HTML & Java Script

Table des matières

Chapitre 1

Introduction

Les langages du Web, introduction à la deuxième édition

HTML est et restera incontestablement le langage de base du Web. Parallèlement à son évolution, apparaissent de nouveaux langages apportant plus d'interactivité aux écrans du Web, et d'offrir une intelligence locale aux postes de consultation; le Web client-serveur devient une réalité.

A l'origine : le langage HTML

C'est un langage de description permettant de structurer et d'afficher différents objets sur un écran (qu'on appellera abusivement "page"). Ces objets peuvent être du texte, des tableaux, des images voire de la vidéo et des sons. Ils peuvent être aussi les éléments devenus classiques des environnements graphiques, à savoir les boutons, pop-listes, listes à ascenseurs, boîtes de dialogue... Sur la plupart de ces objets, il est possible de placer des liens qui vont permettre de se connecter à d'autres pages. Ce langage est donc à l'origine essentiellement statique. Sur le serveur, le fichier HTML décrit une "scène" qui sera envoyée à la demande vers un client (le *browser,* logiciel de navigation présent sur le poste de consultation). Ce client va décoder les instructions HTML pour afficher la scène qui restera ensuite la même ; seule une nouvelle requête vers le serveur sera à même d'en modifier l'apparence.

Le Web, support d'applications

Développer des applications informatiques, c'est aussi se demander qui utilisera ces applications et sur quelles plates-formes. Déployer une application est aussi important et souvent aussi long que de produire l'application elle-même. Le succès du Web et de

l'Intranet est dû pour beaucoup au fait qu'une application qui s'exécute dans le cadre d'un browser Web fonctionnera aussi bien sous Windows 95, Windows NT, MacOS, Unix... On crée l'application sans souci du poste client, celle-ci devenant immédiatement disponible pour tout poste disposant d'un *browser* Web. Si l'on détient enfin le client universel pour toutes les applications, il devient nécessaire de se préoccuper de l'intelligence que l'on peut intégrer. HTML peut alors être insuffisant : il faut programmer...

Le langage JavaScript

Créé à l'origine par Netscape, ce langage de programmation est conçu pour traiter localement des événements provoqués par le lecteur (par exemple, lorsque le lecteur fait glisser la souris sur une zone de texte, cette dernière change de couleur). C'est un langage interprété, c'est-à-dire que le texte contenant le programme est analysé au fur et à mesure par l'interprète, partie intégrante du *browser,* qui va exécuter les instructions.
Ce langage a fait l'objet d'une normalisation sous le nom de ECMAScript.

Comment JavaScript se situe-t-il par rapport à HTML ?

HTML permet de décrire des pages, un peu à la manière d'un traitement de texte :

```
"ici un texte gras, aller à la ligne, ici une image, ici une image
permettant d'activer un lien vers une autre page, etc."
```

JavaScript permet de programmer des actions en fonction d'événements :

```
Si l'utilisateur remplit une case de saisie alors
     tester si les caractères tapés sont numériques
     si oui
        enregistrer la valeur
     si non
        envoyer un message d'alerte
```

ou bien d'effectuer des calculs toujours sans aucun recours au serveur :

```
lire la valeur d'une cellule de saisie, multiplier ce nombre par
1,186, afficher le résultat
```

On nomme *script* l'ensemble d'instructions permettant de réaliser une action.

Les domaines d'applications du langage JavaScript peuvent être classés en plusieurs catégories :

- Petites applications simples (calculettes, petits outils de conversions, édition automatique d'un devis par l'acheteur, jeu, etc.).
- Aspects graphiques de l'interface (modification d'images lors du passage de la souris, gestion de fenêtres, fabrication locale d'une page HTML, etc.).
- Tests de validité des données sur les éléments de l'interface de saisie (pour vérifier

qu'une valeur considérée comme obligatoire a bien été saisie, que le champ saisi correspond bien à une date, etc.), gestion complète de l'interface d'une application complexe (création de fenêtres, modification de menus, aide contextuelle, etc.).

Où s'exécute un script[1] JavaScript ?

Depuis l'origine, il est possible de faire exécuter par les serveurs des programmes dont le résultat est une page Web. Contrairement à ces programmes (les scripts du CGI qui seront étudiés dans ce livre), les "JavaScripts" sont chargés avec la page Web pour être exécutés sur le poste client par le logiciel de navigation (le *browser* Netscape ou Internet Explorer). Cette notion d'exécution locale est importante. En effet, avec HTML le poste client n'avait pas d'intelligence ; il se bornait à afficher une page telle qu'elle était décrite dans le fichier provenant du serveur. Avec JavaScript, on exécute localement un programme téléchargé lui aussi depuis le serveur ; ce programme est par définition une entité capable de prendre des décisions en fonction d'événements survenant localement, une partie de l'intelligence résidant sur le poste client.
Les *browsers* intègrent donc un interpréteur qui décode les instructions et les exécute au moment opportun, soit par exemple au chargement de la page dans le *browser,* soit à l'apparition d'un événement particulier (clic sur un bouton, saisie d'une valeur, mouvement de souris, etc.)

D'où provient le code JavaScript ?

Le code JavaScript est décrit dans le même fichier que le document HTML ce qui augmente encore la simplicité d'écriture. Ce code est par conséquent du langage source[2] et, à l'image du code HTML, il a l'avantage d'être parfaitement lisible par le lecteur curieux de voir comment est codée la page de l'application qui se déroule sous ses yeux.
Lorsque le code HTML d'une page est chargé, les fonctions JavaScript le sont aussi. Elles pourront être réutilisées, sans aucun accès vers le serveur, autant de fois qu'on le souhaite. Imaginons une page qui offre un bouton "calculette"; une fois chargée on pourra utiliser cette calculette en permanence.
La rançon de cette simplicité est qu'aucune analyse préalable du code JavaScript n'est réalisée. Une erreur dans la syntaxe d'une instruction d'un programme mal testé se traduit par l'apparition d'une fenêtre d'erreur...

Le langage JAVA

Nous ne parlerons pas (ou peu) de Java dans ce livre, d'abord parce que Java est un langage qui mérite un livre à lui seul. En outre, écrire une application en Java est très loin du langage HTML. Cela est si vrai que l'affichage produit par une *applet*[3] Java dans la fenêtre du *browser* n'est pas du tout réalisé avec des instructions HTML.

1. Ou un programme, la notion est sensiblement la même.
2. Le langage source ou code source représente en fait les instructions telles que les a tapées le programmeur.
3. Nom donné aux programmes Java s'exécutant dans le cadre d'un browser Web.

Conçu par Sun, puis adopté par la quasi-totalité des constructeurs, Java est un langage de programmation comme JavaScript, mais c'est un langage compilé, c'est-à-dire qu'avant de pouvoir s'exécuter, un programme Java sera préalablement traduit. Cette traduction ou compilation effectue d'abord une analyse syntaxique et vérifie les références ; si le résultat de cette analyse est correct, un code exécutable est généré.
Habituellement, on compile un programme (en langage C, C++, Fortran ou autre) pour une exécution sur une plate-forme et un *operating system* donné. Avec Java, le code produit (appelé le P-code) est indépendant de la plate-forme de production aussi bien que de la plate-forme d'exécution ; tel est le grand avantage. On peut ainsi diffuser des applications sur le réseau sans se soucier du *hardware* qui va les utiliser ! Ce P-code est un code généré non pas pour un processeur (ou une machine) spécifique, mais pour une machine "virtuelle" — la machine Java. Par contre, il est bien évident que cette machine Java est dépendante de la machine physique qui la supporte. Mais, du point de vue du développeur, c'est aussi "transparent" que pour l'interpréteur JavaScript décrit précédemment. C'est le concepteur du *browser* qui a la charge d'implémenter cette machine virtuelle. Le P-code a aussi été conçu pour être transporté correctement sur les réseaux. On peut compiler un programme Java sur un Macintosh qui, une fois téléchargé à travers le réseau Internet, s'exécutera aussi bien sur un *browser* installé sur une machine Unix que sur un *browser* installé sur un PC.
Ce P-code est téléchargé, via une page HTML, depuis le serveur vers le client où il va s'exécuter, mais l'utilisateur devant son écran n'aura plus cette fois la possibilité d'accéder au code source.

Java ou JavaScript ?

Qu'apporte Java en plus de JavaScript ? L'un comme l'autre s'exécutent sur le poste client et sont téléchargés depuis le serveur via une page HTML. Tous deux permettent une programmation événementielle. Au niveau syntaxique, ils ressemblent à C++. L'un est relativement simple et ne demande pas d'outils spécifiques, l'autre est plus complexe et nécessite un compilateur, voire un environnement de développement complet (compilateur, *debbuger* gestionnaire de code, etc).
On considère couramment que JavaScript est un bon assistant des pages HTML, permettant de faire des vérifications ou de petits calculs sur les données d'un formulaire de saisie, d'agrémenter le graphisme d'une page Web ou de charger un *plug-in*[1] du *browser*.
On peut envisager de réaliser de petites applications, mais en JavaScript, la maintenance du code et son *debuggage* sont moins aisés. Enfin, le langage est plus limité que le langage Java (pas de classes et de notion d'héritage).
Java est donc en principe un langage de développeur adapté à des applications plus lourdes (traitement de texte, tableur, etc.). Il est plus riche et plus rigoureux (déclaration obligatoire des variables comme en C ou en C++, possibilité d'accéder aux composants réseaux...). Si une application doit être distribuée sur le réseau Internet avec une contre-

1. Module externe développé selon les règles dictées par le concepteur du *browser* et permettant d'ajouter des fonctionnalités.

partie financière, Java qui ne transfère pas le code source, va permettre d'exclure son "piratage" ou sa copie. Une application en JavaScript peut être plus facilement copiée, conservée, modifiée...

Développer en Java ou en JavaScript présente l'immense avantage de créer des applications indépendantes de la plate-forme d'exécution. Dans un cas, le code est exécuté par la machine virtuelle Java intégrée au browser, dans l'autre, le code est interprété par l'interpréteur JavaScript intégré lui aussi au browser. On laisse ainsi aux éditeurs de logiciels (Netscape, Microsoft, etc.) le travail de portage de leurs *browsers* sur les différentes plates-formes.
Enfin, la question du choix d'un langage pour une application sur le Web ne se pose pas réellement. C'est le mariage des trois technologies HTML, JavaScript et Java qui va permettre de faire de véritables applications pour le Web.

Type de page	HTML	JavaScript	Java	*CGI*[a]
Serveur institutionnel, pages de présentation, publicités informations...	OUI	éventuellement pour enrichir le décor	NON	NON
Pages de formulaires, accès à des bases de données.	OUI	OUI	NON ou éventuellement	OUI
Applications complexes, traitement de texte, tableur, outil de dessin...	OUI	éventuellement	OUI	NON
Exécution par le	client	client	client	serveur

Figure 1 - Types de pages Web et langages nécessaires

a. Le CGI n'est pas un langage, mais la zone du serveur où sont stockés les programmes exécutés par le serveur.

HTML 3.2, 4, Dynamic HTML ?

Le niveau d'HTML que l'on pourrait considérer comme actuellement[1] officiel est le niveau 3.2, mais les principaux *browsers*, comme Netscape ou Internet Explorer, peuvent être en avance sur les normes et implémenter de nouvelles fonctionnalités. La plupart des pages Web qui voient le jour actuellement sont écrites en utilisant ces spécificités, principalement au niveau de la mise en page et des artifices de présentation.
Ce manuel est fondé plutôt sur HTML 3.2. Il inclut ce que nous avons pu tester des feuilles de style et de Dynamic HTML. Nous avons inclus la balise *layer* de Netscape car

1. Décembre 1997

elle nous semble être ce qu'il y a de plus dynamique dans HTML, même si cette approche n'est pas encore "normalisée" ; lors de la première édition, nous nous étions posé les mêmes questions concernant les *frames.* Notre objectif reste toujours de fournir un manuel pratique décrivant des concepts qui fonctionnent réellement sur les browsers de référence.

Comment étudier avec ce manuel (ou sans) ?

On peut distinguer dans cet ouvrage une partie HTML pure ne nécessitant pas de connaissance particulière en informatique, une partie JavaScript nécessitant la compréhension d'un langage de programmation simple, et enfin une partie traitant de la programmation au niveau du serveur, le CGI. Ces trois parties sont clairement identifiées. Dans l'apprentissage des balises HTML, l'aspect programmation JavaScript peut être ignoré par le lecteur ne s'intéressant qu'à la fabrication de pages statiques.
Si vous disposez d'un micro-ordinateur, installez un échantillonnage de logiciels client (Netscape, Internet Explorer par exemple) et testez les différents exemples du manuel et leurs résultats sur ces différents *browsers.* Essayez des exemples, modifiez-les, examinez le résultat de ces modifications ; c'est une excellente méthode d'apprentissage. Il n'est pas nécessaire pour cette étape d'être connecté au réseau Internet. C'est possible tant que nous n'abordons pas la programmation CGI ; par la suite, il sera nécessaire d'être dans l'environnement d'un serveur.
Si vous êtes connecté à Internet, vous pourrez très vite vous passer de ce manuel et examiner, grâce à un *browser,* le code source HTML des millions de pages que vous avez à votre disposition. Accéder à toute la documentation en ligne sur le Web est le plus sûr moyen d'être toujours au fait de ces nouvelles techniques.
Chaque fois que vous trouverez une page sur le Web qui vous séduit ou vous intrigue allez voir le code source, c'est un excellent moyen d'apprentissage !

Comment est décrit JavaScript ?

Nous avons décidé d'intégrer au fur et à mesure de la description des balises HTML l'apport que peut avoir JavaScript sur la balise considérée. Cette partie sera clairement indentifiée par une icône et pourra être ignorée par le lecteur ne voulant pas se préoccuper de l'aspect programmation. Si vous vous intéressez à cet aspect vous devrez avoir lu précédemment le chapitre consacré au langage JavaScript.

Comment est rédigé ce manuel ?

Avec quelques termes anglais. Nous avons préféré le terme *browser* à celui de "navigateur" et nous avons utilisé le terme balise plutôt que *tag* !
Tous les affichages des divers exemples ont été capturés dans un environnement X Window et principalement depuis Netscape sur plate-forme Unix. Nos serveurs sont installés sur des machines Unix et utilisent les serveurs du NCSA et du CERN. Cela explique le choix de nos exemples.

En revanche, c'est dans un environnement Macintosh que nous aborderons l'atelier de fabrication des dessins et des illustrations.

Conventions d'écriture

La syntaxe du langage sera inscrite de cette façon,

```
Les exemples de code de cette façon.
```

Icônes

Un jeu d'icônes présenté sur la page suivante est utilisé pour différencier les langages utilisés, le niveau d'HTML, ainsi que le *browser* utilisé.

Code HTML standard. Cette icône définit un exemple complet pouvant être reproduit et testé sur n'importe quel browser (en principe !).

Code Dynamic HTML . L'exemple inclura des scripts JavaScript.

Code HTML utilisant des extensions propres à Netscape. L'exemple ne pourra être interprété correctement que sur un browser Netscape.

Affichage du fichier HTML à l'aide du browser Netscape.

Affichage du fichier HTML à l'aide du browser Netscape version 4 minimum.

Source d'un programme écrit en langage PERL.

Source d'un programme écrit en langage PLSQL.

Source d'un programme écrit en langage shell (shell, cshell, kshell...).

Code HTML incluant du code JavaScript ou description de fonctionnalités JavaScript.

Source d'un programme écrit en langage C.

Ligne unique de code imprimée sur plusieurs lignes pour des raisons de mise en page.

Environnement de test et de production

Il comprend quatre zones :
La zone 1 se compose d'un simple poste (pas forcément connecté au réseau) sur lequel on trouvera un ou plusieurs browsers installés ainsi qu'un simple éditeur de texte. La machine peut être indifféremment un Mac, un PC, ou un terminal X. Cette première zone est suffisante pour apprendre le langage HTML de base.
La zone 2 est identique à la zone 1, mais les machines sont connectées au réseau. C'est depuis cette zone que l'on va tester les codes HTML et les scripts développés que l'on aura préalablement installés en zone 3.
La zone 3 est la zone serveur. Elle se compose d'une machine Unix sur laquelle se trouvent tous les logiciels nécessaires au fonctionnement d'un serveur Web, tous les répertoires contenant les fichiers HTML, les images, les films, les sons ainsi que la zone où s'exécutent les scripts.
La zone 4 est l'atelier de PAO dans lequel on va réaliser les images, les sons et les films.

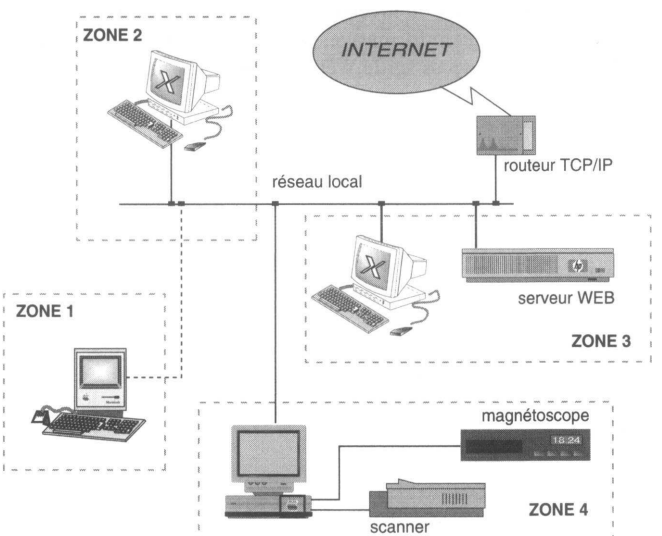


Figure 2 - L'environnement de travail

Faut-il encore apprendre HTML ?

Plus que jamais ! En effet, c'est dans les fichiers HTML et avec des instructions HTML qu'on appelle des *applets* Java et qu'on décrit des scripts. HTML est le liant entre les différentes techniques existantes ou futures et, à ce titre, il nous semble incontournable.

Chaque jour, la presse rapporte que tel traitement de texte offre dans sa nouvelle version la possibilité de sortie de documents en langage HTML. D'autres produits se proposent de convertir tel format de texte en HTML.
Des éditeurs plus ou moins *wysiwyg* apparaissent çà et là ; souvent, ils n'offrent que la possibilité d'insérer des balises grâce à un clic dans un menu, et tous ne font pas une analyse structurelle du fichier, ce qui serait le plus intéressant. Ils dispensent surtout de connaître le balisage et les règles de structuration.
Tous ces produits n'indiquent pas toujours clairement quel niveau d'HTML ils sont capables de traiter ou de produire. Là est peut-être le problème le plus crucial, car lorsque l'on génère de la documentation pour le Web, il est important de savoir à quel niveau d'HTML on va se conformer, niveau que l'on peut d'ailleurs admettre différent selon le type de pages que l'on traite dans son serveur.
Il reste enfin que toute la partie interactive (les formulaires ou la fabrication de scripts) n'est pour l'instant accessible qu'en travaillant au niveau du langage.
Il ne faut pas pour autant négliger ces logiciels, qui permettent souvent de peupler les pages d'un serveur en partant de documents existants.
Tout cela dépend bien sûr du niveau que l'on souhaite donner aux pages de son serveur : pour réaliser des pages informatives, un assistant HTML peut suffire, pour mettre en place de véritables services sur le Web, il vaut mieux maîtriser ces langages.

Chapitre 2

L'hypertexte

On pourrait définir l'hypertexte comme un système qui relie par des liens activables des éléments d'information.
Ce principe est en partie apparu avec l'édition et la consultation de textes sur un écran d'ordinateur. La taille limitée de ces écrans, inférieure à une page imprimée, le passage d'un écran au suivant plus lent que l'action de tourner une page, et la constatation qu'à un moment donné, le lecteur n'a besoin que d'une faible portion de l'information, ont fait naître un système à base de boutons, où un clic sur une partie du texte remplace celle-ci par un ensemble plus détaillé.
La pratique de la lecture sur des médias imprimés constitue la base des pratiques d'écriture hypertextuelle et la métaphore de la page sera souvent employée. Dans le document électronique, cette page pourra avoir en longueur l'élasticité que lui donne l'ascenseur[1] permettant le défilement du texte dans sa fenêtre.
L'auteur d'un document hypertexte sera tenté d'utiliser les mêmes divisions et la même présentation (chapitres, notes, etc.) que lorsqu'il rédige un document destiné à l'impression. Effectuons tout d'abord un rappel sur l'organisation des documents imprimés dans les types ou formats les plus courants :

1. On appel *ascenseur* la facilité, apparue avec les interfaces graphique, de faire dérouler un texte dans une portion d'écran, la fenêtre, à l'aide d'un curseur que l'on déplace en faisant glisser la souris.

Le livre

On accède généralement à un tel ouvrage par la première page, la lecture se poursuivant dans l'ordre chronologique des pages ; le seul déroutement dans la lecture est introduit par les notes de bas de page. Le document se suffit à lui-même, et il n'y aura en principe pas d'accès nécessaire à d'autres ouvrages (éventuellement, on aura recours à un dictionnaire).

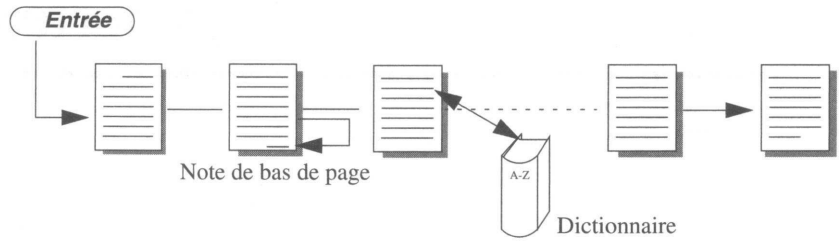


Figure 3 - Accès séquentiel ou linéaire d'un document

Le document technique

Prenons l'exemple de la documentation du traitement de texte Word. A l'exception du manuel d'initiation, il est rare qu'on lise les divers volumes comme un roman. C'est au cours de l'utilisation du traitement de texte que l'on va consulter, selon ses besoins le manuel de référence, celui de l'éditeur d'équations ou encore le fascicule traitant des graphes.

L'accès à l'information va se faire en choisissant d'abord le volume, puis on orientera sa recherche en consultant soit la table des matières, soit l'index pour *naviguer*[1] dans le document. On sera aussi peut-être amené à consulter un glossaire ou diverses annexes afin de compléter la réponse attendue. Une telle structure de document est dite hiérarchique ou arborescente ; sa consultation est plus difficile lorsqu'on travaille sur une édition “papier” que lorsqu'il s'agit d'un document électronique. Ce type de document sera un très bon candidat à une présentation hypertextuelle.

1. *Navigation* est devenu le terme consacré pour illustrer l'action de se déplacer dans un système d'hypertexte.

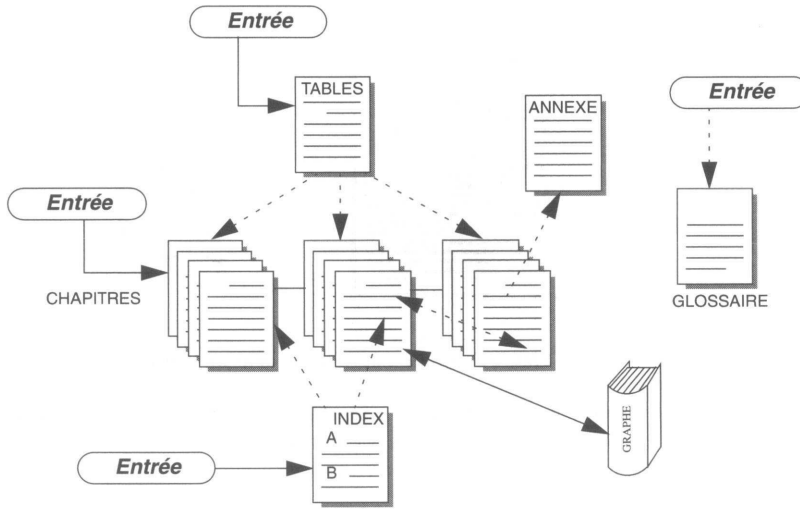


Figure 4 - Structure hiérarchique d'un document

Le journal

Ce type de document correspond à un besoin différent. Nous allons l'étudier, car il est intéressant d'analyser la similitude entre la première page, la “une” d'un journal, et la *home page,* ou page d'accueil, d'un service sur le Web.
La première page d'un journal est conçue pour répondre aux critères suivants :

- identifier le journal (logotype, typographie, mise en page),
- attirer l'œil (couleur, iconographie),
- donner envie d'acheter le journal pour poursuivre la lecture.

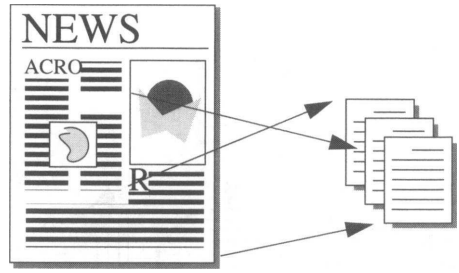


Figure 5 - Liaison des articles dans un journal

On peut lire un journal ou une revue en commençant les articles de première page et continuer la lecture des pages suivantes, en revenant ensuite éventuellement vers la première page pour choisir un nouvel article.

Notions d'hypertexte

Dans les deux derniers types de documents décrits, apparaît le principe de liens entre divers éléments de texte non contigus (une flèche en pointillé représentant les liens à l'intérieur du document, une flèche en trait plein pour des liens externes au document). Dans une forme “papier” des documents, on exécute ces liens en tournant des pages ou en allant chercher sur des rayons des livres au gré des références, notes de bas de page ou autres renvois rencontrés dans la lecture. Il sera chaque fois nécessaire de préserver le retour au point d'interruption de la lecture en marquant la page par un signet. Dans un document électronique, on ira consulter ces liens en cliquant sur un repère spécial ; la sauvegarde du point de retour devra être assurée par une mémoire du système.
On peut considérer qu'hypercard sur Macintosh fut un des premiers systèmes d'hypertexte. On rencontre maintenant des documents hypertexte dans divers environnements :

- sur les CD-Rom où ils sont réalisés avec des logiciels comme Director ou Acrobat,
- dans la documentation en ligne sur les ordinateurs (réalisée avec FrameMaker par exemple sur les stations Unix),
- dans la présentation assistée par ordinateur (PowerPoint),
- sur l'Internet, où l'on peut fabriquer de tels documents à l'aide du langage HTML.

La rapidité de déplacement (un simple clic de souris) à l'intérieur du document et la facilité de retour en arrière (mémoire) permettent une conception nouvelle de la documentation : les pages sont courtes (tenant sur un écran d'ordinateur), vont à l'essentiel, et on offre aussi souvent que nécessaire l'explication d'un mot ou d'un concept en lui attribuant un lien vers une autre partie du document ou vers un autre document. Cette

autre partie peut être un document rédigé par un auteur différent sur un sujet différent. Par exemple, dans un document sur Léonard de Vinci, on trouve un lien vers un document touristique sur Florence.
La première page du document ressemble donc plus à une table des matières où chacun des titres de chapitre est un lien vers le chapitre lui-même. Si le document est complexe (type de document technique), on peut introduire dans la première page une zone de dialogue, dans laquelle le lecteur peut entrer un mot que le système utilisera pour lancer un moteur de recherche automatique dans le document (lien exécutable), et présenter alors une nouvelle page où seront indexées, en liens hypertexte, toutes les occurrences rencontrées.

Ecrire de l'hypertexte

S'il ne possède pas de véritable logiciel de traitement d'hypertexte, l'auteur peut se tourner vers son traitement de texte et concevoir un document selon le schéma habituel. Il peut ensuite tenter une transformation du texte linéaire en hypertexte soit à la main soit avec un logiciel de conversion. Cette méthode très tentante permet de produire simultanément deux versions du document : une version imprimable et une version hypertextuelle.

Du multimédia pour le Web

Pour le Web, on recommande souvent d'offrir une version imprimable (généralement en PostScript ou en PDF) d'un document hypertexte. Si ce document est un manuel technique ou une publication scientifique, il est inutile de se priver de l'aide d'un convertisseur. Mais, traduire directement la plaquette institutionnelle de son entreprise, c'est se priver de toute la richesse que peut apporter un document pensé dans un mode hypertexte (films, sons, navigation à l'aide de boutons, etc.). Ne travailler qu'à l'aide d'un convertisseur de texte, c'est aussi perdre de vue l'interactivité apportée par les formulaires et l'accès aux scripts sur les serveurs Web.
La réalisation de pages pour le Web procède aussi d'un assemblage de matériels issus de divers logiciels : logiciels de dessin (vectoriel et bitmap), logiciels d'acquisition d'images (fixes ou animées) et de sons, traitements de texte, logiciels d'écriture de programmes informatiques, etc., qui seront étudiés par la suite.

RÉSUMÉ

Ce chapitre de généralités sur l'hypertexte introduit la notion importante de **liens :**

Un lien définit une relation entre deux éléments d'information :

- un élément sensible au clic de la souris - une lettre, un mot, un groupe de mots, une image ou une portion d'image. Cet élément est généralement signalé à l'attention du lecteur par un attribut visuel.
- sa cible, qui peut être du texte "plat", un autre document hypertexte, une image, un film ou un son.

On distingue plusieurs types de liens :

- **lien interne** : référence à une partie se trouvant à l'intérieur du même document.
- **lien externe** : référence vers un autre document.
- **lien exécutable** : lien externe déclenchant un programme informatique de traitement de données en réponse à une action effectuée par le lecteur de manière interactive.

Les liens sont la base du Web. La cible ou l'extrémité d'un lien peut se trouver sur n'importe quel ordinateur de la planète.

Chapitre 3

HTML, son environnement

Qu'est-ce que HTML ?

HTML - *HyperText Markup Language* - est un langage simple utilisé pour créer des documents hypertexte pour le Web. Ce n'est pas un langage de description de page tel que PostScript ; HTML ne permet pas de définir de façon stricte l'apparence d'une page. De plus, la présentation de la page peut être dépendante du *browser* utilisé et des options de paramétrage du lecteur.

HTML est une implémentation relativement simple de SGML (*Standard Generalized Markup Language).*

Comme tout langage, il est en constante évolution. La version en cours est la version 3.2, mais il existe déjà un projet pour la version 4.

Sa simplicité est telle qu'il n'est pas nécessaire d'utiliser un éditeur particulier. Sa grande permissivité demande de la rigueur et de l'attention dans l'écriture des documents afin que ceux-ci s'affichent correctement quels que soient le contexte et le *browser* utilisé.

Dans quel contexte s'exécute HTML ?

Le cadre d'exécution du Web et de son langage HTML est celui d'un modèle client-serveur véhiculé sur un réseau informatique comme le réseau Internet.

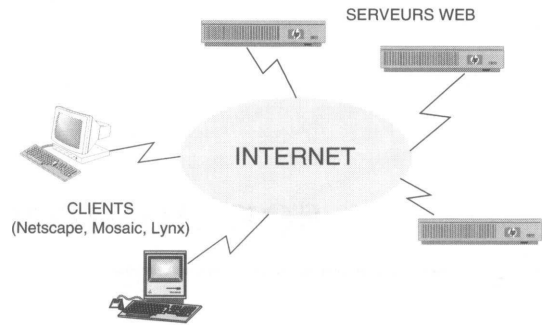


Figure 6 - Clients-serveurs Web

- Le client, ou browser : c'est un logiciel de consultation s'exécutant sur tous les types de plates-formes (Macintosh, station Unix, PC). Il dialogue avec un serveur selon un protocole[1] spécifique (HTTP, *HyperText Transfert Protocol),* qui va lui fournir l'information structurée en code HTML. C'est ce client qui a en charge l'exécution de ce code (le langage source) et qui produira l'affichage du document.

  Le client peut s'entourer d'éléments périphériques pour afficher des documents dont il n'est pas capable d'utiliser le format. C'est le cas pour les films, le son et divers formats d'images dont le format PostScript fait partie.
  Les clients les plus connus sont Netscape et Internet Explorer.

- Le serveur : ce logiciel est en permanence à l'écoute des requêtes que vont formuler les clients. Il en vérifie la validité, s'assure que la demande émane d'un client autorisé à accéder au document, et lui envoie finalement textes et images avant de clore la connexion[2]. Dans le cas d'un lien exécutable, le serveur active un programme dont la sortie sera du code HTML. Ce programme peut être par exemple un module d'interfaçage avec une base de données.

  Un serveur peut être sollicité à chaque instant. Il est donc important que la machine qui l'accueille soit toujours disponible et présente sur le réseau. Bien que l'on trouve des logiciels serveurs pour toutes les plates-formes (y compris Macintosh et PC),

1. Un protocole est un ensemble de régies et de messages régissant le dialogue entre deux processus informatiques.
2. Il est important de savoir qu'avec le Web, lorsque l'on se connecte à un serveur, la connexion établie ne dure que le temps de l'envoi de la page demandée. Pour chaque page demandée au même serveur, on recommence la procédure de connexion.

dans la majorité des cas ceux-ci seront implantés sur des stations de travail (Unix ou Windows NT principalement). La puissance de ce type d'ordinateurs permet d'accepter simultanément un grand nombre de requêtes.
Les logiciels serveurs couramment utilisés sont ceux qui sont fournis gratuitement par le CERN, le NCSA, ou APACHE. Dans la terminologie Unix, on les appelle *démons* (HTTPD, *HyperText Transfert Protocol Daemon*)

Le protocole d'adressage des documents - l'URL

Interconnecter des documents sur toute la planète implique un moyen unique de les identifier sur le réseau Internet. L'adresse d'un document, appelée URL, *Uniform Resource Locator* se compose à l'aide des éléments suivants :

- le protocole d'échange entre le client et le serveur. A ce stade, le seul protocol que nous ayons rencontré est HTTP ; nous verrons plus tard qu'il en existe d'autres ;
- l'adresse Internet du serveur qui diffuse les documents. Cette adresse unique sur tout le réseau est l'adresse TCP/IP de la machine. Elle a la forme d'une suite de nombres telle que 134.158.69.113 ; comme ces nombres ne sont pas facilement mémorisables, un annuaire (DNS) résout généralement la relation adresse numérique, nom de la machine (exemple : 134.158.48.1 est l'adresse de la machine fourviere.lyon.fr, fourviere représente le nom de la machine et .lyon.fr le nom du domaine) ;
- l'arborescence des répertoires (le chemin) qui conduit au document ;
- le nom du document qui aura toujours l'extension .html ou .htm.

Moins fréquemment, cette adresse pourra être complétée par d'autres éléments :

- le port[1];
- des informations d'authentification *(username* et *password)* ;
- des arguments qui seront passés à un programme lors de l'appel de liens exécutables.

La syntaxe minimale utilisée pour représenter l'URL d'un document est la suivante :

protocole://nom_du_serveur/

Lorsqu'aucun nom de fichier n'est précisé, on arrive sur le fichier par défaut du serveur, habituellement la *home-page*.
La syntaxe que l'on rencontre couramment est :

protocole://nom_du_serveur/repertoire/sous-repertoire/nom_du_document

1. Dans le réseau TCP/IP, les différents services du réseau sont identifiés par un numéro. Par défaut, le numéro de port attribué à HTTP est 80. Mais, lors de l'installation du logiciel, il est possible de choisir un autre numéro. Dans ce cas, il devra explicitement être indiqué dans l'URL du document.

La syntaxe complète est :

protocole://username;password@nom_du_serveur:port/sous-repertoire/ ➥ nom_du_document?arguments

On remarque, dans certaines adresses, la présence d'un tilde (~) devant le nom d'un répertoire. Il s'agit de ***home-pages*** personnelles, possibilité offerte aux utilisateurs ayant un compte sur la machine serveur.

Exemples d'URL :

```
http://www.fnac.fr
http://www.ra.net/routing.arbiter/NFSNET/NFS.transition.html
http://www.inter.fr:2000/
http://www.in2p3.fr/~dupont/index.html
```

Où sont stockés les documents HTML ?

Sur un disque de la machine serveur. Il appartient à l'administrateur du système de faire connaître l'organisation des répertoires qu'il a choisie, et de donner les droits nécessaires pour que les personnes autorisées puissent y déposer leurs fichiers (textes, images, etc.).
Lorsque le fichier est placé dans le répertoire, il faut que celui-ci soit en lecture pour tout le monde.
Remarquez, dans l'exemple de la figure 8, concernant un serveur sur une machine Unix, que le "top niveau" de l'espace disque du serveur n'est pas le "top niveau" de l'espace disque réservé au serveur.

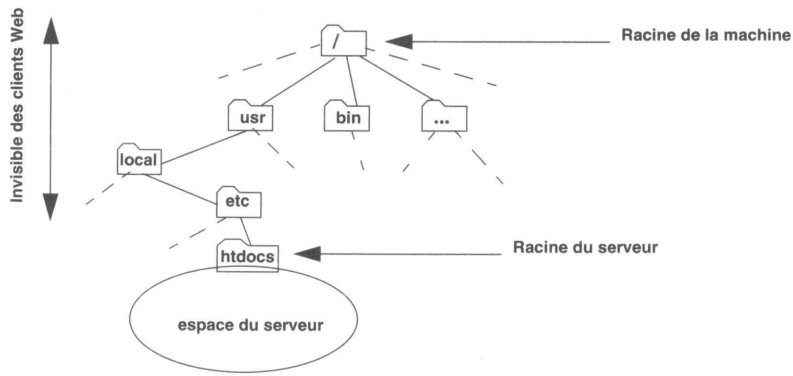


Figure 7 - Localisation des fichiers du serveur dans une machine Unix

Quelles protections pour les documents ?

L'accès à un document (ou à un ensemble de documents) peut être réduit grâce à un système de protection développé pour le Web. Il ne faut pas confondre les protections Unix du fichier et celles du Web : au sens Unix, on permettra toujours la lecture et éventuellement l'exécution du fichier ; au sens Web, le système permettra d'autoriser la lecture d'un document :

- aux lecteurs enregistrés, ayant un *username* et un *password.* Il ne s'agit pas dans ce cas d'avoir un compte (au sens Unix) sur la machine serveur. Le système de protections des serveurs Web permet d'enregistrer des utilisateurs qui auront accès à des documents protégés ;
- aux lecteurs accédant depuis un domaine particulier (au sens TCP/IP) ;
- aux lecteurs accédant depuis une machine particulière.

RESUME

HTML est un langage permettant de décrire la **structure** et la présentation des documents pour le Web.
Ce chapitre définit l'environnement **client-serveur** dans lequel s'exécute ce langage.
Le protocole d'échange des données entre le serveur et les clients s'appelle **HTTP.**
L'adresse **unique** d'un document Web est appelée **l'URL** du document.
Les fichiers ou documents que diffuse le serveur sont stockés dans **des répertoires** du serveur.
Il existe un système de protection des fichiers **propres** au Web. Il comporte **plusieurs** niveaux de protections.

Le langage HTML

Chapitre 4

Structure

Les balises

Dans le chapitre précédent, nous avons défini le langage HTML comme un langage permettant de décrire la structure d'un document Web. Pour ce faire, il faut encadrer chacune des différentes structures du texte par une paire de balises, une au début et une autre à la fin. Ces balises seront bien sûr invisibles au moment de l'affichage du document.
Dans un traitement de texte, lorsque l'on veut mettre un mot en gras, on sélectionne les caractères, puis on clique généralement sur une petite icône qui a pour effet de les faire passer dans la graisse choisie. De façon interne et sans que cela apparaisse à l'écran, le traitement de texte inscrit dans le fichier un code indiquant qu'une zone en caractères gras commence, puis viennent ensuite les caractères en question ; enfin un nouveau code indique le retour aux caractères standard.
Ce sont ces codes que l'on appelle des balises. Dans le langage HTML, la gestion des balises est à la charge de l'auteur.
Les balises sont délimitées par les signes < (inférieur à) et > (supérieur à). Un texte balisé aura donc cette apparence :

...texte courant <balise>texte affecté par la balise</balise> suite du texte ...

Remarquez le caractère / dans la balise de fin.
Voici un exemple de saisie pour passer un groupe de caractères en gras dans le langage HTML (utilisation de la balise <B> pour *bold* ou gras) :

```
texte maigre <B>texte gras</B> texte maigre
```

Voici l'affichage correspondant :

texte maigre texte gras **texte maigre**

Si l'on veut utiliser les deux caractères <> dans le texte courant, il faudra trouver un artifice afin que le browser ne tente pas de les interpréter comme balises. On les remplacera par leurs entités HTML équivalentes :

le caractère > sera remplacé par **>**
le caractère < sera remplacé par **<**

Ainsi,

```
Les signes &lt; et &gt; délimitent les<b>balises</b> HTML
```

produira dans la fenêtre du browser :

Les signes < et > délimitent les balises HTML

Le texte à l'intérieur des balises peut être indifféremment en majuscules ou en minuscu-

<a> ou <A> sont strictement équivalents.

L'accentuation

L'enregistrement des fichiers contenant les documents HTML est effectué sur la machine serveur en code ASCII 7 bits[1], ce qui ne permet pas d'accentuer directement le texte. On utilisera donc des mots-clés pour tous les caractères accentués et autres symboles, de façon identique au traitement des caractères <> décrits au paragraphe précédent.
L'annexe **Caractères accentués, symboles,** page 439 donne les codes des mots-clés pour l'alphabet ISO Latin 1.
Ainsi, la phrase

" Le Naïf " a déjà été créé au théâtre français.

sera codée en HTML par :

1. Le code ASCII 7 bits définit sur les 7 bits de poids le plus faible d'un octet un jeu de 128 caractères incluant les chiffres, les minuscules, les majuscules, un certain nombre de signes ainsi que des caractères de contrôle non imprimables. Le huitième bit sert au contrôle de parité pour valider l'exactitude de la transmission. Il ne reste pas assez de place dans ce code pour représenter les caractères accentués.

"" Le Naïf" a déjà été créé au théâtre français.

Structuration du document HTML

<HTML>

Tous les documents HTML commenceront par la balise <HTML> et se termineront par la balise </HTML>.
La structure de premier niveau du document aura donc la forme suivante :

```
<HTML>
 Corps du document

</HTML>
```

<HEAD>

Le titre du document, les scripts JavaScript seront généralement inclus dans une zone de prologue ou d'en-tête : <HEAD>. Cette zone se termine par </HEAD>. Nous verrons par la suite que nous pouvons trouver d'autres termes dans cette zone d'en-tête.

```
<HTML>
 <HEAD>
     prologue
 </HEAD>
     Corps du document

</HTML>
```

<TITLE>

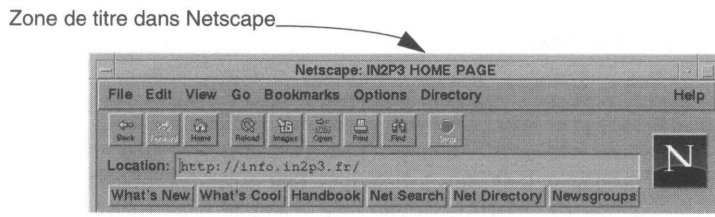
Un texte qui apparaît dans la zone titre du *browser* est défini par <TITLE>. Il faut noter que ce titre est souvent négligé au détriment d'une zone de titre enrichie d'images et de logos, que l'auteur situe au début du document. Le titre défini entre les balises <title> et </title> a, outre un caractère informatif, au moins deux raisons d'être soigné :

- C'est ce texte qui sera stocké dans le fichier *bookmark*[1] que gère le *browser* pour le compte du lecteur. On préférera donc un texte du style « Bienvenue au CNRS » à « Bienvenue », qui ne sera jamais capable de fournir une indication pertinente sur la destination.
- Dans un contexte XWindow, lorsque l'on "iconifie" la fenêtre, c'est ce titre qui sera pris comme nom par l'icône.

1. Ce fichier permet, lorsqu'on est connecté à un document, de conserver son URL dans un fichier de type annuaire afin de pouvoir s'y connecter ultérieurement d'un simple clic.

Le titre doit être suffisamment court pour tenir entièrement dans l'espace que le *browser* lui réserve, sous peine d'être tronqué et ainsi de perdre son sens.

```
<HTML>
  <HEAD>
      <TITLE>Bienvenue au CNRS</TITLE>
  </HEAD>
      Corps du document

</HTML>
```



<BODY>

Entre <BODY> et </BODY> réside tout le reste du document. Nous verrons par la suite (voir **<BODY> pour le décor** !, page 92) que des extensions permettent de compléter l'utilisation de cette balise.


```
<HTML>
  <HEAD>
      <TITLE>Bienvenue au CNRS</TITLE>
  </HEAD>
      <BODY>
          document

      </BODY>
</HTML>
```

Ce dernier exemple donne la structure minimale que doit posséder tout document HTML. Dans ces exemples, on a présenté le code HTML en décalant les divers niveaux de la structure. Cette indentation n'est pas obligatoire, mais elle est fortement recommandée afin de faciliter la lecture et la maintenance du code source. Les *browsers* qui exécutent le code HTML ne tiennent pas compte des retours à la ligne et l'on aurait très bien pu écrire le code de la façon suivante, qui aurait produit le même affichage dans le *brow-*

```
<html><head><title>Bienvenue au CNRS</titlex/head>
<body>
document
 ....</body></html>
```

<ADDRESS>

Le bloc adresse est un élément prévu pour indiquer toute information concernant l'auteur du document (adresse, téléphone, e-mail, etc.). Il peut être inséré à n'importe quel endroit du document ; l'habitude sur le Web est de mettre cette zone adresse à la fin du document. Les *browsers* affichent généralement le texte compris entre les balises <address> et </address> en italique.

Exemple :

```
<address>
 Jean DUPONT - T&eacute;l 78-90-12-34 - e-mail dupontj@sdf.fr
</address>
```

<!-- Commentaires -->

Tout texte commençant par <!-- et se terminant par --> ne sera pas interprété par le *browser,* donc pas affiché. Cela sert à l'auteur du document pour commenter son fichier source.
Exemple :

```
<!-- Document à ne diffuser qu'aux machines appartenant au domaine
 .aramis.fr -->
```

Attributs d'une balise

Nous verrons, au cours de l'étude des différentes balises HTML, que certaines peuvent admettre des attributs, chacun d'eux pouvant avoir une valeur. Au niveau syntaxique, le premier attribut est séparé de la balise par un espace, ainsi que les attributs entre eux.
Si l'attribut doit prendre une valeur, elle est spécifiée après le signe = (égale) suivant l'attribut.
La valeur de l'attribut sera écrite entre guillemets (") si cette valeur est alpha-numérique.

```
<balise attribut_1 =numerique attribut_2="alpha-numerique">
```

Exemple :

```
<pre width=50>
<a href="/home/default.html">
```

Le document HTML minimum

Nous avons maintenant défini suffisamment d'éléments de structure du langage HTML pour lister l'ossature type que nous trouverons toujours dans un document HTML :


```
<HTML>
 <!-- Squelette HTML -->
 <HEAD>
     <TITLE>Emplacement du titre</TITLE>
 </HEAD>
 <BODY>
     Document
     ...
     <ADDRESS>
         Jean Dupont - dupont@cnrs.fr
     </ADDRESS>
 </BODY>
</HTML>
```

Tester les documents HTML

La finalité d'un fichier contenant un document HTML est son installation dans les répertoires du serveur afin qu'il soit offert à la consultation. Ce n'est pas envisageable pendant la phase d'apprentissage du langage ou pendant la mise au point d'un document.
On éditera[1] donc ces fichiers HTML de test et on les stockera dans un répertoire personnel si l'on travaille sur une machine Unix, ou dans un dossier de test si l'on travaille sur un PC ou un Macintosh.

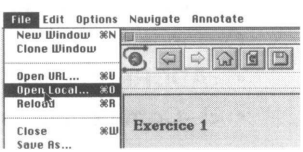
1. On pourra utiliser n'importe quel éditeur : TeachText comme Word feront l'affaire sur un Macintosh ; cependant, si l'on prend un traitement de texte évolué comme Word, on aura soin de sauvegarder le fichier en format "TEXT".

Il suffit que soit installé sur cette machine d'apprentissage un logiciel client comme Netscape. Il n'est absolument pas nécessaire pendant cette période d'avoir accès à un serveur Web ni même d'être connecté au réseau.
En effet, les *browsers* offrent la possibilité d'ouvrir et d'interpréter un fichier local. Il suffit de cliquer sur le menu déroulant **File** du *browser* et de sélectionner l'option **Open file.** On reçoit alors une nouvelle fenêtre permettant de sélectionner le fichier à tester.

Méthode de travail

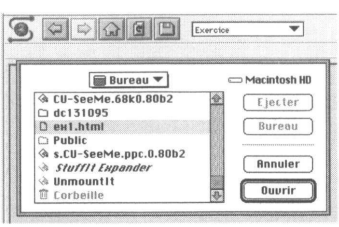
Voici les étapes à suivre pour tester les fichiers HTML avant de les installer dans les répertoires du serveur :

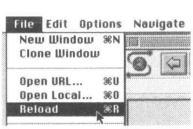
- appeler l'éditeur de texte
- écrire le code html,
- enregistrer sous un nom avec le suffixe **.html,**
- appeler le browser Netscape par exemple,
- utiliser le menu **Open file,**
- sélectionner votre document, qui s'affichera dans le *browser.*



Si vous n'êtes pas satisfait...

- retourner dans l'éditeur de texte sans quitter le *browser,*
- corriger ou modifier votre code html,
- utiliser la commande **reload** et juger vos modifications.





RÉSUMÉ

HTML est un langage balisé où chaque balise s'inscrit entre les caractères < (inférieur à) et > (supérieur à). Les balises peuvent avoir des attributs spécifiant éventuellement des valeurs ou des caractéristiques. On ne peut pas saisir des caractères accentués directement au clavier ; il faut les remplacer par les codes HTML équivalents ; il en va de même pour certains symboles. Les délimiteurs pour ces codes sont le caractère **&** (et commercial) au début du code et le caractère ; (point virgule) à la fin. Ainsi **é** signifie **é** (e accent aigu).

Les éléments de structure définis dans ce chapitre sont de deux sortes :

Obligatoires

- **<HTML>** premier élément encadrant tout le fichier HTML
- **<HEAD>** zone d'en-tête
- **<TITLE>** définition d'un titre
- **<BODY>** corps du document

Optionnels

- **<ADDRESS>** bloc d'information sur l'auteur du document
- **<!--Commentaires--->** commentaires

Chapitre 5

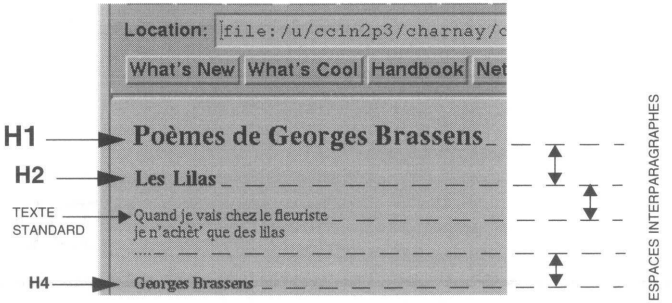
Divisions

Tout texte technique ou scientifique est habituellement divisé selon des paragraphes subissant une numérotation hiérarchique du type 1.1.1, 1.1.2, 1.1.3, 1.2.1, etc. La plupart des traitements de texte gèrent ces formats de numérotations automatiques en affectant la taille des caractères d'un titre en fonction de son niveau dans la hiérarchie.
Dans HTML, encore une fois, tout est à la charge de l'auteur. Celui-ci dispose de six grosseurs de titre qui vont lui permettre de diviser son document et d'un système de liste (voir **Listes,** page 39)

<Hn>

C'est la balise affectant la taille des caractères dans laquelle *n* varie de 1 à 6. Les plus gros ont la valeur 1 et les plus petits la valeur 6. Le texte entre ces balises est traité en caractère gras.
La balise de début et la balise de fin génèrent automatiquement un espace de type nouveau paragraphe (passage à la ligne avec espace correspondant environ au saut d'une ligne).

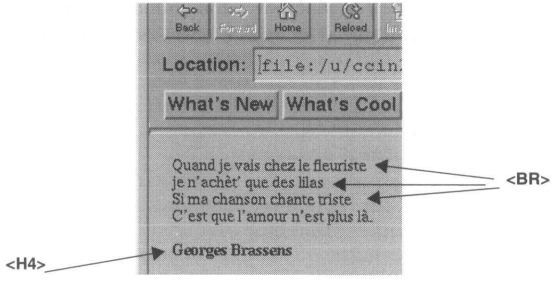
```
<h1>Po&egrave;mes de Georges Brassens</h1>
<h2>Les Lilas</h2>
Quand je vais chez le fleuriste<br>
je n'ach&egrave;t' que des lilas<br>

<h4>Georges Brassens</h4>
```



Lorsqu'ils exécutent du code HTML, les *browsers* traitent le texte au kilomètre dans la largeur de leur fenêtre et ignorent le découpage en ligne tel qu'il existe dans le fichier source. La longueur des lignes sera donc fonction de la taille que fixe le lecteur en dimensionnant la fenêtre de son *browser*. Il appartient à l'auteur du document de provoquer, lorsqu'il le souhaite, un passage à la ligne. La balise
 qui génère ce passage est une balise vide, c'est-à-dire qu'il n'y a pas de balise de fin (</BR> ne s'utilise pas).

```
Quand je vais chez le fleuriste<br>je n'ach&egrave;t' que des
lilas<br>
Si ma chanson
chante triste<br>C'est que l'amour n'est plus l&agrave;.
<h4>Georges Brassens</h4>
```

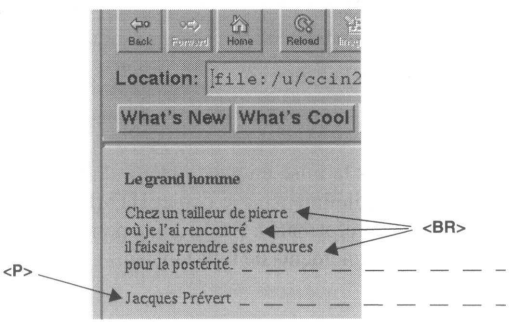
Le retour à la ligne derrière « Si ma chanson » sera ignoré par le browser. En revanche l'absence de balise
 derrière « ...l'amour n'est plus là. » n'est pas gênante car la balise <H4> va provoquer un retour à la ligne et un espacement interparagraphe.



<P>

La balise <P> provoque le passage au paragraphe suivant. Nous verrons par la suite que cette balise peut être comptée par de nombreux attributs. Tout comme
, cette balise est généralement considérée comme une balise vide, bien qu'il soit tout à fait valide de terminer un paragraphe par </P>.
<P> est différent de
 : elle provoque un passage à la ligne et décale la ligne suivante d'un espace d'environ une ligne (espacement interparagraphe).

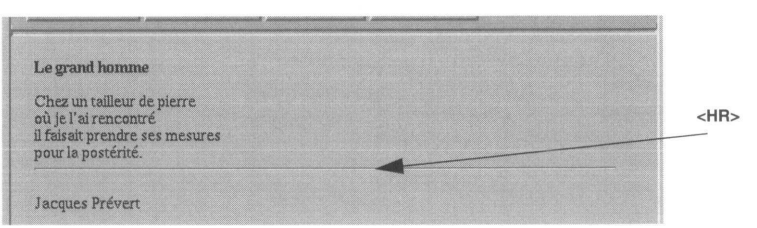
```
<h4>Le grand homme</h4>
Chez un tailleur de pierre<br>o&ugrave; je l'ai rencontr&eacute;<br>
il faisait prendre ses mesures<br>pour la post&eacute;rit&eacute;.
<p>Jacques Prévert
```



<HR>

C'est une balise de division graphique qui provoque un trait sur toute ou partie de la largeur de la fenêtre du browser.

```
<h4>Le grand homme</h4>
Chez un tailleur de pierre<br>où je l'ai rencontré<br>
il faisait prendre ses mesures<br>pour la postérité.<hr>
<p>Jacques Prévert
```



D'autres attributs permettent de régler certains paramètres de cette balise :

L'attribut SIZE

Il définit en pixels l'épaisseur du trait *<HR SIZE=valeur>*.

L'attribut WIDTH

Définit soit en pourcentage de la largeur de la fenêtre, soit en valeur absolue (pixels), la longueur du trait *<HR WIDTH=valeur>*.

Exemple :

```
<HR WIDTH=30%>, <HR WITH=100>
```

L'attribut ALIGN

Dans le cas où l'on spécifie une longueur du trait inférieure à 100%, cela permet de positionner le trait dans la fenêtre *<HR ALIGN=valeur>*. Les valeurs possibles sont LEFT (positionnement à gauche), RIGHT (positionnement à droite), CENTER (positionnement au centre, valeur prise par défaut).

L'attribut NOSHADE

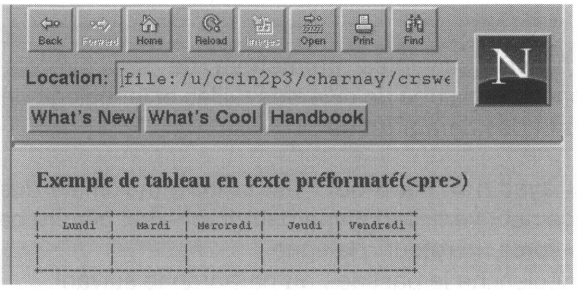
Permet de supprimer l'effet de relief du trait.

<PRE>

Cette balise permet de respecter la mise en page précise d'un texte ou d'une portion de texte, de la façon dont elle a été définie dans le fichier HTML.

Le *browser* interprète ce texte préformaté à l'aide d'une police de caractères non proportionnelle, ce qui autorise des alignements de type tableau.


```
<html>
 <head><title>tableau pr&eacute; formatt&eacute;</title></head>
 <body>
      <h2> Exemple de tableau en texte pr&eacute;for
matt&eacute;(&lt;pre&gt;)</h2>
      <pre>

 I   Lundi    I   Mardi    I Mercredi  I    Jeudi    I  VendrediI

 |           |           |           |            |           |
 |           |           |           |            |           |

      </pre>
 </body>
</html>
```



Les espaces consécutifs sont préservés à l'inverse du texte situé en dehors de ces balises où plusieurs espaces sont compactés en un seul.
On n'utilisera pas de balise de mise en page dans un bloc préformaté (
, <p>, <h>, etc.) et on évitera le caractère de tabulation. En revanche il est tout à fait permis d'inscrire des liens vers d'autres documents, (voir **Créer des liens,** page 59).
La balise <PRE> admet l'attribut WIDTH qui permet de fixer le nombre de colonnes dans lesquelles se situe le texte non formaté (<pre width=40> par exemple). Si cet attribut est absent, la valeur prise par défaut est 80 colonnes. Cet attribut n'est pas interprété par tous les *browsers.*

RESUME

Les balises suivantes permettent de structurer le document par l'adjonction de titres de diverses tailles et le réglage de la longueur des lignes et des paragraphes :

<Hn> avec n = 1 à 6 où 1 est la taille maximale des caractères. Un espacement vertical de type paragraphe est généré par ces balises.
**
** force un retour à la ligne.
<P> déclenche le passage au paragraphe suivant.
<HR> permet de tracer un trait en travers de la page, avec différents réglages rendus possibles par les attributs :

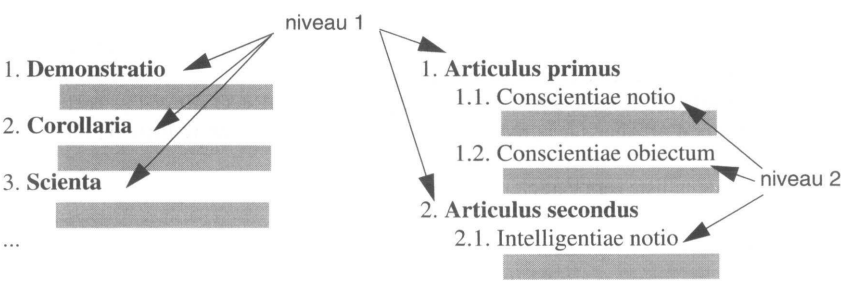
- **SIZE** règle l'épaisseur du trait (en pixels).
- **WIDTH** règle la longueur du trait soit relativement à la largeur de la page (%) soir en valeur absolue (pixels).
- **ALIGN** positionne le trait à droite, à gauche ou au centre (**RIGHT**, **LEFT**, **CENTER**) dans le cas où il est inférieur à la largeur de la page.
- **NOSHADE** supprime l'effet 3D du trait.

<PRE> permet de conserver le formatage du texte délimité par cette balise tel qu'il a été saisi dans le fichier source HTML.

Chapitre 6

Listes

La liste est un élément important dans la structuration d'un document ; elle permet d'organiser tout ou partie d'un document pour le rendre perceptible au lecteur de la façon la plus claire possible. Les listes pourront donc être utilisées pour diviser le document aussi bien que pour effectuer des énumérations d'objets. Mais à l'inverse des traitements de texte comme Word ou FrameMaker, il n'y a pas de numérotation automatique pour des niveaux hiérarchiques différents :



est possible...

n 'est pas possible automatiquement.

Figure 8 - Numérotation des paragraphes dans HTML

HTML définit deux types de listes :

• des listes descriptives, de type glossaire,

• des listes régulières avec ou sans numérotation.

Liste descriptive

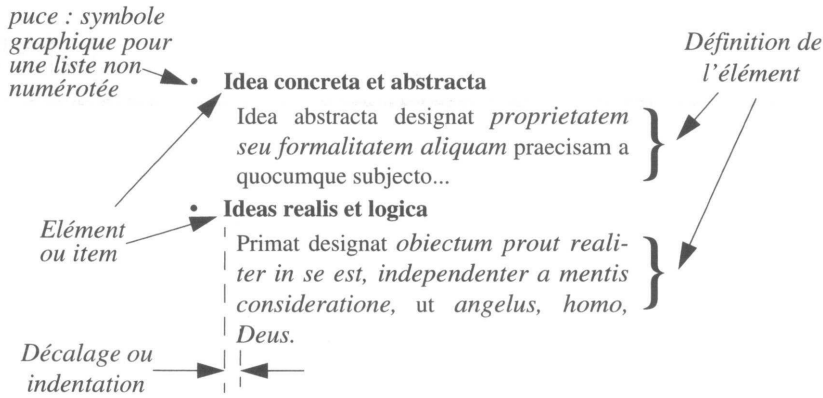


Figure 9 - Exemple de liste descriptive

<DL>

L'élément <DL> ouvre une liste descriptive. Il définit le début de la liste et englobe deux autres balises (DT et DD), dont le rôle est de caractériser, de désigner chacun des éléments, la partie définition et l'élément lui-même.

L'attribut COMPACT associé à la balise <DL> (<DL COMPACT>) permet à certains *browsers* d'afficher sur la même ligne l'élément et la première ligne du bloc de description.

La fin de la liste sera caractérisée par la balise </DL>.

<DT>

L'élément <DT> précède chaque item de la liste, celui-ci ne devant pas excéder une ligne. Il est à noter que le système de liste ne gère pas automatiquement la mise en relief des caractères de l'item. Il appartient à l'auteur du document d'utiliser des balises de styles (voir **Styles,** page 51) ou de titre (<Hn>) pour effectuer cette opération.

Cette balise est une balise vide.

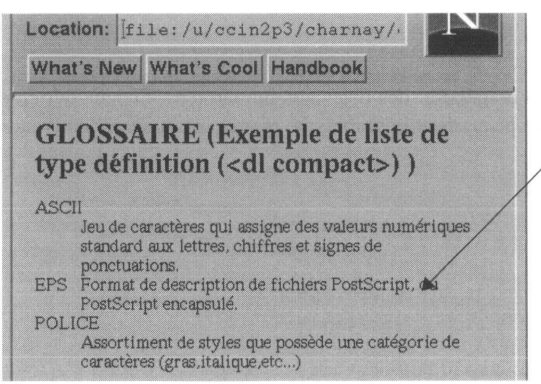
<DD>

La balise <DD> correspond à la zone de définition de l'item. La taille de cette zone n'est pas limitée et chacune des lignes sera décalée vers la droite. Cette balise est vide.
La structure de ce modèle de liste sera donc la suivante :

<DL>
<DT>Identification de l'élément<DD>Description de l'élément
<DT>...<DD>...
</DL>

.html

```
<html>
  <head><title>Listes &lt;dl&gt;</titlex/head>
  <body>
       <h2> GLOSSAIRE (Exemple de liste de type d&eacute;finition
 (&lt;dl compact&gt;) )</h2>
       <dl compact>
          <dt>ASCII<dd>Jeu de caract&egrave;res qui assigne des
valeurs num&eacute;riques standard aux lettres, chiffres et signes de
ponctuations.
         <dt>EPS<dd>Format de description de fichiers PostScript, ou
PostScriptencapsul&eacute;.
          <dt>POLICE<dd>Assortiment de styles que poss&egrave;de une
cat&eacute;gorie de caract&egrave; (gras,italique,etc...)
       </dl>
  </body>
</html>
```



L'attribut **COMPACT** provoque pour l'item EPS un compactage sur la même ligne de l'élément et de son descriptif.
En remplaçant <DL COMPACT> par <DL>, on normalise la présentation de tous les items.

Listes régulières

- cheval
- poney
- mulet
- zèbre

1. Livres
2. Rapports
3. Thèses
4. Périodiques

Figure 10 - Listes non ordonnées et listes ordonnées

<LI>

La balise <LI> est vide et commune à tous les types de listes restants. Elle précédera chaque objet de la liste.

<LH>

Commune aussi à l'ensemble des listes, cette balise vide permet d'affecter un en-tête à l'ensemble de la liste. Le texte associé apparaît de façon identique aux items mais n'est pas précédé de puce ou de numéro. La syntaxe générale des listes sera :

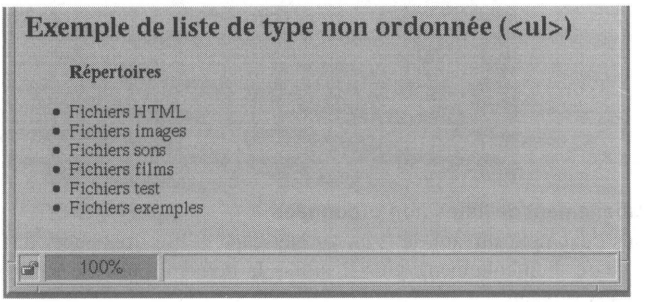
<balise d'ouverture>
<lh>élément d'en-tête
<li>élément de rénumération
<li>..
<balise de fermeture>

<UL>

La balise <UL> permet de générer des listes non numérotées. Chacun des éléments de la liste sera précédé d'une puce, dont le graphisme pourra varier selon le niveau d'imbrication de la liste.


```
<html>
  <head><title>Listes &lt; ol&gt ;</title></head>
  <body>
       <h2> Exemple de liste de type non
              ordonn&eacute;e (&lt;ul&gt;)</h2>
       <ul>
           <lh> <b>R&eacute;pertoires</bxp>
           <li> Fichiers HTML
           <li> Fichiers images
           <li> Fichiers sons
           <li> Fichiers films
           <li> Fichiers test
```

```
            <li> Fichiers exemples
        </ul>
    </body>
</html>
```



L'attribut TYPE dans les listes non ordonnées

Cet attribut permet de contrôler le symbole (la puce) précédant chacun des items de la liste. Il peut être utilisé avec la balise <ul> ; il définit alors le symbole pour toute la liste, sauf s'il apparaît dans une balise <li> où il redéfinit un nouveau symbole pour le reste de la liste.

<OL TYPE=valeur>

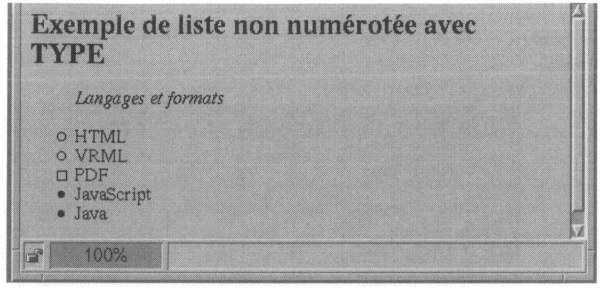
<LI TYPE=valeur>

Les valeurs définissant la puce peuvent être :

- **square,** qui donne à la puce l'aspect d'un rectangle,
- **circle,** qui donne à la puce l'aspect d'un cercle vide,
- **disc,** qui donne à la puce l'aspect d'un cercle plein.

.html

```
<html><head><title>Liste -type</title></head><body>
<h2>Exemple de liste non num&eacute;rot&eacute;e avec TYPE</h2>
     <ul type=circle >
           <lh id=langxi>Langages et formats</i><p>
           <li>HTML
           <li>VRML
           <li type=square>PDF
           <li type=disc>JavaScript
           <li>Java
    </ul>
< / body >< / html >
```



Emboîtement de listes non ordonnées

Dans l'exemple suivant, le *browser* Netscape utilise trois puces différentes en fonction du degré d'emboîtement, puis il utilise la dernière puce (le carré vide) pour tous les niveaux suivants.

.html

```
<html>
  <head><title>Listes &lt;ul&gt;</title></head>
  <body>
       <h2> Exemple de liste de type non ordonn&eacute;e
imbriqu&eacute;e (&lt;ul&gt;)</h2>
       <ul>
           <li> Fichiers HTML
           <ul>
               <li>Exemples
               <ul>
                   <li>listes
                   <li>tableaux
                   <li>Styles
                   <ul>
                       <li>citation
                       <li>texte en emphase
                   </ul>
               </ul>
               <li>En cours
               <li>P&eacute;rim&eacute;s
           </ul>
           <li> Fichiers images
           <li> ...
       </ul>
  </body>
</html>
```

Application (évaluation):

1. Inspectez la page courante dans l'onglet **Élément**.
2. Éditez la page HTML. Modifiez le texte du TD. Rajoutez ou enlevez des balises de la page HTML.
   Note: Pour éditer le HTML, il faut faire clic droit > 'Edit as HTML'.
3. Changez des éléments de style. Par exemple la façon dont les bouts de code en ligne comme `margin`, `padding` sont stylisés. Ou passez à une numérotation binaire des listes d'exercices (`list-style-type: binary` ; Ne marche pas sur Firefox ni IE).
   Notes : Vous pouvez faire ces modifications de style dans la sous-partie **Styles** de l'onglet **Élément** ou directement dans le ficher *CSS* qui se trouve dans l'onglet **Sources**.
4. Rajouter votre premier gestionnaire d'événement (event handler). Pour cela, rajoutez `onclick="alert('À Malibu!')"` comme attribut à l'une des balises HTML. Vous n'avez alors plus qu'à cliquer dessus pour voir le message s'afficher.
   Changez alors l'action JavaScript en `console.log('À Malibu!')` et vérifiez que cela écrit bien dans l'onglet **Console**.

Références :

[1] https://hal.archives-ouvertes.fr/file/index/docid/1356/filename/HTML-JS.pdf

[2] http://romainlebreton.github.io/ProgWeb-ClientRiche/tutorials/tutorial1.html