 1em;    /* alinéa rentrant */
```

```
      margin-top: 0;       /* pas d'interligne */
      margin-bottom: 0;
   }
/* ********** Anglais ********** */
   [lang|="en"] > * { quotes: '\2018' '\2019' '\201c' '\201d'; }    /* guillemets de
citation, 2 niveaux de citation  */
   [lang|="en-US"] > * { quotes: '\201c' '\201d' '\2018' '\2019'; } /* différents en
GB et aux US  */
   p:lang(en) {
      text-align:left;        /* texte aligné à gauche */
      text-indent: 0;         /* pas de retrait de paragraphe */
      margin-top: 1em;
      margin-bottom: 1em;     /* alinéa "pavé" (interligne) */
   }
```

Les guillemets étant des caractères non-ASCII, on les mentionne en CSS en indiquant le code Unicode en hexadécimal après une barre de fraction (U+00ab est le guillemet ouvrant français «, U+00a0 est l'espace insécable…). Si on définit la page de code utilisée dans la feuille de style, on peut utiliser des caractères non-ASCII, mais la solution avec les barres de fraction inversée est universelle.

Vous pouvez tester le résultat avec des paragraphes suffisamment longs.

## Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >

<html lang="fr" xml:lang="fr">

<head>

   <title>
      Variation de mise en forme selon la langue
   </title>

   <meta
      http-equiv="Content-Type"
      content="text/html; charset=ISO-8859-15" />

   <style type="text/css">
      /* ********** Français ********** */
      [lang|="fr"] > * { quotes: "«\00a0" "\00a0\00bb"; } /* guillemets de citation */
      p:lang(fr) {
         text-align: justify; /* texte justifié */
         text-indent: 1em;    /* alinéa rentrant */
         margin-top: 0;       /* pas d'interligne */
         margin-bottom: 0;
      }
      /* ********** Anglais ********** */
      [lang|="en"] > * { quotes: "‘" "’" "“" "”"; }    /* guillemets de citation, 2
niveaux de citation  */
      [lang|="en-US"] > * { quotes: "“" "”" "‘" "’"; } /* différents en GB et aux US
*/
      p:lang(en) {
         text-align:left;        /* texte aligné à gauche */
```

```
        text-indent: 0;         /* pas de retrait de paragraphe */
        margin-top: 1em;
        margin-bottom: 1em;     /* alinéa "pavé" (interligne) */
     }
  </style>

</head>

<body>

<p>
  Ceci est un paragraphe en français
  <q>avec une citation et
  <q>une citation dans la citation</q></q>…
</p>

<p>
  Notez l'alinéa rentrant
  et l'absence de saut de ligne entre les paragraphes…
</p>

<p lang="en" xml:lang="en">
  This is a paragraph in english
  <q>with a quotation and
  <q>a quotation in the quotation</q></q>…
</p>

<p lang="en-US" xml:lang="en-US">
  This is a paragraph in american english
  <q>with a quotation and
  <q>a quotation in the quotation</q></q>…
</p>

</body>

</html>
```

Lorsque vous affichez le résultat de ce code, diminuez la largeur de la fenêtre de votre navigateur afin que chaque paragraphe s'affiche au moins sur deux lignes, pour pouvoir apprécier les différences de mise en page (justifié/aligné à gauche, indentation/saut de ligne).

Vous devriez obtenir à peu près le résultat suivant, si le navigateur interprète bien la feuille de style.

Ceci est un paragraphe en français « avec une citation et « une citation dans la citation » »…

Notez l'alinéa rentrant et l'absence de saut de ligne entre les paragraphes…

This is a paragraph in english ‘with a quotation and “a quotation in the quotation”’…

This is a paragraph in american english “with a quotation and ‘a quotation in the quotation’”…

---

**Note**

Comme on a défini la page de code dans l'en-tête du fichier HTML, on peut utiliser les guillemets « en clair » (caractères non-ASCII) dans la feuille de style intégrée.

---

## Décryptage de la feuile de style

Le CSS possède son propre wikilivre. Toutefois, afin que cela ne soit pas juste du copier-coller et que vous puissiez modifier l'exemple sans lire tout le wikilivre *Programmation CSS*, voici quelques éléments.

Dans l'expression « `[lang|="en"] > * {…}` » :

- `[lang|="en"]` signifie « les éléments dont l'attribut est « `lang="en"` » » ;
- `> *` désigne tous les éléments (*) situés en-dessous (>) de l'élément pré-cité, donc tous les sous-éléments de l'élément ayant pour attribut « `lang="en"` ».

L'expression « `[lang|="en"]` » désigne aussi les éléments ayant les attributs dérivés de `en`, notamment `en-GB` et `en-US` (mais ce cas-là est redéfini après).

L'expression « `p:lang(en)` » désigne les paragraphes (éléments introduits par la balise `<p …>`) dont l'attribut `lang` est `"en"`.

La propriété `quote` indique les guillemets utilisés, sous la forme « `"guillemet ouvrant" "guillemet fermant"` ». Lorsqu'il y a deux paires de délimiteurs, la première paire correspond aux citations de premier niveau, la seconde paire aux citations de deuxième niveau (citation dans la citation).

Les autres éléments sont assez explicite en anglais.

> *Pour plus de détails voir :* ***Programmation CSS/Les sélecteurs#Autres pseudo-classes***.

# Voir aussi

- Internationalisation de logiciel
- RFC 4646
- IANA: Subtag registry (http://www.iana.org/assignments/language-subtag-registry)
- Yoyo design (http://www.yoyodesign.org/)
  - Gestion des guillemets (http://www.yoyodesign.org/doc/w3c/css2/generate.html#propdef-quotes)
  - La pseudo-classe de langue (http://www.yoyodesign.org/doc/w3c/css2/selector.html#lang)

# Balises complémentaires

## Éléments de phrase

Les éléments de phrase ont été mentionnés dans le chapitre *Style de texte*. Il existe d'autres éléments « exotiques », dont la définition en HTML 4.01 est très vague et qui ne sont en pratique pas utilisés (mais on peut toujours les associer à une mise en forme particulière avec du CSS).

Éléments de phrase « exotiques »

| Style | Balise | Mnémotechnique | Rendu par défaut |
|---|---|---|---|
| terme faisant l'objet d'une définition | `<dfn>...</dfn>` | ... | italiques |
| extrait de programme | `<samp>...</samp>` | *sample* (échantillon) | *idem* `<code>` |
| variable ou paramètre d'un programme | `<var>...</var>` | ... | |

## Les balises multimédia

## Les scripts

La balise `<script>` permet d'insérer un script dans un langage spécifié par l'attribut `language`. Le type est également spécifié par l'attribut `type`.

Cette balise peut être placée dans la partie `<head>` ou `<body>` de la page HTML.

Exemple :

```
<script language="Javascript1.2" type="text/javascript">

var resultat = "aucun";

function clic_bouton()
{
  alert("Résultat = " + resultat);
}
</script>
```

Le langage du script peut être quelconque (VBScript, Javascript, ...), mais seul le Javascript est reconnu par la plupart des navigateurs.

### Événements

Le script peut définir des fonctions appelées plus tard, lorsqu'un évènement particulier survient (clic d'un bouton par

exemple).

Les différentes balises HTML, en particulier les champs de formulaire, permettent de faire appel à une fonction javascript pour traiter certains évènements. L'appel se fait en ajoutant un attribut dont le nom commence par "`on`" suivi du nom de l'évènement. La valeur de l'attribut est un code javascript (souvent un appel de fonction), qui doit retourner une valeur booléenne. Si le code retourne faux (`false`), l'action par défaut associée à l'évènement est annulée.

Exemple :

```
<script language="Javascript1.2" type="text/javascript">
function envoi_formulaire()
{
  return confirm("Voulez-vous réellement envoyer les données au serveur ?");
}
</script>
<form action="/envoyer.cgi" onsubmit="return envoi_formulaire();">
<label>
<input type="text" name="nom"></input>
<br />
<input type="submit" value="Envoyer"></input>
</form>
```

### Script dans un lien

Un lien dont l'adresse commence par "`javascript:`" est en fait un code javascript. En général, il s'agit d'un appel de fonction.

Exemple :

```
<a href="javascript:clic_bouton()">Appel à la fonction</a>
```

## Les applets

Une applet Java est une application dont le code est sur le serveur. Il est téléchargé puis exécuté sur le navigateur de l'utilisateur, où il s'affiche dans une zone de la page HTML, définie par la balise `<applet>`.

Cette balise possède les attributs suivants :

**`code`**
: (obligatoire) Nom de la classe principale de l'applet Java.

**`codebase`**
: (optionnel) URL du répertoire où trouver la classe, si elle ne se trouve pas dans le même répertoire que la page HTML.

**`archive`**
: (optionnel) Liste des archives JAR contenant les classes de l'applet.

**`width`**
: (obligatoire) Largeur de la zone d'affichage de l'applet.

**height**
: (obligatoire) Hauteur de la zone d'affichage de l'applet.

**alt**
: (recommandé) Texte alternatif quand les applets ne sont pas autorisées par le navigateur.

Cette balise peut contenir des paramètres à transmettre à l'applet, en utilisant la balise `<param>`. Cette balise a pour attributs `name` pour le nom du paramètre et `value` pour sa valeur.

Exemple :

```
<applet code="Graph.class" width="500" height="300" alt="Graphique du nombre de
fichiers téléchargés dans les 5 derniers jours">
<param name="type" value="bar2D" />
<param name="data" value="0, 10, 15, 7, 16" />
<param name="labels" value="Lun, Mar, Mer, Jeu, Ven" />
</applet>
```

---

### Note sur la sécurité des applets

Étant donné que le code s'exécute sur le navigateur de l'utilisateur, un programme malveillant pourrait faire n'importe quoi. Cependant la machine virtuelle Java n'autorise pas les applets à faire tout ce qu'elles veulent, à moins que l'utilisateur ne lui accorde plus de droits où que l'applet soit signée et acceptée par l'utilisateur. Les navigateurs permettent également d'empêcher toutes les applets de s'exécuter.

---

# Zones de mise en forme

On peut définir une zone sur laquelle on applique une mise en forme particulière. Cette notion est donc liée au CSS, bien que l'on puisse l'utiliser en HTML « pur ».

Il faut s'attacher à la notion d'imbrication de zones : le formattage d'une zone se superpose au formattage environnant, ou éventuellement s'y substitue, selon des règles de priorité (cf. la section *Priorité des règles* du wikilivre sur le CSS).

Par exemple, si l'environnement indique que le texte est centré et le formattage de la zone indique qu'il est en gras, alors on aura du texte centré et gras. Mais si le formattage de la zone indique que le texte est aligné à droite, alors cela annule la consigne générale.

---

**Note**

Cette notion de zonage est fréquemment confondue avec la notion de calque *(layer)*, qui était une notion spécifique à Netscape (introduit en 1997 dans sa version 4.0). Cette confusion est entretenue par certains logiciels de création [1] (http://css.alsacreations.com/Bases-et-indispensables/Quelle-est-la-difference-entre-un-div-et-un-calque). La notion de « calque » n'est pertinente que si l'on utilise l'attribut `z-index` (superposition de couches).

---

## Balises universelles

Toute balise est une zone de mise en forme. En effet, on peut associer un style à toute balise,

- soit en écrivant le style dans la balise, par exemple
  `<q style="font-style: italic;">…</q>` ;
- soit en faisant référence à un style défini dans l'en-tête ou dans un fichier `.css` lié, par exemple
  `<h1 id="menu">…</h1>`
  — ou —
  `<p class="remarque">…</p>`

On a alors une mise en page liée à la fonction de la portion de page.

Il existe de plus, deux balises spécifiques pour créer une zone :

- `<span>…</span>` : balise en-ligne *(inline)*, donc une balise dans un paragraphe ou plutôt dans une balise dite « bloc » ; elle s'utilise au sein d'un bloc pour une modification ponctuelle ;
- `<div>…</div>` : balise de bloc *(block)*, un bloc est créé, il y a un retour à la ligne automatique.

Notons qu'appliquer `span` ou `div` sans attribut à un texte ne change rien : leur intérêt réside dans les paramètres que contiennent ces balises.

On peut ainsi définir une mise en forme sur une portion de page sans qu'elle ait une fonction particulière.

Notons que ces balises sont des « éléments de regroupement » qui permettent d'appliquer un traitement à une zone de manière générale, pas nécessairement une feuille de style.

---

**Note**

Les éléments `span` et `div` ne se substituent pas aux éléments décrivant le contenu. Si le rendu d'un élément ne vous convient pas, il faut le modifier en faisant référence à une classe dans la balise ouvrante ou en y intégrant du code CSS. Mais il serait malvenu de remplacer la balise par un `span` ou `div` qui, eux, ne donnent aucun sens au contenu.

---

**Attention !**

On peut imbriquer des balises en-ligne dans des balises blocs, mais pas l'inverse[2].

# La balise div

## Utilisation

On va entourer la partie souhaitée des balises `<div>` et `</div>` pour délimiter le bloc auquel vont se rapporter les effets de style.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Titre</title>
       <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   </head>
   <body>
     <div>
       <p>Texte</p>
     </div>
   </body>
</html>
```

## Mise en lien avec le style

En effet, un `div` est associé à des informations de style en CSS, qui peuvent être implantées en haut de la page HTML dans un espace entre les balises `<head>` et `</head>` ainsi :

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Titre</title>
       <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
       <style type="text/css">
          ''code CSS''
       </style>
   </head>
   <body>
     <div ''référence au formattage''>
       <p>Texte</p>
     <div>
   </body>
</html>
```

Ou bien dans un fichier séparé, que l'on lie au fichier HTML en remplaçant `<style type="text/css"> </style>` par :

```
<link rel="stylesheet" media="screen" type="text/css" title="Titre"
href="feuilledestyle.css" />
```

Ou encore directement dans la balise `<div>` ainsi :

```
<div style=" ... ''code CSS'' ... ">
  <p>texte</p>
</div>
```

## Caractérisation

Il est conseillé d'opter pour une solution avec feuille de style externe. Mais comment fait-on alors pour personnaliser chaque `<div>` ? Il suffit d'ajouter dans la balise `<div>` une information qui permettra de la caractériser, un `class` ou un `id`. On précise qu'un `id` est à usage unique (il ne peut être utilisé qu'une fois par page), pas le `class` (`id` : identité unique, `class` : classe de style ou multiplicité). L'utilisation de ces balises se fait ainsi :

Utilisation de `class` et de `id`

| **Avec `class`** | **Avec `id`** |
|---|---|
| Code CSS dans l'en-tête ou le fichier lié | |
| `.ma_classe`<br>`{`<br>`  propriété 1 : valeur 1;`<br>`  propriété 2 : valeur 2;`<br>`  ...`<br>`}` | `#mon_id`<br>`{`<br>`  propriété 1 : valeur 1;`<br>`  propriété 2 : valeur 2;`<br>`  ...`<br>`}` |
| Code HTML | |

```
...
   <body>
     <div class="ma_classe">
       <p>Texte</p>
     </div>
   </body>
</html>
```

```
...
   <body>
     <div id="mon_id">
       <p>Texte</p>
     </div>
   </body>
</html>
```

# Fonctions

## Position

La balise `<div>` peut être placée comme on le souhaite grâce au CSS, jugez plutôt l'aspect CSS que l'on peut obtenir : `visibility: visible; position: absolute; top: 20px; left: 20px; width: 100px; height: 100px`. De plus le dernier cadre déclaré recouvre celui d'avant si les deux se superposent.

## Clarification

Lorsque l'on travaille avec un bloc à part avec un `<div>`, on peut définir avec le CSS les différentes balises usuelles dans le `<div>` choisi. D'une manière plus simple il y a une cohérence entre forme et fond.

# Mise en application

Voici un exemple de styles de mise en forme avec un menu, un centre de page et un bas de page. La page se compose de trois calques `<div>` les uns derrière les autres. Les blocs sont modulables : peu importe l'ordre dans lequel on les écrit, on obtient le même rendu final.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >

  <head>
     <title>Titre</title>
     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
     <style type="text/css">
        .un
        {
           position: absolute;
           top: 20px; left: 20px; width: 100px; height: 100px
        }
        .deux
        {
           position: absolute;
           top: 20px; left: 50%; width: 100px; height: 200px
        }
        .trois
        {
```

```
            position: absolute;
            bottom: 20px; left: 20px; width: 100px; height: 20px
         }
      </style>
  </head>

  <body>
     <div class="un">
        <p>menu</p>
     </div>
     <div class="deux">
        <p>centre</p>
     </div>
     <div class="trois">
        <p>bas de page</p>
     </div>
   </body>
</html>
```

# Voir aussi

- Chapitre *Interface HTML* du wikilivre *Programmation CSS*

# XML et XHTML

Le balisage HTML autorise une souplesse de syntaxe :

- ouvrir une balise sans la fermer,
- donner la valeur d'un attribut entre guillemets, apostrophes ou sans délimiteurs (un seul mot),
- ...

Ces problèmes sont gérés par l'interpréteur du navigateur. La gestion de ces erreurs de syntaxe dépend donc du navigateur.

La syntaxe du XML n'autorise pas les erreurs ; elles sont signalées par le navigateur. Un fichier utilisant la syntaxe XML avec des balises HTML utilise en fait le format XHTML.

Avec les navigateurs supportant le format XML, il est possible d'écrire des pages en XML sans utiliser de balises HTML (exemple : <paragraph>, <section>, ...), et d'utiliser une feuille de transformation XSL pour convertir le fichier XML en page HTML.

Certaines balises utilisées en HTML font directement de la mise en forme, ou font un lien vers une feuille de style CSS particulière :

```
<div class=noprint>
Le contenu de ce tableau ne sera pas affiché à l'impression.
</div>
```

# HTML5

La prochaine version de HTML sera la version 5. Elle n'est pas encore finalisée mais rien ne vous empêche de l'utiliser dans vos pages, surtout qu'elle apporte plusieurs nouveautés intéressantes.

## Le doctype

En HTML5, le doctype a été réduit au strict minimum. Cette version du langage ne nécessite plus de versions du DOCTYPE comme en HTML 4.0 et XHTML 1.0.

```
<!DOCTYPE html>
```

## La syntaxe

En HTML5, vous avez le choix entre utiliser une syntaxe SGML (comme HTML 4) ou XML (comme XHTML). Prenez juste garde à être cohérent dans votre choix.

De plus, les balises <style> et <script> n'ont plus besoin systématiquement de l'attribut `type`. S'il n'est pas précisé, la balise <style> accepte le css par défaut, tandis que la balise <script>, elle, accepte le javascript.

## data

L'attribut "data-"[3] permet de renseigner des données dans les balises HTML, séparées par "-", mais on peut aussi leur affecter des valeurs avec "=".

Exemple :

```
<div id="introduction" data-introduction data-paragraphe="introduction">
...
</div>
```

> **Attention !**
>
> Le fait de sectionner des data en JavaScript est plus lent que de sélectionner des classes, lui-même plus lent que d'utiliser les id[4].

## Balises obsolètes

, <font>, <strike>, <tt>.

## Nouvelles balises

<article>, <aside>, <footer>, <header>, <nav>, <section>.

## <canvas>

La balise <canvas> permet de créer une zone de dessin. On utilise ensuite du JavaScript pour créer/animer celle-ci.

Exemple :

```
<canvas id="myCanvas" width="300" height="200" />
<script>
var canvas=document.getElementById("myCanvas");
var context=canvas.getContext("2d");
context.strokeStyle="#900";
context.strokeRect(20,30,100,50);
</script>
```

Ce code affichera : 

Cet exemple est basique, mais canvas devrait permettre de créer des animations qui n'étaient avant possibles qu'avec Flash.

*Voir aussi*

Programmation objet et géométrie#Géométrie avec html5

## <audio>

La balise <audio> permet d'afficher un lecteur audio. L'avantage par rapport aux balises <object> et <embed> est que le fichier est lu directement par le navigateur, il n'y a plus besoin de plugin.

Elle s'utilise comme ceci :

```
<audio src="monfichier.mp3" autoplay controls loop />
```

Les attributs `autoplay`, `controls` et `loop` sont facultatifs. Ils permettent respectivement de lancer automatiquement la lecture, d'afficher les boutons lecture/arrêt et de lire la piste en boucle.

Vous pouvez aussi spécifier plusieurs sources, dans différents formats, comme ceci :

```
<audio>
<source src="monfichier.mp3" />
<source src="monfichier.ogg" />
</audio>
```

Le navigateur choisira alors le format qui lui convient le mieux.

### <video>

La balise <video> s'utilise comme la balise <audio>, sauf qu'il faut préciser les dimensions de la vidéo, ainsi que le type MIME du fichier :

```
<video width="360" height="240">
<source src="movie.ogv" type="video/ogg" />
<source src="movie.mp4" type="video/mp4" />
</video>
```

# Faire un livre dont vous êtes le héros

## Idée maîtresse

L'arborescence classique est :

- ./

./scripts
./styles
./assets <-- c'est ici que viennent les images et les sons
./assets/images

## Source

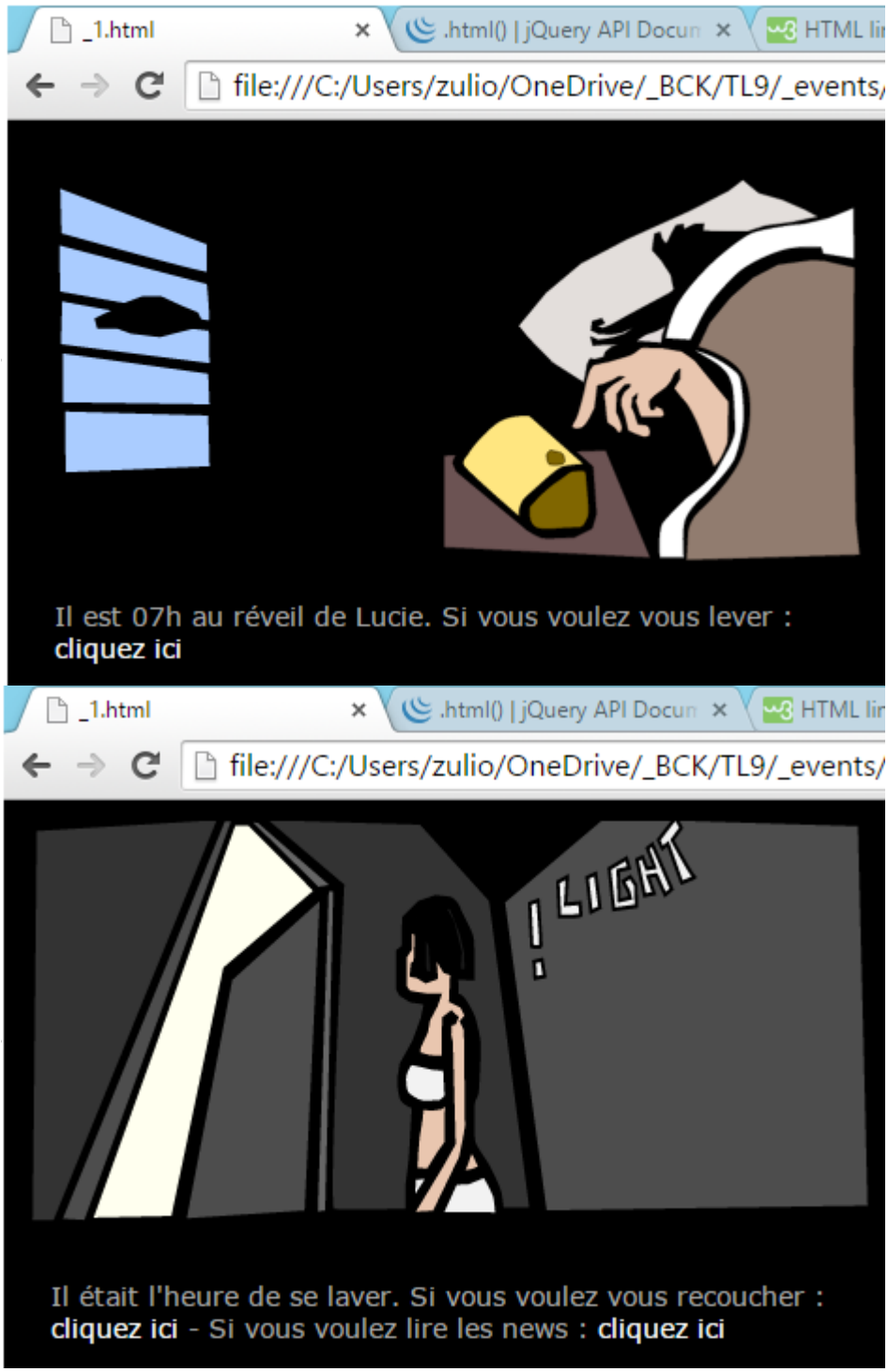
```
<!doctype html5>
<html>
    <!--
        LDVELH(Livre dont vous êtes le héros) -> engine amicaFolio
        +-------------------------------------
        | Cet engin est un single page application
        | cette version de l'engin ne permet de changer qu'un image à la fois par page
        +-----------------------------------------------
        synopsis : j'ai fait cet engin pour une amie (\).(\)
        version : 0.1
        authors : zulad//
    -->
    <head>
        <meta charset="UTF-8" />
        <script src="scripts/jquery-1.11.3.js" ></script>
        <link rel="stylesheet" type="text/css" href="styles/style.css" > </link>
    </head>

    <body onload="fillPage(1);" >

        <div id="main" />

    </body>
```

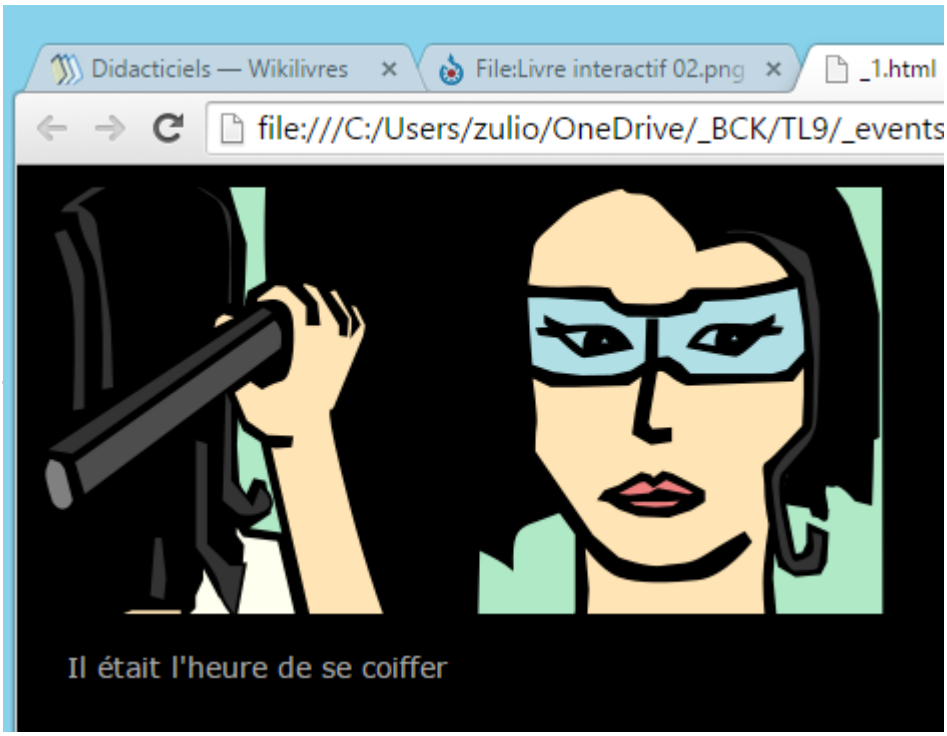
```
</html>
<script>

// Les données peuvent être modifiées en fonction de l'histoire
    var datas = {

        _cpt:1,
        _1:{img:"0_start.svg", txt:"Il est 07h au réveil de Lucie. Si vous voulez vous
lever : {{cliquez ici|2}} "},
        _2:{img:"0_bed.svg", txt:"Il était l'heure de se lever. Si vous voulez vous
laver : {{cliquez ici|3}} - Si vous voulez vous recoucher : {{cliquez ici|1}}"},
        _3:{img:"2_sdb.svg", txt:"Il était l'heure de se laver. Si vous voulez vous
recoucher : {{cliquez ici|1}} - Si vous voulez lire les news : {{cliquez ici|4}}"},
        _4:{img:"3_miroir.svg", txt:"Il était l'heure de regarder les news. Si vous
voulez vous coiffer : {{cliquez ici|5}}"},
        _5:{img:"4_miroir.svg", txt:"Il était l'heure de se coiffer"}

    };

// Ceci est le noyau de l'application. Il est préférable de ne pas le modifier
    function getMatches(string, regex, index) {
      index || (index = 1); // default to the first capturing group
      var matches = [];
      var match;
      while (match = regex.exec(string)) {
        matches.push(match[index]);
      }
      return matches;
    }

    function fillText(str) {
        var re = new RegExp("(\{\{)([ a-zA-Z0-9|]*)(\}\})", "gm");
        matches = str.match(re);

        var matches = getMatches(str, re, 2);

        var lnk = Array();
        for (m in matches)
        {
            lnk[m] = matches[m].split("|");
            str = str.replace(matches[m], "<span class='click'
onclick='fillPage("+lnk[m][1]+")'>"+lnk[m][0]+"</span>");
            str = str.replace("{{","").replace("}}","");
        }

        return str;
    }

    function fillPage(page) {
        if(!page) page = 1;
        var content_1 = '<table id="tbl" width="480"> \
                <tr><td width="10"><img id="imgTrg" src="assets/images
/'+eval('datas._'+page+'.img')+'" onclick="fillPage(target);" /></td></tr> \
                <tr><td id="txt">'+fillText(eval('datas._'+page+'.txt'))+'<td/></tr>
\
            </table>';  $("#main").html(content_1); eval(datas._cpt = page);
    }
```

```
</script>
```

## Résultat

Le résultat HTML peut être porté sur tablette ou smartphone. J'ai prévu les screenshots suivant pour un petit web serial sur smartphone.






## Références

1. https://www.w3schools.com/tags/att_input_disabled.asp
2. http://www.css-faciles.com/bloc-en-ligne.php
3. http://www.w3schools.com/tags/att_global_data.asp
4. http://jsperf.com/id-vs-class-vs-data-attribute

# Conclusion

Le langage de balisage HTML permet de structurer un document et de le mettre en forme. Cependant, ce langage utilisé intensément par les sites internet tend à devenir exclusivement un langage de structuration en se rapprochant du format XML, laissant la mise en forme aux feuilles de styles en cascade (CSS), et les animations au JavaScript.

Afin de protéger le code et lui permettre une plus grande interactivité entre l'utilisateur et les bases de données, le PHP est de plus en plus employé avec l'HTML.



---