LE LANGAGE HTML

TODO :

-

v1.2.1.0 – 07/05/2010
peignotc(at)arqendra(dot)net / peignotc(at)gmail(dot)com

(cc) BY-NC-SA Toute reproduction partielle ou intégrale autorisée selon les termes de la licence Creative Commons (CC) BY-NC-SA : Contrat Paternité-Pas d'Utilisation Commerciale-Partage des Conditions Initiales à l'Identique 2.0 France, disponible en ligne http://creativecommons.org/licenses/by-nc-sa/2.0/fr/ ou par courrier postal à Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA. *Merci de citer et prévenir l'auteur*.

TABLE DES MATIÈRES

TABLE DES ANNEXES

TABLE DES ILLUSTRATIONS

0 PROLOGUE

Veuillez noter que le contenu technique du présent document date du 2nd semestre 2004 [1]. L'essor fulgurant de la planète Internet ainsi que l'avancée inéluctable des technologies informatiques rendent une partie de ces informations obsolètes. Les éléments décrits demeurent fonctionnels, mais certains sont déconseillés dans le cadre d'activités à caractère professionnel, notamment par souci de sécurité et de respect des standards.

On veillera donc à compulser d'autres documents dédiés au langage HTML afin de parfaire ses connaissances quant aux us et coutumes du moment du bon développeur web : évolution du HTML 4 vers le XHTML [2], nécessité de séparer le fond de la forme avec les CSS [3], utilisation des outils de validation des standards [4], etc.

---

[1] La mise en page peut être plus récente. La date spécifiée en page de garde correspond à la date de dernière modification du fond et/ou de la forme.

[2] XHTML : eXtensible HyperText Markup Language (eng) ≡ Langage de Description de l'HyperTexte Extensible (fr), successeur du HTML qui mélange les syntaxes du langage HTML et du langage XML nécessitant plus de rigueur dans son écriture.

[3] CSS : Cascading Style Sheets (eng) ≡ Feuilles de Style en Cascade (fr), définition de styles d'écriture (mise en forme, police, couleur, etc.).

[4] cf. http://validator.w3.org/.

1 INTRODUCTION AU LANGAGE HTML

Le langage **HTML** [1] est un langage utilisé afin de construire des documents logiciels présentant des informations affichées sur l'écran de l'ordinateur. Il s'agit du langage principal utilisé pour construire les pages web [2] visualisables en naviguant sur internet [3].

Les caractéristiques du langage permettent de structurer, d'afficher et gérer différents « objets » à l'écran ; l'ensemble de ces objets constitue ce qu'on appelle une page [4].

Ces objets peuvent être du texte, mais aussi des tableaux, des images, des vidéos, des composants graphiques (boutons, boîtes de dialogue, menus déroulants, …), etc. : soit donc tout objet constituant un élément d'information.

Chacun des objets de la page peut être associé à une autre page (contenant aussi des objets) ou bien à un autre objet à laquelle on accède en cliquant sur l'objet concerné.

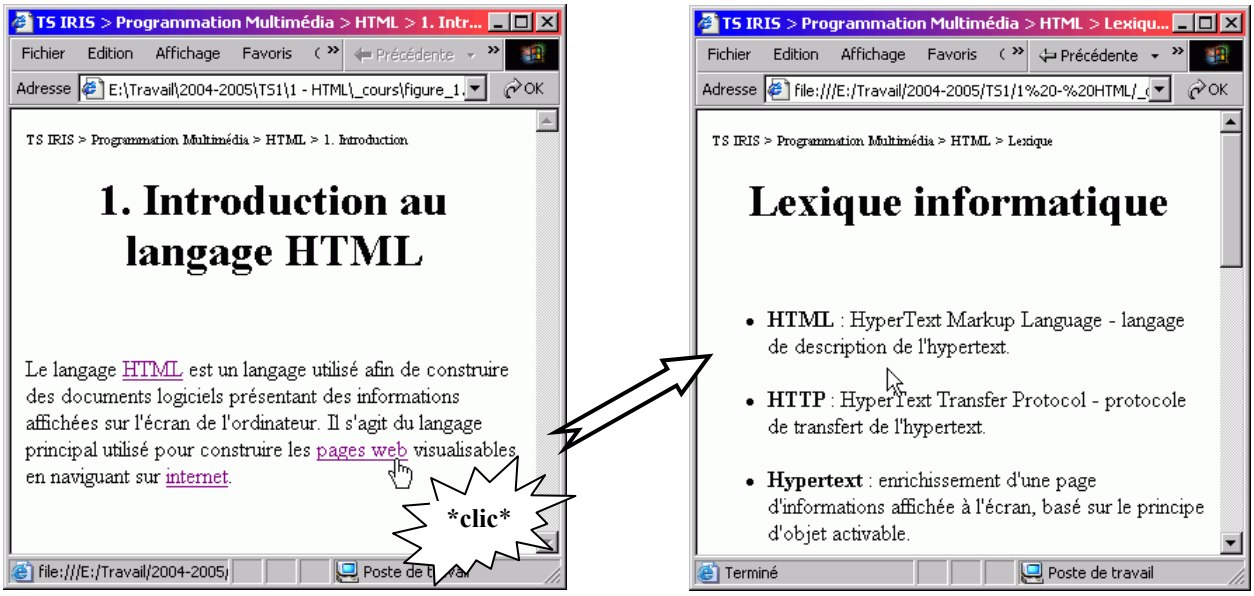


Figure 1.1 : exemple de mise en œuvre de l'hypertexte

Ce principe d'élément d'information accessible par un simple clic en pointant sur un autre élément d'information est appelé **hypertexte** [5] ; et le langage permettant de décrire l'hypertexte est donc le HTML.

Le langage HTML a donc pour principal but de transmettre des informations entre deux ordinateurs [6] :

- un ordinateur qui recherche une information, appelé *client* [7] ;
- un ordinateur qui possède l'information recherchée, appelé *serveur* [8].

---

[1] HTML : HyperText Markup Language (eng) ≡ Langage de Description de l'HyperTexte (fr).
[2] Page web : page accessible via internet ; par extension, on désigne par « site web » un ensemble de pages présentant généralement des informations relatives à un thème précis.
[3] Internet : réseau international formé par l'interconnexion de milliards d'ordinateurs à travers le monde qui permet l'échange d'informations diverses entre particuliers et professionnels (appelé aussi World Wide Web (www), toile, etc.).
[4] Page : affichage structuré d'un ensemble d'objets divers.
[5] Hypertexte : enrichissement d'une page d'informations affichée à l'écran, basé sur le principe d'objet activable.
[6] Même si l'ordinateur n'est qu'un maillon dans la chaîne d'information dont vous – utilisateur – êtes l'une des extrémités.
[7] Ou *machine cliente* ou *ordinateur client*.
[8] Ou *machine serveur* ou *ordinateur serveur*.

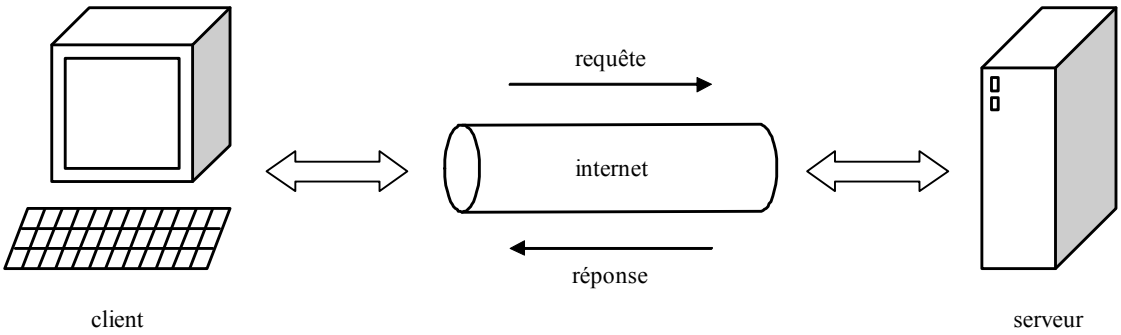


Figure 1.2 : communication entre 2 ordinateurs via internet

Pour transmettre ces informations, il faut bien entendu que ces ordinateurs aient la capacité de communiquer selon un mode commun ; c'est ce qu'on appelle le protocole [1]. Celui qui permet une communication basée sur l'échange de pages écrites en HTML s'appelle le HTTP [2].

L'information est donc « codé » suivant le langage HTML, et le support de cette information est alors un fichier possédant l'extension *.html* ou *.htm*.

Lorsque l'ordinateur client effectue une demande d'information, il demande donc à l'ordinateur serveur de bien vouloir lui transmettre le fichier HTML répondant à ses besoins [3] ; cette demande est ce qu'on appelle une requête [4].

La requête répond donc à 3 questions : *quoi ?* (fichier *.html*), *à qui ?* (serveur), *comment ?* (protocole HTTP).

D'un point de vue logiciel, la spécification de la requête ainsi que l'exploitation des informations reçues s'effectuent à l'aide d'un logiciel installé sur la machine cliente, appelé navigateur [5] ; le navigateur permet d'aller consulter les informations du serveur accessibles via internet. De l'autre côté, pour pouvoir diffuser ces informations sur internet (donner accès à certains fichiers au monde entier), le serveur doit avoir un logiciel installé, appelé serveur web [6].

Pour transmettre la requête entre les 2 ordinateurs, il faut la mettre en forme ; elle devient alors une URL [7] et est constituée par la désignation du protocole (`http`), du nom de l'ordinateur serveur (`nom_serveur`), et du nom du fichier HTML recherché (`fichier.htm`) suivant la syntaxe suivante : `http://nom_serveur/fichier.htm`.

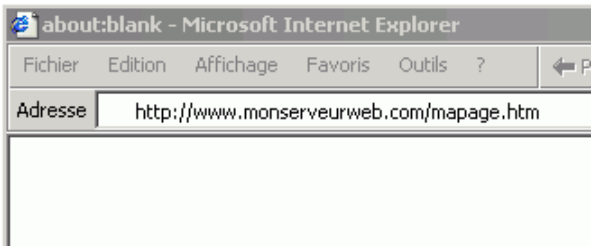


Figure 1.3 : exemple d'une URL dans le navigateur Internet Explorer

L'URL, utilisée par le navigateur, permet donc ainsi de recevoir la « réponse » à la requête – laquelle précise : *quoi ?*, *qui ?*, et *comment ?*.

---

[1] Protocole : ensemble de règles permettant une transmission correcte de l'information.
[2] HTTP : HyperText Transfer Protocol (eng) ≡ Protocole de Transfert de l'HyperTexte (fr).
[3] La question de savoir « comment le client sait quel fichier répond à ses besoins » ne nous intéresse pas ici.
[4] Requête : demande d'informations envoyée par un ordinateur client à un ordinateur serveur.
[5] Navigateur : logiciel installé sur une machine connectée à Internet qui permet d'accéder aux informations accessibles sur la toile (appelé aussi navigateur web ou browser (butineur en français (nda :-|)).
[6] Serveur web : logiciel installé sur une machine hôte (généralement connectée en permanence à internet) qui spécifie les fichiers d'accès public à travers la toile.
[7] URL : Uniform Resource Locator (eng) ≡ Localisateur Uniforme de Ressource (fr) (adresse d'un document consultable via Internet).

2 NOTIONS DE BASE

2.1 LANGAGE INTERPRÉTÉ

Le langage HTML permet la transmission d'informations entre 2 ordinateurs sous forme de pages affichées à l'écran constituées d'un ensemble structuré de divers objets.

HTML est un langage interprété, en opposition à un langage compilé [1].

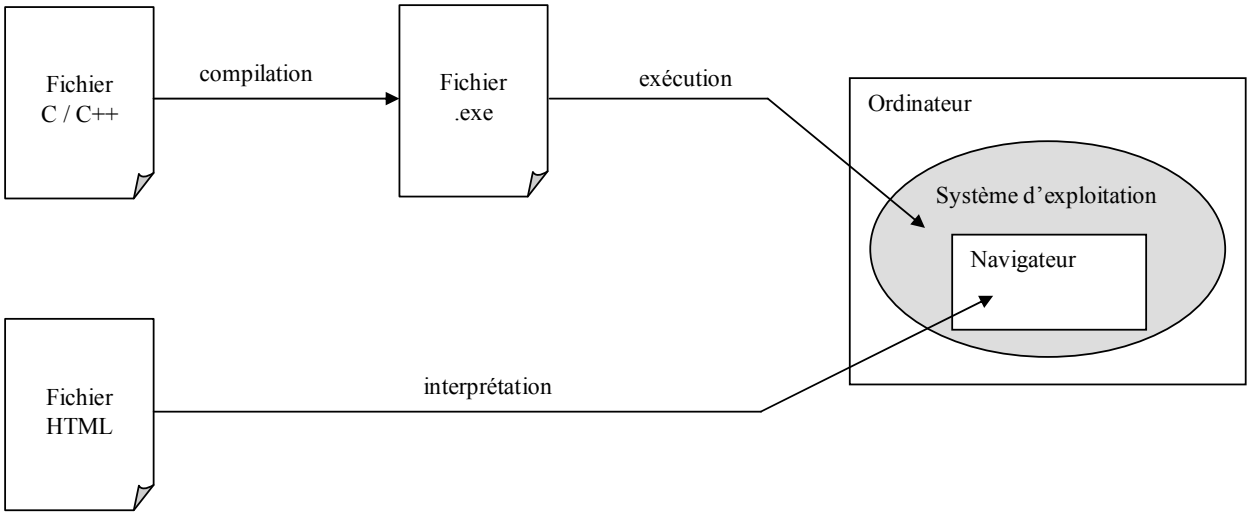


Figure 2.1 : différence entre un langage interprété et un langage compilé

Un langage interprété n'a pas besoin d'être « traduit ».

En effet, le logiciel recevant le code HTML est capable de comprendre directement son contenu ; il affiche alors les informations contenues dans ce code selon une présentation et une mise en page spécifiées dans le code lui-même [2].

Le navigateur web, qui est le logiciel qui permet d'afficher le code HTML, constitue donc un **interpréteur HTML**.

Par conséquent, un simple éditeur de texte (tel que « bloc-notes » ou « wordpad ») suffit à l'écriture d'un programme HTML.

Les informations de présentation contenues dans le code HTML sont spécifiées par ce qu'on appelle des balises [3].

[1] Après écriture d'un programme en langage compilé (langage C ou C++par exemple), celui-ci est traduit en langage « machine » compréhensible par le micro-processeur de l'ordinateur utilisé ; cette opération de traduction spécifique au type de processeur utilisé (x86, Alpha, Mac, etc.) est ce qu'on appelle la compilation.

[2] Puisque le langage HTML a pour principal objectif de présenter des documents.

[3] Balise (fr) ≡ tag (eng).

2.2 LES BALISES

2.2.1 Définition

Une **balise** HTML est un indicateur écrit en texte qui a une signification particulière ; cette signification renseigne sur le traitement de présentation ou de mise en page à appliquer au texte ou à une partie du texte figurant dans le code HTML.

Ex. : `<b>bonjour</b> comment ça va ?`

La balise `<b>` a pour action de mettre en gras. Le début du texte à mettre en gras est indiqué par l'insertion du code `<b>` (appelée balise de début) ; la fin du texte à mettre en gras est indiquée par l'insertion du code `</b>` (appelée balise de fin). Dans l'exemple ci-dessus le texte `bonjour` sera donc affiché en gras à l'écran :

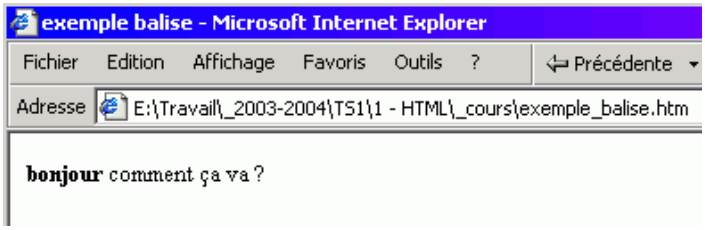


Figure 2.2 : exemple de la balise de mise en page <b>

Une balise se définit toujours de la manière suivante : < mnémonique >. Elle commence donc toujours par le caractère < et se termine toujours par le caractère >.
Entre ces deux caractères figure une mnémonique (lettre ou groupe de lettres) ; chaque fonction de mise en page et de présentation possède une mnémonique unique (`b` pour mettre en gras [1], `u` pour souligner [2], etc.).

Notons, pour débuter, l'existence des balises suivantes, permettant de gérer la mise en page :

- `<h*>` : titre de taille variable ($* \in [1;6]$ – $*=1$: plus grand ; $*=6$: plus petit) ;
- `<br>` : retour à la ligne ;
- `<p>` : nouveau paragraphe ;
- `<hr>` : trait de séparation horizontal.

Ex. : `<h2>bonjour</h2> comment ça va ?`

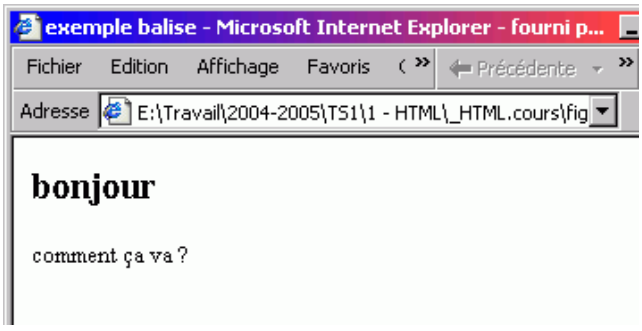


Figure 2.3 : exemple de la balise de mise en page <h>*

La balise `<h2>` définit un titre de taille 2 (assez grand donc). Notez que l'utilisation de cette balise implique forcément un retour à la ligne, et ce, sans utiliser la balise `<br>`. Chaque balise a un caractère bien défini, qui lui est propre ; il convient donc, lorsque l'on utilise une balise, de savoir quelles sont exactement ses propriétés [3].

La balise de fin est identique à la balise de début, à ceci près que l'on rajoute le caractère / (slash) [4].

[1] Bold (eng) ≡ gras (fr).
[2] Underlined (eng) ≡ souligné (fr).
[3] Tester une balise directement dans un fichier HTML peut se révéler très instructif…
[4] La notation </exemple> peut être analysée comme la « négation » de la balise <exemple>, à l'image de l'électronique numérique, où on écrit /A la négation d'une variable booléenne A ; si A vaut 'faux' (0) alors /A vaut 'vrai' (1) et vice-versa (/vrai = faux et /faux = vrai).

Toutes les balises n'utilisent pas le principe de balise de fin. Cela n'est fait que si la signification (en terme de présentation ou mise en page) de la balise le nécessite.

Ainsi mettre une partie de texte en italique, ou bien augmenter la taille de la police pour mettre en valeur un titre, sont 2 actions nécessitant une balise de fin ; en revanche, un retour à la ligne, l'affichage d'un trait de séparation horizontal, sont des actions ne nécessitant pas de balise de fin.

2.2.2 *Les attributs de balise*

On peut préciser l'effet d'une balise par l'ajout d'un **attribut** pouvant prendre différentes valeurs. Un attribut est toujours associé à une balise (utilisé seul, il n'a aucun sens).

Ex. : `<h2 align="center">bonjour</h2> comment ça va ?`

Figure 2.4 : exemple de la balise de mise en page <h> avec un attribut*

L'attribut `align=` permet de préciser le type d'alignement horizontal utilisé par l'objet décrit par la balise. Utilisé avec la balise `<h2>`, il précise ainsi si le titre doit être aligné à gauche (`left` (valeur par défaut)), centré (`center`) ou bien aligné à droite (`right`).

On peut retrouver le même attribut utilisé avec une balise différente ; généralement le sens de l'attribut sera identique.

Autres exemples d'attributs :

- `<hr>` : `size=`, `width=`, `align=`, `noshade` ;
- `<pre>` : `width=` ;
- `<ul>` : `type=` ;
- `<li>` : `type=`, `value=` ;
- `<ol>` : `type=`, `start=`.

Il existe 2 types d'attributs :

- attribut à valeur : on donne la valeur de l'attribut selon une chaîne numérique ou alphanumérique (ex. : `align="center"...`, `size=2`) ;
- attribut booléen : on indique simplement le nom de l'attribut pour signifier l'activation d'une propriété précise (ex. : `noshade`).

Lorsque l'on ne fait pas mention d'un attribut, l'interpréteur utilise une valeur par défaut [1] pour celui-ci. Par exemple, la valeur par défaut de l'attribut `align=` est généralement `left`.

2.3 STRUCTURE D'UN FICHIER HTML

Le fichier doit toujours avoir l'extension *.html* ou *.htm* afin d'être reconnu en tant que fichier HTML par le navigateur [2].

[1] Laquelle n'est pas forcément identique d'un navigateur à un autre.

[2] Par convention un fichier écrit en langage de programmation xxx portera l'extension .xxx (.bas (basic), .c (C), .cpp (C++), .html (html), .java (Java), .js (JavaScript), .php (PHP), etc.).

La structure d'un fichier HTML comprend toujours le code suivant [1] :

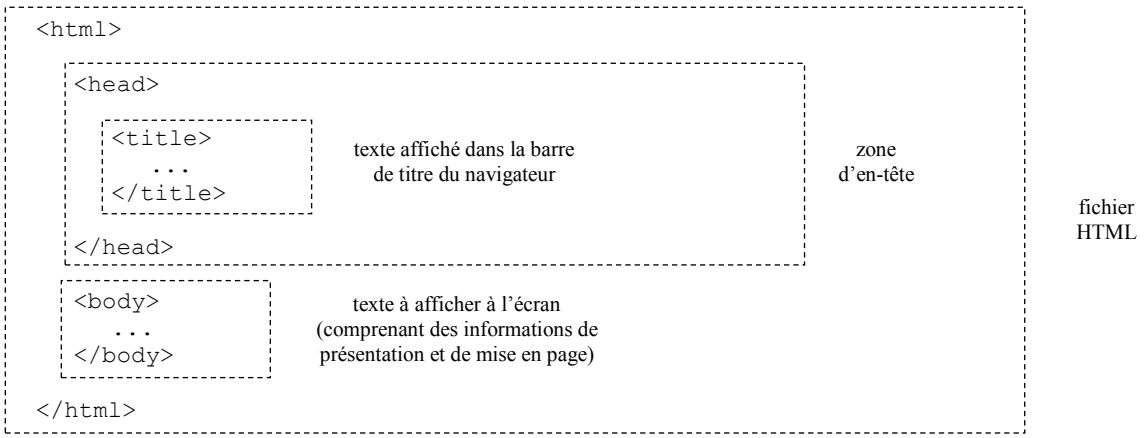


Figure 2.5 : structure du code dans un fichier HTML

La zone d'en-tête est en général assez courte et correspond à des informations générales sur le fichier HTML.

Le texte à afficher est inséré à l'intérieur des balises `<body>` et `</body>`. Le formatage du texte lui-même dans le fichier source n'a aucune influence sur la manière dont celui-ci sera affiché dans le navigateur ; seule l'insertion de balises permet de définir la mise en page et la présentation du texte.

La balise `<address>` permet de préciser éventuellement l'auteur du site (généralement positionnée juste avant la balise `</body>`).

La balise `<!-- commentaire -->` permet d'insérer des commentaires dans le fichier HTML, lesquels ne seront donc pas affichés par le navigateur lorsque celui-ci interprétera le code source.

2.4 BALISES PRINCIPALES

Le langage HTML étant destiné à présenter des informations à l'écran, il existe beaucoup de balises relatives à la mise en page et à la présentation.

2.4.1 Mise en page

Les balises principales de mise en page sont :

- `<h*>` : titre de taille variable ($* \in [1;6]$ – $*=1$: plus grand ; $*=6$: plus petit) ;
- `<br>` : retour à la ligne ;
- `<p>` : nouveau paragraphe ;
- `<hr>` : trait de séparation horizontal ;

2.4.2 Listes

2.4.2.1 Liste simple

Une liste simple présente de manière ordonnée (puces ou numéros) une liste d'éléments. Les balises utilisées pour mettre en forme une liste simple sont :

- `<ul>` : création d'une liste simple avec puces ;
- `<ol>` : création d'une liste simple avec numéros ;
- `<lh>` : en-tête/titre de la liste ;
- `<li>` : élément de liste.

[1] Lequel constitue donc un squelette de fichier HTML.

Ex. :
```
<ol>
   <lh>titre</lh>
   <li>description élément1</li>
   <li>description élément2</li>
</ol>
```

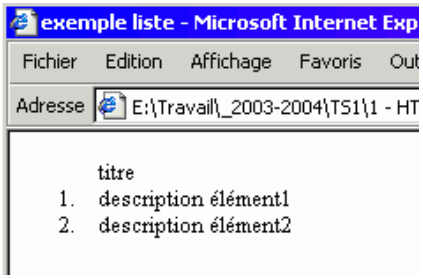


Figure 2.6 : exemple d'une liste simple numérotée

2.4.2.2 Liste descriptive

Une liste descriptive présente une liste d'éléments (sans puces ou numéros). Les balises utilisées pour mettre en forme une liste descriptive sont :

- `<dl>` : création d'une liste descriptive ;
- `<dt>` : titre d'un élément de liste ;
- `<dd>` : description de l'élément.

Ex. :
```
<dl>
   <dt>titre1</dt>
   <dd>description titre1</dd>
   <dt>titre2</dt>
   <dd>description titre2</dd>
</dl>
```

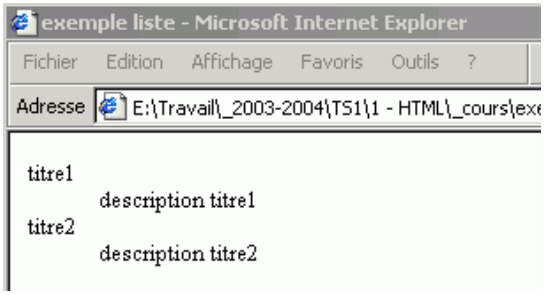


Figure 2.7 : exemple d'une liste descriptive

2.4.3 Styles

- `<b>` : gras ;
- `<i>` : italique ;
- `<u>` : souligné ;
- `<font size=*>` : texte de taille variable (* ϵ [1;7] – *=1 : plus petit ; *=7 : plus grand) ;
- `<font color=nom_couleur>` : texte de couleur (`nom_couleur` : `black`, `red`, `blue`, `green`, etc.) ;
- `<tt>` : police à matrice constante [1] ;
- `<pre>` : respect du formatage du texte tel que défini dans le fichier source (nb : utilisation d'une police à matrice constante) ;
- `<cite>` : ouvrage ou référence ;
- `<blockquote>` : citation ;
- `<code>` : code informatique (police à matrice constante) ;
- `<dfn>` : définition ;
- `<em>` : mise en valeur d'un texte (italique) ;
- `<kbd>` : saisie clavier (police à matrice constante) ;
- `<samp>` : exemple ;
- `<strong>` : mise en valeur d'un texte (gras) ;
- `<var>` : variable informatique (italique).

[1] Une police à matrice constante alloue la même largeur d'affichage à tous les caractères ; par exemple le caractère 'i' sera aussi large à l'affichage que le caractère 'm', ce qui est rarement le cas dans les polices classiques.

2.5 LES TABLEAUX

Pour dessiner un tableau, on dispose des balises suivantes :

- `<table>` : création d'un tableau ;
- `<caption>` : titre du tableau ;
- `<tr>` : création d'une nouvelle ligne dans le tableau ;
- `<th>` : titre d'une colonne d'un tableau (cellule avec écriture en gras centrée) ;
- `<td>` : contenu d'une cellule du tableau.

Ex. :
```
<table border=1>
   <caption>titre_tableau</caption>
   <tr>
      <td>ligne1 colonne1</td>
      <td>ligne1 colonne2</td>
   </tr>
   <tr>
      <td>ligne2 colonne1</td>
      <td>ligne2 colonne2</td>
   </tr>
</table>
```

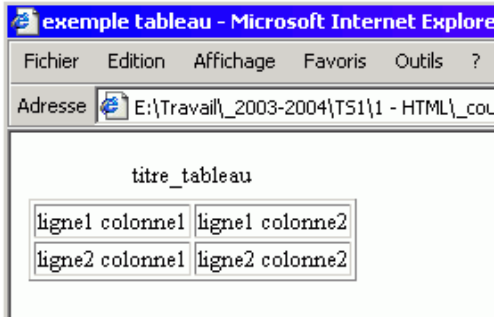


Figure 2.8 : exemple d'affichage d'un tableau

Il existe de nombreux attributs pour les tableaux :

- `<table>`
  - `border=` : épaisseur des traits du tableau (en pixels) ;
  - `width=` : proportion de la largeur de fenêtre du navigateur occupée par le tableau (en pourcentage) ;
  - `cellspacing=` : distance entre 2 cellules (en pixels) ;
  - `cellpadding=` : marge entre le bord d'une cellule et le texte de la cellule (en pixels).

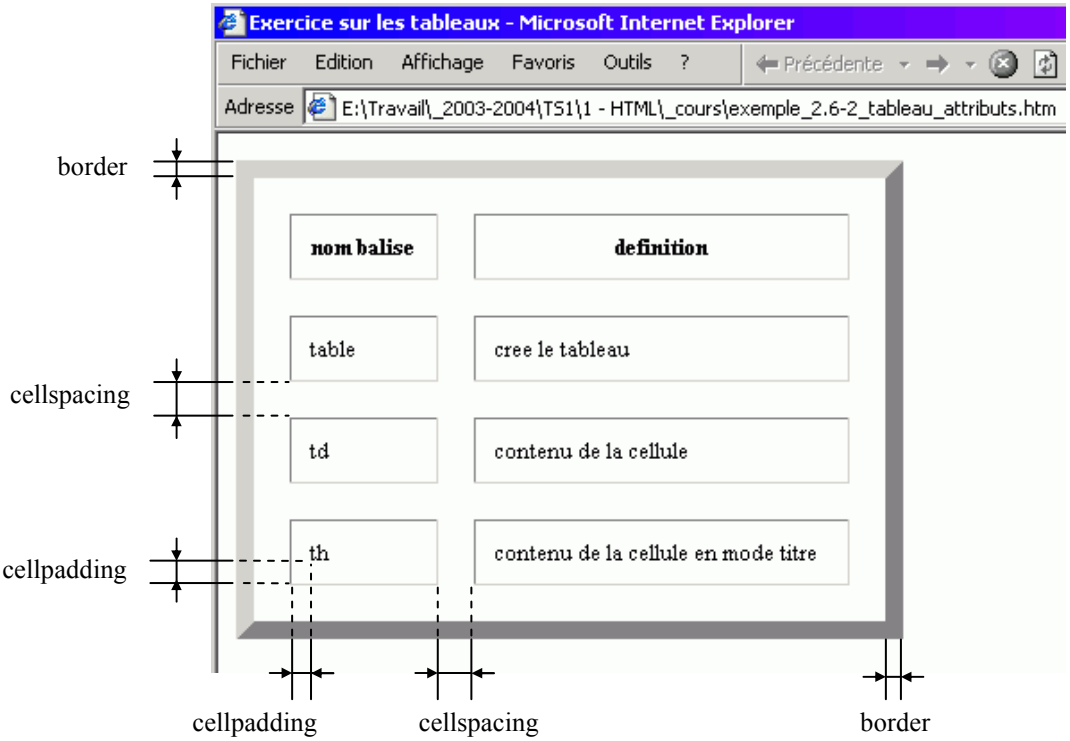


Figure 2.9 : détail des différents attributs de la balise <table>

- `<td>`, `<tr>`, `<th>`
  - `align=` : alignement horizontal ;
  - `valign=` : alignement vertical ;
  - `bgcolor=` : couleur de fond.

- `<td>`
  - `colspan=` : nombre de cellules fusionnées en colonnes ;
  - `rowspan=` : nombre de cellules fusionnées en lignes ;
  - `nowrap` : place le texte de la cellule sur une seule ligne.

Ex. :
```
<html>
   <head>
      <title>Exercice sur les tableaux</title>
   </head>
   <body>
      <table border=1>
         <tr>
            <td>L1C1</td>
            <td>L1C2</td>
            <td>L1C3</td>
         </tr> <tr>
            <td rowspan=2 colspan=3>L2C1</td>
            <td>L2C2</td>
         </tr> <tr>
            <td>L3C2</td>
            <td>L3C3</td>
         </tr>
      </table>
   </body>
</html>
```

Figure 2.10 : exemple de fusion de cellules dans un tableau

3 LES LIENS

3.1 INTRODUCTION

Un lien permet de définir une propriété particulière à un élément affiché sur la page.

Cette propriété est le renvoi vers un autre élément d'information par un simple clic sur cet objet [1].

Le terme de **lien** vient du fait qu'on associe ainsi de manière invisible 2 éléments d'information qui, de prime abord, ne sont pas sensés avoir de relation particulière.

Ex. : Un texte dont certains mots sont cliquables et qui donne alors accès à un lexique expliquant le mot.

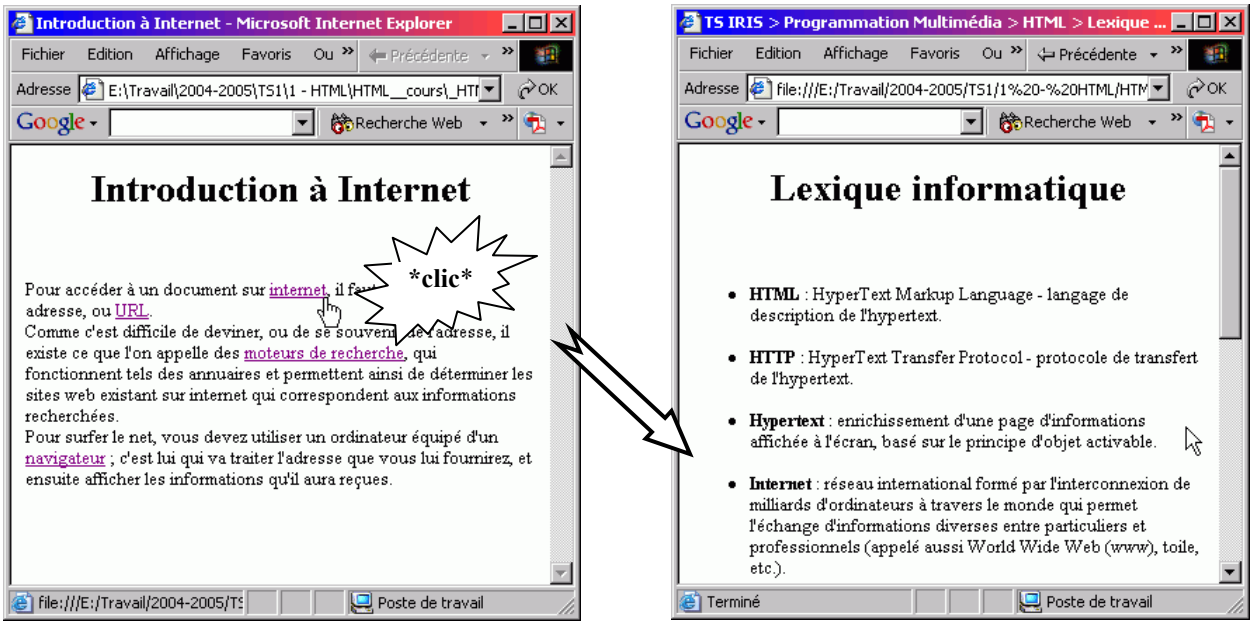


Figure 3.1 : exemple d'un lien

Dans l'exemple ci-dessus, en cliquant sur le mot *internet*, on accède à une autre page donnant une définition de ce mot.

3.2 DÉFINITIONS

Un lien relie 2 éléments d'information distincts appelés alors **ancres** dans le cadre de cette association :

- ancre de départ : zone active du document sur laquelle un clic renverra à l'ancre d'arrivée ;
- ancre d'arrivée : élément d'information obtenu par un clic sur l'ancre de départ.

Dans l'exemple ci-dessus, l'ancre de départ est le mot *internet*, et l'ancre d'arrivée est la page HTML *Lexique*.
Une ancre de départ peut être : un mot, un groupe de mots, une image, une partie d'une image, etc., soit donc tout élément visuel affichable à l'écran.

[1] C'est ce principe de lien qui a donné naissance au terme *navigation* car on « surfe » ainsi sur la toile au gré du « vent » de son bon vouloir.

Une ancre d'arrivée peut être : une page HTML, une image, un son, un mot, groupe de mots ou zone précise de la page HTML courante.

Une ancre de départ ne peut être associée qu'à une seule ancre d'arrivée ; en revanche, une ancre d'arrivée peut correspondre à une infinité d'ancres de départ [1].

On parle de *lien interne* si le lien pointe à l'intérieur de la même page HTML (généralement un mot, groupe de mots, ou paragraphe du texte). On parle de *lien externe* si le lien pointe à l'extérieur de la page HTML courante.

Une ancre d'arrivée correspondant à une page HTML (ou une image, un son, …) appartenant au même site web que l'ancre de départ est considérée comme un lien externe [2].

Un lien se définit par la balise `<a>` ; les attributs associés sont :

- `name=` : définition de la référence[3] de l'ancre d'arrivée associée au lien (cas d'un lien interne uniquement).
  syntaxe : `<a name=etiquette>ancre_d_arrivee</a>`.
- `href=` : désignation de l'ancre d'arrivée associée au lien ;
  syntaxe : `<a href=ancre_d_arrivee>ancre_de_depart</a>` (lien externe) ;
  `<a href=#etiquette>ancre_de_depart</a>` (lien interne).

Ex. :

- Lien interne
  […] `un document sur <a href="#inte">internet</a>, il faut` […]
  `...`
  […] `<a name="inte">Internet</a> : réseau international` […]
- Lien externe
  […] `Le langage <a href="lexique.htm">HTML</a> est un langage` […]

On peut aussi « mélanger » les deux types de liens.

Ex. : […] `un document sur <a href="lexique.htm#inte">internet</a>, il faut` […]

Une ancre peut être à la fois de départ et d'arrivée.

Ex. : […] `un document sur <a name="bak.inte" href="#inte">internet</a>, il faut` […]
`...`
[…] `<a name="inte" href="#bak.inte">Internet</a> : réseau international` […]

Nb : L'ancre de départ est repérable à l'écran grâce à la modification du curseur de souris lorsque l'on « survole » la zone cliquable ; de plus, s'il s'agit d'un texte, celui-ci est en général souligné. Cette modification du comportement est paramétrée par le navigateur. Par conséquent, elle ne sera pas forcément similaire d'un client web à un autre.
Pour les ancres d'arrivée de type vidéo et audio, il faut disposer soit d'un logiciel adapté installé sur le poste client (ex. : winamp), soit d'un plugin adapté ajouté au navigateur (ex. : quicktime pour les .mov).

3.3 IMAGE CLIQUABLE ET IMAGE À ZONES CLIQUABLES

On insère une image avec la balise `<img>`, selon la syntaxe : `<img src=fichier_image>`.

Ex. : `<img src="wikipedia.png">`

Il est alors très facile de définir une image comme l'ancre de départ d'un lien.

Ex. : `<a href="http://fr.wikipedia.org"><img src="wikipedia.png"></a>`

On peut aussi délimiter différentes zones dans l'image, chacune pointant sur une ancre d'arrivée distincte.

Pour cela, il faut préciser que l'image suit un plan de découpage précis en utilisant l'attribut `usemap=` de la balise `<img>` (ex. : `<img src="image.gif" usemap="#plan_découpage">`).

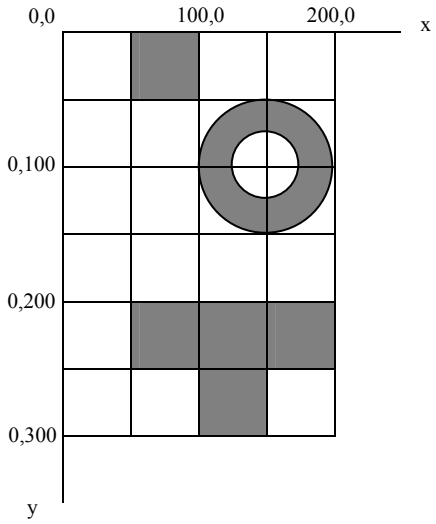
[1] On imagine par exemple facilement que multitudes de sites web renvoient vers le site *fr.wikipedia.org*.
[2] Il n'y a en effet aucune différence structurelle entre une page web appartenant au même site web que la page courante et une page web appartenant à un autre site que celui de la page courante, puisqu'on y accède grâce à l'URL.
[3] La référence du lien interne est une étiquette, elle est librement choisie par le programmeur (nda : choisir un nom évocateur évidemment).

Ensuite, il faut définir précisément le découpage de l'image par zones avec la balise `<map>` (ex. : `<map name="plan_découpage">`).

Enfin, on va décrire chacune de ces zones avec la balise `<area>` et ses attributs :

- `href=` : définition de l'ancre d'arrivée associée au lien ;
- `nohref` : lien ne pointant sur rien (vient en remplacement de `href=`) [1] ;
- `shape=` : forme générale de la zone, parmi 3 types prédéfinis :
  - `"rect"` : rectangle ;
  - `"circle"` : cercle ;
  - `"poly"` : polygone.
- `coords=` : ensemble des coordonnées dessinant la zone conformément à la forme définie (dans le plan et référentiel de l'image) :
  - rectangle : on indique les coordonnées (x,y) de 2 sommets opposés ;
  - cercle : on indique les coordonnées du centre (x,y) puis le rayon (r) ;
  - polygone : on indique les coordonnées de chacun des points successifs de la forme.

Ex. : Une image de 200x300 pixels dans laquelle on veut définir 3 formes (1 carré, 1 cercle, 1 forme quelconque).




```
<img src="formes.gif" usemap="#figures">
<map name="figures">
    <area shape="rect" coords=50,0,100,50 href="r.gif">
    <area shape="circle" coords=150,100,25 nohref>
    <area shape="circle" coords=150,100,50 href="c.gif">
    <area shape="poly" coords=50,200,200,200,200,250,
    150,250,150,300,100,300,100,250,50,250 href="p.gif">
</map>
```

Figure 3.2 : exemple de découpage de zones cliquables dans une image

3.4 LES FRAMES

3.4.1 Principes

Il est possible de proposer un système multifenêtrage dans l'afficheur du navigateur ; c'est-à-dire découper cet afficheur en plusieurs zones bien distinctes, chacune de ces zones affichant alors un document HTML unique. On appelle ces zones, des **frames**, ou cadres en français.

Chaque document HTML affiché dans un cadre est un document HTML classique, pouvant être interprété tout seul par le navigateur sans aucune modification.

[1] Utile pour définir une zone non-active à l'intérieur d'une zone active ; exemple : le pourtour d'un rond.

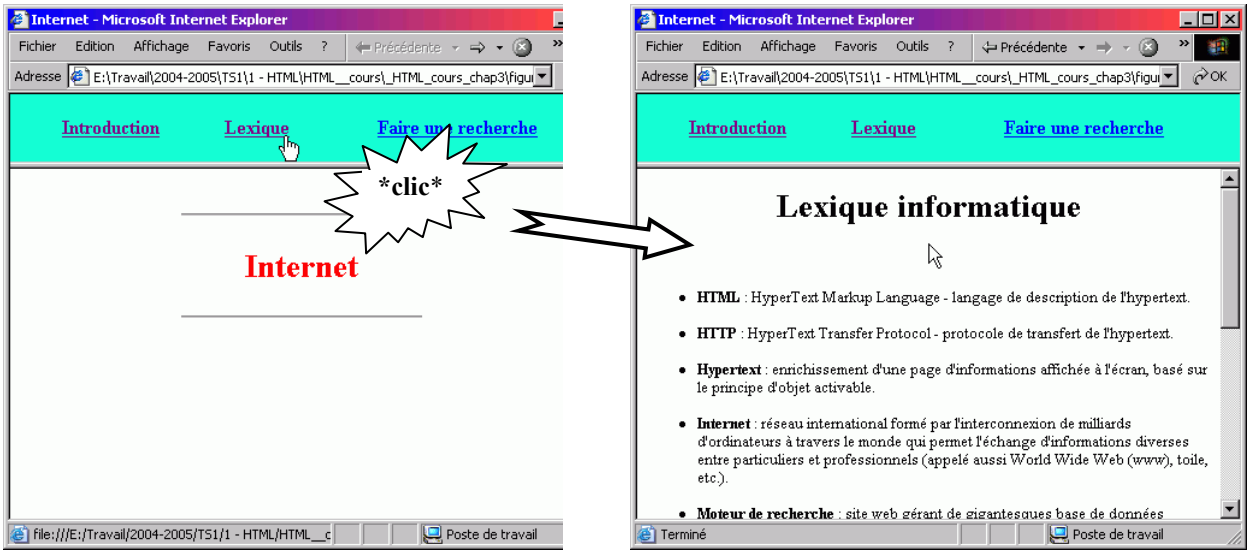


Figure 3.3 : exemple d'utilisation des cadres pour créer un menu

Pour lier ces documents HTML en 1 seule et même page découpée en cadres, il faut écrire un document HTML supplémentaire spécifiant le nom des cadres, leur taille, position, etc. ; ce fichier a pour but de rassembler tous les documents décrivant les différents cadres en 1 seul document affiché dans le navigateur.

Ce fichier HTML principal ne fait que décrire le découpage en cadres désiré, et préciser éventuellement le ou les nom(s) des fichiers HTML chargé(s) à l'initialisation dans chacun des cadres. Il ne contient pas de texte à afficher à l'écran.

3.4.2 *Les balises de gestion de frames*

Pour créer un découpage en cadre, on va utiliser les balises `<frameset>`, `<frame>` et `<noframes>`.
La balise `<frameset>` vient en remplacement de la balise `<body>` dans le document HTML principal.

- `<frameset>` : création d'un découpage en cadres dans la zone courante, avec indications du nombre de cadres, leur disposition, et leur proportion ou leur taille ;
  Les attributs sont :
  - `rows=` : dimension de chacun des cadres disposés en horizontal ;
  - `cols=` : dimension de chacun des cadres disposés en vertical [1].
    Les dimensions peuvent être exprimées selon 3 modes :
    - `X` : dimension (en pixels) ;
    - `X%` : proportion (en pourcentage) ;
    - `n*` : proportion du cadre parent occupé (en nombre de parts, où une part (`*`) correspond à la proportion par rapport au nombre total de parts obtenus en additionnant la proportion de chaque cadre), ex. : `rows="40%, 60%"` est identique à `rows="2*, 3*"` (2 * + 3 * = 5 *, donc 1 * = 1/5 = 20%).
- `<frame>` : caractérisation d'un cadre (fichier HTML chargé au départ, nom, etc.) ;
  Les attributs sont :
  - `src=` : nom du fichier *.html* chargé au départ dans le cadre (nom en chemin relatif ou absolu) ;
  - `name=` : nom du cadre.
- `<noframes>` : texte à afficher à l'écran dans le cas où le navigateur utilisé ne saurait pas gérer les frames [2].

Il faut donc insérer autant de balises `<frame>` que vous avez de cadres définis par la balise `<frameset>`.

Ex. :
```
<frameset cols="25%, 75%">
    <frame src="frame1.htm" name="cadre_menu">
    <frame src="frame2.htm" name="cadre_principal">
</frameset>
<noframes>changez de navigateur, vite!</noframes>
```

[1] On ne peut pas utiliser les attributs `rows=` et `cols=` en même temps.
[2] Obsolète, mais sait-on jamais…

Il est possible d'imbriquer les découpages en sous-cadres : soit le document HTML chargé dans l'un des cadres correspond alors à un autre fichier de description de découpage en cadres ; soit on insère directement le second découpage à l'intérieur du premier.

Ex. :
```
<frameset cols="25%, 75%">
<frame src="frame1.htm" name="cadre_menu">
   <frameset cols="25%, 75%">
      <frame src="frame2.htm" name="cadre_primaire">
      <frame src="frame3.htm" name="cadre_secondaire">
   </frameset>
</frameset>
<noframes>changez de navigateur, vite!</noframes>
```

3.4.3 Gestion des cadres avec les liens

Le nom de chacun des cadres (attribut `name=` de la balise `<frame>`) permet de les référencer et de s'y référer lors de l'affichage d'un lien. En effet, que ce soit avec ou sans l'utilisation de frames, le comportement par défaut d'un lien fait que l'ancre de destination vient s'afficher en lieu et place du document dans lequel figure l'ancre de départ.

Or, il est possible, avec les frames, de préciser quel cadre doit accueillir l'ancre de destination. On introduit pour cela un nouvel attribut de la balise `<a>`, il s'agit de l'attribut `target=`.

Ex. : *index.htm*

```
<html>
   <head>
      <title>Internet</title>
   </head>
   <frameset rows="*, 5*">
      <frame src="menu.htm" name="cadre_menu">
      <frame src="titre.htm" name="cadre_principal">
   </frameset>
   <noframes>changez de navigateur, vite!</noframes>
</html>
```

menu.htm

```
<html>
   <head>
   </head>
   <body>
      <a href="introduction.htm" target="cadre_principal">Introduction</a>
      <a href="lexique.htm" target="cadre_principal">Lexique</a>
      <a href="www.google.fr" target="cadre_principal">Faire une recherche</a>
   </body>
</html>
```

Ainsi, en cliquant sur le lien *Lexique* on demande la page `lexique.htm` mais qui se chargera dans le cadre nommé `cadre_principal`.

Lorsque l'on ne précise pas le nom du cadre de destination (pas de balise `target=`), c'est le cadre source qui sera utilisé.

4 LES FORMULAIRES

4.1 INTRODUCTION

Jusqu'à présent, nous n'avons travaillé qu'avec des transmissions de données dans un seul sens, s'effectuant du serveur vers le client [1].

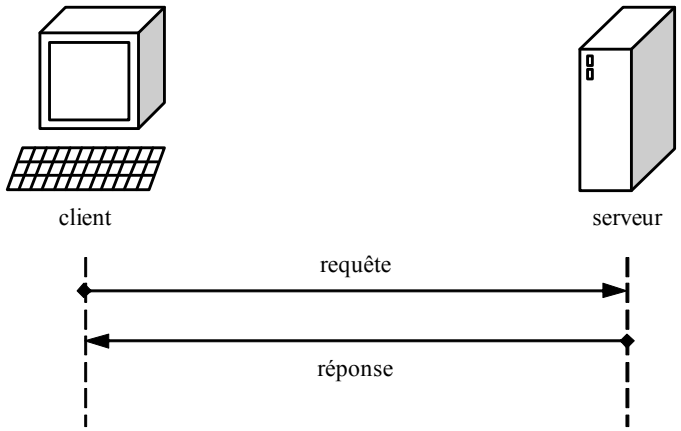


Figure 4.1 : communication client-serveur classique

Pourtant, les documents HTML ne constituent pas forcément qu'une communication unilatérale. En effet, il est possible d'envoyer des données du client vers le serveur.

Cela permet ainsi de pouvoir adapter la réponse en fonction de chaque requête (et de fait pour chaque client).

Ex. :

- site web du type « dictionnaire » : l'utilisateur faire connaître au serveur le mot dont il recherche la signification ;
- site marchand à base de catalogue et panier de commande : l'utilisateur choisit les produits proposés dans le catalogue qu'il désire acheter ;
- demande d'envoi de documentation : l'utilisateur transmet ses coordonnées postales ;
- inscription sur une mailing-list : l'utilisateur transmet son adresse email ;
- etc.

[1] On considère que l'envoi de la requête ne constitue pas une transmission d'informations en tant que telle ; certes, physiquement, des octets sont bien transmis en premier lieu du client vers le serveur, mais ceux-ci ne sont pas porteurs d'informations personnalisées.

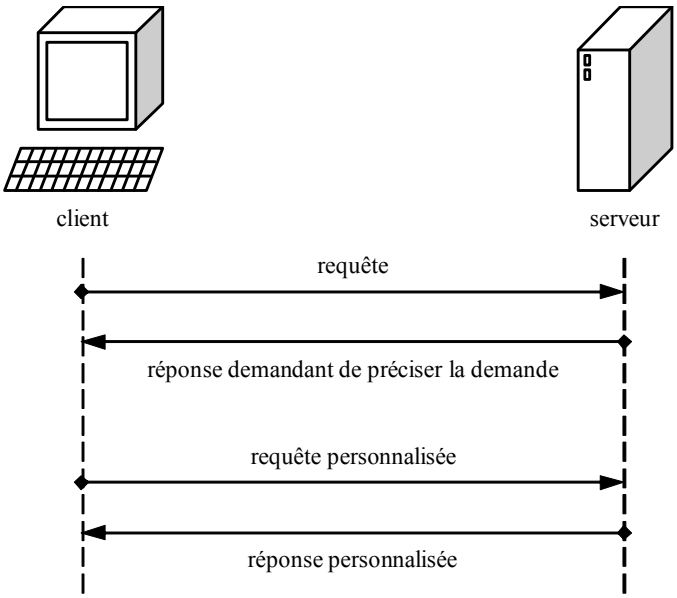


Figure 4.2 : communication client-serveur avec réponse personnalisée

Pour effectuer une requête personnalisée, le langage HTML utilise le principe de formulaire.

4.2 FORMULAIRE ET CHAMPS

À l'instar de sa version « papier », un **formulaire** HTML est une page web constituée de différentes rubriques à compléter afin de communiquer des éléments d'informations au serveur. En informatique, ces rubriques sont appelées **champs**.

On insère un formulaire au sein d'une page web en utilisant la balise `<form>`.

Ex. :
```
<body>
    <form action="hello.php">
       Entrez votre nom<br>
       <input type="text" name="votrenom">
       <br>
       <input type="submit" value="Valider">
    </form>
</body>
```

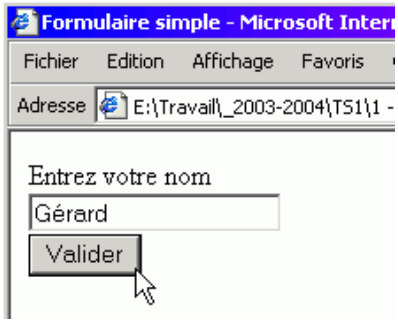


Figure 4.3 : exemple de formulaire simple

À l'intérieur d'un formulaire, on peut placer à loisir les champs que l'on veut ; la seule condition à respecter est de toujours positionner un bouton de validation. Lorsque l'utilisateur clique sur ce bouton de validation, l'ensemble des valeurs des autres rubriques [1] qui constituent le formulaire est alors transmis au serveur.

Les champs peuvent être de 3 types :

- `<input>` : zone de texte ou bouton ;
- `<select>` : liste ou menu (associée avec la balise `<option>`) ;
- `<textarea>` : grande zone de texte.

[1] Telles qu'elles ont été entrées, modifiées ou laissées par défaut par l'utilisateur.

Les différents attributs de balises utiles sont :

- `<form>`
  - `action=` : URL de la page à afficher ou du script à exécuter après validation du formulaire ;
  - `target=` : nom du cadre de destination pour l'URL de la balise `action=` (en cas d'utilisation des frames) ;
  - `name=` : nom du formulaire (permet à l'URL d'identifier le formulaire l'ayant appelé [1]) ;
  - `method=` : méthode de transmission des valeurs des champs au serveur (valeurs `get`/`post`).
- `<input>`
  - `type=` : type du champ ;
    - `"text"` : zone de texte ;
    - `"password"` : zone de texte masquée à l'affichage (`*****`) ;
    - `"hidden"` : zone de texte invisible à l'écran [2] ;
    - `"checkbox"` : case à cocher ;
    - `"radio"` : bouton de sélection unique parmi plusieurs (bouton radio) ;
    - `"reset"` : bouton de réinitialisation du formulaire ;
    - `"submit"` : bouton de validation du formulaire et envoi des champs au serveur ;
    - `"button"` : simple bouton, pas d'action particulière (utilisé en JavaScript principalement).
  - `checked` : case initialement cochée (cas `type="checkbox"` ou `type="radio"`).
- `<input>`, `<select>`
  - `name=` : nom du champ (permet au serveur de récupérer la valeur du champ) ;
  - `value=` : valeur initiale du champ.
- `<select>`
  - `multiple` : liste multi-choix [3] ;
  - `size=` : nombre d'éléments de liste à afficher simultanément à l'écran.
- `<option>` : définition d'un élément de liste ou de menu
  - `selected` : élément de liste ou de menu initialement sélectionné.
- `<textarea>`
  - `rows=` : nombre de lignes de la zone de texte ;
  - `cols=` : nombre de colonnes de la zone de texte.

Une case à cocher (`type="checkbox"`) prend la valeur spécifiée par l'attribut `value=` lorsqu'elle est cochée ; si cet attribut n'est pas spécifié, la valeur prise est celle par défaut (`value="on"`). Lorsque la case n'est pas cochée, la valeur prise par la rubrique est une chaîne vide (`value=""`).

Un bouton radio n'est jamais utilisé seul (sélection unique parmi plusieurs) ; il fait donc toujours parti d'un groupe de boutons radio dont le nom est indiqué par l'attribut `name=` [4] pour chaque bouton y appartenant. La valeur prise par le champ est alors celle correspondant à l'attribut `value=` du bouton radio sélectionné ; si aucun n'est sélectionné, la rubrique prend la valeur chaîne vide (`value=""`).

Bien évidemment, il peut exister plusieurs groupes de boutons radio différents à l'intérieur d'un même formulaire ; chaque groupe se rapportant ainsi à un choix différent.

La balise `<select>` sert à insérer une liste ou un menu, selon la valeur de l'attribut `size=` ; lorsque cet attribut n'est pas spécifié ou a une valeur de `1`, on est en présence d'un menu ; dans le cas contraire il s'agit d'une liste.

La balise `<option>` ne peut être utilisée qu'associée avec la balise `<select>`. Elle possède une balise de fin.

La balise `<textarea>` possède une balise de fin. Le texte initial affiché dans cette zone de texte (le cas échéant) doit être placé entre la balise de début et la balise de fin (ex. : `<textarea>texte initial</textarea>`).

[1] Dans l'absolu, comme une URL peut être l'ancre d'arrivée d'une infinité de lien, il peut être utile de connaître l'ancre de départ.
[2] Utile pour passer des valeurs de manière invisible à l'URL mentionnée (cas de formulaires enchaînés).
[3] Pour faire une sélection multiple, il faut appuyer et maintenir la touche <maj> enfoncée pendant la sélection de chaque élément de liste.
[4] Lors de la sélection d'un bouton radio, cela provoque alors automatiquement la désélection du précédent choix du groupe auquel appartient le bouton radio.

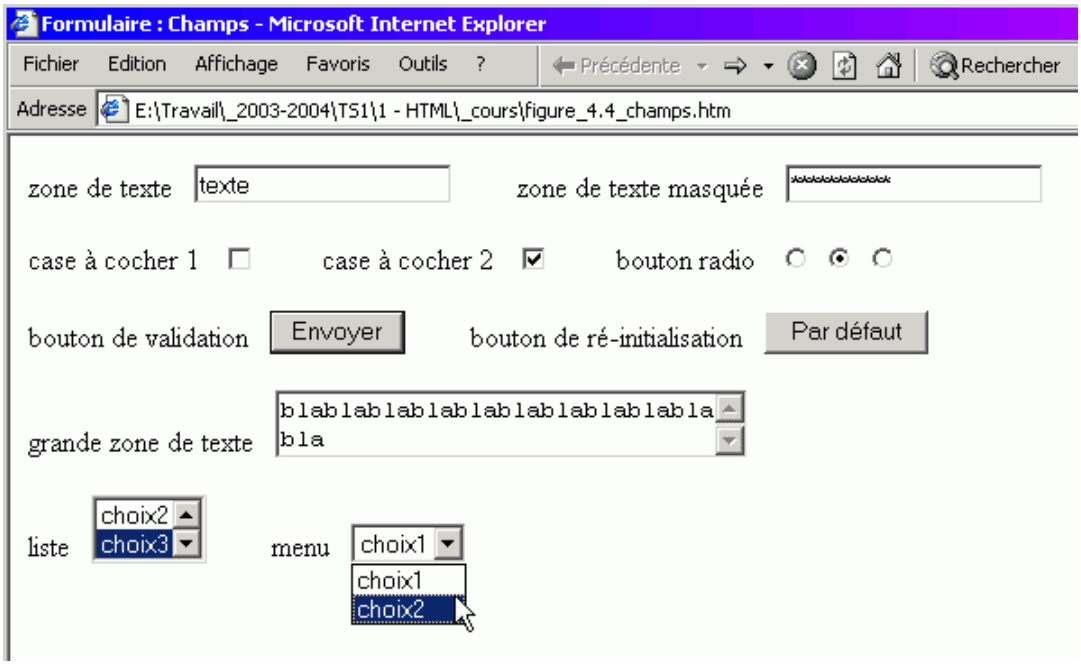


Figure 4.4 : les différents types de champs

Ex. :
```
<form action="hello.php">
      zone de texte
         <input type="text" name="txt" value="texte">
      zone de texte masquée
         <input type="password" name="pwd" value="textemasqué"><br><br>
      case à cocher 1
         <input type="checkbox" name="chkbx1" value="chkbx1_sel">
      case à cocher 2
         <input type="checkbox" name="chkbx2" value="chkbx2_sel" checked>
      bouton radio
         <input type="radio" name="grouperad" value="radio1_sel1">
         <input type="radio" name="grouperad" value="radio1_sel2" checked>
         <input type="radio" name="grouperad" value="radio1_sel3"><br><br>
      bouton de validation
         <input type="submit" name="valid" value="Envoyer">
      bouton de ré-initialisation
         <input type="reset" name="rst" value="Par défaut"><br><br>
      grande zone de texte
         <textarea rows=2 cols=30>blablablablablablablablabla</textarea><br><br>
      liste
         <select size=2 multiple>
            <option>choix1</option>
            <option>choix2</option>
            <option selected>choix3</option>
         </select>
      menu
         <select>
            <option>choix1</option>
            <option>choix2</option>
         </select>
   </form>
```

4.3 EXPLOITATION DES DONNÉES DU FORMULAIRE

Une fois les champs remplis, le formulaire est validé par action sur le bouton de validation.

Cette validation a pour conséquence 2 actions :

- l'ensemble des valeurs de chacun des champs qui constituent le formulaire est transmis au serveur web ; ces valeurs vont ainsi pouvoir être utilisées ;
- le document mentionné par l'attribut `action=` de la balise `<form>` est exécuté ; la valeur de cet attribut est un nom de fichier pouvant correspondre à :
  - un document HTML : le document HTML est alors appelé et affiché ;
  - un fichier script.

Un document HTML décrit une page web en précisant le contenu ainsi que sa mise en page ; la structure de celle-ci sera donc fixe quel que soit l'utilisateur qui la visualisera via internet [1]. On parle dans ce cas de langage **statique**, de programmation statique, et de site web statique.

Un **fichier script** est un fichier écrit dans un langage de programmation, dit **langage de script**, qui permet de réaliser des actions, prendre en compte certains paramètres [2], traiter des informations, effectuer des opérations, etc. ; ce qui permettra ainsi de mettre en place des pages personnalisées. On parle alors de langage **dynamique**, de programmation dynamique, de site web dynamique.

Les deux types de programmations sont indissociables l'une de l'autre. Le langage HTML a besoin d'un langage de script pour traiter des informations ; le langage de script a besoin du langage HTML pour afficher les informations traitées.

Lors de l'utilisation de langages de script associés à un formulaire, on utilise un fichier script qui est exécuté par l'ordinateur serveur. On parle alors de **langage de script serveur**. Lorsque le fichier script est écrit dans un langage exécuté par le client, on parle dans ce cas-là de **langage de script client**.

Ex. : différents langages de script serveur : PHP, script CGI, ASP, JSP, Perl, ...
différents langages de script client : JavaScript, VBScript.

Il existe 2 méthodes différentes pour transférer les valeurs des champs du client vers le serveur. On spécifie la méthode employée grâce à l'attribut `method=` de la balise `<form>` qui peut prendre les valeurs suivantes :

- `get` : les valeurs des champs sont transmises directement au fichier script via l'URL (ex. : `http://www.monserveur.com/monscript.php?nom=Durand&prenom=Jean`) ;
- `post` : les valeurs des champs sont transmises au script par une séquence spéciale [3].

La méthode `post` est la plus sécurisée et est généralement préférable.

Le fichier script récupère ainsi les valeurs des champs qui ont été transférées au serveur. Ces champs sont représentés sous la forme de variables affectées d'une valeur.

Pour chaque balise `<input>` (hormis les boutons) ou `<select>` du formulaire, on dispose alors d'une variable dont le nom correspond à l'attribut `name=`, et qui a pour valeur la valeur du champ lors de la validation du formulaire.

Ex. :
```
<body>
   <form action="hello.php">
      Entrez votre nom<br>
      <input type="text" name="votrenom"><br>
      <input type="submit" value="Valider">
   </form>
</body>
```

Dans le script *hello.php*, on va donc avoir à sa disposition 1 variable [4] qui aura pour nom `votrenom` et dont la valeur sera la valeur lors de la validation du formulaire (en l'occurrence `Gérard` dans notre exemple).

Ce qui, en algorithmique correspondrait à l'affectation suivante : `var votrenom ← "Gérard"`.

Parmi les différents langages de script existants, on peut faire ressortir le langage PHP qui est à la fois, simple, puissant et standard.

[1] Aux différences d'affichage des différents navigateurs près.
[2] Au contrario du langage HTML qui lui n'est que descriptif.
[3] Laquelle ne sera pas précisée ici.
[4] Les boutons servent à la gestion du formulaire, et aucunement à envoyer des données ; ils ne renvoient donc pas à des variables dans le script.

A BIBLIOGRAPHIE

Wazir Dino, *Cours HTML*, TS IRIS – LEGT Louis-Modeste Leroy – Évreux, 2002 ;

CommentÇaMarche.net, http://www.commentcamarche.net/, 2004 ;

Wikipedia – l'encyclopédie libre, http://fr.wikipedia.org/, 2005.