

Création et conception de site web

Auteur : Alain Seigneur, Seigneur@math-info.univ-paris5.fr

Droits de propriété intellectuelle : Alain Seigneur, UFR Mathématiques et Informatique Paris 5

Module offert au premier semestre en L2 et L3 de la licence MIA

Description du module

- Dernière date de la modification : 30 août 2008 (Version 4.0)
- Volume de la formation : 18 heures de cours et 24 heures de TP
- Contrôle des connaissances : un examen de TP (30%) + 1 **projet** final (70%)
- Scénarisation : Présentiel (cours, TDM), contenu du cours en ligne

- Objectifs : savoir concevoir et publier des pages web informatives et attrayantes.

- Résumé : Ce module offre un apprentissage du langage HTML et initie aux langages CSS et Javascript.
- Mots clés : HTML, XHTML, hyperlien, balise, tableau, formulaire, styles, CSS, Javascript

Feuille de route de la formation :

<http://www.math-info.univ-paris5.fr/~seigneur/csw.htm>

La navigateur de référence pour l'examen de TP et le projet final est Mozilla.

Merci de signaler à l'auteur les coquilles ou autres erreurs par courrier électronique

Pourquoi cet enseignement ?

- Pour favoriser la maîtrise des technologies de l'information et de la communication qui permet à l'étudiant d'acquérir les savoir-faire **indispensables à la poursuite d'études supérieures**.
- Pour amener l'étudiant à afficher sa **personnalité** et son savoir-faire à travers un média, le web, plus particulièrement à lui faire **concevoir et publier un site web personnel**.
- Pour construire des sites web qui servent non seulement les intérêts de l'individu, mais aussi ceux des entreprises. Derrière, ce sont de nombreux métiers qui sont offerts : webmaster, webdesigner, graphiste, hébergeur,
- Pour accompagner, **en citoyen**, l'entrée dans la société de l'information qui voit naître de nombreuses applications internet dans la vie courante comme dans la vie professionnelle (téléprocédures, cours en ligne, recherche d'emploi sur le web, ...).

Plan du cours

L'environnement de travail de l'étudiant

I Introduction

Qu'est-ce qu'un site web ?
 Les étapes de la création d'un site
 Des pages web aux applications web

II Les premiers pas en HTML, plutôt en XHTML

Infrastructure et premiers éléments du langage HTML
 Ecrire et tester du code HTML
 Les images et les liens

III La représentation visuelle d'un document

Définitions : le langage CSS et les règles de style
 L'intégration des styles et les propriétés de style
 Le système de boîtes et le positionnement

IV Plus avant avec HTML

Les tableaux
 Les formulaires

V Premières applications de Javascript

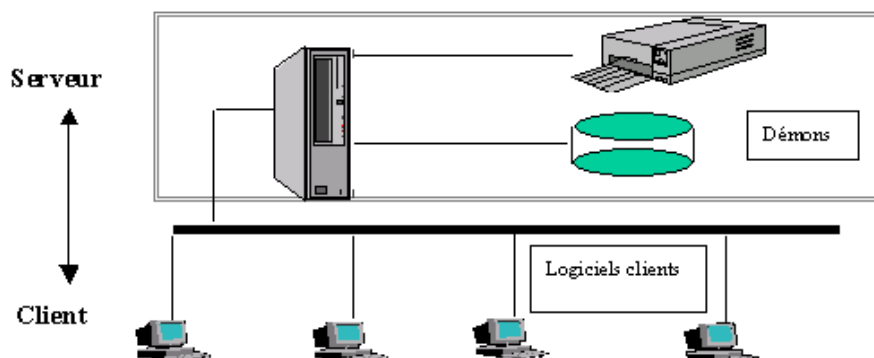
Le langage Javascript
 Objets et événements
 Applications aux formulaires

VI Conception de sites web

Conception adaptée au public et à l'écran
 Structuration et navigation
 Publication d'un site web

L'environnement de travail de l'étudiant

L'étudiant est positionné dans un **environnement client-serveur**. Le modèle client-serveur est la mise en présence d'un ordinateur-serveur (plus puissant et à l'écoute en permanence) et d'un ou plusieurs ordinateurs-clients (ordinateurs des étudiants en réseau).

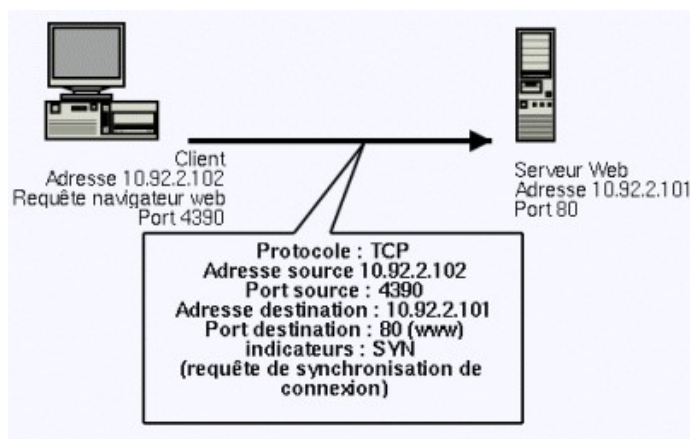


Les processus serveurs sont souvent appelés "*démons*" (programmes `ftpd`, `httpd`) qui réalisent respectivement les **services** FTP, HTTP,

Le poste utilisateur (étudiant) possède deux systèmes d'exploitation (Linux et Windows). Il est en réseau TCP/IP et possède sa propre adresse IP (c'est un ordinateur du réseau Internet) que l'on peut obtenir en mode console sous Windows ou sous Linux (`ipconfig`). La politique informatique de l'UFR consiste à mettre toutes les applications sur le poste étudiant (client FTP, navigateur, suite bureautique, ...) et le "compte" de l'étudiant sur un poste serveur (diamant en l'occurrence).

Chaque étudiant inscrit administrativement à l'université dispose automatiquement d'un **compte informatique** créé après la validation de l'inscription. Chaque compte est unique durant toute la période d'études et accessible par login (fourni par l'UFR de type *h_+5 chiffres*) et mot de passe (secret, choisi par l'étudiant). Le compte informatique permet d'accéder aux ordinateurs de l'UFR et à l'ensemble des services informatiques proposés (services de l'internet, bureautique, espace fichiers, espace web stocké dans le dossier spécifique **public_html**, ...).

Dans le cadre de ce cours, on se restreint principalement à des échanges avec des serveurs web pour nous envoyer des pages web.



Les outils logiciels dont a besoin pour cette formation

- des navigateurs (Internet Explorer et Mozilla) pour lancer des requêtes via la barre d'adressage et pour interpréter le code de la page html renvoyée.
- des éditeurs pour le code html à écrire (`gedit`, bloc-notes, ...) ou d'autres plus spécifiques

- un logiciel de traitement d'images (Gimp)
- un logiciel de compression et décompression (Powerarchiver)
- une suite bureautique (OpenOffice)
- un logiciel ftp (WinScp3, gftp)

Des ressources à disposition

- références pour le langage HTML : <http://www.w3.org/TR/html4/>,
- cours d'html : <http://nephi.unice.fr/CoursHTML/>, et l'excellent <http://www.mcli.dist.maricopa.edu/tut/lessons.html>
- un « dictionnaire » html et javascript : www.allhtml.com, <http://fr.selfhtml.org/>
- un cadre juridique et éthique : <http://www.cnil.fr/>
- un dictionnaire technique : www.commentcamarche.com
- un guide pour les actions de recherche et de référencement : www.abondance.com
- des règles pour la netiquette : <http://www.sri.ucl.ac.be/SRI/rfc1855.fr.html>
- une encyclopédie **wikipedia** : <http://fr.wikipedia.org/wiki/Portail:Informatique>
- un moteur de recherche **Google** ... , sa recherche avancée et ... ses autres services

L'approche projet

On a essayé dans cette formation d'avoir une attitude pré-professionnelle, c'est-à-dire de **sensibiliser** l'étudiant au monde de l'entreprise qui a ses **exigences** de qualité, d'adaptabilité et d'**autonomie**, de **relationnel** aussi.

Pour aider à cette "autonomie", l'apprentissage est organisé dans un environnement spécifique : vous avez des outils et des ressources, vous travaillez à un **projet** final. Un projet se définit à travers un **cahier de charges** à respecter. Ce dernier inscrit le projet dans son contexte (l'existant en particulier), décrit son objectif, présente ses fonctionnalités (ce qu'il apporte), décrit les aspects techniques de sa réalisation dans les **délais** prévus [CF Annexes].

La netiquette

Avant de vous plonger rapidement dans la publication de pages web, l'étudiant doit connaître un minimum de règles quant au contenu, au statut étudiant, au fournisseur d'accès. En voici quelques unes :

- Souvenez-vous que tous les services trouvés sur le web appartiennent à quelqu'un d'autre. Les gens qui paient les factures établissent les règles qui en régissent l'usage. L'information peut être libre ou peut ne pas l'être ! Vérifiez bien.
- Ne supposez qu'**aucune** information que vous trouvez est à jour et/ou exacte. Souvenez-vous que les nouvelles techniques permettent à n'importe qui de devenir un éditeur, mais tout le monde n'a pas découvert les responsabilités liées à la publication.
- Souvenez-vous que, à moins d'être sûr qu'une technique de sécurité et d'authentification est utilisée, toute information que vous soumettez à un système est transmise "en clair" sur l'Internet, sans protection contre les "renifleurs".
- Comme l'Internet embrasse le globe, souvenez-vous que les services d'information peuvent refléter des cultures et styles de vie franchement différents de ceux de votre communauté. Des choses que vous trouvez choquantes peuvent provenir de régions où elles sont acceptables. Restez sans parti pris.
- N'utilisez pas le site FTP de quelqu'un d'autre pour déposer des affaires que vous désirez voir repris par d'autres gens. Cela s'appelle du "dumping" et n'est pas un comportement acceptable.
- Lorsque vous avez des problèmes avec un site et demandez de l'aide, veillez à fournir autant d'information que possible afin d'aider à résoudre le problème.
- Lorsque vous installez votre propre service d'information, tel qu'une page d'accueil, veillez à vérifier avec le gestionnaire de votre système local, en quoi les règles de conduite locales sont concernées.

I Introduction

La création et la conception d'un site web est un processus qui implique plusieurs étapes qui peuvent, suivant l'importance du site, mettre en oeuvre des moyens importants et des compétences diverses.

I.1 Qu'est-ce qu'un site web ?

1 Définition

Un **site Web** est un ensemble de pages web et d'éventuelles autres ressources hyperliées en un ensemble **cohérent**, pouvant être consulté avec un **navigateur** à une adresse web donnée.

Un site est publié (entendons par là qu'il est mis en ligne sur un **serveur web**) par un propriétaire (une entreprise, une administration ou un particulier). Le propriétaire du site choisit l'adresse Web à laquelle le site est accessible. Il peut créer et maintenir le site à jour lui-même ou faire appel à une entreprise spécialisée, une agence web.

2 Les technologies mises en oeuvre pour un site web

Protocole de communication : HTTP, HTTPS

***http** est un protocole qui permet le transfert de fichier (essentiellement au format HTML) localisé grâce à une chaîne de caractères appelée url entre un navigateur (le client) et un serveur.*

Formats de données

C'est la manière utilisée en informatique pour représenter des données sous forme de nombres binaires. C'est une convention utilisée pour représenter des informations représentant un texte (UTF-8, ASCII), une image (GIF, JPEG, et PNG), un son, un fichier exécutable. Lorsque ces données sont stockées dans un fichier (unité informationnelle stockée), on parle de **format de fichier**.

On distingue un format dont la spécification est publiquement accessible, un **format ouvert** (créé dans un but d'interopérabilité) d'un **format propriétaire** que seul un logiciel spécifique est capable d'exploiter (MSWord, PDF)

Formats de documents : HTML, XHTML, CSS, Javascript,

3 Histoire

Tim Berners-Lee, l'inventeur du World Wide Web, a mis en ligne le tout premier site internet en 1991, faisant de lui le tout premier concepteur de site (web designer). Il utilisait le maillage de l'internet, non pas uniquement à travers ses différents chemins (gain en tolérance de panne) **mais pour ses nœuds**. Il proposait une ramification de documents stockés sur des **serveurs** et accessibles par des liens **hypertexte** (dispositif offrant une organisation non linéaire des informations).

Il a créé un système standard basé sur **HTML, http et l'hypertexte**. **HTML** est un dialecte du langage existant mais très complexe SGML (Standard Generalized Markup Language) véhiculant deux idées fondamentales : une approche descriptive, une structuration par classes de documents.

HTML est beaucoup plus simple : il structure grossièrement un document (titres et paragraphes), intègre des hyperliens et en permet une représentation à l'aide d'un navigateur.

Au fur et à mesure que l'Internet et la conception de site progressaient, le langage qui formait les pages, HTML devint plus fourni et flexible. Un nouvel élément, les **tableaux**, dans lesquels on peut afficher des données, furent vite détournés de leur objectif initial et furent utilisés comme moyen de mise en page.

Avec l'apparition du **CSS** (1996) et des feuilles de styles, la mise en page avec les tableaux fut vite considérée comme obsolète. Les technologies modernes qui utilisent des bases de données ont diversifié les outils de conception de site.

Javascript a été créé en 1995. C'est un langage de script côté client dont l'objectif est de manipuler de façon simple des objets fournis par une application hôte (exemple : contrôle des données saisies dans des formulaires).

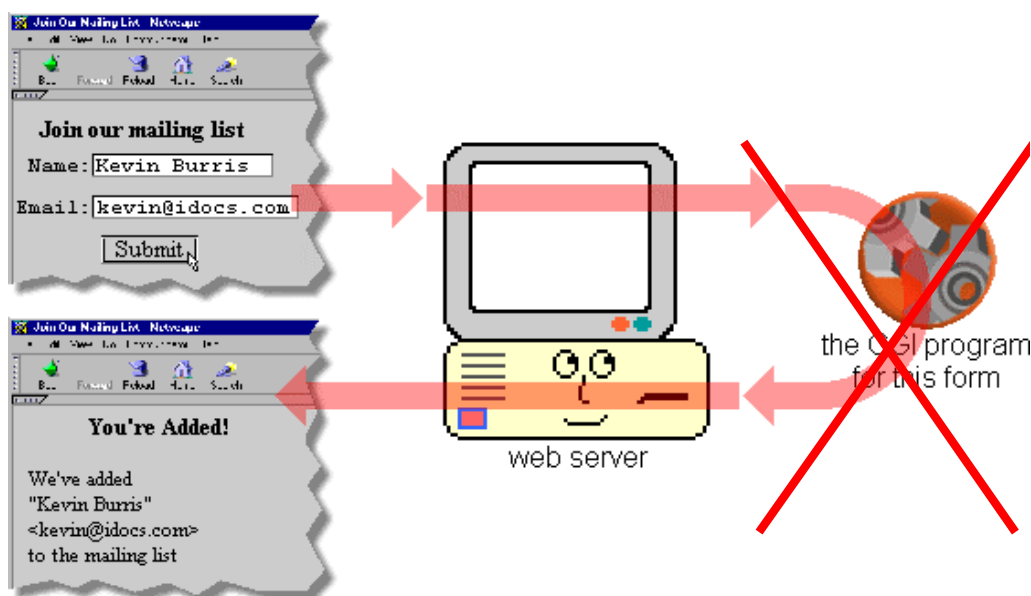
L'apparition de **Flash** (1996 pour la création d'animations vectorielles interactives, Macromédia) a également changé l'apparence d'Internet, en offrant de nouvelles cartes aux concepteurs de sites. Toutefois, Flash est beaucoup plus restrictif que le HTML car c'est un format propriétaire nécessitant un plug-in pour être affiché.

Une technique assez récente appelée le codage à distance a permis une utilisation d'Internet de façon plus dynamique, sans nécessiter de plug-ins ou d'application spécialisées. Le chef de file de ces technologies est AJAX, mais ce n'est pas la seule technologie existante.

4 Les constituants des pages et le périmètre du contenu de ce cours

- La **structure de contenu** de la page va être décrite en XHTML
- La **présentation** (l'affichage) est assurée via des feuilles de style CSS.
- Des objets multimédias peuvent être ajoutés et intégrés (GIF, JPG, PNG, FLASH)
- Une première **manipulation de scripts côté client** à l'aide de Javascript

Ce cours se limite à une formation au langage HTML, englobant un langage de script (Javascript) et l'utilisation de feuilles de styles CSS. La programmation côté serveur (Java, PHP, ...) ne relève pas de cet enseignement.



I.2 Les étapes de la création

En fonction du type de site, du contexte et des moyens disponibles pour le mettre en oeuvre certaines de ces étapes peuvent être optionnelles voire inutiles.

- Réflexion sur l'**objectif** du site, sa rentabilité, les moyens financiers à engager ...
- Réflexion sur l'**autonomie** souhaitée et le type de moyens humains pour faire la mise à jour. Choix éventuel d'une agence web (*qui peut s'occuper de tout ce qui suit*).

- Dépôt d'un nom de domaine, choix éventuel d'un hébergeur.
- Etablissement d'une structure de pages (**contenu éditorial**).
Choix et installation d'un système de gestion de contenu (comme SPIP).
- Mise au point d'une **charte graphique**.
- Optimisation : préparation du site pour les moteurs de recherche.
Référencement : on doit le préparer dès le début et l'affiner à la dernière étape..

L'accessibilité comme le référencement du site sont à prendre en compte à chaque étape à partir de la mise au point de la structure des pages.

1.3 Des pages web aux dernières applications web

La métaphore de la page

Il y a plusieurs types de **pages web** : des **sites institutionnels**, des **sites d'e-commerce électronique**, des annuaires et moteurs de recherche, des messageries web, des forums internet, des wikis, des blogs,

*Un **wiki** est un **système de gestion de contenu** de site web qui rend les pages web librement et également modifiables par tous les visiteurs autorisés. Les wikis sont utilisés pour faciliter l'**écriture collaborative** de documents avec un minimum de contrainte*

*Un **blog** est un site web sur lequel une ou plusieurs personnes s'expriment de façon libre, sur la base d'une certaine périodicité. Son expression est décomposée en **unités chronologiques** ; chaque unité est susceptible d'être commentée par les lecteurs. "Blog" est un mot-valise (néologisme), né de la contraction de « [web log](#) » (c'est-à-dire carnet de bord Web.) Contrairement au site personnel, le blog bénéficie d'une structure éditoriale pré-existante, sous la forme d'outils de publication plus ou moins formatés.*

Pour les sites informatifs, les pages web se succèdent les unes aux autres dans la logique de l'hypertexte : un navigateur web contacte un serveur web, lui demande une page et ce dernier lui renvoie.

Une autre approche est celle d'envoyer des informations, via un formulaire par exemple, au serveur qui les traite à l'aide d'une application PHP (remote scripting). C'est le cas des sites de e-commerce. L'information est personnalisée, mais toujours reçue à travers des pages. Ce type d'applications, certes fort répandues, est limité par la latence et la fiabilité du réseau, par le fait que la requête est toujours à l'initiative de l'utilisateur (le protocole HTTP de requête / réponse est à sens unique). Mais on peut aller plus loin en ouvrant une session (le serveur mémorise des informations),

On vous propose ici comme ouverture technologique à ce cours de repenser une application web, de décrire très sommairement un nouveau mode de conception. Aussi les quelques paragraphes en italiques qui suivent « Ajax et le client riche » ne sont pas au programme.

Ajax et le client riche

Un client riche est un client riche en ce sens qu'il offre une variété d'interactions (méthodes de saisie en particulier) qu'il est intuitif et réactif. Un exemple de bonne qualité d'interaction est le tableur (ou encore le traitement de texte). Mais ce n'est pas un client riche : la logique et le modèle des données sont enfermés dans un environnement (du moins jusqu'à l'arrivée d'XML).

*Une réponse aux limites des applications web dont nous avons parlé serait de rendre les réponses aux requêtes indépendantes de l'activité réseau par une **interaction asynchrone**.*

Prenons un exemple de la vie quotidienne, le début de la journée : réveiller les enfants, leur préparer le petit déjeuner, s'habiller soit-même et partir ... Vaste programme !

Première option

La mère reste à côté des enfants pour surveiller le « décollage ». Une fois la toilette et l'habillement fait, elle prépare le déjeuner et les fait manger, s'habille pendant qu'ils mangent et avale son café. Départ ... tendu ! Les enfants font de la résistance passive, la mère perd son temps, s'habille mal et se brûle en prenant le café !!

Deuxième option

Les enfants s'habillent seuls, la mère prend son temps pour s'habiller et préparer le petit déjeuner. Ensuite tous déjeunent ensemble. Départ organisé.

Où est la différence ? En termes informatiques, on désynchronise les activités : les enfants se préparent seuls et la mère accomplit ses tâches (les enfants se débrouillent en arrière plan). Les enfants seront prêts quand ils demanderont leur petit déjeuner

Une solution offrant un client riche se développe vite : c'est **AJAX**. **Asynchronous Javascript And XML**, est un acronyme désignant une méthode informatique de développement d'application web (la mise à disposition de services affichés dans des fenêtres).

Ajax n'est pas une technologie monolithique mais un terme qui évoque l'utilisation **conjointe** d'un ensemble de technologies déjà couramment utilisées sur le Web, incluant plus particulièrement celles qui nous intéressent pour cette formation :

- **XHTML** pour la structure sémantique des informations ;
- **CSS** pour la présentation des informations ;
- **DOM** et **Javascript** pour afficher et interagir dynamiquement avec l'information présentée ;
- XML et XSLT qui seront vu en master 1^{ère} année
- ... et le point fort **XMLHttpRequest** pour échanger et manipuler les données de **manière asynchrone** avec le serveur web donc traité ici car nous restons côté client.

Pour information,

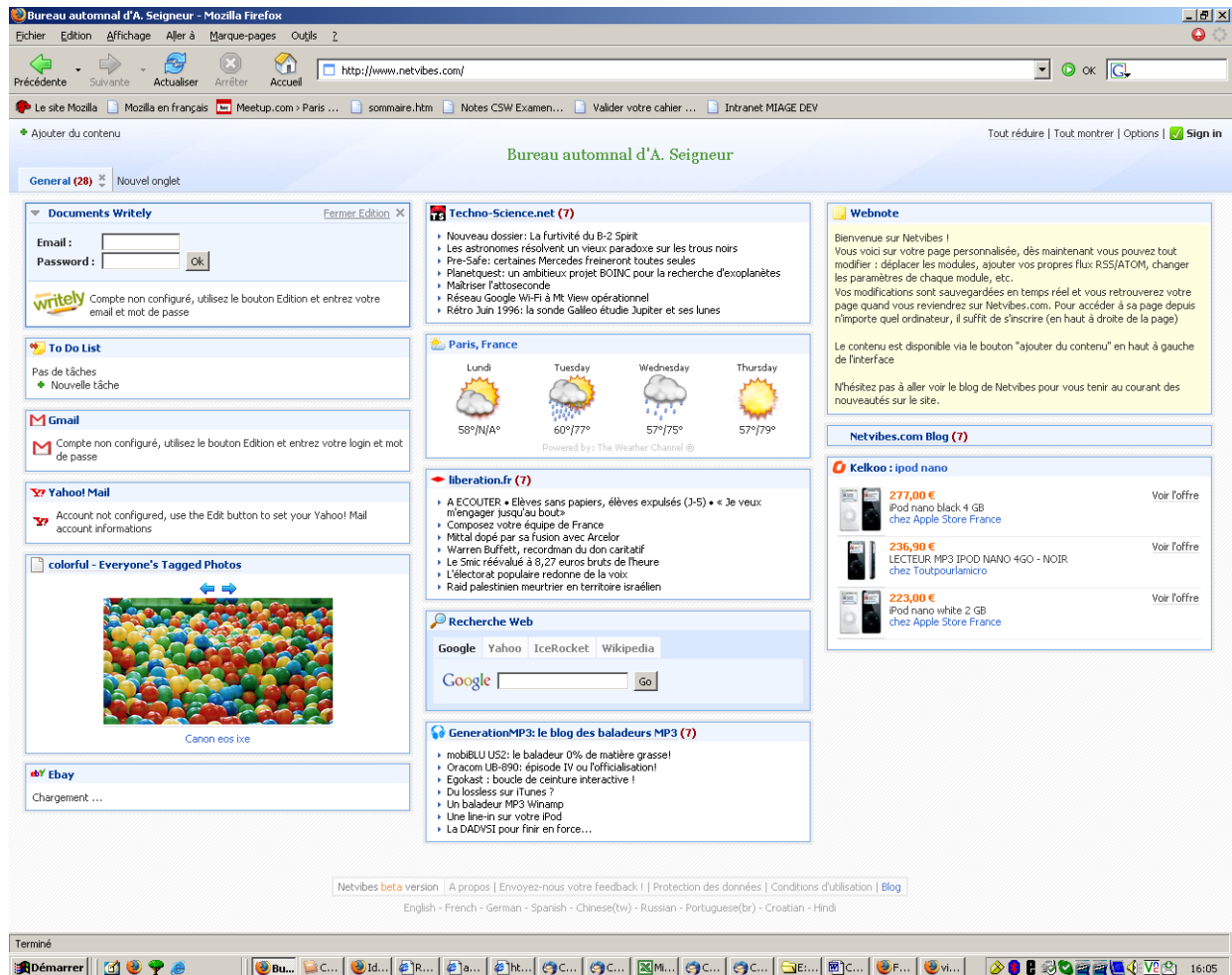
DOM présente la structure des pages web sous forme d'arbre dont les noeuds peuvent être manipulés par Javascript. Ainsi une application Ajax peut modifier l'interface utilisateur « à la volée » en redessinant les éléments visuels de la page affichée.

XMLHttpRequest permet d'extraire des objets de données sur le serveur web dans le cadre d'une activité en arrière plan. Le format de données est XML (ou bien HTML « xmlisé ») ou encore du texte simple.

Les applications AJAX sont alors plus réactives, la quantité de données échangées entre le navigateur et le serveur HTTP étant fortement réduite. Le temps de traitement de la requête côté serveur est également légèrement réduit, une partie du traitement étant réalisé sur l'ordinateur d'où provient la requête. En contrepartie, le chargement de la première page peut être pénalisé si l'application utilise une bibliothèque AJAX volumineuse. De plus le référencement n'est plus assuré compte tenu que la page est générée en ligne. L'accessibilité pose aussi problème.

Pour illustrer ce cours, nous avons choisi une page web interactive personnelle, - un bureau qui peut être ouvert en n'importe quel point de la planète et à partir de n'importe quel navigateur -, conçu avec Ajax. Vous pouvez vous même le tester et l'utiliser en vous connectant sur <http://www.netvibes.com/>

Ce qu'on veut montrer ici, c'est qu'avec des technologies simples (XHTML, CSS, Javascript) que nous allons découvrir, on peut construire des applications puissantes.



Regardons le code source de cette page à deux endroits : tout ce dont nous avons parlé y est.

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

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

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```
<title>Netvibes</title>
```

```
<link rel="shortcut icon" href="/favicon.ico" >
```

```
<link rel="stylesheet" type="text/css" href="/style/styles.css?v=32" media="screen" />
```

```
<script>
```

```
function netvibesError (errmsg, url, lineNumber) {
    var prepend = "", append = "", b = "", b_e = "", br = "\n";
    if (document.getElementById("startMsg") &&
        document.getElementById("startMsg").style.display != "none") {
        b = "<" + "b>"; b_e = "</" + "b>";
        br = "<" + "br />";
        prepend = "<" + "div style='padding-left: 5px'>";
        append = "</" + "div>";
    }
}
```

```
var msg1 = _("Oh no! A JavaScript error occurred!");
var msg2 = _("Please include this information in your error report (feedback@netvibes.com), e.g. via a screen shot.");
var msg = msg1 + br
+ b + msg2 + b_e + br + br
+ 'Error: ' + errmsg + br
+ 'URL: ' + url + br
+ 'Line: ' + lineNumber + br
```

```

+ 'Client: ' + navigator.userAgent + br
+ 'Cookie: ' + getCookie("activeUserID") + br
+ 'Language: ' + (App.lang||navigator.language) + br
+ 'Date: ' + (new Date()).toLocaleString() + br
;
return true; // enough error messages for now
        if (b == "") {
            alert(msg);
        } else {
            document.write(msg);
        }
    }
    return true;
};

```

.....

</head>

<body>

<div id="userPageFrame" style="text-align:center;padding:2px;background:#EFF5FF;border-bottom:1px solid #36C;display:none"></div>

<div id="startMsg">

<p>Loading...</p>

<p>

We currently support [Mozilla Firefox 1](http://www.getfirefox.com/), [Microsoft Internet Explorer 6](http://www.microsoft.com/ie), [Opera 8](http://www.opera.com/), and [Safari 2](http://www.apple.com/safari).

</p>

<p>Stuck at loading? Please try again using the [error-reporting](http://www.netvibes.com/?err=1) enabled version of

Netvibes and report the message as described.**</p>**

<noscript>

<p>Welcome to Netvibes.com</p>

<p>This page is designed for JavaScript-only browsers. In case you have disabled JavaScript, please enable it and reload this page. Otherwise consider installing a JavaScript capable browser such as

[Mozilla Firefox](http://www.getfirefox.com/) or

[Opera](http://www.opera.com/).**</p>**

</noscript>

</div>

<div id="application">

<div id="topPart">

<div id="topLinks">

<table cellpadding="0" cellspacing="0" width="100%">

<tr>

<td style="text-align:left;width:30%"><div id="contentLink">

<div class="addLink"><a id="addContentLink"

href="javascript:App.Nav.openCloseSelection()"></div>

<!--<div class="optionsLink"><a id="appOptionsLink"

href="javascript:App.Nav.openOptions()"></div>-->

</div></td>

<td style="text-align:center;width:25%;">

<div id="player">

<object type="application/x-shockwave-flash"

data="player/player.swf" width="2" height="2">

<param name="movie" value="player/player.swf" />

</object>

</div>

</td>

.....

.....

Voici du bon grain HTML à moudre

II Les premiers pas en HTML, plutôt en XHTML

II.1 Infrastructure et premiers éléments du langage HTML

HTML est un langage de balisage universellement reconnu et diffusé. Il se distribue en flux « **texte** » écrits à partir d'un jeu de caractères particuliers.

1 A quoi sert HTML

Pour publier de l'information avec la distribution la plus large possible, on a besoin d'un langage universel que tous les ordinateurs puissent comprendre via une application : le navigateur.

HTML est ce langage hôte. Il donne les moyens de :

- publier des documents en ligne
- retrouver de l'information en ligne via un dispositif de cliquage et de liens
- concevoir des formulaires pour conduire des transactions avec des services éloignés (comme une opération de réservation)
- d'inclure des scripts
- d'inclure des applications directement dans un document (téléchargement par exemple)

Un document HTML est un **flux de données textuelles**, éventuellement **combiné** avec d'autres flux de données qu'il **réfère** alors (le plus souvent, il est multimédia). Quand on veut insérer une image dans le document, on utilise un "pointeur" vers cette image.

Un **document**, étymologiquement, est un **support** pour l'instruction, plus généralement, pour l'information (pour se documenter). Il a une **identité** (pour la traçabilité), un **fond** (de l'information "pure"), une **forme** (papier par exemple) et un **format** (pour nous ici HTML).

2 HTML, langage de balisage et son dernier avatar XHTML

HTML est un langage de **balisage** (de marquage, markup language) **recommandé** par le **World Wide web Consortium (W3C)**.

Le contenu textuel d'un document HTML est enrichi par des **balises** de la forme **<xxx>**. Ce marquage avec des chevrons est une directive que **le navigateur doit interpréter de manière appropriée**.

Le contenu textuel renferme les **données**, le balisage apporte les **métadonnées**. Ces dernières sont des informations qui permettent la description de tous types de données : la nature, la définition, l'organisation, la présentation,

Syntaxiquement, une balise se présente dans le document entre les chevrons **<** et **>**.

Certaines balises délimitent une portion de texte ; dans ce cas elles sont formées d'une **balise ouvrante** **<portion>** et d'une **balise fermante** **</portion>**. La portion délimitée (balises comprises) se nomme un **élément**. Exemple **<h1>** GROS TITRE **</h1>**

D'autres balises sont "**vides**", en ce sens qu'elles n'encadrent pas un contenu textuel mais pointent vers un contenu autre, par exemple une image **** ou un passage à la ligne **
**. Dans ce cas, le standard HTML préconise l'emploi d'un marquage à la fois ouvrant et fermant, avec un espace avant le caractère barre oblique (comme indiqué ci-dessus).

Chaque balise a un **nom**, éventuellement ou obligatoirement précisé par des **attributs**. Dans l'exemple **** : **img** est le nom de la balise, **src** est le nom de l'attribut obligatoire de la balise img, **fleur.gif** est la **valeur** de l'attribut src qui est en même temps le nom du fichier contenant l'image.

Le langage HTML a considérablement évolué depuis sa première formalisation (1992) et a montré ses limites : il a méprisé la notion de classes de documents, il a sous-estimé la complexité des informations à représenter dans un document, il est tombé dans le piège de la compatibilité pour devenir ... très flexible sinon très tolérant.

Concluant que la meilleure façon d'avancer avec HTML (qui restera encore longtemps présent sur le web) consiste à le reconstruire en tant qu'application du langage XML, le W3C a créé la norme **XHTML** en 2000. Sa version 1.0 étend HTML 4.0 et c'est sa syntaxe que nous retiendrons pour l'écriture d'un code correct :

- le document commence par une DTD « déclaration de type de document » : DOCTYPE
- les attributs ont toujours une **valeur** suivant un signe = et placée entre **guillemets**
- les attributs sont séparés du nom de la balise, et séparés entre eux, par un espace
- les balises sont **bien imbriquées** (derrière, il ya la construction d'un arbre)
- les balises et les attributs sont écrits en minuscules
- toute balise ouverte doit être fermée même celle qui n'a pas de fermeture ()
- l'attribut **id** prend de l'importance comme identifiant et tend à remplacer l'attribut name.
- Pour tous les caractères spéciaux (&, < ...) destinés à être interprétés (comme le symbole inférieur par exemple), on utilise les "entités caractères" correspondantes(& , < ...).

On peut se trouver dans des situations très particulières, par exemple

- *pour insérer un caractère particulier le signe inférieur < dans un cours de maths. Il va falloir le déguiser pour qu'il ne soit pas pris comme le chevron ouvrant une balise. On l'insère comme **référence d'entité caractère** en écrivant **<**;*
- *pour afficher une lettre particulière. On va l'insérer dans une **entité numérique** marquant sa numérotation (décimale ou hexadécimale) dans un jeu de caractères unicode. Le caractère cyrillique pour le i majuscule est **И** И*

Conversion de HTML en XHTML

Les syntaxes de HTML et XHTML sont très proches, celle de XHTML héritant de XML l'obligation de réaliser des documents bien formés, vérifiant en particulier les règles énoncées ci-dessus et la conformité à une DTD (définition de Type de Document). Ce qui implique la déclaration suivante (toujours la même) en haut du document

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

suivi d'un élément racine précisant un espace de nom et une langue.

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

3 Structure d'un fichier HTML

Un fichier HTML commence par la balise particulière **<html>** et se termine par la balise **</html>**. L'intérieur de ces balises englobantes est structuré sur deux niveaux :

- d'abord l'**en-tête** compris entre les balises **<head>** et **</head>** qui contient
 - le titre (title) du document,
 - des informations méta-textuelles (des informations **sur** le document) introduites par une balise **<meta>**,
 - des scripts javascript, des styles, des liens vers d'autres fichiers de styles ou de scripts
- le **corps** du document compris entre les balises **<body>** et **</body>** où réside (en principe tout) le contenu du document.

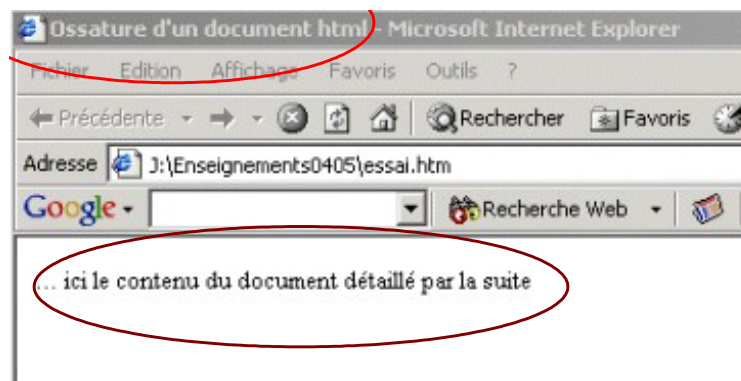
Remarques :

- *Ce qui figure dans le corps du document est visible dans la fenêtre d'affichage.*
- *Le titre, lui, vient s'inscrire dans la barre de titre de la fenêtre du navigateur*
- *HTML est un conteneur pour d'autres langages (Javascript, CSS, PHP ...)*

On arrive ainsi à l'ossature classique d'un document HTML :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
  <head>
    <title>Ossature d'un document html</title>
    <meta name="keywords" content="HTML, balise, tableau, cadre" />
    <meta http-equiv="content-type" content="text/html" ; charset="ISO-8859-1" />
    <meta name="keywords" content="html, head, body" />
    <script> function ..... </script>
    <link rel="stylesheet" type="text/css" href="/style/styles.css?v=32" media="screen" />
  </head>

  <body>
    ... ici le contenu du document détaillé par la suite
  </body>
</html>
```



Remarques

- Les deux premières lignes ne sont pas à apprendre par coeur : elles sont à recopier.
- Le contenu de la balise <title> n'est pas considéré comme du contenu textuel : il est affiché comme titre de la fenêtre du navigateur ou encore comme en-tête de feuille papier.

4 Caractères et encodages

Un document HTML contient du texte constitué à partir d'un **jeu de caractères** bien défini (si ce n'est que par défaut !). Les jeux utilisés sont le plus souvent une extension du code ASCII sur 7 bits qui recouvre 128 caractères. Le plus connu est **Unicode** qui représente tous les caractères écrits du monde et qui a deux encodages principaux : UTF-8 (choisi par défaut) sur un à quatre octets ; UTF-16 sur deux octets. CF. <http://www.unicode.org/> ou encore une section concerne plus spécifiquement l'euro : http://www.unicode.org/unicode/reports/tr8/#Euro_Sign

Unicode

Some (or all) of text styles can represent Unicode now!
 In this demo such style has name "Unicode".
 Below are some phrases from several national search engines:

ノート デスクトップ MS (Japan,
<http://www.e-sekai.com>)
 현대중-전자 분쟁 극적 타결 전망 (Korean,
<http://altavista.co.kr>)
 京报披露：湖南有线网络大战伤亡百人 (simplified Chinese, <http://www.sina.com.cn>)
 千禧七夕情聚兩地 (traditional Chinese, <http://www.sina.com.tw>)

Un texte sur le web doit être transporté et reconnu sous sa forme initiale. Ce qui nécessite de donner à chaque caractère, à travers des tables de correspondances (encodages), un numéro "universel".

Par exemple, nos navigateurs sont pré-adaptés à la langue française et, quand on entre du code HTML, on ne ressent pas le besoin d'encoder les accents é, è, à. Si on le fait (par exemple avec l'encodage ISO - 8859 -1), le navigateur étranger fera la conversion. Si on ne le fait pas, sur un navigateur pré-adapté en anglais par exemple, le texte accentué ne s'affichera pas comme souhaité.



Pour préciser l'encodage en HTML, on utilise une balise meta :

```
<meta http-equiv="content-type" content="text/html ; charset="iso-8859-1" />
```

5 Codage des couleurs

Une couleur qui s'affiche sur l'écran est le résultat d'un mélange de trois couleurs (canaux chromatiques) : rouge, vert et bleu (RVB, RGB en anglais). Chacune d'elles a une plage d'intensité codée sur un octet, c'est-à-dire allant de 0 (absence de couleur) à 255 ("plein phares").

Remarque : si les trois couleurs sont à zéro, on obtient du noir. Si tout est illuminé (les 3 couleurs à 255), on obtient un blanc ... éclatant.

Le langage HTML utilise des valeurs hexadécimales pour exprimer les trois couleurs fondamentales (RVB). Ainsi, les valeurs chromatiques sont représentées par 6 chiffres hexadécimaux (les deux premiers pour le rouge, ...) précédés du symbole # (dièse).

On peut nommer les couleurs et les navigateurs reconnaissent les 16 couleurs ci-dessous.

	white	FFFFFF
	silver	C0C0C0
	gray	808080
	black	000000
	red	FF0000
	maroon	800000
	yellow	FFFF00
	lime	00FF00
	green	008000
	olive	808000
	aqua	00FFFF
	teal	008080
	blue	0000FF
	navy	000080
	fuchsia	FF00FF
	purple	800080

Exemple la codage **#fc0277** mélange les trois couleurs de base et donne comme résultat

composantes :	Rouge	Vert	Bleu
#	FC	02	77
couleur :			
résultat :			

6 Exercices immédiats

De quel langage HTML est-il le sous-ensemble ?

Qu'est-ce qu'une balise obsolète ?

Que distingue principalement HTML d'XML ?

Donner des noms de couleurs

Si vous codez les couleurs sur 16 bits, de combien de couleurs disposerez-vous ?

Ecrire le code pour que l'en-tête du fichier comporte les informations suivantes :

le nom de l'auteur du document et les mots clés de la page

II.2 Ecrire et tester du code HTML

Editeur et navigateur sont deux outils liés : l'éditeur permet la création de code HTML, mais ne le teste pas. C'est le travail du navigateur qui est encore malheureusement très changeant d'un poste à l'autre. Aussi une approche correcte de l'écriture web consiste à visualiser le résultat du code sur les principaux navigateurs.

1 Les éditeurs HTML et les navigateurs

Un fichier HTML est un fichier texte, éditer du code HTML se fait donc en utilisant un éditeur de texte. Un simple éditeur (bloc-notes sous Windows ou encore gEdit sous Linux) ne dispose pas de certaines facilités (recherche/remplacement ou dictionnaire), mais il permet d'éditer un texte propre, "manuel" (se souvenir que l'on doit savoir intervenir manuellement quand on retouche du code!).

Nous recommandons dans un premier temps l'utilisation de ces éditeurs de texte avant de passer à des éditeurs spécialisés (comme Dreamweaver, par exemple, un des plus connus). Mais vous pouvez même utiliser tout traitement de texte qui vous convertit son format en HTML. Pour votre information, un simple affichage de salutation (Bonjour) "pèse" 19456 octets avec Word et 44 octets en Ascii !. Se souvenir alors que du code HTML est fait pour transiter sur les réseaux !!

Comment développe-t-on du code HTML ?

D'abord ouvrir un simple éditeur de texte. Une fois la première mouture de code rédigée, enregistrer, dans un dossier choisi, le fichier **avec** l'extension .htm ou .html. **Attention aux sur-extensions .txt, à la casse et au nom des fichiers mêmes.** Ces derniers, pour être valables sur tous systèmes, doivent avoir moins de 255 caractères et éviter certains caractères : / \ < > : * .

Puis lancer un navigateur (Mozilla, Internet Explorer ou encore un autre) et sélectionner votre fichier avec la commande Fichier Ouvrir : le contenu **interprété** de votre fichier va s'afficher dans la fenêtre du navigateur.

Si vous n'êtes pas satisfait de l'affichage, retournez dans l'éditeur sans fermer le navigateur et corrigez le code HTML (Internet Explorer permet la correction directe avec affichage de la source). Reprendre la fenêtre du navigateur et utiliser la commande (recharger ou actualiser) pour apprécier l'effet de vos modifications.

Remarques :

- ***l'indentation***, dans l'écriture du code HTML, si elle n'est pas obligatoire, est fortement recommandée pour une meilleure lisibilité.

- même si votre code est incorrect, il a toutes les chances d'être réparé par votre navigateur, mais passera-t-il aussi bien sur les autres agents utilisateurs (PDA, téléphone, ..) ?

Une première bonne pratique de développement de code HTML consiste à ouvrir une page web et à regarder son code source (clic droit sur le contenu et valider Afficher la source).

Une autre, pour rendre le code produit lisible, consiste à insérer des commentaires. Ces derniers ne sont pas interprétés par le navigateur (donc non affichés) et sont placés entre les séquences de

caractères `<!--` et `-->` (le texte du commentaire ne doit pas comporter en son milieu une de ces deux dernières marques).

2 Les premières balises de structuration du texte

Une page web est avant tout un contenu textuel à diffuser sur un écran (ce qui en rend plus fastidieux la lecture). Mais remplir la page web ne se fait pas comme pour une page imprimée. Originellement, le flux textuel est sur une seule ligne ! Aussi va-t-il falloir, pour une meilleure lisibilité, penser à une "aération" du texte, c'est-à-dire à laisser des espaces libres.

Organiser un texte consiste d'abord à le structurer hiérarchiquement, à l'aide de titres, en parties, chapitres, sections, paragraphes. HTML dispose de six formats de titres (headings) : h1 à h6, du plus grand au plus petit.

Exemple : `<h1>` Voici un gros titre! `</h1>`. `<h3>` Voici un titre moyen `</h3>`

Remarque : il n'y a pas automatiquement de numérotation hiérarchique du genre I.1, I.2, I.3. Si vous voulez la créer, il faut la coder ... en dur.

Passer à la ligne nécessite la balise vide `
` (break).

La balise `<p />` est différente de `
` : elle provoque un passage à la ligne et, en plus, décale la ligne suivante (espacement inter-paragraphes).

Gestion des espaces (blancs) ou "aération"

On appelle espace ou blanc les caractères "espace" (Code Unicode décimal 32), "tabulation" (9), "nouvelle ligne" (10), "retour chariot" (13).

Par défaut, les navigateurs fusionnent les séquences de blancs (3 espaces + une tabulation = 1 espace). Si l'on veut interdire la fusion, c'est-à-dire écrire directement le texte pré-formaté, on utilise les balises `<pre>`.

Si l'on veut "aérer" le texte, on utilise l'entité ` ` pour insérer volontairement un espace supplémentaire (pour un alignement par exemple).

Exemple :

```
<html> <body>
  <h1>Voici un titre imposant</h1> suivi de sa "traîne"
  <hr size="5" width="70%" align="center" />
  <h2>et un autre plus modeste </h2> <hr size="2" width="30%" /> <p />
  Voici le chiffre 1 suivi de 6 espaces et du chiffre 2 : <br />
  1&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;2 <p />
  <pre>
Voici un texte indenté
  &lt;html&gt;
  &lt;head&gt;
  &lt;/head&gt;
  &lt;body&gt;
    bonjour le contenu de cet exemple de code
  &lt;/body&gt;
  &lt;/html&gt;
  </pre> </body> </html>
```

ce qui donne l'affichage suivant

Voici un titre imposant

suivi de sa "traîne"

et un autre plus modeste

Voici le chiffre 1 suivi de 6 espaces et du chiffre 2 :

1 2

Voici un texte indenté

```
<html>
<head>
</head>
<body>
    bonjour le contenu de cet exemple de code
</body>
</html>
```

Remarques :

- *center* est l'alignement par défaut d'un trait de division
- une balise `<hr>` provoque un passage à la ligne **et** un espacement d'interligne
- les entités sont interprétées à l'intérieur de la balise `pre`
-

3 Les listes

Les traitements de texte actuels manipulent les énumérations facilement, avec puces ou numéros :

- Toyota
- Nissan
- Renault

- 1 Toyota
- 2 Nissan
- 3 Renault

La balise `` (unordered list définit une liste non numérotée. Les listes numérotées, elles, sont encadrées par `` (ordered list). Ces deux balises prennent pour attribut **type** qui indique soit le type de la puce, soit le type du numéro (chiffres arabes, romains, ..). Pour chaque type d'énumération, s'il y a un **en-tête**, il est inséré entre les balises `<lh>` et `</lh>`. Chaque **article** (item) qui suit est inséré entre les balises `` et ``.

Schéma syntaxique d'une liste

`<balise d'ouverture de la liste ul ou ol avec l'attribut type="valeur">`

`<lh> élément d'en-tête </lh>`

` élément de l'énumération `

``

``

`<balise de fermeture ul ou ol>`

Dans les énumérations ordonnées, on peut ajouter des attributs

- **start** à la balise `` pour définir la valeur de départ

- **value** à la balise `` pour interrompre la séquence et l'initialiser à une nouvelle valeur

voici un exemple de liste non numérotée :

```
<p/>
<ul>
  <lh>Marques d'autos</lh>
  <li>Renault</li>
  <li>Peugeot</li>
  <li>Toyota</li>
  <li>Nissan</li>
</ul><!-- fin de liste, fin de fichier --> <br />
</body> </html>
```

Il existe un autre type de liste, la liste de glossaire annoncée par `<dl>` : un mot (introduit par la balise `<dt>`) suivi de sa définition (introduite par la balise `<dd>`) comme dans un dictionnaire

Exemple

Deux types de listes

Les listes descriptives

Ce sont des listes de type glossaire avec plusieurs articles (item) et leur définition. En voici un exemple :

```
dl
  dt
    Ouvre une liste descriptive
  dt
    Inscrit un article dans la liste
  dd
    Donne la définition d'un article
```

Les listes d'énumération numérotées ou non

Voici un exemple de liste non numérotée :

```
Marques d'autos
• Renault
• Peugeot
• Toyota
• Nissan
```

4 Les balises `<div>` et ``

Pour structurer un contenu, sans passer par les balises classiques du HTML, il existe des balises génériques créées spécialement pour être utilisées avec des feuilles de style : elles vont tout simplement "déconnecter" certains morceaux de paragraphe ou encore de mots sur une ligne.

Ce sont les balises : `<div>` et ``.

- `<div>` peut contenir toutes les autres balises HTML et donc servir à mettre en forme toute une section d'un texte (un paragraphe par exemple)
- `<div>` avec l'attribut `class` peut servir à faire des "custom tags" (voir plus loin)
- `` sert à changer la présentation d'une séquence de caractères ou de mots à l'intérieur d'une balise

La balise ` ... ` permet d'appliquer des styles à des éléments de texte d'un paragraphe

ou si vous préférez à un morceau de paragraphe. Ainsi pour afficher « Un monde de **géants** » il faut écrire

```
<html>
<head>
  <STYLE type="text/css"> .element {font-size: x-large; color: navy} </STYLE>
</head>
<body>
```

```
<p>Un monde de <span class= « element »>géants</span>.</p>
</body> </html>
```

La balise <div> ... </div> permet de regrouper plusieurs paragraphes ou si on préfère, de délimiter une zone comportant plusieurs paragraphes.

```
<htm> <head>
<STYLE type="text/css"> .zone {font-size: x-small} </STYLE>
</head>
<body>
La balise &lt;DIV&gt;
<div class= « zone »>
<p>Commentaire :</p>
<p>N'oubliez pas l'attribut class!</p>
</div>
</body> </html>
```

Donne comme résultat :

- *La balise <DIV>*
- *Commentaire :*
- *N'oubliez pas l'attribut class!*

5 Balises de blocs et balise en ligne

Extrait de la spécification du HTML (W3C) : en HTML, on utilise des balises pré-définies afin de préciser à l'intérieur d'un fichier texte des éléments tels les titres, les paragraphes,

Au sein de celles-ci, on distingue les balises définissant un **bloc** des balises **en-ligne** par :

- le modèle de contenu
 - les éléments de bloc peuvent contenir à la fois données, éléments de bloc et éléments en-ligne ;
 - les éléments en-ligne ne peuvent contenir que des éléments en-ligne et des données. «L'idée inhérente à cette distinction structurelle, c'est que les éléments de bloc créent des structures « plus grandes » que les éléments en-ligne.»

le formatage

«Par défaut, les éléments de bloc sont formatés différemment des éléments en-ligne. En général, les éléments de bloc commencent sur une nouvelle ligne, et non les éléments en-ligne.»

Les feuilles de style fournissent les moyens de spécifier la restitution d'éléments arbitraires, y compris si l'élément est rendu comme étant de type bloc ou de type en-ligne.»

Exemples de balise de bloc : <p>, <pre>, <div>

Exemples de balise en ligne : , ,

6 Exercices immédiats

Que va afficher le code suivant :

```
<html> <head> <title> Essai </title> </head>
<body><!-- Essai--></body></html>
```

- Le mot Essai dans la page du navigateur
- Le mot Essai dans la barre d'état du navigateur
- Le mot Essai dans la barre de titre de la fenêtre du navigateur

Quels sont les avantages de l'examen d'un code source sur le web ?

Citer trois résolutions d'écran.

Comment va-t-on écrire **correctement** l'expression mathématique $0 < 1$? (0 inférieur à 1)

- a) 0<1 b) 0 < 1 c) 0<1

Quelle est l'entité correspondant au & (et commercial) ?

- a) il n'y en a pas b) && c) &

Quelle ligne de code est correcte pour écrire *La naïveté du mâle*

- a) La naïveté du mâle ?
 b) La naïveté; du mâle ?
 c) La naïveté: du mâle ?

Le code HTML suivant est-il correct :

`<html><body><!-- ?--></body></html>`

- a) Oui b) Non

La balise body affecte :

- a) l'arrière-plan et le texte seulement
 b) l'en tête (la partie déclarative)
 c) tout ce qui est affichable dans le navigateur

L'adresse de certaines pages web se termine par aucune extension (.htm, .html, .php, .jsp, ...) : vrai ou faux ?

Comment allez-vous entrer dans le code de votre page web le caractère **ç** si vous avez un clavier anglais ?

Comment code-t-on les couleurs noir et blanc ?

Ecrire le code qui affiche à l'écran : 'La balise <body> ouvre le corps du document'

Donnez deux ou trois exemples des polices les plus répandues sur le web.

Dans la balise , quel attribut permet de changer la taille du texte :

- a) width b) fontsize c) size d) height

Comment décrivez-vous rapidement la couleur cyan ?

Qu'est-ce qu'une police sans Serif ?

Créer une page web simple en suivant les instructions suivantes :

- le fond de la page est bleu clair
- le titre de la page s'intitule "Première page en HTML"
- la police générale est de couleur bleu marine
- la marge de droite est positionnée à 40 pixels

Square est-il

- a) la valeur d'un attribut ? b) le nom d'un attribut ? c) le nom d'une balise ?

Quelle balise sert à introduire un libellé dans une liste de définition :

- a) dd b) dl c) dt d) li

Ecrire une énumération ordonnée et numérotée avec des chiffres romains.

Ecrire le code pour deux énumérations ordonnées imbriquées.

II.3 Les images

1 les formats d'image

Les images numériques destinées à être affichées sur l'écran d'un ordinateur se divisent en deux catégories : les images matricielles et les images vectorielles (en Flash).

L'image matricielle (image bitmap) se représente sous forme d'une matrice de points allumés ou non, chacun d'une seule couleur ou de plusieurs. Le mécanisme d'affichage est simple et le calcul matriciel facilite le traitement de l'image. Ce type d'image contient un nombre fixé de points rapporté à une unité de longueur appelée **résolution**. La modification spatiale d'une image matricielle peut poser problème (perte d'information, effet d'escalier).

Les images matricielles sont très « lourdes » en stockage mémoire. Aussi, pour en réduire la taille mémoire, on les compresse. Ce processus de compression a été normalisé et a permis d'arriver aux formats du web : JPEG, GIF et PNG

Le format GIF

Il est doté d'une profondeur chromatique de 8 bits (256 couleurs possibles).

Il autorise la transparence dans les zones désignées (avec un éditeur) comme telles.

Il peut réunir plusieurs images accompagnées d'un minutage pour élaborer une animation (on peut trouver divers logiciels d'animation GIF sur le site www.download.com, rubrique gif animator).

Il n'est pas libre de droit.

Le format JPG ou JPEG

Il est doté d'une profondeur chromatique de 24 bits (16 millions de couleurs possibles) Il permet des compressions importantes et de bons dégradés. Il est parfaitement adapté aux photographies.

Le format PNG

C'est un format spécialement conçu pour le web, libre de droit. Il possède de très bons taux de compression et autorise la transparence. Il est capable d'intégrer une chaîne de caractères au sein même d'une image à des fins d'indexation (tatouage). Il n'est **pas encore bien pris en compte** par les navigateurs.

Remarque : ces formats d'images sont indépendants de la plateforme (comme html lui-même). Ceci veut dire qu'ils sont compris par tout navigateur.

2 Quelques points à savoir avant d'utiliser les images

Si vos pages web doivent inclure des images, vous avez à tenir compte de ce qui suit :

- de grandes images peuvent être "super", mais elle peuvent frustrer l'internaute impatient ! Limitez vous à 50Ko pour chacune.
- Tout le monde n'a pas un écran de 21 pouces ! Limitez vos dimensions d'images à une largeur de 700 pixels (points écran) et à une hauteur de 450 pixels pour éviter de forcer l'utilisateur à "scroller".
- Une grande palette de couleurs peut faire beau, mais souvenez-vous que les images ne sont pas si bien compressées !
- Plutôt que d'afficher toutes les images sur votre page, mettez-les en fichiers attachés (liés). (voir ci-après)
- Le plus important, assurez-vous que les images apportent un plus à la compréhension de vos pages web.

Remarques

- ici, on utilise les attributs **vspace** et **hspace** pour spécifier les possibilités d'écartement de l'espace entre le texte et l'image
- si l'on veut que l'image soit seule "sur une ligne", on insère sa balise `` au sein des balises de paragraphe `<p>`.

4 La largeur et la hauteur d'une image

On peut forcer les dimensions d'une image, en sachant que ceci peut nuire à sa qualité, avec les attributs **width** et **height** prenant pour valeurs un nombre de pixels. Ceci se fait alors en proportion.

```

```

On utilise des programmes graphiques pour redimensionner les images.

On peut aussi dimensionner les images en pourcentage de la taille de la fenêtre du navigateur

```

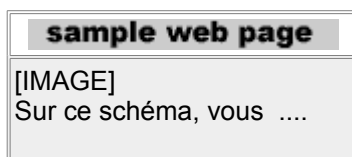
```

On peut aussi choisir une image comme fond de page. On utilise alors l'attribut **background** de la balise `<body>`. Syntaxe : `<body background="image.gif">`

Remarque : quand on choisit un fond de page, le choix de l'image est important. La texture de l'image doit être simple et relativement monochrome pour préserver la lisibilité du texte affiché. De plus, la taille en octet du fichier de l'image doit être faible pour ne pas retarder le chargement.

5 Les attributs alt et title

Pour les utilisateurs qui n'auraient pas de navigateur graphique, pour éviter un vide béant (la place de l'image), il est recommandé de spécifier, dans la balise d'insertion de l'image, un texte substituable à l'image. Ceci se fait comme valeur de l'attribut **alt** (alternative) : en lieu et place de l'image, l'internaute trouvera un texte.



pour le code ``

Si l'on veut afficher une bulle lorsque la souris s'arrête sur l'image, on utilise l'attribut **title** avec pour valeur le contenu textuel de la bulle

Exemple : ``.

Remarques : ces deux attributs peuvent améliorer le référencement de la page.

6 Exercices immédiats

Que signifie "image en ligne" ?

Que font les attributs alt et title ?

Donner le nom de deux logiciels de traitement d'images.

A l'affichage d'une page web, on peut parfois voir à la place d'une image un petit carré avec un dessin affiché à l'intérieur. Pourquoi ?

Mettre dans une page web une image comme fond d'écran.

Si l'on redéfinit la hauteur d'une image, qu'arrivera-t-il à sa largeur ?

Un pouce (inch) vaut : a) 3,1 cm ? b) 2,57 cm ? c) 2,54 cm ?

Sur combien de pixels va s'afficher une image de 2,5x2,5 pouces de résolution 100 dpi sur un écran de définition 80 dpi : a) 250 x 250 points ? b) 40 x 40 points ? c) 200 x 200 points ?

Combien peut-on définir de couleurs avec le modèle RGB ?

a) 256 b) 65536 c) 16777216

Quel le rapport entre la longueur et la hauteur de l'affichage à l'écran

a) 1,33 b) 1,2 c) 1,5

Que signifie le code ``

- a) que , quand l'utilisateur appuie sur la touche alt, le mot voiture est affiché ?
- b) que l'image a une hauteur de 20 pixels seulement ?
- c) que si l'image ne s'affiche pas, voiture s'affiche à sa place ?

Quelle est la bonne réponse ?

- a) le format JPG permet d'obtenir des images transparentes et très compressées
- b) le format JPG peut dégrader des images et le format GIF permet les images animées
- c) le format GIF ne dégrade pas les images et ne permet pas les images transparentes.

Votre écran de téléviseur est un 82 cm. Sa largeur (horizontale) est environ :

- a) 70 cm b) 60 cm c) 50 cm

*Remarque finale : on peut aussi intégrer du son à une page web avec la balise **embed** et ses attributs `src = « son.wav »` ; `autostart`, `loop` et `hidden` prenant la valeur « true » ou « false », `width` et `height` en pixels et `control= « console »`.*

II.3 Les liens

1 L'hypertexte

Les chapitres précédents étaient essentiellement axés sur la structuration et l'intégration d'éléments textuels et d'images au sein d'un document HTML. On restait en dimension 1 ("sur papier") alors que le web est multidimensionnel : lorsque l'internaute consulte le Web, il veut surtout **naviguer** d'un document à un autre. Il peut le faire par le biais d'un simple clic sur des **liens activables** dénommés **hyperliens** qui sont **externes** s'ils pointent vers d'autres pages ou **internes** si l'on reste au sein d'une même page.

L'idée de l'hypertexte est qu'un document contiennent des **liens** vers d'autres document , c'est-à-dire un **point de départ** (source ou zone activable par un clic de souris) vers un **point de destination** (ressource pointée par le lien qui peut être un texte, une image, un son ...).

La zone activable de départ peut être un caractère, un mot, un groupe de mots ou encore une image. Elle est matérialisée par le navigateur qui lui donne un aspect particulier (**couleur bleu et soulignement pour du texte**, bordure bleue pour une image) et "cache" sous elle un lien vers une ressource désignée par une **url**.

L'élément HTML qui correspond à la notion de lien est l'**ancree**. Elle est désignée par le marquage `<a>` et peut encadrer presque n'importe quel type de contenu. L'uri de la destination est précisée dans la balise par l'attribut obligatoire **href** (hyper-reference).


En voici la syntaxe : ` zone source cliquable `

Remarques

- Les liens hypertextuels sont habituellement en bleu et soulignés.

- Une image peut être un hyperlien. Pour ce faire, on code ` `
Pour aller au document X ``

Attention, le fichier image doit se trouver dans le dossier du fichier html. Sinon vous risquer de voir

afficher l'icone suivante : 

2 Liens vers des fichiers "locaux"

On veut "poser une ancre " vers une autre page qui se trouve sur la même machine que la page appelante "volc.html".

Si cette deuxième page est dans le même dossier, il suffit d'écrire

```
<a href="page2.html">texte cliquable</a>
```

Ancre vers une image

```
<a href="image1.gif">texte cliquable</a>
```

 si l'image est dans le même dossier.

Maintenant, si le fichier appelant volc.html est dans le dossier de travail (work area) et l'image appelée, msh.gif, est dans le dossier pictures, on doit écrire le code suivant :

```
<a href="pictures / msh.gif">texte cliquable</a>.
```



Ancre vers un dossier parent

Les liens que nous construisons sont des liens **relatifs** : l'url est basée sur la localisation de la page html du lien appelant volc.html.

Ici, le lien invoqué nous conduit dans un dossier parent.



```
<a href="../../home.html">Retour à la page d'accueil</a>
```

```
<a href="../../pictures / msh.gif">vers une image placée dans le dossier pictures<a>
```

Remarques :

- on a pris l'habitude de placer les images dans un autre dossier parent ou non ; tout dépend de l'usage fait.

- si l'on change volc.html en index.html, un lien pointant vers ce fichier est tout simplement

www.xxx.edu/volcano. Ceci parce que les serveurs web ont un nom standard pour la page par défaut d'un dossier.

3 Lien vers une ressource n'importe où sur l'internet

On utilise une adresse absolue :

`http://www.nomdedomaine.domaine.racine/dossier/sous-dossier/nomdelapage.htm`

` Accéder à nos formations Bac+3, 4 ou 5 `

`Belle image`

Un lien peut aussi pointer vers un autre service de l'Internet (messagerie, transfert de fichiers, ...) que le web via de nouveaux url (pseudo-url) :

- ♦ **ftp** : cette url permet d'offrir à travers un serveur Web une connexion vers les serveurs de fichiers anonymes. Exemple : `Télécharger un fichier du répertoire pub`. Ce lien va aller chercher un fichier dans le dossier public de l'Ecole Nationale des Ponts et Chaussées.
- ♦ **mailto** : cet url lance un agent de mail. Lorsqu'elle est appelée, une zone de saisie est proposée à l'internaute lui permettant de rédiger un courrier électronique, d'en donner l'objet et éventuellement les autres destinataires. Exemple : ` Pour envoyer un email`

4 Lien vers une section de la page appelante

Une **ancree nommée** est un pointeur vers une section particulière de la page. On l'utilise dans le cas de longues pages pour accéder directement à une section souhaitée

Pour poser une ancre nommée, on la nomme ! `Contenu recherché`

Pour écrire le lien vers l'ancree nommée, on écrit `Pour aller vers le contenu recherché`.

Le symbole #indique au navigateur de rechercher dans le document une ancre nommée appelée "nomAncre". Une fois le lien activé, on est téléporté vers le contenu recherché.

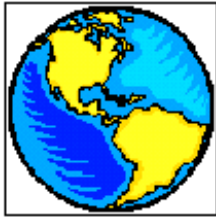
5 Base de résolution des liens hypertextes

On a vu qu'il y a deux façons de définir un lien hypertexte. Soit on spécifie un lien **absolu** (par exemple "<http://www.exemple.com/index.html>"), le chemin mène directement au bon endroit. Soit on définit un lien hypertexte en **relatif**, le chemin mène au dossier du fichier html appelant (page2.htm). Il est possible que tout lien relatif soit résolu non plus en rapport à la localisation de la page appelante, mais en rapport à une autre **base de résolution de lien** (un autre référentiel pour les liens).

Pour ce faire, il vous suffit d'inscrire une balise `<base>` dans l'en-tête avec l'attribut **href** qui détermine la **base de résolution** des liens du document qui ne sont pas complètement spécifiés (lien relatif). A titre d'exemple, considérons une page quelconque contenant le lien hypertexte "page1.html". Si dans cette même page est mentionnée `<base href="http://www.serveur.fr"/>` et si on active le lien, on est redirigé vers le document ciblé par l'url <http://www.serveur.fr/page1.html>.

6 Exercices immédiats

Comment faire pour supprimer la bordure hypertexte autour d'une image ?



Que signifie le code ` `

- a) que l'image image.gif apparaît si doc.htm n'est pas trouvé ?
- b) qu'un lien est créé vers la page doc.htm ?
- c) qu'un lien est créé vers l'image ?

Qu'est-ce qu'une adresse relative

- a) une adresse vers une ressource externe ?
- b) une adresse vers une ressource qui se trouve sur le même serveur ?
- c) aucune de ces réponses ?

Réaliser un mécanisme de notes de bas de page.

Exemple de TP à faire : une image réactive

On peut souhaiter, en cliquant sur des zones spécifiques d'une image (représentant une carte géographique par exemple), obtenir différentes informations liées à l'endroit du clic. On utilise alors ce qu'on appelle une **image réactive**. Pour la construire, le premier travail consiste à délimiter chaque zone cliquable en la repérant dans le référentiel habituel (origine en haut et à gauche, séries de points définis en pixels).

La délimitation des zones se fait à l'aide de formes géométriques prédéfinies (**shapes**) :

- **Rectangle** (rectangle dont on donne les sommets supérieur gauche et inférieur droit),
- **Circle** (cercle dont on donne le centre et un point quelconque de la circonférence),
- **Poly** (ligne polygonale fermée dont on donne les coordonnées des extrémités des côtés).

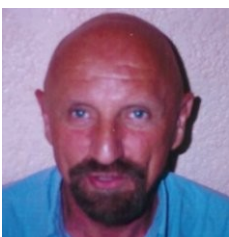
Pour ensuite coder, on définit l'image réactive avec la balise `` dotée d'un attribut supplémentaire `usemap` : ``

Le code de description de la carte se situe entre les balises `<map name="nomCarte">` et `</map>`.

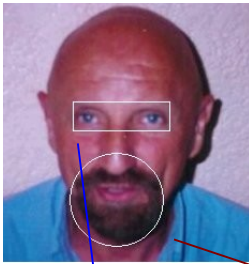
Entre ces deux balises, on dispose de plusieurs balises `<area>`

`<area shape="TypeFigure" href="url" coords="x0, y0, x1, y1," />`

Pour réaliser concrètement une image réactive, deux éléments sont indispensables : un fichier image (.gif ou .jpg) et la définition des zones cliquables. Pour ces dernières, on utilise un logiciel spécifique (gimp ou mieux mapedit).



On choisit de cliquer sur les yeux (et d'ouvrir une image, yeux_bleu.jpg) ou de tirer la barbe (et d'ouvrir une nouvelle page, barbe.html)



CLIC !

yeux_bleu.jpg

barbe.htm

C'est un patient qui doit subir une intervention chirurgicale très lourde genre quadruple pontage coronarien et il est mort de trouille. Son chirurgien qui s'en aperçoit décide de le rassurer avant que l'anesthésiste ne l'endorme. - Ne vous en faites pas, tout va très bien se passer. Tenez, vous voyez cette barbe ? (Nota : le chirurgien est barbu) Et bien quand vous vous réveillerez, vous la retrouverez.

Sur ce, un peu rassuré, notre patient se laisse endormir en pensant à la barbe. À son réveil, notre patient aperçoit, dès qu'il ouvre l'oeil, une silhouette penchée sur lui. Et, il lui semble reconnaître une barbe!! Alors, il se rappelle de ce que lui avait dit son chirurgien et, fou de joie, commence à caresser la barbe.

- Oh, merci docteur ! Je savais que je pouvais vous faire confiance. Je suis si heureux de retrouver cette barbe.

- ça va, ça va. Lâchez-moi... Et puis d'abord, je ne suis pas docteur, je suis Saint Pierre...

Pour obtenir cet effet, voici le code html :

```
<html>
<head> <title> Image cliquable ou réactive </title> </head>
<body>
  
  <map id="idAS" name="idAS">
    <area shape="rect" #Quels beaux yeux bleux coords="67,94,157,121"
      href="yeux_bleu.jpg" title="Quels beaux yeux bleux" />
    <area shape="circle"#Barbant coords="108,186,44" href="barbe.htm"
      title="Barbant" />
    <area shape="default" nohref="" alt="" />
  </map>
</body>
</html>
```

Les coordonnées des points de l'image sont définies en pixels suivant les axes x et y, l'origine étant le point en haut et à gauche de l'image.

III La représentation visuelle d'un document avec CSS

III.1 Définitions

1 le document HTML et ses éléments

On a vu :

* Un document HTML est un **flux de données textuelles**, éventuellement **combiné** avec d'autres flux de données qu'il **réfère** alors (le plus souvent, il est multimédia). Quand on veut insérer une image dans le document, on utilise un "pointeur" (sous forme d'élément vide).

* Le document est structuré en **éléments** (terme issu de SGML) composés :

- d'une balise ouvrante,
- d'un contenu
- d'une balise fermante.

.Exemple d'élément en HTML : `<p> Ceci est un contenu de paragraphe. </p>`

Un élément unique introduit et encapsule le document : c'est l'élément **racine** qui lui donne une cohésion sémantique.

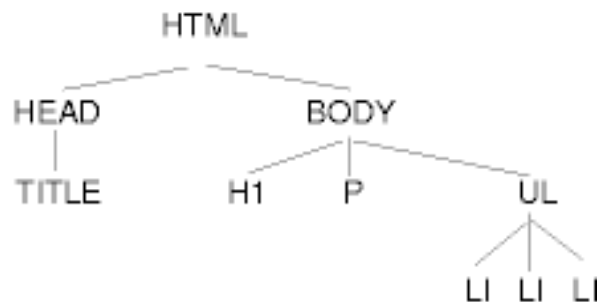
2 L'arbre de document

Ayant un **seul** élément racine car doté d'une bonne imbrication des éléments, un document peut avoir une **représentation arborescente et ordonnée** définie à partir de la structure de ses éléments.

Voici un exemple de document source dans le langage HTML :

```
<html>  <head> <title>Ma page personnelle</title> </head>
<body>
  <h1>Ma page personnelle</h1>
  <p>Bienvenue sur ma page personnelle ! Voici mes compositeurs préférés :</p>
  <ul>
    <li> Elvis Costello</li>
    <li> Johannes Brahms</li>
    <li> Georges Brassens</li>
  </ul>
</body> </html>
```

et maintenant sa représentation arborescente :



Les éléments se positionnent les uns par rapport aux autres dans l'arbre, dans l'ordre où ils apparaissent dans le document et suivant leur **imbrication**.

L'arbre se dessine bien car chacun des éléments a exactement un **seul parent** à l'exception de l'élément **racine** qui n'en a pas.

Ma page personnelle

Bienvenue sur ma page personnelle ! Voici mes compositeurs préférés :

- ♦ Elvis Costello
- ♦ Johannes Brahms
- ♦ Georges Brassens

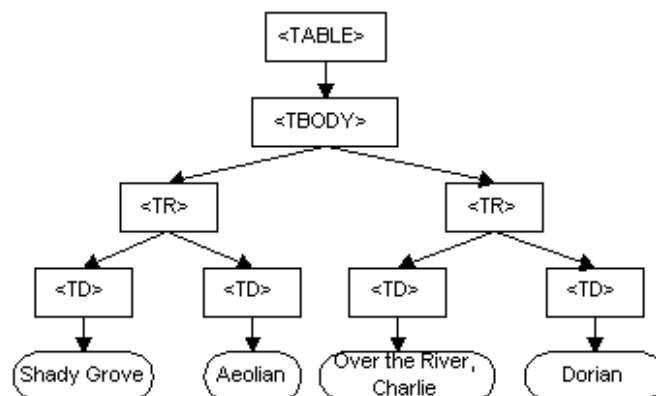
3 Le Modèle Objet Document *(en deuxième lecture pour une utilisation ultérieure avec Javascript)*

Le Modèle Objet de Document (DOM) est une interface de programmation d'applications (API) pour documents HTML et XML. Il permet de reconstruire en mémoire la structure logique d'un document et fournit une batterie de fonctionnalités pour le manipuler.

Cette interface ressemble étroitement à la structure des documents qu'elle modélise. Par exemple, si l'on considère le tableau suivant, extrait d'un document HTML :

```
<table>
  <tr>
    <td> Shady Grove </td>
    <td>Aeolian</td>
  </tr>
  <tr>
    <td>Over the River, Charlie</td>
    <td>Dorian</td>
  </tr>
</table>
```

DOM le représente comme ceci :



Le nom "Modèle Objet de Documents" a été choisi parce qu'il s'agit d'un "modèle objet" au sens traditionnel de la conception orientée objet. Le modèle ne contient pas uniquement la structure du document mais également **son comportement et celui des objets dont il est composé**. En d'autres termes, les noeuds du dessin ci-dessus ne représentent pas une structure de données, ils représentent des objets ayant des fonctions et une identité.

En tant que modèle objet, DOM identifie :

- les interfaces et les objets utilisés pour représenter et manipuler un document
- la sémantique de ces interfaces et objets - incluant le comportement et les attributs
- les relations et collaborations entre ces interfaces et ces objets

4 L'utilisation du DOM

Avec le Modèle Objet de Document, les programmeurs peuvent construire des documents, naviguer dans leur structure, et ajouter, modifier, ou supprimer soit des éléments soit du contenu. Tout ce qui peut être trouvé dans un document HTML ou XML peut être accédé, changé, détruit, ou ajouté en utilisant le Modèle Objet de Document, à quelques exceptions près.

Nous en verrons une implémentation avec Javascript.

Symboliquement, on pourrait changer la valeur de la deuxième cellule de la première ligne dans l'exemple ci-dessus : `nomTable.NomTBody.TR[1].TD[2].valeur="White shade of pale"`

III. 2 Le langage CSS

L'idée directrice de ce chapitre est la séparation du contenu et de sa présentation (sur différents médias).

1 Le langage CSS

Dans ce cours, on a retenu le langage CSS, bien implémenté dans les dernières versions des principaux navigateurs et utilisable avec les deux principaux langages de balisage, HTML et XML. Ici, il s'intègre à son langage hôte, HTML, soit dans une balise de style (insertion de règles de style), soit via un fichier associé.

Le langage HTML "classique" les valeurs par défaut des propriétés de style du langage CSS.

CSS signifie Cascading Style Sheets. Lorsqu'un élément est défini avec un certain style, ce style se répercutera non seulement dans toutes les occurrences de l'élément mais encore dans la plupart des éléments enfants (héritage au sens de l'inclusion des balises). Dans le cas où plusieurs entrent en jeu dans la présentation de la page, des règles de priorité jouent (lesquelles ?).

Le langage CSS est un **langage déclaratif non sensible à la casse** qui a sa propre syntaxe. Le W3C en a publié deux versions, **CSS1**, en 1996 qui a défini beaucoup de règles de styles et **CSS2**, en 1998, qui a ajouté une prise en charge des styles autres que visuels et placé HTML et XML sur le même pied d'égalité.

Il permet la composition de fichiers texte sauvegardés avec l'extension **.css**

Les commentaires s'affichent à l'aide des marqueurs `/* */` comme en PHP et Java.

2 Un langage de style pourquoi faire ?

Avant tout, pour fournir un outillage destiné à représenter un contenu sur des médias en continu ou paginé.

Les feuilles de styles offrent une nouvelle façon d'intervenir dans la présentation d'une page Web. Le langage HTML intégrait auparavant des balises de présentation typographiques ou de mise en couleur (``, ``, ...) pour les pages web. Cette façon de procéder était dispersée, dépendante du navigateur (par exemple, le style `<h1>` marquant un titre de niveau 1 s'affiche par défaut dans une police Times Roman de 24 pts) et limitée à quelques ornements (gras, italiques, titres, couleurs).

L'intérêt premier des feuilles de styles est la **séparation du contenu et de son rendu** : l'utilisateur reçoit l'information en fonction du terminal utilisé (un contenu pourra avoir plusieurs présentations suivant sa destination : téléviseur, imprimante, téléphone, ...).

Un second apport est de donner via des fichiers de styles .css une **charte graphique** à l'ensemble d'un site web et partant, non seulement une cohérence graphique aux pages web mais aussi une symbolique pour l'entreprise.

Un troisième point est le **positionnement** des éléments dans une page à l'aide de boîtes et conteneurs.

3 Le modèle visuel

Ce modèle va transcender les modèles précédents qu'étaient l'utilisation de la structure de tableaux en tant que grille de la page ou encore l'utilisation de pages de frames. Ces derniers modèles étaient rigides, mais surtout liés directement au contenu du document.

Le nouveau **modèle de boîtes** décrit des rectangles générés pour les éléments de l'arbre document en accord avec le **modèle de rendu visuel**.

Dans ce modèle, chaque élément de l'arbre produit zéro, une ou plusieurs boîtes. La mise en place des boîtes est gouvernée par

- la dimension de la boîte,
- le type de la boîte,
- son schéma de positionnement (flux normal, positionnement flottant, absolu ou fixe),
- la position des éléments dans l'arbre du document et les informations externes

Ces éléments de contrôle dans la génération des boîtes seront vus plus loin. D'ores et déjà, on peut dire que l'élément racine génère une boîte qui sert de conteneur initial pour les autres boîtes à venir.

En CSS2, les positions et les tailles d'un grand nombre de boîtes sont calculées en fonction des bords d'une boîte rectangulaire ; on l'appelle un bloc conteneur. La plupart des boîtes générées se comportent comme des blocs conteneurs pour les boîtes qui en sont les descendantes. L'expression "le bloc conteneur d'une boîte" signifie "le bloc conteneur dans lequel se trouve la boîte", et non la propre boîte que celle-ci génère.

4 Les agents utilisateurs

On donne le nom **d'agent utilisateur** à tout logiciel capable d'interpréter un document dans un langage donné et de mettre en œuvre les feuilles de style qui lui sont associées selon cette spécification. Celui-ci peut afficher un document, le lire en synthèse vocale, l'imprimer, le convertir vers un autre format, etc.

Les agents utilisateurs pour médias continus offrent généralement une **zone de visualisation** (une fenêtre ou une autre zone d'affichage sur un écran) grâce à laquelle l'utilisateur peut consulter un document. Les agents utilisateurs peuvent changer l'affichage du document quand cet espace est redimensionné.

5 Exercices immédiats

Positionner des boîtes dans une zone de visualisation (reprendre l'exemple de « ma page personnelle ci-dessus »).

Décrivez vos boîtes (texte, image) et leur positionnement dans le flux du document

III.3 Les règles de style

Le mécanisme de base pour associer une partie sélectionnée du document à un rendu visuel stylisé repose sur un **jeu de règles de styles**.

1 A telle partie, telle apparence

Une **règle de style** se compose :

- d'un **sélecteur** qui indique l'élément auquel elle s'applique,
- d'une ou de plusieurs couples **propriété / valeur** entre accolades.

```
sélecteur {propriete1: valeur ;
           propriete2: valeur ; ...
           proprieten: valeur ;
           }
```

Les couples **propriete: valeur** forment ce que l'on appelle la **déclaration de style**. On peut mettre entre accolades autant de déclarations que nécessaire en les séparant par des points-virgules.



Remarques :

- A tout élément de la structure du document, on peut associer sa ou ses déclarations de style.
- Attention : on ne met plus de guillemets autour des valeurs !

2 Sélection et sélecteur du passage à styliser

Il s'agit de découper le document (fabriquer des sélecteurs) sans impliquer obligatoirement et uniquement les balises HTML vues jusqu'ici. Comme on l'a vu, pour délimiter à volonté la partie à "styliser", on va utiliser les deux balises génériques (**div** et **span**) qui vont spécifier n'importe quelles divisions logiques au sein d'un document et les doter de noms et de styles propres.

<div> agit au niveau de blocs délimités par des retours à la ligne (comme **<p>**, **<h1>** ...).

**** agit dans le flux du texte (comme ****, **<i>**, ****).

Par exemple, pour mettre en rouge le mot danger dans la phrase "Attention, il y a **danger**ici ...", on utilise la règle de style **span {color: red;}**

Les deux balises (**<div>** et ****) vont introduire les deux types principaux de boîtes.

Les attributs id et class

Ces attributs vont permettre de spécifier les éléments de "découpe", de les affiner. De plus, la **valeur nominative** de l'attribut va permettre de regrouper un certain nombre d'éléments, ceux qui justement appartiennent à la même "classe" (*attention : le nom donné à cet attribut n'a rien à voir avec une classe en Java*).

Prenons un exemple. Certains paragraphes du document qui ont un fond bleu sont repérés dans le document par des balises dotées de l'attribut class prenant comme valeur « enbleu » :

```
<p class= "enbleu">
```

Dans la définition de la règle de style correspondante, on note ce sélecteur par un point suivi immédiatement du nom de la classe.

```
<style>
```

```
  p.enbleu {background-color: blue;} /*p.enbleu en un seul tenant */
```

```
</style>
```

Remarque : on pouvait aussi le faire directement en utilisant **<p style="background-color: blue">**, mais si ce type de paragraphe est répété dix fois, si l'on veut par la suite que le fond soit vert, il faudra modifier le codage à dix endroits. D'où l'intérêt d'un regroupement en classe.

La définition d'une classe peut être encore insuffisante si l'on veut régler un grand nombre de paramètres. Imaginons que l'on veut définir deux types de paragraphe de la même classe (premier)

dont seule la couleur de la police diffère. On va faire appel à un autre attribut **id** qui va autoriser une particularité ou une exception dans une classe (il est unique).

Exemples :

`<p class="premier">` Ce texte sera en caractères de 20 points et en gras `<p>`

`<p class="premier" id="vert">` Ce texte sera en caractères de 20 points et en gras, mais aussi en vert `<p>`.

Dans la définition de la règle de style correspondante, on introduit ce sélecteur par le caractère **#** (dièse) suivi immédiatement de la valeur nominative de l'attribut **id**.

Au niveau de la définition du style (dans la balise ou le fichier), pour reprendre le dernier exemple, on écrira :

```
<style>
  .premier {font-size: 20pt ; font-weight: bold;}
  #vert    {color: green}
</style>
```

3 Regroupement de sélecteurs ou de déclarations

Pour condenser la rédaction d'un style et en accroître la lisibilité, on peut regrouper les sélecteurs en les séparant par une virgule : **h1, h2, h3, .premier** {font-family: arial}

Les textes contenus entre les balise **h1, h2, h3** ou **div class="premier"** sont affichés en Arial.

De la même façon on peut regrouper plusieurs déclarations de styles.

```
h1 {font-weight: bold; font-size: 12pt; line-height: 14pt; }
```

Il est possible aussi de préciser le contexte dans lequel le style va s'appliquer, par exemple uniquement à **"l'intersection"** de deux sélections : **b i** {color: blue}

Un élément en italique (i) est coloré en bleu si cet élément est lui-même situé au sein d'un élément en gras (b) (autrement dit si l'élément italique (i) est un descendant de l'élément gras (b)).

Remarque : s'il s'agit de l'intersection de deux sélecteurs, le séparateur est l'espace et non plus la virgule.

4 Sélections particulières (à l'aide de « pseudos » ou d'attributs)

Les pseudo-éléments et pseudo-classes

Un pseudo-élément permet de sélectionner non des éléments entiers mais des **parties d'élément** comme par exemple la première lettre d'un paragraphe (**p:first-letter**)

La syntaxe d'utilisation est : **sélecteur:pseudo-element** { propriété:valeur }.

Exemple :

```
<html>
<head>
  <style type="text/css">
    p:first-letter {float: left; font-size: 300%; }
    span {text-transform: uppercase }
  </style>
</head>  <body>
  <p> <span>Les premiers</span> mots d'un article de journal, les voici :
jdhu zdndheouidh hofhoi iodjdjeidfeifj f,nijeic rjfr"jr'oàfjkop
ckfrof kropfk koperkfopkvopevk kfrookojk lefpelkpêlkdcvpêkl kf'riopjfer dk
riojerv ifjioj jkvjhnrei ovjn rifj njev iervjioev .....  </p>
</body>  </html>
```

LES PREMIERS mots d'un article de journal, les voici : jdhuzd ndheouidh hofhoi iodjdjeidfeifj f,nijeic
 rjfr"jr'oàfjkop ckfrofropfk koperkfopkvopevk kfrookojk lefpelkpêlkdcvpêkl kfriopjfer dk riojerov
 ifjioj jkvjhnreiovjn rifj njev iervjioev

Une pseudo-classe permet **d'attacher un événement à un élément**, par exemple le passage de la souris sur une ancre (**a:hover**).

Autre exemple de style : **a:visited** {background-color :navy ;color :gray ;}

La sélection à l'aide d'un attribut

pour sélectionner un élément qui possède un attribut, on met ce dernier entre crochets.

Pour faire apparaître en bleu les éléments p ayant un attribut align, on écrit : **p[align]** {color:blue}
 apparaissent en bleu

Sélection d'un élément qui possède un attribut de telle valeur : **p[align="center"]** {color:blue}

5 Exercices immédiats

Nommer les deux composants d'une règle de style

Nommer les deux composants d'une déclaration de style

Quel est l'intérêt majeur de l'utilisation d'un langage de style ?

Pourquoi utiliser une règle de style pour spécifier des types de caractères plutôt qu'une balise ?

Que signifie CSS ?

- Creative Style Sheets
- Computer Style Sheets
- Colorful Style Sheets
- Cascading Style Sheets

Pourquoi n'utilise-t-on pas en CSS la même syntaxe de commentaire qu'en HTML ?

Ecrire la règle qui met en rouge les mots en italiques des titres de niveau 2.

Ecrire la règle qui met en gras et en rouge un seul mot d'un texte donné.

Ecrire une règle qui formate les éléments <i> en rouge uniquement lorsqu'ils sont présents dans une balise <p>

Ecrire une règle qui formate un titre de niveau 1 en inversé (blanc sur fond noir).

Quelle différence voyez-vous entre les balises <div> et ?

Quelle différence voyez-vous entre la déclaration d'un sélecteur lié à l'attribut id et à l'attribut class ?

III.4 L'intégration des styles dans une page web

Il s'agit ici de savoir associer des styles au langage HTML.

1 Par l'utilisation d'une balise de style

C'est l'utilisation classique. La balise de style s'écrit dans l'en-tête du document : elle associe sélecteurs à des déclarations de style pour la page web associée.

Exemple :

```
<html> <head>
  <style type="text/css">
```

```

    .special {color: red}
    .titre {font-size: 24}
</style> </head>
<body bgcolor="yellow">
  <span class="titre">TITRE DU JOUR</span><br />
  Ce livre est d'une<span class="special">couleur magnifique</span>.
  Il est ... <br />
</body> </html>

```



On voit affiché "une**couleur**" Comment décoller le e du c?

2 Par l'association d'une feuille de style

C'est l'utilisation à privilégier pour homogénéiser la présentation des pages web. Une feuille de style est un document externe au document web : c'est un fichier **.css** relié au fichier HTML par la balise **<link>** qui doit se trouver dans l'en-tête du fichier HTML. C'est une approche plus globale (le même fichier .css peut être relié à plusieurs pages) qui permet de construire une charte graphique pour un site entier.

Cette balise prend plusieurs attributs : rel, type et href.

- **rel** indique soit la feuille par défaut (stylesheet) ou une feuille accessoire (alternate stylesheet) car il peut y avoir plusieurs feuilles de styles associées.
- **type** indique le type MIME de la feuille de style.
- **href** indique l'url de la feuille de style.

Exemple de fichier HTML

```

<html> <head>      <title>Fichier avec CSS</title>
  <link rel="stylesheet" type="text/css" href="style1.css" /> </head>
<body>
  <font size="5">Voici un exemple de fichier HTML avec<br /> une
  déclaration de style dans un fichier externe</font> <p/>
  <h1>Ce texte de niveau 1 est en rouge</h1>
  <h3>Ce texte de niveau 3 est en vert</h3>
  <h2 id="bleu">Mais ce texte de niveau 2 est bleu </h2>
</body> </html>

```

Exemple de fichier (texte) de style associé : style1.css

```

/* style1.css */
h1    {color: #ff0000}
#bleu {color: blue}
h3    {color: green}

```

Remarques :

- Dans une feuille de style .css, seules sont inscrites les règles de style : il n'y a pas de balise style à ajouter.
- Plusieurs feuilles de styles peuvent être reliées à un même document (une pour l'affichage, une pour l'impression, ...). Dans ce cas, certaines règles peuvent entrer en concurrence.

3 Utilisation de l'attribut style **Cette utilisation est désuète.**

Si, par exemple, à un seul paragraphe comportant des consignes très dangereuses, on veut donner un style particulier, on dote la balise du paragraphe d'un attribut style :

```
<p style="font-size :30; color :red"> DANGER </p>
```

Cette utilisation de l'attribut style est désuète.

4 Règles de priorité

Il se peut qu'avec l'introduction de styles à plusieurs niveaux il y ait des déclarations de style contradictoires comme dans l'exemple suivant :

```
<html>  <head> <title>Fichier avec CSS</title>
  <link rel="stylesheet" type="text/css" href="style1.css">
<!--dans le feuille de style, les titres h1 sont déclarés être mis en rouge -->
</head>
<body>
Voici un exemple de fichier HTML avec
<br /> une déclaration de style dans un fichier externe
<br /> et une autre locale <p/>
  <h1 style="color:yellow">Ce texte de niveau 1 n'est donc plus en rouge ! </h1>
  <h3> Ce texte de niveau 3 est en (vert) fuchsia</h3>
  <h2 id="bleu">et ce texte de niveau 2 est en bleu </h2>
</body> </html>
```

avec le fichier de style style1.css

```
h1 {color: #ff0000} // couleur rouge
#bleu {color: blue}
h3 {color: green}
```

ce qui donne

Voici un exemple de fichier HTML avec
une déclaration de style dans un fichier externe
et une autre locale

Ce texte de niveau 1 n'est donc plus en rouge !

Ce texte de niveau 3 est en fuchsia

Mais ce texte de niveau 2 est en bleu

Remarque : la déclaration de style la plus proche de la portion de texte sur laquelle elle porte a priorité.

5 Exercices immédiats

Reprenez le fichier HTML suivant en supprimant les balises obsolètes et en respectant la syntaxe XHTML

```
<html> <head> <title>Fontes</title> </head>
<body>
Ce texte est affiché dans la police par défaut du navigateur avec sa taille
courante <font size="1">Affichage de la plus petite taille</font>
<p><font size="7">Affichage de la plus grande taille</font>
<p><font size="3" color="#FF0000">Texte en taille 3 et en rouge</font>
<p><font size="+2" face="arial">Augmentation relative de la taille et
passage en arial</font>  </body> </html>
```

Quelle est la différence d'effet entre **p:first-letter** {float:left; font-size:48pt} et span { font-size:48pt }

Pour se référer à une feuille de style, où placer le code dans le document HTML?

- A la fin du document
- Dans le corps du document <body>
- En haut du document
- Dans l'en-tête du document <head>

Ecrire le code à l'aide déclarations de style pour l'image suivante :



III.5 Les premières propriétés de style

Objectif : connaître quelques propriétés de styles les plus usuelles (il y en a plusieurs centaines)

1 Les unités dans les valeurs de déclaration de styles

On a vu que pour attribuer une valeur à une propriété dans le langage CSS, on utilise la symbole deux-points (:) et non pas le symbole = comme on le fait pour les valeurs d'attributs. La valeur affectée à une propriété peut être numérique, alphanumérique ou spécifique. On ne la place pas en général entre guillemets **sauf** par exemple si la valeur attribuée est en plusieurs mots.

Le langage CSS propose différentes unités de mesure (système décimal avec point ; valeur absolue ou relative) :

Unités absolues : on peut utiliser **mm** (millimètre), **cm**, **inch** (25,4 mm), **pt** (point) qui vaut 1/72 d'inch, mais la plus répandue est **px (pixel)**.

Unités relatives : **%** (pourcentage) de quelque chose défini dans son 'parent', **em** qui représente la taille de la police courante (en général la largeur du m minuscule) ce qui peut être utile pour indenter des textes.

Nombres hexadécimaux : pour les couleurs (codées sur trois octets. Exemple : color:**#FF0000**).

2 Les propriétés typographiques

Le contenu HTML est essentiellement du texte. Il est donc normal qu'une partie importante de CSS soit consacrée à sa mise en forme. On va donc s'intéresser successivement aux familles de polices, aux graisses de police, aux hauteurs de ligne, au crénage, aux retraits de texte, aux couleurs

Familles de polices

font-size permet de déterminer la taille de la police de caractères.

font-family permet de définir une famille de police de caractères qui sera utilisée pour un élément

Exemple : **p {font-family: arial, "courrier new"}**

Celle-ci doit être installée sur l'ordinateur. Dans la négative, le navigateur utilise sa police par défaut à moins que l'on mentionne une police de substitution (comme ici Courier New).

Graisses de police

font-weight permet de définir la "graisse" d'une police (de 100 à 900) ou avec des valeurs prédéfinies. Exemple : font-weight : bold;

*Remarque : on peut utiliser en raccourci la propriété font avec plusieurs valeurs des propriétés précédentes : **font : 20pt bold arial***

Hauteurs de ligne.

Une valeur absolue ou un pourcentage (du corps de la police employée) permet de spécifier la hauteur d'un ligne. Exemple : **line-height :30pt ;**

Crénage

Le crénage désigne en imprimerie l'espace entre les lettres : **letter-spacing: 2 pt;**

Retraits de texte et alignements

text-align permet de déterminer l'alignement horizontal du texte d'un élément : left, center, right.

text-indent permet de déterminer le retrait de la première ligne d'un paragraphe (valeur ou %)

Couleurs et arrière-plans

color définit la couleur du texte

background-color définit la couleur de l'arrière plan

background-image propose via son url une image d'arrière plan.

Background-repeat répète ou non l'image dans le fond.

margin permet de définir la largeur des marges invisibles autour des quatre côtés d'un bloc. Elle peut prendre une des deux valeurs (largeur en px ou auto, option par défaut) répétée de une à quatre fois. Exemple : p {margin : 15px 25px}. Cette propriété peut être déclinée individuellement : margin-top, margin-left, ...

padding permet de définir l'espace qui se trouve entre le bord extérieur d'un élément et sa bordure (même utilisation et syntaxe que margin).

La plupart des règles de bordure prennent en charge trois propriétés (style, width et color) et peuvent admettre de une à quatre formes suivant qu'elles s'appliquent aux quatre côtés en même temps (1), aux côtés deux par deux (bas-haut, droite-gauche) (2), au haut, aux cotés droite-gauche et aux bas (3), aux quatre côtés pris individuellement dans le sens des aiguilles d'une montre (4).

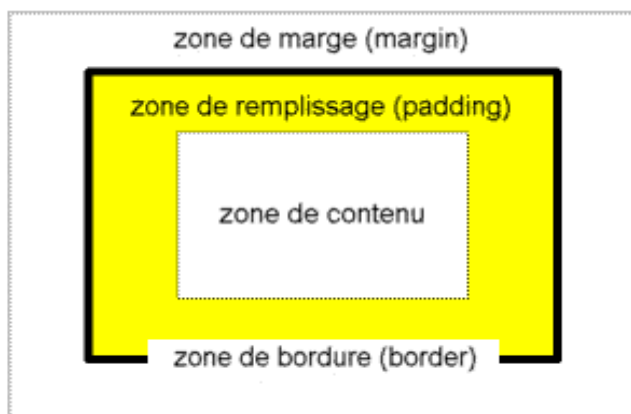
border-color définit la couleur de la bordure du bloc : h2 {border-color: red}

border-style définit le style de la bordure : double, none, solid, (même syntaxe)

border-width définit l'épaisseur de la bordure : thin, medium, thick, n pixels (même syntaxe)

border seulement avec 3 arguments successifs : la largeur, le style et la couleur (le plus utilisé).

width, height permettent de définir respectivement la largeur et la hauteur d'un bloc



Autres exemples :

```
.p {background-color: yellow ; width:200}
```

```
h1 {background-image:url(arrierePlan.jpg);background-repeat:no-repeat;}
```

Remarque : quand on utilise des couleurs, il faut savoir que 5 à 10% des visiteurs distinguent mal les couleurs.

Ceci n'est pas une liste exhaustive, il y a encore d'autres propriétés :

font-style définit un style (italique par exemple).

text-decoration permet de présenter le texte de l'élément biffé (line-through), souligné (under-line), surligné (over-line) ou clignotant (blink). Exemple : p {text-decoration: underline}

text-transform permet le passage de majuscules (uppercase) en minuscules (lowercase)

3 Exercices immédiats

Quand on donne une valeur a une propriété, doit-on utiliser des guillemets ?

Qu'apporte la propriété background-repeat ? Sur quels éléments s'applique-t-elle ?

Laquelle de ces syntaxes CSS est correcte?

- body:color=black
- {body:color=black(body)}
- body {color: black}
- {body;color:black}

Comment insérer un commentaire dans une "partie" CSS?

```
// Ceci est un commentaire //
' Ceci est un commentaire '
/* Ceci est un commentaire */
// Ceci est un commentaire
```

Comment ajouter une couleur de fond pour tous les éléments h1 de votre page ?

```
h1.all {background-color:#FFFFFF}
h1 {background-color:#FFFFFF}
all.h1 {background-color:#FFFFFF}
```

Comment changer la couleur de texte d'un élément textuel ?

```
text-color:          text-color=          color:          fgcolor:
```

Quelle est la syntaxe correcte pour mettre en gras le contenu de l'élément <p>?

```
<p style="text-size:bold">          <p style="font-size:bold">
p {text-size:bold}                  p {font-weight:bold}
```

Comment changer la marge de gauche d'un élément?

```
- text-indent:          - margin:          - indent:          - margin-left:
```

4 Ressources : index des principales propriétés

Toutes les propriétés en CSS ont une valeur par défaut.

Guide de références

<http://www.htmlhelp.com/reference/css/>

Propriété	Valeurs	Description
<u>BACKGROUND</u>	background-color background-image background-repeat background-position	Définition des caractéristiques du fond d'écran Répétition : repeat-x, repeat-y, no-repeat Position d'une image de fond : taille, pourcentage, top, left ...
<u>BORDER</u>	border-width, border-style border-color, ...-top, ... spacing right, , ... top-color, top-style, ...	Définition de l'ensemble des caractéristiques d'une bordure
<u>BOTTOM</u>	Taille, Pourcentage, auto	Spécification de l'alignement par rapport au bas
<u>CAPTION-SIDE</u>	top, bottom, left, right	Définition d'une légende pour un tableau
<u>CLEAR</u>	none, left right, both	Spécification du comportement d'un élément par rapport à une boîte flottante
<u>CLIP</u>	auto	Définition d'une région de découpage pour l'affichage d'un élément
<u>COLOR</u>	Forme	Spécification d'une couleur de texte
<u>DIRECTION</u>	Couleur ltr, rtl	Spécification de la direction du texte
<u>DISPLAY</u>	none, inline block, list-item run-in, compact marker, table inline-table, table-column-group table-column table-cell table-caption	Définition de l'affichage des éléments
<u>EMPTY-CELLS</u>	show hide	Définition des cellules vides

<u>FLOAT</u>	left right none	Définition du flottement des éléments
<u>FONT</u>	<font-style>, <font-weight> <font-size>/<line-height> <font-family> , caption status-bar	Définition des caractéristiques de la police
<u>FONT-FAMILY</u>	nom_de_la_police nom_famille_police, nom_police_générique	Définition de la famille de la police
<u>FONT-SIZE</u>	xx-small, x-small small, medium large, x-large, x-large Taille, Pourcentage	Définition de la taille de la police
<u>FONT-STYLE</u>	normal italic oblique	Définition du style de la police
<u>FONT-VARIANT</u>	normal italic oblique	Définition de caractères en petites majuscules
<u>FONT-WEIGHT</u>	normal, bold, bolder, lighter 100 à 900	Définition de l'épaisseur des caractères
<u>HEIGHT</u>	auto, Longueur, Pourcentage	Définition de la hauteur d'un élément
<u>LEFT</u>	Taille, Pourcentage, auto	Positionnement d'un élément par rapport à la gauche de la fenêtre
<u>LETTER-SPACING</u>	normal, Taille	Définition de la taille de l'espace entre les lettres
<u>LINE-HEIGHT</u>	normal, Nombre Longueur, Pourcentage	Définition de la hauteur des lignes
<u>LIST-STYLE</u>	list-style-type list-style-position list-style-image	Définition du style de puces d'une liste
<u>LIST-STYLE-IMAGE</u>	Adresse URL none	Définition d'une image pour les puces d'une liste
<u>LIST-STYLE-POSITION</u>	inside outside	Spécification de la position de la liste
<u>LIST-STYLE-TYPE</u>	disc, circle, square decimal lower-roman, upper-roman lower-greek, lower-alpha lower-latin, upper-alpha upper-latin, hebrew, etc... none	Définition du type de puces d'une liste
<u>MARGIN</u>	Taille, Pourcentage, auto	Définition des marges d'un élément
<u>MAX-HEIGHT</u>	auto, Longueur, Pourcentage	Définition d'une hauteur, largeur maximum pour un élément
<u>MAX-WIDTH</u>		
<u>MIN-HEIGHT</u>	Longueur	Définition d'une hauteur, largeur minimum pour un élément
<u>MIN-WIDTH</u>	Pourcentage	
<u>OVERFLOW</u>	auto, visible, hidden, scroll	Définition du comportement d'un contenu en cas de dépassement
<u>PADDING</u>	Taille, Pourcentage	Définition de l'espace de remplissage
<u>PAGE</u>	auto Identificateur	Spécification d'un type particulier de page
<u>PAGE-BREAK-AFTER</u>	auto, always, avoid, left, right	Définition d'un saut de page après (avant)
<u>PAGE-BREAK-INSIDE</u>	auto avoid	Définition d'un saut de page à l'intérieur
<u>PAUSE</u>	Temps Temps Pourcentage Pourcentage	Définition d'une pause
<u>PAUSE-AFTER</u>	Temps Pourcentage	Définition d'une pause après
<u>PAUSE-BEFORE</u>	Temps Pourcentage	Définition d'une pause avant

<u>PLAY-DURING</u>	Adresse URL Adresse_URL mix Adresse_URL repeat auto, none	Lecture d'un fichier audio en fond
<u>POSITION</u>	static, relative absolute, fixed	Définition de la position d'un élément
<u>QUOTES</u>	Chaîne de caractères none	Définition des guillemets
<u>RIGHT</u>	Taille, Pourcentage, auto	Positionnement d'un élément par rapport à la droite de la fenêtre
<u>SIZE</u>	auto, Taille, portrait landscape	Définition de la taille d'un élément
<u>TABLE-LAYOUT</u>	auto, fixed	Définition du type d'affichage d'un tableau
<u>TEXT-ALIGN</u>	left, right, center, justify	Spécification de l'alignement du texte
<u>TEXT-DECORATION</u>	none, underline, overline line-through, blink	Définition du type de décoration du texte
<u>TEXT-INDENT</u>	taille, pourcentage	Définition de l'indentation d'un texte
<u>TEXT-SHADOW</u>	none, {Couleur Taille_à_droite Taille_en_bas rayon, Couleur Taille_à_droite Taille_en_bas rayon}	Définition d'un ombrage du texte
<u>TEXT-TRANSFORM</u>	none, capitalize uppercase, lowercase	Spécification de la casse du texte
<u>TOP</u>	Taille, Pourcentage, auto	Positionnement d'un élément par rapport au haut de la fenêtre
<u>VERTICAL-ALIGN</u>	baseline, sub, super, top, text- top, middle, bottom, text-bottom, Longueur, Pourcentage	Alignement vertical du texte
<u>VISIBILITY</u>	visible, hidden, collapse	Définition de la visibilité d'un élément
<u>VOLUME</u>	Nombre, Pourcentage silent, x-soft soft, medium, loud, x-loud	Définition du volume sonore
<u>WHITE-SPACE</u>	normal, pre, nowrap	Définition d'un espace blanc
<u>WIDOWS</u>	Nombre entier	Spécification du nombre de lignes en haut de la page
<u>WIDTH</u>	auto, Longueur, Pourcentage	Définition de la largeur d'un élément
<u>WORD-SPACING</u>	normal, Taille	Définition de la taille de l'espace entre les mots
<u>Z-INDEX</u>	auto Nombre entier	Définition d'un niveau de l'élément dans la pile d'affichage

Sites intéressants

www.csszengarden.com

www.openweb.org

Document sur les propriétés CSS www.jahpil.com/HEI/3eme/archi/CoursCSS.pdf

<http://colorweb.free.fr/captures/aides/univbret.htm>

III.6 Le système des boîtes

1 Génération de boîtes

La manière dont les boîtes s'agencent pour composer le rendu graphique d'une page Web dépend de

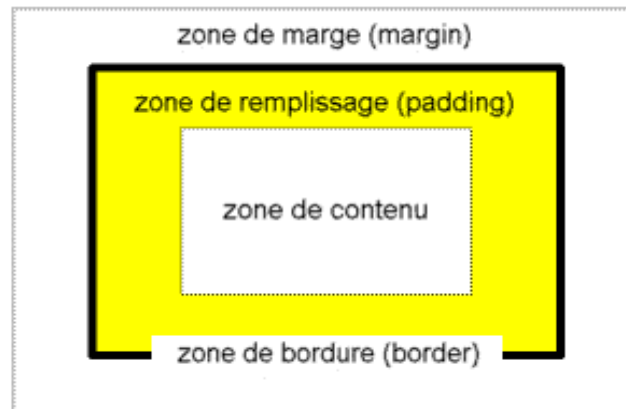
- de données « extérieures » :
 1. l'organisation du document HTML, c'est à dire la succession des éléments générateurs de boîtes ;
 2. des dimensions déterminées : la taille de la zone d'affichage (la fenêtre), la taille des images ...
- des facteurs CSS propres :
 1. le type de boîtes : bloc, en ligne, en enfilade ;

2. le flux dans lequel sont placées les boîtes

2 Les boîtes

Comme on l'a vu, une boîte CSS est constituée :

- d'un contenu (le texte d'un paragraphe par exemple) ;
- d'un rembourrage (*padding*) : l'espace entourant le paragraphe ;
- d'une bordure (*border*) : encadrement du paragraphe ;
- d'une marge (*margin*) : espace entourant le tout.



Il y a plusieurs types de boîtes dont les principaux sont les boîtes en ligne et les boîtes bloc.

* Les principaux éléments créant des **boîtes en ligne** sont : l'élément `span` ; le lien `a` ; l'image `img`.

* Les principaux éléments créant des **boîtes bloc** sont : `div` ; les titres `h1`, `h2`, `h3`, `h4`, `h5`, `h6` ; le paragraphe `p` ; les listes et éléments de liste `ul`, `ol`, `li`, `dl`, `dd` ; le bloc de citation `blockquote` ; le texte pré-formaté `pre` ;

Les propriétés CSS permettent de déterminer :

- ◆ les dimensions de la zone de contenu, son arrière-plan (image, couleur) et sa couleur d'avant-plan ;
- ◆ les dimensions, la couleur et le style des bordures ;
- ◆ les dimensions de la zone de remplissage ;
- ◆ les dimensions de la marge.

*En CSS2, les positions et les tailles d'un grand nombre de boîtes sont calculées en fonction des bords d'une boîte rectangulaire ; on l'appelle un **bloc conteneur**. La racine de l'arbre du document génère une boîte qui fait office de **bloc conteneur initial** pour les constructions subséquentes.*

On peut spécifier la largeur (ou la hauteur) du bloc conteneur initial de l'élément racine avec la propriété `width` (`height`). Quand celle-ci a la valeur "auto", c'est l'agent utilisateur qui fournit cette largeur (hauteur) initiale.

3 Les boîtes en flux normal

Le flux normal, c'est un traitement linéaire (dans l'ordre d'arrivée des balises) du contenu de la page. L'alignement des boîtes bloc est vertical ; l'alignement des éléments en-ligne dans les boîtes bloc est horizontal.

Prenons par exemple deux blocs différenciés par leur couleur décrits en HTML par :

```
<p class="jaune">Une boîte jaune</p>
<p class="verte">Une boîte verte</p>
```

et en CSS par

```
.jaune { background-color: #ffff00; }
.verte { background-color: #00ff00; }
```

Ceci donne :

Une boîte jaune

Une boîte verte

Le navigateur traite successivement les deux éléments rencontrés. Comme il s'agit d'éléments de type bloc, il aligne la marge gauche de chaque élément sur la marge gauche de l'élément conteneur, c'est à dire ici le bloc conteneur initial.

Par défaut, les **boîtes de type en-ligne** sont affichées dans une succession horizontale. Prenons par exemple deux portions de texte différenciées par leur couleur décrites en HTML par :

```
<p>
  <span class="jaune">Une boîte jaune</span>
  <span class="verte">Une boîte verte</span>
</p>
```

Et en CSS par :

```
.jaune { background-color: #ffff00;}
.verte { background-color: #00ff00; }
```

Ceci donne :

Une boîte jauneUne boîte verte

4 La propriété display

Elle sert à afficher ou non une boîte et aussi à transformer une boîte bloc en boîte en ligne et réciproquement. Elle prend principalement les valeurs block, inline, none, list-item, table ...

block force la génération d'une boîte bloc. Ce peut être utile pour transformer une séquence de liens en un menu avec des options encadrées.

inline force la génération d'une boîte en ligne. Ce peut être utile pour afficher les éléments d'une liste en une seule ligne.

none empêche l'affichage d'une boîte

5 Exercices immédiats

Présenter des boîtes en ligne avec marges, padding et bordures. Les valeurs des paramètres sont différentes pour chaque boîte.

Présenter des boîtes bloc avec marges et padding. Leurs textes ont des couleurs différentes

Afficher les articles d'une liste sur une seule ligne et une série de liens dans des boîtes bloc.

A partir du code

```
<h1>Gros titre :</h1>
<p>paragraphe en embuscade ... </p>
```

comment afficher : Gros titre : paragraphe en embuscade ... ?

III.7 Le positionnement en CSS

Objectif : savoir positionner les boîtes engendrées par les éléments de différentes façons dans la zone de visualisation.

1 Les schémas de positionnement

En CSS2, cinq schémas de positionnement peuvent déterminer l'emplacement d'une boîte avec les propriétés de style **position** ou **float** : le flux normal, la position relative, les flottants, la position absolue, la position fixe

Remarques :

Chaque boîte se voit attribuer une position vis-à-vis de son bloc conteneur.

Le bloc conteneur initial ne peut pas être positionné ou être flottant (c.à.d., les agents utilisateurs ignorent les propriétés ['position'](#) et ['float'](#) de l'élément racine).

Les valeurs de la propriété **position** sont :

static (par défaut) : La boîte est placée selon le flux normal (haut/bas, gauche/droite pour nous européens de l'ouest). Les propriétés ['left'](#), ['top'](#), ['right'](#) et ['bottom'](#) ne s'y appliquent pas ;

relative : l'emplacement de la boîte est calculé selon le flux normal. Ensuite la boîte est déplacée relativement à cette position dans le flux normal. Quand une boîte B a une position relative, l'emplacement de la boîte **suivante** est calculé comme si B n'avait pas été déplacée ;

absolute : l'emplacement de la boîte (et éventuellement sa taille) est déterminé par les propriétés [left](#), [right](#), [top](#), [bottom](#). Celles-ci spécifient les déplacements en fonction du bloc conteneur. Les boîtes en position absolue se situent **hors** du flux normal. Elles n'ont ainsi aucune influence sur la mise en forme des autres éléments de même degré de parenté. Bien que les boîtes en position absolue aient également des marges, celles-ci ne fusionnent pas avec d'autres marges.

fixed (encore mal implémenté) : l'emplacement de la boîte est calculé comme pour 'absolute', mais la boîte est en plus fixe par rapport à une référence donnée.

Pour les médias continus, par rapport à la zone de visualisation (la boîte ne bouge pas lors d'un défilement). Pour les médias paginés, par rapport à la page, même si celle-ci apparaît dans une zone de visualisation (par exemple lors d'un aperçu avant impression).

Les valeurs de la propriétés **float** (voir développement plus loin) sont [left](#) et [right](#).

2 Les décalages des boîtes

On dit qu'un élément est *positionné* quand la valeur de sa propriété [position](#) est autre que 'static'. Ces éléments génèrent des boîtes positionnées qui sont disposées selon les quatre propriétés suivantes : **top**, **right**, **bottom** et **left** (décalage en haut, à droite, en bas ou à gauche).

Ces propriétés prennent les valeurs longueur, pourcentage ou auto :

longueur : le décalage est représenté par un nombre fixe à partir du bord de référence ;

pourcentage : le décalage est représenté par un pourcentage de la largeur du bloc conteneur (pour les propriétés [left](#) ou [right](#)) ou la hauteur de celui-ci (pour les propriétés [top](#) ou [bottom](#)). Si la hauteur du bloc conteneur n'est pas spécifiée explicitement (par exemple celle-ci dépendant de la hauteur du contenu), les propriétés 'top' et 'bottom' sont censées avoir la valeur 'auto' ;

auto : l'effet de cette valeur dépend des propriétés associées, lesquelles ont aussi cette même valeur 'auto'

3 Le fonctionnement en position relative

Le positionnement relatif permet d'inscrire un contenu en flux normal, puis de le décaler horizontalement et/ou verticalement. Le contenu suivant n'est pas affecté par ce déplacement, qui peut donc entraîner des chevauchements.

Soit par exemple un positionnement relatif indice qui stipule un décalage vers le haut de 5 pixels et un arrière-plan jaune. En HTML :

```
<p>Lorem
```

```
  <span class="jaune">
```

```
    boîte en position relative
```



```
</span> ipsum dolor
</p>
```

et en CSS :

```
.jaune { position: relative; bottom: 5px; background-color: #ffff00;}
Ce qui donne
```

Lorem **boîte en position relative** ipsum dolor

Pour illustrer le risque de chevauchement, ajoutons un décalage vers la droite. En CSS :

```
.jaune { position: relative; bottom: 5px; left: 3em; background-color: #ffff00;}
Ce qui donne :
```

Lorem **boîte en position relative** ipsum dolor

Autre exemple avec div.

```
<div style="position:absolute; top:100px; width:300px; height:200px ;">
  Texte de référence .....
  <div style="position:relative; top:20px; left:20px; color:red ;">
    Texte décalé deux fois de 20px par rapport au texte de référence
  </div>
</div>
```

Texte de reference

Texte decale deux fois de 20px par rapport au
texte de reference

4 Le fonctionnement "flottant"

Une boîte flottante est **retirée du flux normal**, et placée **le plus à droite** (float: right) ou **le plus à gauche** (float: left) possible dans son conteneur. **Le contenu suivant cette boîte flottante s'écoule le long de celle-ci**, dans l'espace laissé libre. Exemple en HTML :

```
<p class="jaune">Une boîte jaune flottante</p>
<p class="verte">Une boîte verte doté d'un contenu plus long...</p>
```

et en CSS :

```
.jaune { background-color: #ffff00; float: right; width: 100px; margin: 0;}
.verte { background-color: #00ff00;}
Ce qui donne
```

Une boîte verte dotée d'un contenu plus long...
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudinum lectorum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem.

Une boîte
jaune flottante

Le navigateur *place* tout d'abord la boîte jaune, alignée sur le bord droit de cette partie de notre page, puis reprend le cours normal du flux dans l'espace laissé libre à sa gauche pour afficher la boîte verte. Le flux occupe tout l'espace disponible: la boîte verte reprend donc toute la largeur de la page sitôt *passée* la limite inférieure de la boîte flottante jaune.

Une autre propriété clear peut interdire à un contenu d'être adjacent à une boîte flottante. Ses valeurs sont *left*, *right*, *none* (qui n'impose pas de poursuite dessous, réglage normal).

Exemple :

```
<html><head><title>clear</title></head>
<body bgcolor="FFFFFF" text="#000000">
  <div style="float:left; margin-right:20px; margin-bottom:20px; border:solid
```



```

    1px red; text-align:center; font-size:20pt; color:red">
    Bloc<br />    flottant<br />    à gauche <br />
</div>
<p style="font-size:120%">Cette partie de place à droite du bloc flottante</p>
<p style="clear:left">Avec clear:left, le texte se replace en dessous du bloc
flottant déclaré avec float.</p>    <!--attribut style obsolète -->
</body></html>

```

Bloc
flottant
à gauche

Cette partie de place à droite du bloc flottante

Avec clear:left, le texte se replace en dessous du bloc flottant déclaré avec float.

5 Le fonctionnement en absolu

Une boîte en positionnement absolu peut être placée n'importe-où dans le code HTML et s'afficher à l'endroit de votre choix. Ceci s'avère très utile en particulier pour :

- placer les menus de navigation en fin de page, pour améliorer l'accessibilité de votre site en donnant un accès immédiat à son contenu dans les navigateurs textes, tout en les faisant apparaître en haut de page ou encore dans une colonne pour les navigateurs graphiques ;
- créer plusieurs colonnes au positionnement indépendant de l'ordre dans lequel elles se trouvent en HTML.

Le positionnement absolu « retire » totalement du flux le contenu concerné : sa position est déterminée par référence aux limites du conteneur. Celui-ci peut-être :

- une boîte **elle-même positionnée** (position relative ou absolue) ;
- le bloc conteneur initial, à défaut de boîte positionnée, c'est à dire en pratique le plus souvent la fenêtre du navigateur.

Définissons un conteneur verte en position relative et un positionnement absolu jaune :

```

.verte { position: relative; background-color: #00ff00; width: 15em;}
.jaune { position: absolute; top: 1em; right: 1em; background-color: #ffff00;}

```

Et appliquons ces styles :

```

<div class="verte">
  <p>
    ...
  </p>
  <p class="jaune">
    Boîte jaune en position absolue
  </p>
</div>

```

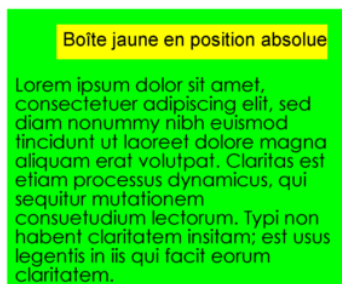
Ce qui donne

Lorem ipsum dolor sit amet,
con Boîte jaune en position absolue
diar
tincidunt ut laoreet dolore magna
aliquam erat volutpat. Claritas est
etiam processus dynamicus, qui
sequitur mutationem
consuetudium lectorum. Typi non
habent claritatem insitam; est usus
legentis in iis qui facit eorum
claritatem.

Ménager un espace pour la boîte en position absolue

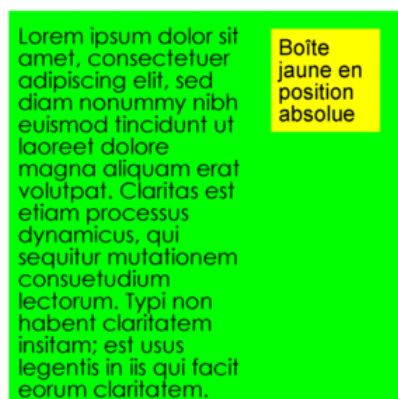
Inévitablement, la boîte en position absolue recouvre le contenu de notre paragraphe. Pour l'éviter, dotons la boîte conteneur verte d'un remplissage supérieur suffisant :

`.verte { padding-top: 5em;}`



Nous pourrions aussi bien fixer la largeur de la boîte jaune et doter le texte de la boîte verte d'un remplissage à droite : `.verte { padding-right: 7em;}.jaune { width: 4em;}`

Ce qui donne



Utilisation : une mise en page à trois colonnes

La position absolue offre donc une alternative aux flottants pour réaliser des mises en pages à plusieurs colonnes. Le cas typique est celui des trois colonnes dont voici le code :

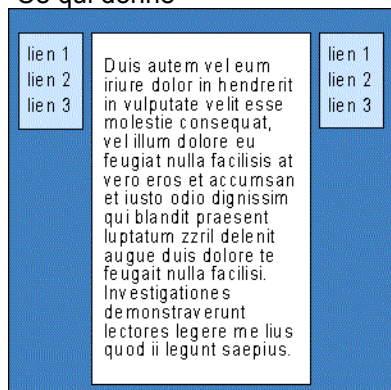
En CSS :

```
.centre { margin: 1em 25%;}
.gauche {position: absolute; top: 1em; left: 1%; width: 18%;}
.droite {position: absolute; top: 1em; right: 1%; width: 18%;}
```

En HTML :

```
<body>
  <div class="centre">    ... </div>  <div class="gauche">    ... </div>  <div class="droite">    ... </div>
</body>
```

Ce qui donne



6 Le positionnement fixe *Éventuellement en seconde lecture*

Comme dans un positionnement absolu, le contenu concerné est retiré totalement du flux. Mais il est cette fois **positionné uniquement par rapport aux limites de la zone de visualisation**, autrement dit la fenêtre du navigateur. Le défilement de la page n'a aucun effet sur un contenu en position fixe.

Remarque : le positionnement fixe n'étant pas reconnu par Internet Explorer 5 et 6, un positionnement absolu doit s'y substituer dans ce navigateur. Le code est :

```
.fixe { position: absolute; top: 10px; left: 10px;}
```

```
html>body .fixe { position: fixed;}
```

Le navigateur MSIE ne lira que la première définition de .fixe et ignorera la seconde car il ne comprend pas la syntaxe html>body. Les navigateurs supportant la position fixe liront les deux positionnements, mais le second se substituera au premier car il est placé après celui-ci dans la feuille de style.

Tout comme le positionnement absolu, le positionnement fixe est susceptible de provoquer des chevauchements. On emploiera donc des méthodes similaires pour l'exploiter.

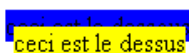
7 Autres propriétés (superposition, visibilité, dépassement)

Une propriété permet de rendre visible ou invisible une boîte : respectivement **visibility** et **hidden**). Cette propriété est proche de display, mais contrairement à display :none, visibility :hidden met de côté la zone de contenu : elle ne l'ignore pas complètement.

Une autre, **z-index**, permet de déterminer l'ordre d'empilement des éléments : 1, 2, 3 Le plus haut (donc visible) est celui d'index le plus élevé.

Exemple :

```
<html> <head>
  <Style type="text/css">
    .dessous {position: absolute; top: 100px ; left:20px; z-index:1 ;
              background-color: blue}
    .dessus {position: absolute; top: 115px ; left:30px; z-index:2 ;
            background-color: yellow}
  </style>   </head>
<body>
  <div class="dessous"> ceci est le dessous </div>
  <div class="dessus"> ceci est le dessus </div>
</body> </html>
```



Overflow contrôle la hauteur et la largeur d'un bloc avec les valeurs scroll, auto, visible ou hidden : elle définit ce qui se produit lorsque le contenu d'un champ dépasse les bords.

III.8 Les différentes sorties média (*éventuellement en seconde lecture*)

Objectif : savoir fournir une couche de présentation associant du contenu balisé à un média spécifique.

1 Les différents média

Une des propriétés les plus importantes des feuilles de styles est de spécifier comment un contenu est représenté sur divers média (papier, écran, lecteur braille, ...). Ceux actuellement reconnus sont :

All	tous types de média
Aural	synthétiseurs vocaux

Braille	périphériques tactiles	et Embossed	imprimante Braille
Handheld	ordinateurs de poche		
Print	imprimantes		
Projection	vidéoprojecteur		
Screen	écrans couleur		
Tty	certaines dispositifs tels que télex		
Tv	téléviseurs		

Par exemple, à l'aide d'une feuille de style aural, on peut « lire » une page web de vive voix à partir de son téléphone portable.

Remarques :

- un médium CSS est un ensemble de propriétés. Certaines de ces dernières peuvent être partagées en plusieurs média sans prendre les mêmes valeurs (exemple de font-size)
- il y a deux façons de d'indiquer les médias dans les feuilles de styles. Elles sont indiquées dans ce qui suit immédiatement.

2 Médiatisation à l'aide de l'attribut media dans une balise link

Une feuille de style va être appelée par une balise link qui va déjà préciser le médium visé grâce à la valeur de l'attribut media. . En voici un exemple :

```
<html>  <head>    <title>Un lien vers les média cibles</title>
  <link rel="stylesheet" type="text/css"  media="print" href="impression.css" />
  <link rel="stylesheet" type="text/css"  media="screen" href="ecran.css" />
  <link rel="stylesheet" type="text/css"  media="aural" href="son.css" />
</head>
<body>
  <p>Le corps du document sera affiché, imprimé ou entendu comme indiqué
    dans la feuille de style respective.</p />
</body> </html>
```

L'intérêt d'une feuille de style pour l'impression est d'avoir une page imprimée sémantiquement claire et graphiquement sobre (on peut vérifier le résultat avec les options du menu Fichier : imprimer ou aperçu).

Remarque : si on déclare des feuilles de styles "alternate" avec un attribut title, elles peuvent être identifiées par le navigateur (Mozilla ou Firebird) et même appelées (respectivement option du menu Affichage / Feuille de style ou icône dans la barre d'état).

3 Médiatisation avec une autre syntaxe CSS (*en seconde lecture*)

La directive **@media** permet de cibler différents média dans une feuille de style ou au sein d'une balise de style. Voici un exemple dans ce dernier cas :

```
<html>  <head>    <title>Un lien vers les média cibles</title>
  <style type="text/css" >
    @media print {margin-left:10px; background-color:white;color:black}
  </style></head>
<body>
  <p>L'impression se fera en noir sur blanc <p />
  dans la feuille de style respective.
</body> </html>
```

Avec la directive **@media**, on peut non seulement définir le média selon le type de périphérique, mais aussi selon les caractéristiques de rendu : continuous (en flux comme une page web ou des paroles) ou paged (en pages distinctes comme pour l'impression), visual (pour tout ce qui est visualisable).

Remarque : on peut préciser l'impression avec une autre directive @page qui permet de définir par exemple la taille des feuilles de papier, la première page ...

@page {size :21cm 29.7cm ;margin :2.5cm}

Autres exemples :

La directive `@import` permet de gérer l'importation des feuilles de style. Elle s'inscrit soit dans une balise de style, soit dans une feuille externe comme suit : `@import url (direct.css)`.

On peut utiliser deux éléments stylisés différemment suivant le media de sortie :

`@media screen { H1#first { position: fixed } }` ou `@media print { H1#first { position: static } }`

4 Exercices immédiats

Quelle est la syntaxe HTML correcte qui désigne l'importation d'un fichier CSS?

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

Attacher deux feuilles de style à une même page et changer les avec le navigateur.

Ecrire une feuille de style pour imprimer économiquement une page web (suppression des images et des couleurs).

III.8 Problèmes / Ateliers sur les feuilles de styles

1 Enoncés

Donner le code pour afficher la figure suivante dans tout une fenêtre de navigateur

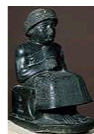


Ecrire le code HTML correspondant à la page Web "La Mésopotamie au Musée du Louvre" :

MUSEE DU LOUVRE Arts de le Mésopotamie

Le département des Antiquités orientales conserve des oeuvres provenant d'un immense territoire qui s'étendait de l'Indus à la mer Méditerranée, et où, depuis le néolithique aux environs de 9500, se sont succédées de nombreuses civilisations et cultures dont les plus anciennes remontent à 6 000 ans avant Jésus-Christ.

C'est dans cette région que nous voyons apparaître pour la première fois une administration politique, militaire et religieuse gérant ces structures complexes que sont les villes. C'est en Mésopotamie, plus précisément à Uruk, vers 3300 avant J.-C., que l'écriture fit son apparition pour la première fois au monde, écriture pictographique dans un premier temps, "dessinant" les éléments du monde réel. L'écriture cunéiforme fera son apparition au début du 3e millénaire.



Statue assise de Gudea,
prince de Lagash Tello,
ancienne Girsu Epoque néo-sumérienne
(vers 2125-2110 avant J.-C.)

Les collections sont réparties selon trois grands ensembles géographiques et culturels : - La Mésopotamie (notamment avec les civilisations du pays de Sumer, de Babylone, d'Assur et d'Anatolie). - L'Iran (Suse, le plateau iranien et les confins orientaux de l'Iran). - Les pays du Levant (la côte syro-palestinienne et aussi Chypre).

Ecrire à l'aide de styles le code HTML d'une page Web de texte dont les paragraphes débutent par une lettrine comme suit

MESOPOTAMIE Le département des Antiquités orientales conserve des œuvres provenant d'un immense territoire qui s'étendait de l'Indus à la mer Méditerranée, et où, depuis le néolithique aux environs de 9500, se sont succédées de nombreuses civilisations et cultures dont les plus anciennes remontent à 6 000 ans avant Jésus-Christ. C'est dans cette région que nous voyons apparaître pour la première fois une administration politique, militaire et religieuse gérant ces structures complexes que sont les villes. C'est en Mésopotamie, plus précisément à Uruk, vers 3300 avant J.-C., que l'écriture fit son apparition pour la première fois au monde, écriture pictographique dans un premier temps, "dessinant" les éléments du monde réel. L'écriture cunéiforme fera son apparition au début du 3e millénaire.

Créer un effet de filigrane avec une image .gif

2 Mini-projet

Représentez la page suivante (orangée et grise) uniquement à l'aide de CSS.

Éléments fournis :

Les couleurs :

- Orange clair : #FFA500
- Orange foncé : #D18800
- Orange du formulaire : #FFE3AF
- Gris : #DEDEDE

Largeur de la colonne de gauche : 200

Largeur de l'ensemble de la page : 750

Les images de la page :

Le logo :

Le bandeau de pub :

Les produits :

Les fonts :

A vous de choisir la taille des polices.

Toutes les polices sont en Arial et doivent être définies dans une feuille de style CSS.



Trouver une astuce pour centrer ses pages web et atténuer le rétrécissement de la fenêtre

IV Plus avant avec HTML

IV.1 Les constutants d'un tableau

1 Les éléments du tableau

La notion de tableau rassemble trois conteneurs : le tableau, la ligne et la cellule. Conceptuellement, un tableau se construit comme **empilement de lignes**, ces dernières divisées en cellules (un même nombre sur toutes les lignes du tableaux avant une éventuelle fusion de cellules).

C11	C12	C13	C14
D11	D12	D13	D14

Tableau de deux lignes comprenant chacune 4 cellules

L'élément HTML **<table>** contient des éléments HTML **<tr>** (table row) marquant les lignes qui à leur tour contiennent des éléments **<td>** (table data) marquant les cellules. Une autre balise **<caption>** (placée juste après **<table>**) ajoute un titre (par défaut en bas du tableau).

Pour obtenir le tableau ci-dessus, on doit écrire le code suivant :

```
<center>      <!-- ou mieux <div align="center"> -->
<table border="2">
  <caption> Tableau de deux lignes comprenant chacune 4 cellules </caption>
  <tr >
    <td>C11</td> <td>C12</td> <td>C13</td> <td>C14</td>
  </tr>
  <tr>
    <td>D11</td> <td>D12</td> <td>D13</td> <td>D14</td>
  </tr>
</table>
</center>
```

Remarques :

- la balise **<td>** peut être remplacée par **<th>** (table heading) pour les en-têtes de colonne (première ligne uniquement). Dans ce cas, le contenu de la cellule est centré et en gras.
- Attention : **laisser une cellule vide** dans un tableau peut entraîner un problème d'affichage. Y mettre par exemple un espace insécable ` `.

2 Les attributs des différents éléments d'un tableau

Les attributs d'un tableau permettent d'en préciser les propriétés. Ils s'appliquent aux trois conteneurs que sont la tableau, la ligne et la cellule. Ce sont :

align	aligne horizontalement le contenu d'un conteneur (valeurs : left, center, right)
valign	aligne verticalement le contenu d'un conteneur (valeurs : top, middle, bottom)
background	spécifie l'image de fond du tableau
bgcolor	spécifie la couleur de fond du conteneur
height	règle la hauteur du tableau ou de la ligne en pixels ou pourcentage de la taille de la fenêtre
width	règle la largeur du tableau en pixels ou pourcentage de la taille de la fenêtre

Exemple de code (partiel)

```
<table border="2" align="center" cellpadding="20" cellspacing="10" width="400">
  <caption align="top">Titre du tableau</caption>
  <tr >
    <th>Titre de la première colonne</th>
    <th>Titre de la deuxième colonne</th>
    <th>3ème colonne</th>
  </tr>
  <tr>
```

```

        <td>Contenu 1</td>
        <td>Contenu 2</td>
        <td>Contenu 3</td>
    </tr>
</table>

```

donne

Titre du tableau

Titre de la première colonne	Titre de la deuxième colonne	3ème colonne
Contenu 1	Contenu 2	Contenu 3

Remarques :

- si l'on ne précise pas de largeur, l'ajustement est automatique (ci-dessus la troisième colonne est moins large).
- quand un même attribut voit plusieurs valeurs rentrer en concurrence (pour un fond de couleur par exemple), la valeur qui l'emporte est celle dont l'attribut concerne le conteneur le plus petit.

3 Marges et bordures

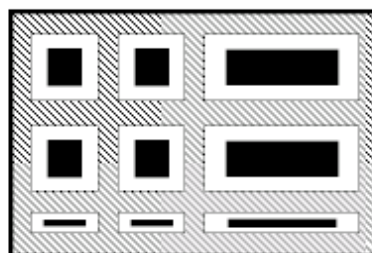
Deux attributs précisent les marges intérieures et extérieures aux cellules. Un autre permet de tracer la bordure autour d'un conteneur (table, ligne, cellule).


cellspacing insère un espace entre les cellules du tableau évalué en pixels

cellpadding insère un "rembourrage" à l'intérieur des cellules

border délimite le tableau (épaisseur du bord en pixels)

Table border 



Cellspacing 

Cellpadding 

Cell content 

4 La fusion des cellules

Les cellules peuvent fusionner, c'est-à-dire s'étendre sur plusieurs "cases" du tableau.

Deux attributs sont utilisés pour spécifier "l'envergure" des cellules :

colspan spécifie le nombre de colonnes sur lesquelles s'étend la cellule

rowspan spécifie le nombre de lignes sur lesquelles s'étend la cellule


```

<tr>
  <td>Row 1 col 1</td>
  <td align=center colspan=2>
    Row 1 col 2-3</td>
</tr>

```

Row 1 col1	Row 1 col2-3	
Row 2 col1	Row 2 col2	Row 2 col3
Row 3 col1	Row 3 col2	Row 3 col3

```

<tr>
  <td>Row 2 col 1</td>
  <td valign=top rowspan=2>
    Row 2 col 2</td>
  <td>Row 2 col 3</td>
</tr>

```

Row 1 col1	Row 1 col2-3	
Row 2 col1	Row 2 col2	Row 2 col3
Row 3 col1		Row 3 col3

Voici un exemple de tableau :

Tête de col 1	Tête de col 2	Tête de col 3
Deux cellules fusionnées verticalement	Deux cellules fusionnées horizontalement	
	Cellule en couleur	Fin

Exemple de tableau à 3 ligne et 3 colonnes



son code est :

```

<html>  <head>  <title>Tableau</title>  </head>
<body>
  Voici un exemple de tableau :  <p />
  <table border="3" cellspacing="3" cellpadding="4" width="100%">
    <caption align="bottom">
      Exemple de tableau à 3 lignes et 3 colonnes
    </caption>
    <tr>
      <th>Tête de col 1</th>
      <th>Tête de col 2</th>
      <th>Tête de col 3</th>
    </tr>
    <tr>
      <td rowspan="2" align="center">Deux cellules fusionnées verticalement</td>
      <td colspan="2" align="center">Deux cellules fusionnées horizontalement</td>
    </tr>
    <tr align="center">
      <td bgcolor="#FAE1A1">Cellule en couleur</td>  <td>Fin</td>
    </tr>
  </table>  </body>  </html>

```

Remarque : nous nous sommes contentés ici de mettre du texte dans les cellules des tableaux. Mais on peut y insérer aussi des images, des liens, des listes

		A droite, hippopotame du moyen empire
		A gauche, chapelle de Sakkara 5ème dynastie

Quelques tableaux du Musée du Louvre

5 Positionnement de tableaux au sein d'une fenêtre

Les attributs width et length prennent des valeurs absolues ou relatives suivant le degré de contrôle souhaité. Si la valeur de la largeur est relative (exprimée en pourcentage de la taille de la fenêtre), le tableau va s'adapter à la largeur de la fenêtre. Exemple : si l'on choisit width="100%", le tableau va s'étaler sur toute la largeur de la fenêtre. Si la valeur de la largeur est fixe (exprimée en pixels), le tableau est mieux contrôlé mais il peut déborder de la fenêtre.

Remarques :

- si des valeurs d'attributs *border*, *cellspacing* ne sont pas précisées, des espacements par défaut sont intégrés au tableau.
- pour des raisons de rapidité d'affichage et de gestion du code, il est préférable d'empiler deux tableaux que d'en faire un très grand. Dans ce cas, il faut optimiser les valeurs de l'attribut *height*.
- on peut aussi imbriquer deux tableaux, c'est-à-dire insérer un deuxième tableau dans une cellule du premier (code lourd).
- centrer un tableau fixe le rend moins sensible aux changements de résolution.

Bannière publicitaire			
Bouton1	Bouton 2	Bouton 3	Bouton 4

6 Exercices immédiats

Est-il possible de définir la couleur de fond d'une ligne de tableau? a) Oui b) Non

Soit le code partiel

```
<table bgcolor="blue"> <tr bgcolor="red"> <td bgcolor="yellow"> 1 </td><td> 2 </td></tr></table>
```

De quelle couleur sera la cellule qui contient '1' ? a) Bleu b) Rouge c) Jaune d) Vert

Peut-on :

- a) mettre une liste dans un tableau uniquement
- b) mettre un tableau dans une liste uniquement
- c) mettre l'un dans l'autre dans les deux cas

Quel sous-élément de tableau présente directement le contenu en gras et centré ?

Quel attribut permet le "rembourrage" des cellules ?

Quel valeur prend l'attribut *colspan* pour que le code suivant soit correct :

```
<tr> <td>C1</td> <td>C2</td> <td>C3</td> </tr>
<tr> <td>C1</td> <td colspan=" ?? ">C2</td> </tr>
```

Quel attribut s'utilise avec la balise `<caption>`

Quel est l'avantage de l'empilement des tableaux plutôt que l'usage d'un seul ?

Quel attribut permet d'aligner un contenu sur le haut d'une cellule ?


IV.2 La construction d'un tableau

La balise `table` possède des attributs qui permettent au tableau de s'étendre sur toute la fenêtre du navigateur : il n'a alors qu'une seule cellule qui peut être centrée horizontalement et verticalement. Il peut servir à modéliser, compartimenter la page (on le voit bien quand on scrute la source de certaines pages web, celles de journaux en particulier). Cette approche est obsolète.

Construisons pas à pas un tableau

1 Partons d'un modèle couché sur papier

Voici le modèle à optimiser pour un écran de résolution 800x 600 :

Bannière publicitaire		
Lien de navigation		Liens publicitaires
	Texte 1	Texte2

On peut le concevoir comme un tableau unique de trois lignes et quatre colonnes :
une bannière ligne horizontale en haut, deux colonnes latérales et une partie centrale divisée en deux.
Soit, structuré grossièrement :

L1C1	L1C2	L1C3	L1C4
L2C1	L2C2	L2C3	L2C4
L3C1	L3C2	L3C3	L3C4

de code

```
<html> <head> <title>Tableau élémentaire de 3 lignes sur 4 colonnes</title> </head>
<body> <table border>
<tr><td>L1C1</td><td>L1C2</td><td>L1C3</td><td>L1C4</td></tr>
<tr><td>L2C1</td><td>L2C2</td><td>L2C3</td><td>L2C4</td></tr>
<tr><td>L3C1</td><td>L3C2</td><td>L3C3</td><td>L3C4</td></tr>
</table>
</body> </html>
```

On peut élargir le tableau à tout l'écran avec une largeur fixe width="750".

Remarque : Compte tenu de l'affichage écran le plus répandu, l'optimisation se fait pour un écran 800 x 600 : on se donne d'abord un premier tableau de dimensions 750 x 300 (limites extérieures) et y insère un autre tableau de largeur et hauteur 100%).

2 Extension des lignes et des colonnes

On procède alors à l'extension :

- de L1C1 sur 4 colonnes
- de L2C2 sur deux colonnes
- de L2C1 et L2C4 sur deux lignes

On arrive à la figure suivante

3 Affinement et tests

A ce niveau, on décide de la largeur des colonnes et de l'alignement vertical des cellules `valign="top">`. On peut ajouter une ligne de contrôle pour définir la largeur des colonnes

```
<tr height="0"> <td width="25%>&nbsp;</td> <td width="25%>&nbsp;</td> <td width="25%>&nbsp;</td> <td width="25%>&nbsp;</td> </tr>
```

<tr

Ensuite on place du contenu dans le tableau pour tester l'ensemble sur les principaux navigateurs.

4 Un contenu toujours centré qui ne rétrécit pas

On peut utiliser une table pour centrer un contenu quelque soit le navigateur :

```
<html>  <head>    <title>Always Centered</title>    </head>
<body bgcolor="#000000">
<table border="0" width="100%" height="100%">
  <tr>
    <td align=center valign=middle>
      
      <p> <font color="#FFFFFF"> Aucune importance, c'est stretch
                                         et toujours centré ! </font>
    </td>
  </tr>
</table>  </body>    </html>
```

Quand la fenêtre rétrécit, tant qu'elle le peut, l'image reste centrée. Quand la fenêtre a trop rétréci, l'image qui a des dimensions fixes va "sortir" de la fenêtre et faire apparaître des barres de scrolling.

Jusqu'ici, les pages HTML sont des documents passifs, à consulter. L'utilisateur parcourt des pages, mais ne peut pas écrire, fournir des informations et les transmettre. Il ne peut que faire des choix de "navigation" proposés sous forme d'hyperliens. Les formulaires offrent des interfaces de saisie de données.

IV.3 Les principes du formulaire

1 Un nouveau type de page

Les formulaires HTML (forms en anglais) proposent des pages ressemblant au contenu de l'écran d'un client de base de données. Pour entrer des informations, le formulaire va utiliser un nouvel ensemble de composants, appelés aussi champs ou **contrôles** qui permettent à l'utilisateur d'exprimer ses choix, de **saisir des informations** sous plusieurs formes :

- champ de saisie de texte court (quelques caractères) ou spécifiques (mot de passe),
- cases à cocher ou boutons radio (choix exclusif)
- liste déroulante ou non
- zone de saisie pour texte long
- bouton d'exécution

Nom Prénom

Mot de passe e-mail

Vous souhaitez commander une laisse pour :

☒ chien
☐ chat
☐ crocodile

Paiement par :

☒ carte bleue
☐ carte American Express
☐ chèque à la livraison

Livraison

Adresse de livraison :

Bien sûr les réactions à apporter à l'entrée de ces différents types d'informations doivent être prévues à l'avance, donc programmées. Ce sera l'apport des scripts sur le serveur.

2 Le traitement des informations saisies par formulaire

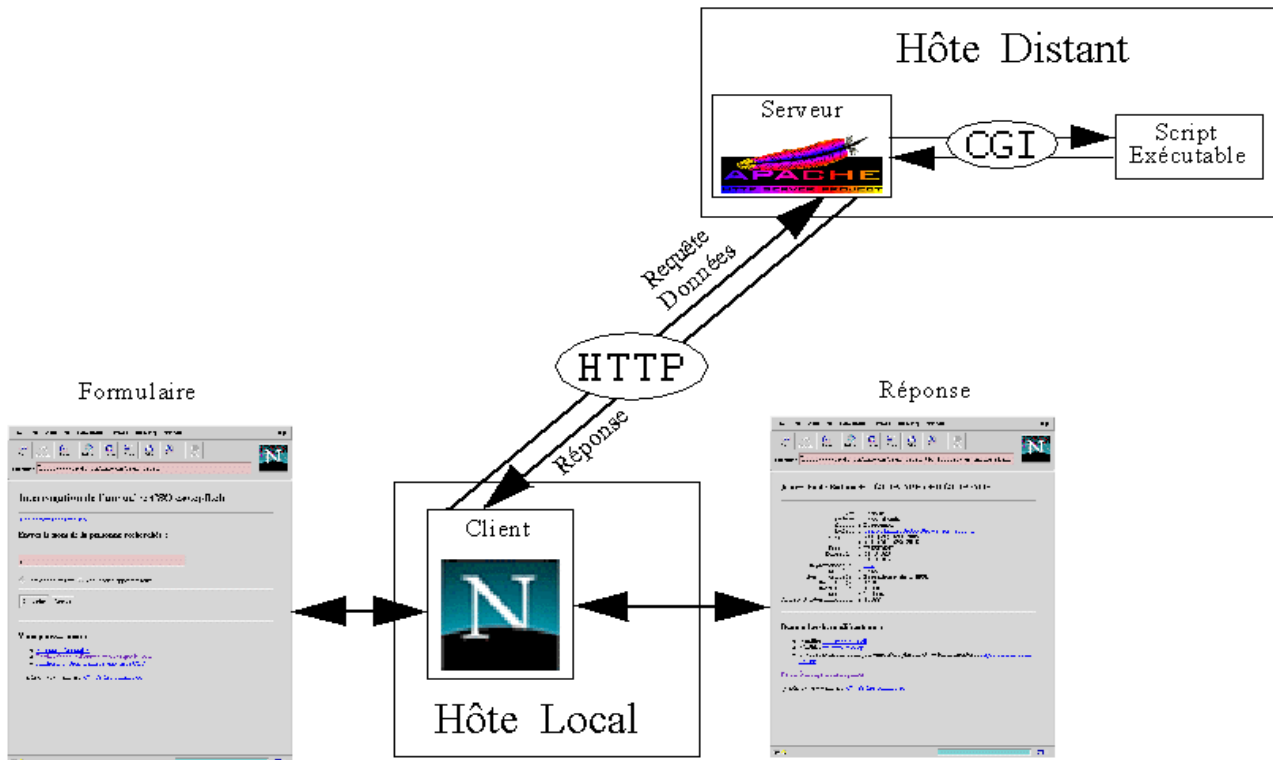
Deux types de programmes (on parle le plus souvent de script) peuvent traiter les informations d'un formulaire : ceux téléchargés avec la page du formulaire (scripting client avec Javascript) et ceux placés sur le serveur (scripting serveur avec PHP). Les deux types de programmes sont complémentaires et vont permettre de réaliser des applications client-serveur avec de "l'intelligence".

Les scripts Javascript sont inscrits dans la page HTML et rendent cette dernière interactive. Il peut s'agir d'un refus d'aller plus loin au sortir d'un champ mal saisi ou encore d'un non envoi du formulaire après vérification globale (champ obligatoire non rempli). Ces scripts ont l'avantage de limiter les accès au serveur en favorisant des traitements locaux.

Les seconds sont des programmes stockés sur un serveur dans un répertoire indiqué lors de la configuration du serveur web. Ils sont apparentés aux scripts **CGI** (Common Gateway Interface). Pourquoi cette notion d'interface ? Parce que le programmeur devra écrire un script permettant d'établir le dialogue entre le serveur recevant une requête http du navigateur (la soumission du formulaire) et une application spécifique installée sur la machine serveur, un moteur de base de données par exemple.

Les programmes CGI peuvent être écrits en Perl, C, Fortran. Par la suite, dans notre approche client-serveur, nous utiliserons le couple PHP – MySQL sur un serveur Apache.

Remarque : il est logique que les pages Web stockées sur le serveur ne puissent pas être modifiées par le client. L'idée de ce type de programmation est de construire un document HTML au moment où l'on clique sur le lien pour le charger. Le client indique un nom de fichier à l'aide d'une url, non pour recevoir une nouvelle page Web, mais pour demander son exécution (on parle alors de lien exécutable) sur le serveur. Ce dernier exécute le programme indiqué et renvoie au client la sortie standard de ce programme sous forme de document HTML.



3 Les données transmises au serveur

Les données saisies dans le formulaire peuvent être transmises au serveur si elles sont accouplées (successful), c'est-à-dire si un nom de contrôle est associé à une valeur (un bouton radio non sélectionné n'est pas transmis).

Lors de la soumission, on a successivement les étapes :

- identification des contrôles "successful"
- formation d'une chaîne de caractères contenant l'ensemble des couples (nom du champ, valeur saisie). Chaque couple est séparé du suivant par un & (et commercial), la séparation entre le champ et sa valeur se fait à l'aide du signe égal :
nom=Dupond&Prenom=Pierre
- encodage de la chaîne : les espaces y sont remplacés par des + et les caractères spéciaux par une valeur hexadécimale de l'encodage (%xx).

Exemple : recherche à partir de la barre Google sur le mot formulaire : <http://www.google.fr/search?sourceid=navclient&hl=fr&ie=UTF-8&q=formulaire>

IV.4 Le balisage des formulaires

L'objectif ici est de créer des formulaires en y insérant différents types de champs et de savoir les envoyer uniquement au serveur pour traitement (qu'on n'étudie pas dans ce cours).

1 Les attributs de l'élément <form>

L'élément <form> est un conteneur pour les contrôles. Il possède cinq attributs concernant le dialogue : qui et avec qui, sous quelle forme (contenu et transport), où insérer la réponse ?

L'attribut **name** identifie le formulaire (s'il y en a plusieurs, par exemple, non imbriqués)

L'attribut **action** adresse le formulaire, s'il y a lieu, vers le programme destinataire sur le serveur qui va l'exploiter. Il précise l'action à entreprendre, le script à exécuter

Une fois construite, la chaîne de données est envoyée au serveur selon la valeur donnée à l'attribut **method**. Il y a deux méthodes de transport : **post** et **get**.

- La méthode **get** ajoute directement à la chaîne contenant l'ensemble des couples l'url référençant le script à exécuter (séparation par le caractère ?). Attention comme la plupart des clients web affichent l'url courante, cette chaîne sera donc visible dans la barre d'adresse du navigateur.
- La méthode **post** est la plus utilisée : la chaîne de caractères n'est pas accolée à l'url mais envoyée par une séquence http spéciale.

L'attribut **target** indique le nom de la page (la cible du boomerang) dans laquelle va venir se nicher la page réponse. Cette nouvelle page peut venir remplacer la page qui contenait le formulaire ou encore s'inscrire dans une nouvelle fenêtre.

Exemples :

```
<form name="PremierForm" method="get" action="/cgi-bin/mailer">
<form action="mailto:webmaster@allhtml.com" method="post">
```

Un élément form d'un document contient de l'information (contenu), du balisage et des composants spéciaux, les contrôles : cases à cocher, boutons radio, menus. L'utilisateur manipule ces contrôles avant d'envoyer (soumettre) le formulaire à un programme qui va le traiter.

2 Les éléments composants introduits par la balise input

Ce sont les principaux réceptacles de données. La balise vide **<input>** qui les introduit est identifiée par son attribut **name** et caractérisée par son attribut **type**. Si l'on veut entrer :

- un champ de saisie de texte, l'attribut **type** prend la valeur **text**
- un champ de saisie de texte caché, l'attribut **type** prend la valeur **password**
- un bouton de soumission, l'attribut **type** prend la valeur **submit**
- une case à cocher, l'attribut **type** prend la valeur **checkbox**
- des boutons radio, l'attribut **type** prend la valeur **radio**
- un simple bouton, l'attribut **type** prend la valeur **button**

La balise ainsi « *typée* » prend aussi les attributs suivants :

- **size** qui définit, s'il y a lieu, la taille visible du champ (en nombre de caractères).
- **maxlength** qui définit, s'il y a lieu, la taille maximale de caractères autorisés
- **value** qui pré-inscrit un texte dans le champ d'entrée (valeur par défaut)

Exemple :

Un champs texte nommé "text1" large de 30 caractères.

```
<input type="text" name="text1" size="30">
```

Un champs texte nommé "text2" large de 30 caractères mais n'acceptant que 20 caratères.

```
<input type="text" name="text2" size="30" maxlength="20">
```

Un champs texte nommé "text3" large de 40 caractères avec une valeur par défaut.

```
<input type="text" name="text3" size="40" value="Nous vaincrons les formulaires">
```

La balise `<input type="password">` est utilisée pour saisir une information confidentielle. Dans cette zone, les caractères tapés sont remplacés par des étoiles (même utilisation que les champs texte).

La balise `<input type="submit">` définit un bouton qui déclenche l'envoi de tous les champs avec leurs valeurs vers l'url destinataire. Il peut y avoir plusieurs boutons submit dans un même formulaire pour effectuer des opérations différentes (bancaires par exemple : pour consulter son compte, faire un virement, commander un chéquier). La balise prend les attributs `name` (qui nomme et identifie le bouton s'il n'est pas seul) et `value` (qui spécifie le texte **inscrit** sur le bouton).

Exemple :

```
<body>
L'accès de ce site est réservé aux acteurs du projet.<br>
Veuillez entrer le mot de passe.<br>
<table>
  <form action="" method="post">
    <tr>
      <td align="center"> <b>Mot de passe :&nbsp;  </b> </td>
      <td> <input type="password" name="identité" value="">
      <br /> </td>
    </tr>
    <tr>
      <td colspan="2" align="center"> <input type="submit"
        value="entrer" name="VALIDATION"> </td>
    </tr>
  </form>
</table>
</body>
```

L'accès de ce site est réservé aux acteurs du projet.
Veuillez entrer le mot de passe.

Mot de passe :

La balise `<input type="reset">` permet une remise à zéro (effacement des données en cours de saisie). Ce sont les mêmes attributs que pour le bouton submit.

Exemple : `<input type="reset" name="initialisation" value="Reset" />`

La balise `<input type="button">` crée d'autres boutons non préprogrammés comme submit ou reset. Ils n'ont de sens qu'associés à un programme (une fonction Javascript par exemple).

Exemple : `<input type="button" value="Lancer" name="lancement" onClick="code" />`

La balise `<input type="checkbox">` définit une case à cocher. Elle possède les trois attributs `name` (qui nomme la case à cocher), `value` (qui spécifie la valeur **envoyée**) et `checked` (qui permet de positionner par défaut la case en mode coché).

Exemple : `<input type="checkbox" name="micro" value="PC" />`

☒ Macintosh

☒ PC

☐ Station Unix

La balise `<input type="radio">` définit un **ensemble** de boutons radio pour ne définir qu'un choix à la fois. Ce sont les mêmes attributs que pour checkbox.

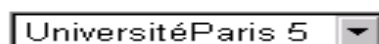
list (1 ou rien) et **multiple** (qui autorise, dans une scrolled-list, la sélection simultanée de plusieurs articles).

Les attributs de la balise `<option>` sont **value** (qui définit la **valeur passée au script**) et **selected** (qui définit une option sélectionnée par défaut).

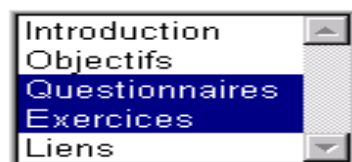
Exemple :

```
<html>   <head>           <title>Listes d'entrée</title>   </head>
<body>   <form name="essai" method="post" >   <h2>Pop-list</h2>
  <select name="sites">
    <option value="www.univ-paris5.fr">Paris 5</option>
    <option value="www.ratp.fr">RATP</option>
    <option value="ww.thot.cursus.edu">THOT</option>
  </select> <p /> <hr />
  <h2>Scrolled-list</h2>
  <select name="menu" size="5" multiple="multiple">
    <option value="intro">Introduction</option>
    <option value="FAQ">Objectifs</option>
    <option value="QCM">Questionnaires</option>
    <option value="Exo">Exercices</option>
    <option value="Liens">Liens</option>
  </select> <p /> <hr />
</form>   </body>   </html>
```

Pop-list



Scolled-list



Remarque : la plupart des éléments d'un formulaire se contente d'une hiérarchie à deux niveaux, sauf Select.

La balise `textarea` est utilisée pour réaliser une boîte de texte multiligne, un commentaire par exemple. Ses attributs sont **name** (qui définit le nom de boîte de données), **rows** (qui définit le nombre de lignes dans la boîte de saisie), **cols** (qui définit le nombre de colonnes (en caractères) dans la boîte de saisie), **readonly** (qui spécifie que la boîte multiligne est en mode lecture seulement) et **wrap** (qui empêche le texte de sortir de la zone en fin de ligne).

Remarque : si du texte se trouve à l'intérieur des balises `<textarea>` et `</textarea>` il va alors apparaître comme texte par défaut. Le bouton reset n'efface pas le texte pré-inscrit.

4 Donner le « focus » à un élément

En HTML, un élément reçoit le **focus** de l'utilisateur pour le rendre actif (activation d'un lien, remplissage d'un contrôle).

On peut donner le focus de deux façons à un élément :

- en cliquant dessus
- en appuyant sur la touche tabulation (à gauche, vers le haut du clavier)

5 Exercices immédiats

Avez-vous une idée pour intégrer, à partir d'un champ de formulaire, une recherche sur Google ?

Donner trois types de balises de formulaire permettant de saisir du texte

Donner deux types de balises de formulaire permettant un choix unique.

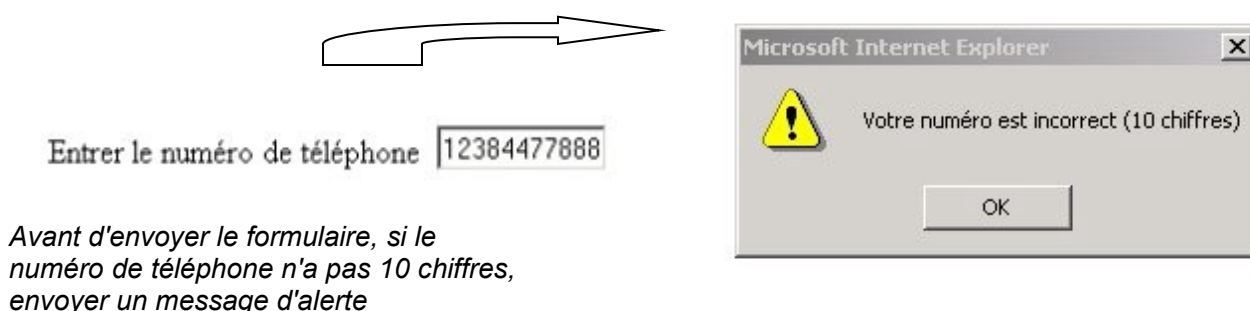
V Premières applications de Javascript

Pour gérer les pages web, on utilise des petits programmes spécialisés que l'on peut trouver sur le serveur et/ou sur le client.

Côté serveur, **PHP**, permet d'aller plus loin : des pages personnalisées vont être générées dynamiquement par le serveur qui autorise éventuellement l'accès à une base de données. C'est l'utilisateur, qui au fil de la consultation, va "aménager" les pages qu'il reçoit.

Côté client, les scripts écrits en **Javascript**, intégrés au code HML ou liés à la page web, servent à limiter les appels au serveur (le nombre de requêtes http). Pour ce faire, on fait exécuter quelques "travaux" directement sur le poste client.

Prenons un exemple simple : pourquoi envoyer un numéro de téléphone manifestement mal composé au serveur ? Il vaut mieux le vérifier avant de l'envoyer : cette vérification est l'objet d'un script écrit en Javascript et interprété par la navigateur.



V.1 Qu'est-ce que Javascript, à quoi sert-il, comment s'insère-t-il dans HTML ?

1 Qu'est-ce que Javascript ?

Créé à l'origine en 1995 par Netscape et conçu pour être multi-plateformes, Javascript est un langage de script **côté client** intégré dans le langage hôte HTML. Sa simplicité et sa « souplesse » conceptuelle et syntaxique lui ont valu une expansion rapide. Dans cette partie, nous allons voir à quoi il sert, comment il s'insère dans une page web et comment en écrire des scripts.

Javascript est un langage **indépendant du format** : on peut ajouter autant de « vides » (espaces, lignes, tabulations) qu'on le souhaite. Mais il est **sensible à la casse** : ceci signifie qu'on fait la différence entre minuscules et majuscules.

2 A quoi sert Javascript ?

HTML est un langage de description sans aucune dynamique propre : Javascript va la lui apporter. Il permet l'écriture et l'exécution de petits programmes appelés **scripts**.

Par exemple, Javascript peut :

- Adapter la mise en page d'un document en cours de chargement
- Ouvrir de nouvelles fenêtres
- Capturer des actions que l'utilisateur effectue
- Effectuer des tâches de retour d'information et de calculs
- Mieux prendre en charge les formulaires

Le **code Javascript** est **interprété** par le navigateur, c'est-à-dire que ses instructions sont décodées les unes après les autres par le **module d'interprétation Javascript** (si ce module existe et est activé, ce qui est automatique dans les dernières versions des navigateurs) **dans l'ordre** où elles se présentent dans le navigateur.

Le code ne peut pas faire grand-chose par lui-même : il travaille avec un programme hôte qui lui fournit objets et données.

3 Y a-t-il un rapport direct entre Javascript et Java ?

L'un et l'autre s'exécutent sur le poste client. Tous deux permettent une programmation événementielle et sont **syntactiquement proches du C++**.

Java est un langage de programmation complet avec un environnement de développement. Il est utilisé dans le contexte du Web sous forme d'applets côté client et de servlets côté serveur, mais ne se limite pas à cela. C'est un langage compilé (production de P-Code), indépendant de la plate-forme de production aussi bien que celle d'exécution. Le code est généré non pour un processeur, mais pour une **machine virtuelle** implémentée par le concepteur du navigateur.

Javascript est plus simple, plus modeste et profondément intégré à HTML : il va permettre de faire des vérifications ou des calculs sur les données d'un formulaire ou encore d'agrémenter le graphisme d'une page Web. Il ne demande pas d'outils spécifiques et offre son code source.

4 L'insertion d'un script dans une page web

Javascript va s'intégrer au langage HTML, comme le langage CSS, sans en altérer sa simplicité. Il s'en remet à l'hôte qui l'héberge pour toutes ses fonctions d'entrées / sorties.

Il peut le faire de plusieurs façons dont la principale est l'insertion d'une balise **<script>** se mettant soit dans la partie d'en-tête, soit dans le corps HTML. Cette balise **<script>** a deux attributs :

- language ou type** pour préciser éventuellement le langage de script : c'est Javascript par défaut,
- src** pour indiquer l'url du fichier Javascript

Exemples

```
<script language="javascript" > // tout simplement <script>
    code javascript
</script>
```

```
<script src="nomfichier.js" />
```

Remarques

*Un script peut aussi être placé dans un fichier à part (ici, nomfichier.js) d'extension **.js**. Dans le fichier externe, on ne re-balise pas avec **<script>** ; on écrit directement le code*

*Un commentaire est introduit par **//** s'il est mono-ligne, est encadré par **/*** ***/** s'il est multiligne*

5 Une utilisation « classique », celle de l'ajout d'une fonction javascript

Pour exécuter une action (sur événement par exemple) au sein d'une page web, on utilise une fonction javascript. Cette dernière se déclare conventionnellement dans l'en-tête et s'appelle dans le corps de la page web.

```

<html>
<head>
  <script>
    function nomFonction() // déclaration de la fonction
    {code javascript}
  </script>
</head>

<body>
  <script>
    nomFonction() // appel de la fonction
  </script>
</body>
</html>

```

6 Une autre façon d'insérer du code grâce à l'url Javascript

Une url est habituellement constituée à partir des schémas d'accès habituels (http, ftp, ..), mais peut aussi l'être à partir du **pseudo-protocole javascript**. On obtient alors une **adresse url-Javascript** qui ne contient pas une adresse conduisant vers un fichier, un compte de courrier électronique, mais provoque l'exécution d'une portion de code Javascript (**une seule ligne**).

Et, ceci de façon interactive ou comme source de document :

Adresse url interactive

L'utilisateur saisit la commande Javascript dans le navigateur comme il le ferait avec n'importe quelle adresse : la fenêtre peut alors afficher soit le résultat d'un calcul (javascript:2+2), soit une aide au débogage.

En tant que source d'un document

On peut remplacer comme suit une url normale par une url javascript :

```

<html>  <head>
  <script>
    function meilleur()
      {resultat = 'excellent score';    //Définition de la fonction
      return resultat}
  </script> </head>
<body>
  Bonjour <br />
  <a href="javascript:meilleur()">Améliorer vos résultats </a>
</body> </html>

```

Si l'on clique sur le contenu de la balise <a>, une nouvelle page avec ce que renvoie la fonction (ici 'excellent score') est créée.

7 Exercices immédiats

Quand l'interpréteur rencontre du code Javascript, l'exécute-t-il immédiatement ?

Quels sont les points communs entre Java et Javascript ? En quoi diffèrent-ils ?

Donner un cas d'application de Javascript

V.2 Variables, types et opérateurs du langage

Objectif : connaître les fondamentaux de la programmation du langage Javascript (variables, constantes, instructions).

1 Edition des scripts

Les scripts Javascript sont de simples textes tapés dans un éditeur de texte. Les mots clés n'utilisent que les caractères Ascii 7 bits. Ce sont principalement :

des structures algorithmique break continue if else while for return function ...
 des objets Number document Boolean Array
 des méthodes et des propriétés typeof do instanceof void write() **getxxx set** ...

Suivant les versions des principaux navigateurs, des aides de mise au point de scripts sont proposées. Sur Mozilla, Menu Outils / Développement web / Console Javascript ; sur Internet Explorer, un débogueur s'ouvre à partir de la barre d'état.

Nous allons reprendre les différents éléments constitutifs d'un langage de programmation : variables, données, types, tableaux, instructions, blocs, commentaires, expressions, conditions, opérateurs, contrôles de flux.

2 Variables et types simples

Les données sont représentées syntaxiquement soit par la variable qui les contient, soit directement en tant que littéral (codées en dur, fixées).

La déclaration d'une variable s'effectue avec le mot clé **var** en lui donnant un nom symbolique sous lequel on désigne un élément quelconque intervenant dans un script et pouvant recevoir différentes valeurs.

Le nom de la variable est composé de caractères non accentués, du caractère de soulignement (underscore) et de chiffres : **le premier caractère n'est pas un chiffre**. Le nom de la variable ne peut pas être un mot clé.

Exemples :

```
var nouvelleValeur ; //variable non initialisée
var cout= null ;
var resultat1 = 10.9 ;
var resultat2 = 'mauvais' ;
var chaine= 'résultat inconnu' ;
```

*Remarque : var ne doit pas **obligatoirement** être mis devant la variable à déclarer, mais il vaut **mieux** le faire. Par exemple, déclarer une variable dans une fonction sans var et sans l'affecter d'une valeur peut conduire à une erreur d'exécution, avec var à une évaluation undefined ou NaN.*

Une variable a un **type**, celui de son contenu. Ici, la variable resultat1 un contient un nombre réel, la variable chaîne une chaîne de caractères.

Javascript est un langage **pauvrement typé** : il ne connaît que **cinq types primitifs simples** de données :

- numérique
- booléen
- chaîne de caractères
- et
- **null**, initialisation marquant une absence de valeur
- **undefined** dénotant une valeur indéterminée (variable non initialisée)

mais il est **dynamiquement** typé : dès qu'une variable est déclarée, le type de la valeur qu'elle contient peut à tout moment être modifié. Aussi, quand on déclare une variable, on n'a pas à en spécifier le type : les variables s'adaptent, se convertissent dynamiquement.

Les nombres (Number)

Javascript gère les valeurs entières et réelles sans grande distinction. Les valeurs entières s'expriment à travers trois systèmes de numération : décimal (1223), octal (0776) et hexadécimal (0x12ED). Les valeurs réelles, en virgule flottante, utilisent le point décimal et les symboles E et e pour indiquer une puissance de 10. Exemples : -2.5, 2.5e-3.

Les valeurs booléennes (Boolean)

Il n'y a que deux valeurs booléennes, les littéraux true (vrai) et false (faux)

Les chaînes de caractères (String)

Les chaînes de caractères se définissent entre **apostrophes ou guillemets**. Le type String autorise le traitement de deux catégories de variables théoriquement distinctes : les caractères (uniques) et les chaînes de caractères.

Remarque : En plus des caractères ordinaires, on peut insérer des caractères spéciaux dans une chaîne. On utilise l'anti-slash comme caractère d'échappement. Ainsi :

\' permet d'insérer une apostrophe

\\ permet d'insérer un back slash

\uXXXX permet d'insérer un caractère via sa référence hexadécimale Unicode

Exemple

```
<html>  <head>
<script language="javascript" >
  var a=225 ; citation = "ils dirent \"Ca va ?\" \" Oui \" " ;
  document.write("<h2>Affiche avec la méthode write de l'objet document</h2>");
  document.write("La racine carrée de " +a+ " est "+Math.sqrt(a)+"<br />") ;
  document.write("FIN  <br />") ; document.write(citation) ;
</script>  </head>
<body>
  <h3>Voici un bel exemple d'intégration</h3>
</body>  </html>
```

Affichage avec document.write

La racine carrée de 225 est 15

FIN

ils dirent "Ca va ?" "Oui "

Voici un bel exemple d'intégration

Pour séparer deux instructions, on utilise le point-virgule.

Dans les chaînes de caractères, avec la méthode write le code HTML est interprété.

Attention : Cet exemple utilise :

- la **méthode write de l'objet document** (notation pointée document.write) permet un affichage dynamique (interprétation du script au moment du chargement de la page).
- Le signe **+** marque une concaténation de chaînes de caractères et d'une variable.
- la **méthode de l'objet Math (Math.sqrt())** qui calcule la racine carrée. En parlant de méthodes, nous sommes en avance sur le cours !

3 Transtypage Les conversions de types

Le tableau suivant indique comment javascript gère automatiquement les conversions de type de variables

	Nombre	Booléen	Chaine
Nombre non nul		true	le nombre en chaîne
Nombre nul		false	'0'
NaN (Not a number, 1/0)		false	'NaN'
Booléen true	1		'true'
Booléen false	0		'false'
Chaine non vide	un nombre ou NaN (*)	true	
Chaine vide	0	false	
Undefined	NaN	false	"undefined"
Null	0	false	< null >

Les conversions peuvent être problématiques. Dans les cas d'utilisation conjointe de chaînes et nombres, l'opérateur + pousse la conversion en chaînes, l'opérateur – vers celle en nombre.
 "33" – 10 retourne 23, "33"+7 retourne 337.

4 Les tableaux (en seconde lecture)

En Javascript, il y a deux *principales* catégories de variables primitives : les variables **simples** représentant des valeurs individuelles (les chaînes de caractères, les nombres et les booléens) et les (variables) **tableaux** représentant une collection structurée de données.

La déclaration d'un tableau se fait sous la forme d'un littéral : `var tab = [12,14, 6 ,4]` (tableau à 4 éléments).

Elle peut aussi se faire en instanciant un nouvel **objet**, **Array**, à l'aide de l'instruction **new** sans préciser le contenu : `var tab = new Array(12)` (tableau à 12 éléments).

On numérote les éléments d'un tableau à partir de zéro. Ainsi un tableau `tab` qui contient trois éléments se compose de `tab[0]`, `tab[1]`, `tab[2]`. 0, 1, 2 sont reconnus comme **indices**.

On peut modifier la taille d'un tableau en enregistrant une valeur dont l'indice est supérieur à l'indice existant le plus élevé..

Exemple :

```
<html><body>
<script language="javascript1.3">
  var Tableau = new Array()
  Tableau[0] = 'debut'; Tableau[1] = 'milieu'; Tableau[2]= 'fin'
  document.write('La taille de ce tableau est : ' + Tableau.length)
</script>    <p />
  Ceci est un <script language="javascript1.3">
    document.write(Tableau[0] + '!!')</script> </body> </html>
```

La taille de ce tableau est : 3

Ceci est un debut!

*Remarque : on utilise ici la propriété **length** (longueur du tableau) de l'objet **Array**.*

A la place d'une valeur d'indice numérique, on peut donner un nom à la « case » du tableau : `tab[alain]`.

Les tableaux de Javascript n'ont qu'une seule dimension, mais on peut passer outre en incluant un tableau dans un tableau : `var matrice=new Array(2) ; matrice[0]= new Array(2) ; matrice[1]=new Array(2) ; matrice[0][0] = 1 ; matrice[1][1] = 1 ;`

Un tableau peut être hétérogène, c'est-à-dire contenir des éléments de types différents :
 var t = [2, « lettre », [a,b]]

Remarque : les tableaux, en Javascript, peuvent avoir une utilisation spécifique, à savoir servir à regrouper des éléments (des objets) de même type (tableaux des images, des formulaires).

5 Opérateurs principaux, expressions et conditions

Une **expression** est une combinaison de valeurs, de variables, d'opérateurs et d'autres expressions que l'interpréteur doit **évaluer** afin **d'obtenir une seule valeur**.

Une **condition** compare des expressions 'valorisées' et renvoie une constante booléenne.

Les combinaisons des expressions ou des conditions se font au moyen **d'opérateurs** qui interviennent en suivant des règles de **priorité**. Pour être actif un opérateur a besoin d'opérandes : 1, 2 ou 3 (respectivement pour un opérateur unaire, binaire, ternaire).

Opérateur de chaînes de caractères

L'opérateur de concaténation, noté +, permet "d'accoler" ou "d'enchaîner" plusieurs chaînes.
 Exemple 'Bonjour' + ' Madame' donne comme résultat 'Bonjour Madame'.

Opérateurs arithmétiques

Ce sont +, -, *, /, % (modulo).

Opérateurs booléens (logiques)

Le ET logique se transcrit par &&

Le OU logique se transcrit par || (*caractère clavier AltGr 6*)

Le NON logique (unaire) se transcrit par !

Remarque : quand les opérations deviennent complexes, il vaut mieux ajouter des parenthèses.

Opérateurs de comparaison (relationnels)

Ils permettent de créer des expressions booléennes et peuvent opérer aussi bien sur des valeurs numériques que sur des chaînes de caractères.

==	égal à
!=	différent de
< ou <=	inférieur ou inférieur ou égal)
> ou >=	supérieur ou supérieur ou égal)
===, !==	strictement égal ou différent sans conversion (même valeur et même type)

Remarques :

Il faut toujours mettre des parenthèses autour d'une condition.

Le signe = est réservé à l'affectation des variables

Exemples :

```
45 % 9   retourne 0 (% retourne le reste de la division entière) ; var w= ('alain' && 68) // true
Tab = new Array() ; if (!Tab[0]) faisCeci()      // La fonction faisCeci va être exécutée
```

Opérateurs unaires divers

Ce sont les opérateurs d'incrémentation / décrémentation préfixés ou post fixés : ++a, a++, a--, --a, le moins unaire et le + unaire. a++ et ++a remplacent simplement a = a + 1.

Lorsque l'opérateur ++ est placé devant, la variable augmente d'une unité et la nouvelle valeur est reprise dans toutes les expressions dont elle fait partie. En revanche, si l'opérateur est placé après, c'est l'ancienne valeur (avant incrémentation) qui sera utilisée dans l'expression, et une fois ce processus achevé, l'incrémentation sera appliquée a posteriori.

```
<html> <body> <script> var x=10 ;
  document.write(++x, '<br />')      // va afficher x = 11
  document.write(x, '<br />')        // va afficher x= 11 (x ne bouge pas)
```

```
document.write(x++, '<br />') // va afficher x = 11
document.write(x, '<br />') // va afficher x = 12 (x a bougé)
</script> </body> </html>
```

*Remarque : les éléments à afficher sont séparés par des virgules (on aurait pu concaténer ce qu'il faut afficher). Le passage à la ligne se fait en ajoutant la chaîne '
' interprétée comme on l'a vu par la méthode document.write.*

6 Les autres opérateurs (éventuellement en seconde lecture)

Opérateur d'affectation (=). Son utilité n'est autre que de mettre une valeur dans une variable.

D'autres opérateurs d'affectation ont une écriture minimale : **x += y** pour $x = x + y$, **x /= y** pour $x = x / y$.

Les opérateurs ternaire conditionnel, new, typeof, instanceof et this.

L'opérateur ternaire conditionnel est un substitut pour l'instruction « if ». **x = (a>b) ? c : d**

x va adopter la valeur c si $a > b$, sinon la valeur d. Exemple : statut = (age >=18) ? Adulte : Mineur

L'opérateur **new** permet de créer une **instance** d'objet (on a vu avec la création d'un tableau, on le reverra plus loin) à partir de type d'objet prédéfini (Array, Date, Image, Option, String, ...).

L'opérateur **typeof** sert à identifier un type de données

Pour chaque affectation var x = « rien » ; var x ; var x =5, document.write(typeof(x)) va respectivement afficher string, undefined, number. Pour typeof(Tab.length), il va retourner number

*L'opérateur instanceof retourne true si l'objet spécifié est du type spécifié : si Tab est un tableau, **Tab instanceof Array** va retourner true.*

7 Priorité des opérateurs

La **priorité** indique l'ordre dans lequel sont impliqués les opérateurs pour évaluer une expression.

Voici l'ordre de priorité, de la plus basse à la plus haute :

Virgule	,
Affectation	=
OU logique	
ET logique	&&
Egalité	== !=
Comparaison	< > <= >=
Addition/soustraction	+ -
Multiplication/division	* /
Négation ou incrément.	! ++ --
Appel de fonction	()
Nouvelle instance	new
Membre	[]

8 Exercices immédiats

L'expression numérique suivante est-elle correcte : 0x56789Q a) Oui ? b) Non ?

L'expression numérique suivante est-elle correcte : 1A

a) Oui ? b) Non ?

Quelle est l'expression booléenne en a, b, e (exprimant les propriétés d'appartenance respective à A, B, E) équivalente à l'expression ensembliste $A \cup B = \bar{E}$ (A union B est égal au complémentaire de E)

a) $a \&\& b = !e$ b) $a || b = !e$ c) $a || b != e$

L'expression $2 + 2 == 4$ est-elle (prend-elle une valeur)
 a) booléenne ? b) algébrique ? c) arithmétique ?

L'index suivant d'un tableau Tab est-il correct : $\text{Tab}[8*3/2]$
 a) Oui ? b) Non ?

Quel est le résultat de l'évaluation $2 == "2"$?
 a) true b) false c) null

Le code suivant

```
<html> <body> <script>
var attention = 'ne mettez pas </script> littéralement dans une chaîne' ;
</script> </body> </html>
```

génèrera l'affichage de a) 'ne mettez pas' b) 'littéralement dans une chaîne' ;
 c) aucun affichage

Quel type de valeur contient la var z déclarée par $\text{var z} = ('0xFF' + 0xF0)$?
 a) Booléenne b) numérique c) chaîne

Du code javascript externe doit contenir la balise <script> a) Faux b) Vrai

V.3 Contrôle de flux et instructions

1 Généralités

Les structures syntaxiques de contrôle de flux permettent de spécifier la nature et le moment d'exécution des instructions Javascript. Sans elles, les instructions seraient exécutées linéairement (les unes après les autres).

Les instructions peuvent être regroupées dans un bloc (une fonction) et traitées à leur tour comme une instruction autonome.

2 L'alternative

L'instruction if ... else

Cette structure d'instructions offre une alternative entre deux blocs d'instructions suivant que la condition est vraie ou fausse.

```
if (condition) {
    instruction1
    .....
    instructionp }
else {
    instruction1
    .....
    instructionp }
```

Remarques :

- On peut utiliser cette structure en cascade (ifelse if else)
- il existe une forme raccourci du couple if ... else. : **condition logique? action1 : action2.**
- si on ne met pas les accolades autour d'une instruction, il faut mettre un point virgule entre le if et le else. Exemples :

```
if (n>=0) {alert("Nbre positif ou nul")} else {alert("Nbre négatif")};
ou encore n>=0?alert("Nbre positif ou nul"):alert("Nbre négatif");
```

```
<html>    <head>
<script>
```

```

function verif3()
{
    var troisCar="AAaa" ; // on aurait du utiliser ici un champ de saisie
    if (troisCar.length == 3) {return true } // On continue ...
    else {alert("Entrer exactement 3 caractères. <br />"
        +troisCar+ " n'est pas valide") ; return false }
}
</script> </head>
<body>
Début
<script>
    verif3() ;
    document.write("fin")
</script>
</body> </html>

```

Remarque : alert() est une méthode de l'objet window (objet le plus haut) qui ouvre une fenêtre modale avec le contenu passé en paramètre et un seul bouton (OK).

3 L'instruction while (tant que)

L'instruction while permet de répéter l'exécution d'un bloc d'instructions tant qu'une condition d'entrée est vérifiée : **while (condition) { instruction1 ; ; instructionp }**. Exemple :

```

var num = 150 - 1 ; var fin = false ;
while (fin == false)
{
    if (150 % num ==0) {document.write("Le plus grand facteur de 150 est " + num) ; fin = true ; } ;
    num = num - 1 ;
}

```

A chaque itération num va baisser de 1. Dès que num divise 150, la boucle s'arrête.

Remarque : attention aux boucles infinies : while (true) { alert("Hello, world") }

4 L'instruction do .. while

L'instruction do ... while permet de répéter l'exécution d'un bloc d'instructions **tant qu'une** condition, testée **après** le bloc, est vérifiée (valeur booléenne true). L'exécution du bloc d'instruction a toujours lieu au moins une fois

```

do { instruction1 ; ..... ; instructionp ; }
while (condition)

```

Exemple

```

var num = 150 - 1 ; var fin = false
do {
    if (150 % num ==0) {document.write("Le plus grand facteur de 150 est "
        + num ) ; fin = true ; } ; --num }
while (fin==false)

```

5 L'instruction for

L'instruction for permet de répéter l'exécution d'un bloc d'instructions en précisant une commande à exécuter avant la première itération, une commande à exécuter après chaque itération et une condition de sortie.

```

for (initialisation ; test ; incrémentation)
{ instruction1 ; ..... ; instructionp ; }

```

Exemple :

```

<html>    <head>    <script language="javascript" >

```

```

var i ;
  for(i=5 ; i<=10 ; i++) {document.write(i + "<br />")}
</script>      </head>
<body>  et voilà  </body>  </html>

```

6 break et continue

Deux instructions peuvent modifier le cours de l'exécution d'une boucle :

- **break** permet d'interrompre l'itération en cours et de passer à l'instruction **après** la boucle

```

for(i=5 ; i<=10 ; i++)
  {if (i==7) break ;
    document.write(i + "<br>") ;}

```

va afficher

5
6

- **continue** permet d'interrompre l'itération courante et de passer à la suivante.

```

for(i=5 ; i<=10 ; i++)
  {if (i==7) continue           // itération suivante
    document.write(i) }        // va afficher 568910

```

*Remarque : attention, l'instruction break n'a rien avoir avec la balise
 appelée aussi break*

7 L'instruction switch ... case

Elle permet de sélectionner un bloc d'instructions à exécuter en fonction de la valeur d'une expression passée en paramètres

```

switch (expression) {
                                case val1: instruction1; break ;
                                .....
                                case valn: instructionn; break ;
                                default: instructiondefaut }

```

Exemple

```

<body>
  <script>
    var chiffre = 2
    switch (chiffre) {
      case 1: document.write('Un'); break ;
      case 2: document.write('Deux'); break ;
      case 3: document.write('Trois'); break ;
      default: document.write('Ce n\' est pas un chiffre < 4') }
    </script>
  </body>

```

8 Exercices immédiats

Comment débiter l'instruction conditionnelle : par

a) if i=5 then b) if (i==5) c) if i=5 d) if i==5 then

Comment débiter l'instruction conditionnelle (cas ou i ≠5) : par

a) if (i != 5) b) if (i <> 5) c) if <>5 d) if != 5 then

Quelle est la bonne syntaxe de début d'une boucle ?

a) for (i <= 5; i++) b) for i = 1 to 5 c) for (i = 0; i <= 5) d)for (i = 0; i <= 5; i++)

Comment ajouter un commentaire dans du code javascript?

a) 'Ceci est un commentaire' b) <!-- Ceci est un commentaire --> c) //Ceci est un commentaire

Qu'affiche ce code dans une page web

```
<html> <body> <script>
var alaligne='<br />';
for(boucle1=0 ; boucle1 < 5 ; boucle1++)
{for(boucle2=0 ; boucle2 <= boucle1 ; boucle2++)
{document.write('T');}
document.write(alaligne) ;}
</script> </body> </html>
```

a)	T TT TTT TTTT TTTTT	b)	Talaligne TTalaligne TTTlaligne TTTTalaligne TTTTTlaligne	c)	T TT TTT TTTT TTTTT
----	---------------------------------	----	---	----	---

Que fait ce code dans une page web

```
<html> <body> <script>
for(boucle=1000 ; boucle < 1299 ; boucle++)
{if (boucle % 99 == 0) break ;}
document.write(boucle) ;
</script> </body> </html>
```

a) il affiche les nombres compris entre 1000 et 1089
b) il affiche 1089
c) il affiche les multiples de 99 compris entre 1000 et 1299.

Que fait ce code dans une page web :

```
<html> <body> <script>
var boucle=0 ;
while (++boucle < 200)
{if (boucle % 7 != 0) {continue ;}
document.write(boucle + ' ');}
</script> </body> </html>
```

a) il affiche tous les multiples de 7 inférieurs à 200
b) il affiche tous les nombres non divisibles par 7
c) il affiche 7

Quelles sont les valeurs de n prises pendant l'exécution du programme suivant ?

```
i = 0
n = 0
while (i < 5)
{ i++ ; if (i == 3) continue ; n += i}
```

a) 1, 2, 3, 4 b) 5 c) 1, 3, 7, 12

Ecrire un programme qui affiche les n premiers entiers et en calcule la somme

V.4 Les fonctions

1 Définition et déclaration des fonctions

Comme l'interpréteur javascript du navigateur exécute une instruction **dès qu'il l'a lue**, si nous voulons que cette instruction dépende d'une circonstance particulière, on doit l'inscrire dans une **fonction** destinée à être **invoquée** (appelée) au bon moment.

Une fonction est donc un ensemble (un bloc) d'instructions regroupées pour effectuer une action à un moment donné. La déclaration d'une fonction se fait avec le mot clé **function** suivi du nom de la fonction, d'une liste d'arguments de la fonction mis entre parenthèses et séparés par des virgules, des instructions spécifiques de la fonction.

function nomfonction(arg1, ..., argn)

```
{
  instruction1 ;
  .....
  instructionp ;
return valeurareturner ;
}
```

S'il y a une valeur à retourner, elle suit le mot-clé **return** ; s'il n'y en a pas, on peut supprimer la ligne commençant par return (c'est le cas d'une "procédure", d'une sous-programme) ou la laisser : return se comporte alors comme un break.

Remarque : les paramètres sont passés par valeur. Si la fonction change la valeur d'un paramètre, ce changement n'est pas reflété globalement.

2 l'appel des fonctions

Déclarer une fonction ne l'exécute pas : il faut l'invoquer par son nom avec ses paramètres (entre parenthèses), pour exécution :

à l'intérieur d'une balise script

```
<html> <head> <script> function salutation(){document.write("Bonjour!")} </script> </head>
<body> Voici un résultat : <script>salutation()</script> </body>
</html>
```

Bonjour s'inscrit dans la page

- comme pseudocode sur un lien (adresse url javascript)

```
<html> <head> <script type="text/javascript" >
function salutation(){document.write("Bonjour!")}</script> </head>
<body>Voici un résultat:<a href="javascript:salutation()">plus ?</a>
</body> </html>
```

Attention : quand on exécute la fonction en cliquant sur le lien, le texte "Bonjour" s'affiche dans une nouvelle fenêtre."

3 Portée des variables

Une variable **globale** est déclarée en début de script et est accessible à n'importe quel endroit du programme.

Une variable **locale** est déclarée à l'intérieur d'une fonction et n'est utilisable que dans la fonction elle-même (et même moins encore).

Pour éviter des interactions indésirables entre variables, il faut mieux les déclarer en local, en particulier pour celles qui servent de compteur ou de variables temporaires.

Exemple

```
<script>
var x='dehors' ;
function essai()
{var x='dedans' ; document.write(x) ;}
document.write(x) ; essai() ;
</script>
```

Ce script va afficher successivement dehors dedans.

4 Le tableau des arguments d'une fonction (en seconde lecture)

Les arguments d'une fonction sont systématiquement stockés dans un tableau **arguments[]**.

Ainsi le premier argument passé à une fonction est arguments[0]. Le nombre total d'arguments est indiqué par **arguments.length**.

En Javascript, le nombre d'arguments d'une fonction n'est pas déterminé d'avance.

Prenons l'exemple d'une fonction qui concatène plusieurs chaînes. Le premier argument formel passé est le séparateur des éléments à concaténer, ... mais ce n'est pas le seul.

function myConcat(separateur)

```
{resultat="" ;           // Initialisation de la boucle
  for (i=1; i<arguments.length ; i++) { resultat += arguments[i] + separateur }
  return resultat }
```

Application : myConcat(", ", "red", "orange", "blue") retourne "red, orange, blue, "

Et si on ne voulait pas voir apparaître une virgule à la fin ?

5 Fonctions Javascript natives

Javascript a plusieurs fonctions prédéfinies : eval, isFinite et isNaN, parseInt et parseFloat,

Code d'évaluation

La fonction eval(chaine) reçoit une chaîne et l'évalue (la calcule). Exemple :

```
var e = eval("1+a") ; document.write(e) ;
va afficher 1a.
```

Détection de valeurs exceptionnelles

isNaN(x) et isFinite(x). Ces deux fonctions évaluent respectivement leur paramètre x pour dire si ce n'est pas un nombre ou si c'est un nombre fini (elles renvoient un booléen).

Contrôle de conversion de types

parseInt(chaine) et parseFloat(chaine,base) servent à extraire des valeurs de types numériques contenues dans des chaînes et **placées devant**. Exemple : parseInt (« 12RT »,10) va afficher 12.

Remarque : Les fonctions parseFloat et parseInt retournent "NaN" quand elles évaluent une chaîne qui ne contient pas de nombres. Elles sont utiles pour "parser" les dates, les écrire en millièmes de secondes.

6 Exercices immédiats

Quel est le résultat de l'évaluation de l'expression (a*b - 5 < x) && (b < 5)

- quand a vaut 10, b vaut 2 et x vaut 1 ?
- quand a vaut false, b vaut 9 et x vaut true ?

Comment intégrer un script hors du code source html ? Comment intégrer à l'aide de Javascript un même texte (à mettre à jour régulièrement par exemple) en plusieurs endroits ou documents ?

Que va afficher le code suivant

```
<html><body>
  <script>
    function combien()
      {document.write("Appel avec " +arguments.length +
        " arguments" + "<br />")}
    combien(1,2);   combien("55", 56, true, 0)
  </script> </body> </html>
```

Remarque : attention, si on définit quatre arguments et qu'on en utilise que trois, le quatrième prend la valeur undefined.

Que va afficher le code

```
<html><body> Un mot
<script> var x = 1; var result=0 ; chaine="if(x==1) {result=9}" ; eval(chaine); document.write(result);
</script> </body> </html>
```

V.5 Javascript sur évènement

1 Définition d'un évènement

Un **évènement** peut survenir du fait de l'interpréteur Javascript (une erreur), au bout d'un certain laps de temps (minuterie) ou le plus souvent consécutivement à une action voulue de l'utilisateur : touche de clavier enfoncée, clic de souris, remplissage d'un formulaire. C'est l'**information détectée** par le navigateur qui va apporter de l'interactivité à l'application.

Un **gestionnaire d'évènement** est la prise en main de l'évènement par un script qui s'exécute **en réaction** au dit évènement. La syntaxe de gestion est : **<nomBalise on ... = "code à exécuter" >**.

Un **modèle évènement** décrit la façon dont les évènements sont créés et gérés par le programme sur un navigateur. Voici les composants du **modèle évènement HTML** :

- condition spéciale qui provoque l'évènement
- attributs spéciaux dans les balises HTML pour chaque évènement.
- script destiné à gérer l'évènement
- objet Javascript sur lequel un évènement donné agit
- contrôle transmis temporairement à l'interpréteur
- données accompagnant l'évènement

Prenons un exemple :

```
<html> <head> <title> Envoi direct </title> </head>
<body>
  <form name="f1" action="mailto:seigneur@math-info.univ-paris5.fr" method="post" >
    <b>Saisissez votre message</b> :<br />
    <textarea name="message" cols="40" rows="3"
      <b>onchange="document.f1.submit()"> </textarea> <p />
  </form>
</body> </html>
```

Ce script appelle plusieurs explications :

Ce qui provoque l'évènement est le changement de contenu dans le champ texte multiligne (textarea), changement matérialisé **quand on sort** de ce champ (par la touche Tab ou en cliquant en dehors du textarea).

L'attribut spécial **onchange** va permettre l'exécution du script qui porte sur l'objet formulaire dont l'interpréteur va demander la soumission.

Le script est composé d'une seule instruction **document.f1.submit()**

Le contrôle est donné à l'interpréteur qui va faire envoyer le mail.

L'adresse électronique (attribut action de la balise form) est une donnée accompagnant l'évènement

2 Liste des évènements les plus courants

Tout un arsenal de commandes permet de capter ces évènements et d'en faire usage dans des scripts :

Attribut spécial	Description de l'évènement	Objet sur lequel il porte
------------------	----------------------------	---------------------------

onBlur	Cet événement se produit lorsque l'utilisateur sort d'un champ de formulaire HTML, d'un cadre ou d'une fenêtre (window), en appuyant sur la touche TAB du clavier ou en utilisant la souris. La fonction appelée peut servir à soumettre ou valider les informations saisies par l'utilisateur.	Button Checkbox Password Radio Reset Select Submit Text Textarea window
onChange	Cet événement se produit lorsque l'utilisateur modifie la valeur d'une boîte liste, d'une boîte à liste déroulante, d'une boîte texte ou d'une boîte texte multiligne. La fonction appelée peut servir à valider ou soumettre les informations entrées dans les champs d'un formulaire	FileUpload Select Text Textarea
onClick	Cet événement se produit lorsque l'utilisateur enfonce puis relâche le bouton de la souris sur un bouton, une option dans un formulaire, un hyperlien ou sur le document.	Button Checkbox document Radio Reset Submit
onDbClick	Cet événement se produit lors d'un double-clic suivi d'un relâchement sur un élément de formulaire ou un hyperlien	Area document
onError	Cet événement se produit lorsqu'il y a une erreur au cours du chargement d'une image ou d'un document.	Image et window
onFocus	Cet événement se produit lorsqu'un élément (champ de formulaire, cadre, etc.) ou un objet (window) obtient le focus. Le focus peut être obtenu par l'intermédiaire de la souris, de la touche TAB du clavier ou par une méthode de focus.	Button Checkbox Password Radio Reset Select Submit Text Textarea window
onMouseOut	Cet événement se produit à chaque fois que l'utilisateur déplace le pointeur de la souris hors d'une zone contenant un hyperlien (images en coordonnées, lien hypertexte).	Area, Image
onMouseOver	Cet événement se produit à chaque fois que le pointeur de la souris entre dans une zone contenant un hyperlien (lien hypertexte ou images en coordonnées).	Area, Image
onMove	Cet événement se produit lorsque l'utilisateur ou un script déplace une fenêtre ou un cadre.	window
onReset	Cet événement se produit lorsque l'utilisateur active le bouton de type reset d'un formulaire	Form
onResize	Cet événement se produit lorsque l'utilisateur modifie les dimensions de la fenêtre du navigateur.	window
onSelect	Cet événement se produit lorsque l'utilisateur sélectionne du texte dans une boîte texte ou dans une boîte texte multiligne.	Text Textarea
onSubmit	Cet événement se produit lorsque l'utilisateur envoie les données d'un formulaire.	Form
onLoad (onUnload)	Cet événement se produit au moment où le document HTML courant est chargé (ou retiré de la fenêtre au profit d'un autre).	Image window

Remarques :

- ◆ rien n'empêche de spécifier plus d'un gestionnaire d'événement dans une même balise.
- ◆ cette liste n'est pas exhaustive

3 Exemple détaillé sur les événements

Il s'agit tout d'abord de recréer une page identique au modèle ci-dessous mais simplifiée (la page se compose de 4 cellules d'un tableau comprenant des images ; les cellules sont positionnées autour d'une grande cellule centrale).

A titre d'utilisation de l'attribut onLoad, on demande un redimensionnement automatique de la fenêtre au chargement (400 pixels de large sur 200 de haut ; cellules latérales 100x100, cellule centrale 200x200).

Au passage de la souris sur une des quatre images latérales, celle-ci change et passe en couleur. Quand le curseur ne pointe plus l'image, celle-ci repasse en noir et blanc. Si on clique sur une des images, celle-ci s'affiche en grand format dans la cellule centrale.



- Dessiner d'abord douze images avec les chiffres :
 1. quatre en noir et blanc 100x100 nommées image_i.gif,
 2. quatre autres sur des fonds de couleur différents 100x100 nommées image_iC.gif,
 3. quatre agrandies 200x200 nommées image_iG.
- 2 Ecrire le code du tableau avec les cinq cellules (celle nommée "centrale" 200x200), puis y positionner les images ... sans oublier de les mettre dans une ancre (lien).
- 3 Rajouter dans les cellules le code événementiel pour le survol ou le clic
- 4 Rajouter le code pour tailler la fenêtre d'ouverture

Voici une proposition de codage :

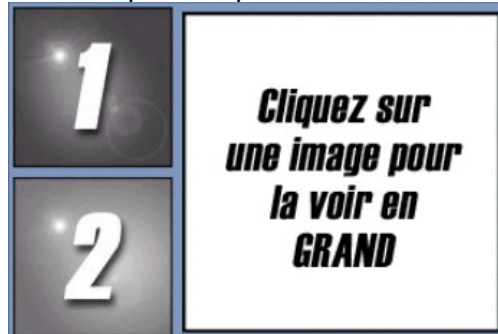
```
html> <head> <title>rollover</title>
<script> function redim() {self.resizeTo(400, 200);} </script> </head>
<body onload= redim()>
<table bgcolor="#FF6600" align="center">
<tr>
<td><a href="#" onClick="document.centrale.src='image_1G.gif'"
onMouseOver=document.image1.src="image_1C.gif"
onMouseOut=document.image1.src="image_1.gif">
</a></td>
<td rowspan="2"></td>
<td><a href="#" onClick="document.centrale.src='image_3G.gif'"
onMouseOver=document.image3.src="image_3C.gif"
onMouseOut=document.image3.src="image_3.gif">
</a></td>
</tr>
<tr>
<td><a href="#" onClick="document.centrale.src='image_2G.gif'"
onMouseOver=document.image2.src="image_2C.gif"
onMouseOut=document.image2.src="image_2.gif">
</a></td>
<td><a href="#" onClick="document.centrale.src='image_4G.gif'"
onMouseOver=document.image4.src="image_4C.gif"
onMouseOut=document.image4.src="image_4_nb.gif">
</a></td>
</tr> </table> </body> </html>
```

Remarque : on insère les images dans une ancre pour que, quand la souris passe dessus, le curseur de la souris se transforme en une main. La balise td peut prendre comme attribut onclick, mais on ne saura pas qu'il faut cliquer !

4 Exercices immédiats

Qu'est-ce qui fait que le champ d'un formulaire perd le focus ? Comment se manifeste le « focus » ?

Refaire l'exemple ci-dessus et le réaliser plus simplement. Tableau à 3 cellules :



Au lieu de mettre l'image en couleur, mettre un chiffre dans une cellule avec un fond coloré. Changer, au survol, la couleur du fond de la cellule.

Quand on clique sur la cellule, afficher le double des chiffres des cellules de gauche dans la grande cellule

Pourquoi l'auteur du cours écrit-il parfois onmouseover et d'autres fois onMouseOver alors que Javascript est sensible à la casse ?

Ecrire le code qui place une image dans une page web et qui ouvre une fenêtre d'alerte ("tu me fais de l'ombre") quand on la survole et qui ouvre une autre fenêtre d'alerte ("Arrête ça fait mal") quand on click dessus.

V.6 Objets, propriétés et méthodes (en grande partie, en seconde lecture)

Objet : savoir manipuler certains objets prédéfinis.

1 : Définitions

Qu'appelle-t-on objet ? Grossièrement, c'est une certaine quantité de données associée à une ou plusieurs fonctions afin de donner une entité facilement manipulable. Il peut exister plusieurs objets semblables (plusieurs objets window, frame form.), chacun étant une entité distincte.

Partons d'un exemple :

```
var personne = new Object ;
    personne.nom = 'dupont' ; // cette personne a une propriété nom qui prend la valeur dupont
    personne.age = 32 ;
    personne.auto = new Object ;           // Ce nouvel objet a ses propres propriétés
    personne.auto.nom = 'renault' ;
```

La conception Objet permet d'accéder intuitivement aux propriétés des objets par la **notation pointée** : **nomObjet.nomPropriete**. Si on veut lui attribuer une valeur, on écrit : **nomObjet.nomPropriete = valeur ;**

On remarque dans notre exemple que l'objet auto est rattaché à son **objet parent** personne.

Un objet peut-être constitué de propriétés ou se voir attribuer des méthodes.

Une **méthode** est introduite par une fonction. Prenons un exemple :

```
<html> <head> <script> function f_disbonjour() {document.write(" Bonjour, <br /> ") ;}
    var personne = new Object ;           // Création d'un objet
    personne.nom = "Alain" ;              // Définition d'une propriété
    personne.disbonjour = f_disbonjour ; // Définition de la méthode (1)
```

personne.disbonjour()

// Appel des méthodes (2)

Que remarque-t-on ?

- La propriété `disbonjour` de l'objet `personne` est affublée d'un **nom** de fonction (1). Elle est définie comme méthode
- A la fin de l'exemple (2), on applique la méthode `disbonjour()` (avec parenthèses) à l'objet `personne`.

Une méthode se définit donc en deux étapes :

- la définition d'une fonction décrivant la méthode elle-même
- la définition de la méthode comme **propriété** : le nom de la propriété devient le nom de la méthode, sa valeur est le nom de la fonction sans parenthèses (le nom seulement **car il s'agit d'une référence vers la fonction et non pas d'un appel**) : **objet.nomMéthode = nomFonction**

La syntaxe d'appel est : **nomObjet.nomMéthode(arg1,..., argn)** où `argi` sont les arguments de la méthode.

2 Les types primitifs Number et Boolean et l'objet tableau

Ils peuvent être traités comme des objets, surtout pour en utiliser les propriétés et méthodes prédéfinies :

```
var data0 = new Boolean(false) ;
var data1 = new Number("5") ;
```

L'objet Boolean est utilisé comme conteneur pour une variable booléenne. Il n'a qu'une seule méthode : `toString()` (... et elle ne sert pas à grand chose car la méthode `document.write` sait faire !)

```
<html>   <body>   <script> var data0 = new Boolean(" True ") ;
      document.write(data0.toString()) ; </script>   </body>   </html>
... tout juste pour afficher le booléen true !
```

L'objet Number possède les propriétés telles que `MAX_VALUE`, `NaN` à assigner à des constantes.

JavaScript prédéfinit l'objet Array et ses méthodes pour manipuler les tableaux.

Pour créer un objet Array, on donne la liste de ses éléments : `nomTableau = new Array(element1, element2, ..., elementn)` ou encore sa longueur initiale : `nomTableau = new Array(arrayLength)`.

On référence les éléments d'un tableau par son indice (attention, le premier élément est `monTableau[0]`).

Les principales méthodes de l'objet Array sont :

<code>concat()</code>	concatène deux tableaux (concatène les "listes" de leurs éléments)
<code>join()</code>	place tous les éléments du tableau dans une chaîne avec un séparateur
<code>pop()</code>	retire le dernier élément du tableau et le retourne
<code>shift()</code>	retire le premier élément du tableau et le retourne
<code>reverse()</code>	transpose le tableau (le premier élément devient le dernier)
<code>sort()</code>	trie les éléments d'un tableau ... suivant l'ordre ASCII

Exemple : Si `monTableau = new Array[A, C, B]`,

`monTableau.join()` = "A,C,B" `monTableau.pop()` = B `monTableau.reverse()` fait que `monTableau[0]` = B, `monTableau.sort()` fait que `monTableau[1]` = B, ...

Remarques : Objets et tableaux sont interchangeables.

Les deux notations sont équivalentes. Si `Tab = new Array(3)`, `Obj = new Object`, on peut écrire `Tab.x = 1` pour `Tab['x'] = 1` et `Obj['x'] = 1` pour `Obj.x = 1`.

Exemples : avec la propriété `action` d'un formulaire, on peut écrire indifféremment la forme de tableau ou de nom d'objet : `forms[1].action = " "` ou `formul.action` si le deuxième formulaire de la page a pour nom `formul`.

Attention : les noms des propriétés ne peuvent pas être numériques. Il faut nécessairement utiliser la forme du tableau.

Attention, `length` ne s'applique qu'à un tableau déclaré comme tel.

3 : L'objet String

Javascript n'est pas à l'aise dans les calculs, mais en ce qui concerne les chaînes de caractères (String) il dispose de nombreuses méthodes. On peut définir une chaîne de caractères de façon implicite `citation = "Eurêka"` comme on peut créer un objet String : `citation = new String("Eurêka")`. Les méthodes s'appliquent à ces deux définitions ; elles peuvent être inspirées d'HTML ou s'attacher directement à la manipulation de chaînes.

Les méthodes inspirées d'HTML

`bold()` affichage en gras ``
`italics()` affichage en italiques `<i>`
`sub(), sup()` affichage comme indice ou exposant `<sup>` `<sub>`
`fontcolor()` affichage de caractères en couleur (la couleur est passée en paramètres)
 s'inspire de ``
`fontsize()` affichage des caractères suivant la taille, s'inspire de ``

Remarque : on peut appliquer plusieurs méthodes à une chaîne de caractères.

Exemple : `document.write(citation.bold().fontsize(10))`

Les méthodes de manipulations intérieures aux chaînes

`charAt()` sert à extraire un caractère situé à une certaine position
`indexOf()` recherche si une chaîne contient une autre chaîne et renvoie
 l'emplacement le plus à gauche de la sous-chaîne diminué de 1
`substring()` sert à extraire une sous-chaîne dont on indique les positions initiale et finale
`toLowerCase(), toUpperCase` sert à passer un chaîne en minuscules (majuscules)
 Exemples avec la chaîne Eureka : `citation.charAt(4) = k`, `citation.indexOf("ka") = 4`,
`citation.indexOf("ki") = -1`, `toLowerCase("Eureka") = "eureka"`
 Quelle valeur prend `var sousdata = citation.substring(1,4) + citation.charAt(5)` ? *ureka*
`citation.split("t")` va scinder la chaîne (à l'aide de la lettre t) citation en un tableau de trois éléments :
 "ci", "a", "ion"

Remarques : il y a bien d'autres objets prédéfinis en Javascript (Math, Date, ...)

4 Exercices immédiats

Le code suivant est-il correct ? 1) Oui 2) Non

```
<html> <body> <script language="javascript">
  if (Math.random()) {document.write("ça marche!")} </script>
</body></html>
```

A quel endroit le gestionnaire d'évènements `onLoad` doit-il être placé?

1) dans la balise `<head>` 2) dans la balise `<body>` 3) dans l'une et l'autre

Comment arrondir 7.25 au nombre entier le plus proche?

`round(7.25)` `Math.round(7.25)` `Math.rnd(7.25)` `rnd(7.25)`

Comment déterminer le plus grand nombre parmi 2 et 4 ?

`Math.ceil(2,4)` `ceil(2,4)` `top(2,4)` `Math.max(2,4)`

Ecrire le code qui permet de trouver un nombre compris entre 1 et 100 tiré au sort avec des indications telles que "trop grand" ou "trop petit".

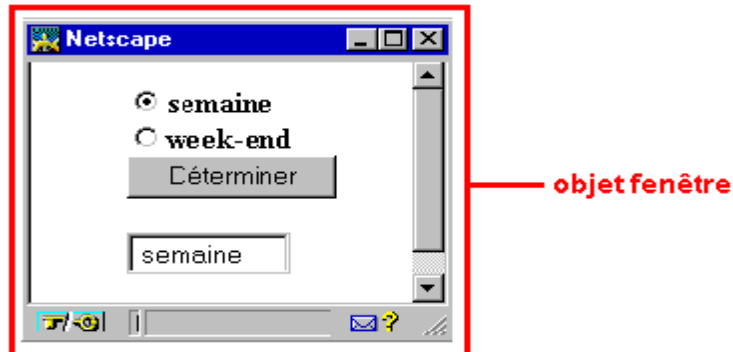
V.7 Les objets hôtes / modèles objets navigateurs

Objectif : comprendre la hiérarchie des objets hôtes du modèle navigateur et savoir manipuler les objets `window`.

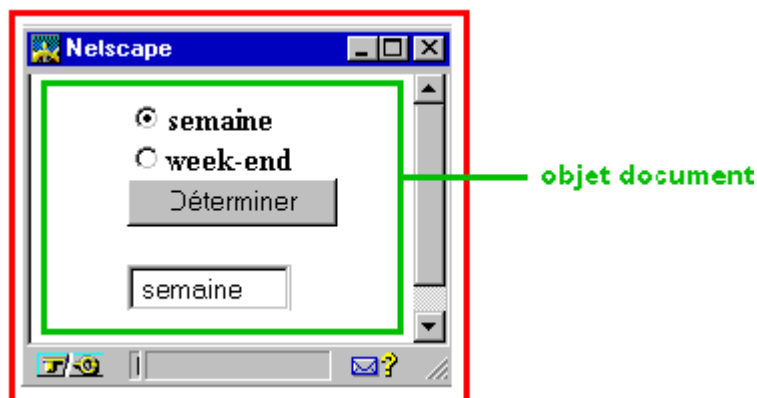
1 Mécanisme d'instanciation des objets hôtes

Lorsque le navigateur charge une page web, l'interpréteur Javascript crée un ensemble **d'objets hôtes** basé sur le contenu du code de la page et récupère les propriétés du navigateur et de l'écran. Les propriétés des objets hôtes sont alors accessibles et modifiables avec les méthodes disponibles.

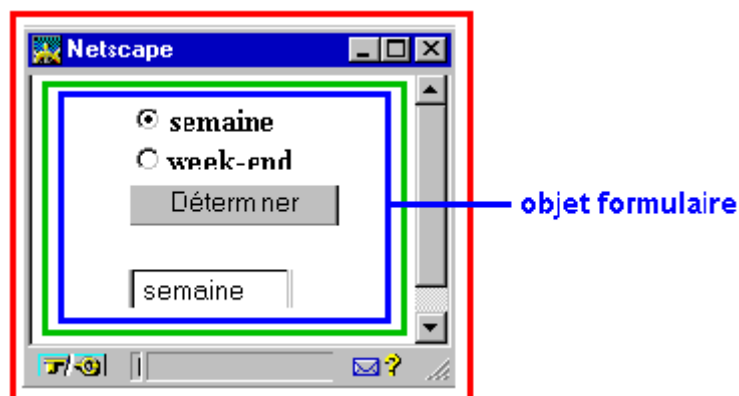
L'élément racine <html> va instancier un **objet window**, représentée par la fenêtre dans laquelle le fichier HTML va s'afficher



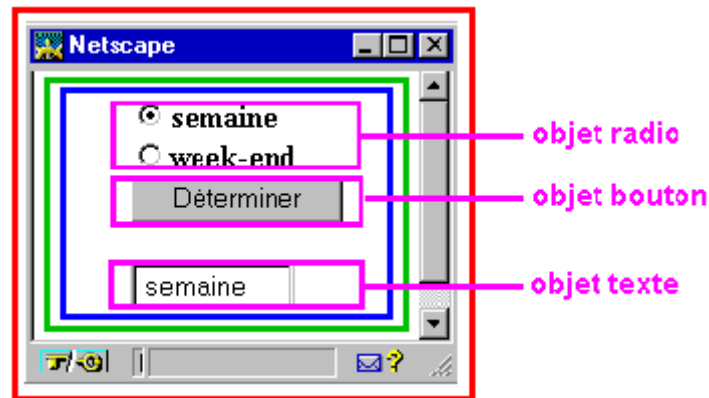
L'élément body va afficher un contenu en instanciant un **objet document**.



Dans cet exemple, le corps du fichier HTML contient un élément form qui va instancier un **objet formulaire**.



L'élément formulaire contient trois sous-éléments input de types radio, button et text : trois nouveaux objets sont donc créés respectivement **un objet radio** (représenté comme un tableau), un **objet bouton**, un **objet champ texte**.



Autrement dit l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour une hiérarchie d'objets :

window **document** **forms[0]** **elements[3]**

Remarque : ce ne sont pas les seuls objets que l'on peut créer.

2 le modèle objet navigateur : hiérarchie générale

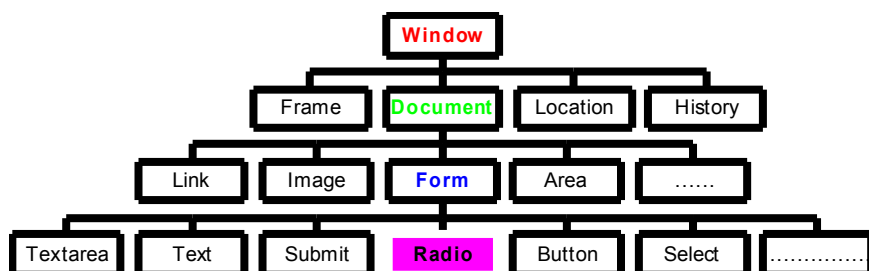
L'interpréteur Javascript est partie intégrante d'un navigateur. C'est pour cette raison que l'on parle de **modèle objet navigateur** définissant les objets hôtes Javascript. Malheureusement apparaissent suivant les navigateurs des différences au niveau de l'interprétation du code Javascript.

L'interpréteur Javascript non seulement crée des objets au chargement de la page, mais il les intègre dans une hiérarchie, celle du modèle navigateur. Ceci va permettre, à travers une notation dite "pointée" de définir pratiquement toutes les propriétés et méthodes des objets de la page web et partant de la modifier.

Exemple pour un formulaire nommé formu avec un premier bouton radio que l'on souhaite voir sélectionné par défaut : **(window).document.formu.radio[0].defaultChecked="true"**

Attention : nous avons mis l'objet window entre parenthèses : comme il occupe la première place dans la hiérarchie, il est repris par défaut par Javascript et devient donc facultatif.

Voici un schéma (approximatif) de hiérarchie générale avec ses objets le plus courants :



Attention cette hiérarchie n'est pas exhaustive : il n'y a pas tous les objets .

Pour écrire du code Javascript, il est essentiel de bien comprendre l'articulation de cette hiérarchie en objets car pour référencer un objet on va le désigner en spécifiant ses ascendants. Et partant, il est tout aussi important de bien imbriquer les éléments dans une page HTML

Une fois l'objet créé, il peut être atteint et transformé par du code javascript.
Donnons un exemple : `document.bgColor="red"` met le fond du document en rouge

3 les objets navigateurs et screen (en seconde lecture)

Avant même que la page s'affiche, deux objets sont instanciés : le navigateur (**navigator**) chargeur et l'écran (**screen**). Quel en est l'intérêt ? Comme on l'a dit, il y a une différence d'interprétation suivant les navigateurs et il peut être utile de connaître le navigateur utilisé pour optimiser le code.

L'objet **navigator** a plusieurs propriétés dont :

appName	retourne le nom complet du navigateur
appVersion	retourne le numéro de la version
language	retourne le code langage du navigateur
platform	retourne le nom du système d'exploitation

Exemple :

```
appCodeName=Mozilla
appName=Netscape
appVersion=5.0 (Windows; en-US)
language=en-US
platform=Win32
oscpu=Windows NT 5.0
product=Gecko
userAgent=Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.4) Gecko/20030624
cookieEnabled=true
javaEnabled= function javaEnabled() { [native code] }
preference= function preference() { [native code] } appCodeName : Mozilla
```

S'y ajoutent deux méthodes, `javaEnabled()` et `préférence()`.

Aucun événement ne peut être associé à l'objet navigator.

Exemple de script qui permet d'afficher les caractéristiques du navigateur (cas d'Internet Explorer et Netscape ; Opera est difficile à reconnaître)

```
<html>  <head>    <Title>Quel est le navigateur ? </title>
  <script language="javascript">
    function quelnavigateur()
      {
        navigateur = "Inconnu" ; langue = "Inconnue"
        if (navigator.appName.indexOf("Microsoft") == -1)
          {
            langue = navigator.language
            version = navigator.appVersion.substring(0,4)
            if (navigator.appVersion.indexOf("[") == -1)
              navigateur = "Netscape Navigator ou Mozilla"
            else navigateur = "peut-être Opera"
          }
        else {navigateur = "Internet Explorer"
              langue = navigator.userLanguage
              var i = navigator.appVersion.indexOf("MSIE")
              version = navigator.appVersion.substring(i+5, i+8)
            }
        document.write(navigateur + "<br /> Version " + version +
          "<br /> Langage "+ langue + "<br />")
      }
  </script> </head> <body onLoad="quelnavigateur()"> </body> </html>
```

Remarque : il existe des méthodes de détection du navigateur plus efficace

L'objet **screen** possède des propriétés concernant le système d'affichage du visiteur.

Pour Mozilla, ce sont : top (coin supérieur gauche de la fenêtre), left, width et height, availWidth et availHeight (availHeight et availWidth contiennent la taille en pixels utilisable pour l'affichage de la page). Internet Explorer ne reconnaît pas top et left.

Voici un exemple de centrage d'une popup (*un peu en avance sur le cours*)

```

<html> <head> <title>Pop-up centrée</title>
<script language="javascript">
function popupcentree(page, largeur, hauteur)
{
    var top=(screen.height-hauteur)/2;
    var left=(screen.width-largeur)/2;
    window.open(page, "", "top="+top+",left="+left+",width="+largeur);
    // ouverture d'une fenêtre popup (en avance sur le cours)
}
</script> </head>
<body> Lien pour l'ouverture de la fenêtre centrée
<a href="javascript:popupcentree('Paradis.jpg',300,200)">Popup</a>
</body> </html>

```

4 Exercices immédiats

Donner des noms d'objets hôtes

Donner des noms de propriétés de ces exemples

A partir d'un exemple, donner un chemin pour atteindre un objet instancié dans une page web

V.8 Des objets de haut niveau, les fenêtres

Objectif : savoir manipuler avec Javascript les objets **window** (qui représente une fenêtre au sens large, fenêtre ou frame) et leurs propriétés et méthodes et des sous-objets.

1 Les propriétés de l'objet window

Les principales propriétés sont :

defaultStatus prend pour valeur le texte par défaut qui apparaît dans la barre d'état.

name prend pour valeur le nom de la fenêtre.

status prend pour valeur le texte qui doit apparaître dans la barre d'état (prend le dessus sur defaultStatus).

Exemples :

```

<html><head><title>Test</title>
</head><body>
<a href="fichier.htm" onmouseover="window.status='Ceci est un lien';return
true">Lien</a>
</body></html>

```

Attention : si vous supprimer return true ou si vous le modifiez en return false, vous n'aurez pas le même résultat. Que faire pour annuler par la suite le message status dans la barre d'état ?

2 Les premières méthodes de l'objet window

Elles sont nombreuses :

focus() : Donne le focus sur la page (page activée)

blur() : Retire le focus de la page

moveBy(x,y) : Déplacement relatif par rapport à la position de la fenêtre

moveTo(x,y) : Déplacement vers un point précis

resizeBy(x,y) : Redimensionnement relatif

resizeTo(x,y) : Redimensionnement à une taille fixe

print() pour imprimer une page

back(), **forward()**, **stop()**, **home()** simulent un clic respectivement sur les boutons Précédent, Suivant, Stop ou Accueil du navigateur (pour revenir sur la page précédente par exemple).

Exemple

```

<html> <head> <script>
function shake(n)

```

```

{  if (self.moveBy)
    {  for (i = 10; i > 0; i--)  // pas de tremblement allant en diminuant
        {for (j = n; j > 0; j--)
            {self.moveBy(0,i);
             //le coin supérieur gauche qui était en (0,1) passe en (0,10)
             self.moveBy(i,0);
             self.moveBy(0,-i);
             self.moveBy(-i,0);    }      }
        }
    }
}
</script> </head> <body><form>
<input type=button onClick="shake(10)" value="Secouez l'écran!"></form>
</body> </html>

```

3 Les Minuteries (en seconde lecture)

Les minuteries permettent de déclencher une action au bout d'un intervalle de temps fixé. Cette action peut être unique ou répétitive, arrêtée ou en continu. On utilise, suivant le cas, **setTimeout**(chaîne de la fonction, intervalle de temps en ms), **setInterval**(chaîne de la fonction, intervalle de temps en ms), **clearTimeout**(nom de la minuterie), **clearInterval**(nom de la minuterie).

Exemples de minuteries :

1 Tester vos réflexes

```

<html> <head><script>
function start()
{comp=window.setTimeout("alert('Vos réflexes sont faiblards')",2000); }
</script></head>
<body>
<form name="formulaire">
<input type="button" value="Mise en route" onClick="start()">
<input type="button" value=" Stop test " onClick="clearTimeout(comp)">
</form></body></html>

```

Avec l'instruction **comp=window.setTimeout("alert('Vos réflexes sont faiblards !')",2000)**, on initialise une minuterie qui va déclencher la fonction **start()** après un délai de 2 secondes. Avec **clearTimeout(comp)**, on va arrêter avant terme le compteur dont le nom est **comp**.

*Remarque : on place dans une variable la minuterie. Ceci va permettre de la passer en argument de la méthode **clearTimeout**.*

2 Changez de fond d'écran

```

<html><head><title> Changement de fond d'écran toutes les secondes</title>
<script>
function Couleur()
{  if(couleur==1) { document.bgColor="yellow"; couleur=2; }
   else { document.bgColor="aqua"; couleur=1; }
   i = i + 1;
   if(i >= 10)   window.clearInterval(activ) ;   }
</script> </head>
<body> <script type="text/javascript">
var activ = window.setInterval ("Couleur()",1000);
var i = 0, couleur = 1;
</script></body></html>

```

4 Les méthodes **open()** et **close()** (en seconde lecture)

La méthode **open(url, nomFenêtre,options)** ouvre une nouvelle fenêtre, un (ou une) **popup**. Elle attend au moins deux paramètres, le troisième est facultatif :

url donne l'adresse du fichier qui doit être chargée dans la nouvelle fenêtre.

nomFenêtre sert à nommer la fenêtre pour éventuellement la prendre pour cible de chargement.

options (facultatif) insère une chaîne de caractères portant sur l'apparence de la fenêtre (taille et

propriétés de la fenêtre séparées par des virgules).

La méthode **close()** ferme une fenêtre. Syntaxe : windowRef.close()

Remarques :

- Une fenêtre est reconnue en HTML par son nom, nom pouvant servir à indiquer une cible par exemple (target).
- Une fenêtre est reconnue en Javascript par sa référence (nom de la variable).
- Attention, si on ne ferme pas une fenêtre secondaire avant la fenêtre qui l'a appelée, elle reste sous la forme réduite dans la barre des tâches.

Prenons un exemple : créons d'abord deux fenêtres secondaires qui vont afficher les fichiers premier.htm et deuxieme.htm quand on les appellera :

```
<html> <body> Première fenêtre </body> </html>
<html> <body> Deuxième fenêtre </body> </html>
```

Voici le code du fichier appelant les fenêtres secondaires

```
<html> <head> <title>Fenêtre principale </title> </head>
<body>
  FENETRE PRINCIPALE<br />
  avec ouverture automatique d'une première fenêtre secondaire
  <script>
    var win1 = open("premier.htm","premier", "resizable=no,width=200,height=200");
    // pour ouvrir une seconde fenêtre secondaire
    function montre()
      {var win2 = window.open("deuxieme.htm","deuxieme",
        "resizable=no,width=100,height=100") ;};
  </script>
</p><a href="#" onClick="montre() ;"> Cliquez </a>
</body> </html>
```

Notons que :

- l'appel de la méthode open() se fait comme suit :
nomVariable = open(« url », « nomFenêtre », « option=val1, option=val2, ... »)
 Si la fenêtre existe, on pourra la ré-ouvrir avec open(", 'son nom') ;
 - Il ne faut pas confondre les propriétés Javascript de la fenêtre avec les propriétés dites ici de mise en page (faisant l'objet du troisième paramètre) de la méthode open qui sont les suivantes :
 - la barre d'état en utilisant l'option status et les valeurs= "no" ou "yes" (par défaut)
 - les barres de défilement en utilisant l'option scrollbars
 - les barres d'outils en utilisant l'option toolbar
 - les barres de menus en utilisant l'option menubar
- Dans l'exemple donné, quand on a cliqué sur le lien, trois fenêtres sont ouvertes, mais pas forcément visibles. Attention : une peut cacher les autres (on le voit dans la barre des tâches : regarder derrière !).

Remarque : en ce qui concerne les options (le troisième paramètre de la méthode open), la barre de titre d'une fenêtre et les boutons qui y sont inclus sont fournis par le système d'exploitation, donc inaccessibles depuis Javascript. Les composants du 3^{ème} argument de la méthode open() doivent être séparés par des virgules.

- la méthode open() renvoie de nouvelles instances window placées dans les variables win1 et win2 ; elle donne en plus un nom à toute nouvelle fenêtre indépendamment du nom donnée par title (propriété de l'objet document).
 La variable win2 est locale à la méthode montre() : elle disparaît donc quand la méthode se termine. L'objet window est précisé mais ce n'est pas nécessaire car la fenêtre courante est l'objet qui gère l'événement (un script s'exécute toujours dans un objet : lequel ici ?).
- L'url « # » est une astuce qui empêche la page d'être remplacée.

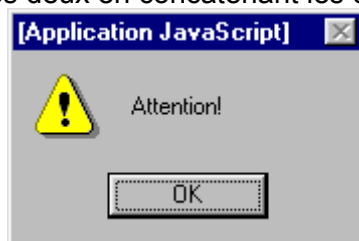
Remarque générale : en informatique, le créateur ou le conteneur d'un objet peut nommer sa descendance, mais n'appelle son ascendance qu'avec un nom générique (père). C'est le cas de la hiérarchie des fichiers : `cd nomfils` pour descendre, mais `cd ..` pour remonter.

5 Les boîtes de dialogue

Une **boîte de dialogue** est une fenêtre qui s'affiche au premier plan (fenêtre modale) suite à un événement, et qui permet soit d'avertir l'utilisateur, soit de lui demander de compléter un champ pour récupérer une information.

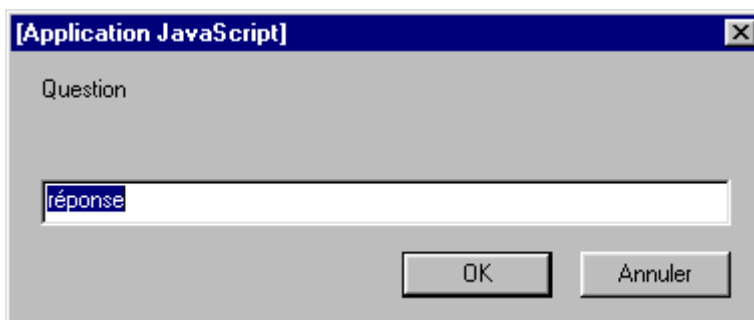
Elles s'ouvrent en faisant appel à des méthodes de l'objet *window*. Elles affichent une mention Javascript inchangeable, par exemple " Application Javascript " pour permettre à l'utilisateur de savoir qu'il s'agit d'une boîte d'invite de la page en cours et rien de plus ! Elle n'interprète pas le HTML.

La méthode **alert()** permet d'afficher dans une boîte toute simple composée d'une fenêtre et d'un bouton OK le texte qu'on lui fournit en paramètre. Dès que cette boîte est affichée, l'utilisateur n'a d'autre alternative que de cliquer sur le bouton OK. Son unique paramètre est une chaîne de caractère, on peut donc la lui fournir directement entre guillemets, soit lui fournir une variable dont il affichera le contenu, ou bien mêler les deux en concaténant les chaînes grâce à l'opérateur +.



Remarque : la chaîne de caractère peut contenir des caractères marqués d'un antislash (\). Par exemple, si vous voulez écrire "Message d'alerte" puis exécuter un passage à la ligne, puis écrire "Au feu!!", il faudra écrire le script suivant : `alert('Message d\'alerte \nAu feu!!');`

La méthode **prompt()** est un peu plus évoluée que la précédente puisqu'elle fournit un moyen simple de récupérer une information provenant de l'utilisateur, on parle alors de boîte d'invite. La méthode `prompt()` requiert deux arguments : le texte d'invite et la chaîne de caractères par défaut dans le champ de saisie



Sa syntaxe est donc la suivante : Sa syntaxe est : `prompt('Posez ici votre question', 'chaîne par défaut');` Cette boîte d'invite retourne la valeur de la chaîne saisie par l'utilisateur, elle retourne la valeur *null* si jamais aucun texte n'est saisi...

Prenons un exemple :

`<html> <body>`

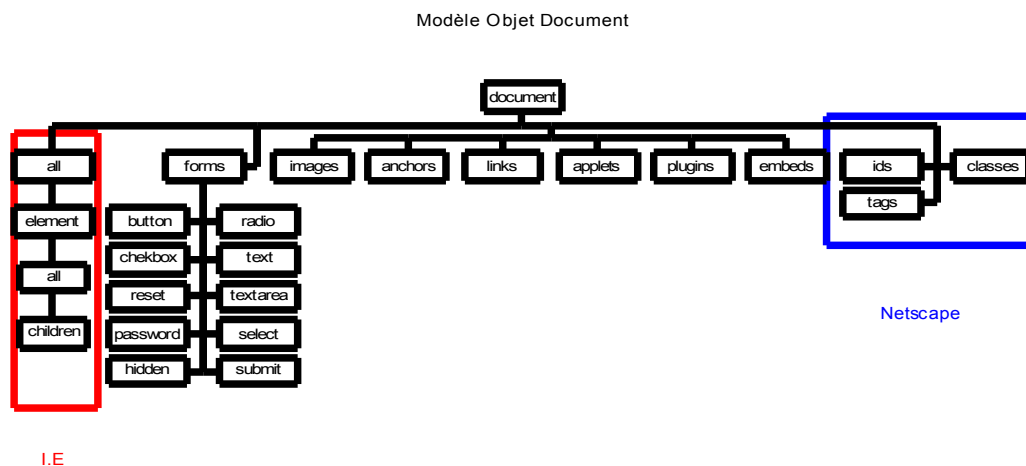
```
<script>
    accord=prompt("Vous aimez Javascript ? ") ;
    // Réponse ou non
    if (accord) {alert("c\'est super")}
    else {alert("Shame at you !")}
</script>
```


Saisir dans deux fenêtres prompt une voyelle et un mot. Ecrire une fonction qui recherche les occurrences de cette voyelle dans le mot et qui affiche leurs positions.

V.9 La structure de l'objet document

1 L'objet document

Le sous-objet document de l'objet window représente le contenu HTML lui-même. Il donne accès à son tour à des sous-objets représentant ses éléments (formulaires, image, ...). Lorsque le navigateur charge une page HTML, l'interpréteur Javascript **instancie** un certain nombre d'objets et de **collections** (tableaux) en relation directe avec le code HTML inclus. Ces objets sont conservées dans une hiérarchie du type suivant (elle évolue un peu avec la sortie de nouvelles versions de navigateurs) :



2 Les propriétés de l'objet document

Elles sont nombreuses et vous pouvez les afficher (rappelez-vous la fonction DisPropriétés d'un objet). Les principales sont :

alinkColor : Couleurs des liens actifs (bouton de souris enfoncé)

linkColor : Couleur des liens non actifs et non visités.

vlinkColor : Couleur des liens visités

bgColor : Couleur de fond

fgColor : Couleur du texte

lastModified : Date de dernière modification du document source.

title : titre inscrit dans la barre de titre de la fenêtre

location : nom du fichier contenant le document

Attention : il y a location sous-objet de l'objet window et location propriété de l'objet document.

3 Les méthodes de l'objet document

La méthode **write()** est la plus courante : elle affiche du texte et du code HTML dans le document en cours. En voici un exemple qui reprend deux propriétés.

```

<script language="javascript">
    document.write("L'URL courante est : "+document.location+"<br>");
    document.write("Dernière modification : "+document.lastModified);
</script>
  
```

Attention : si vous ne mettez pas les caractères + indiquant qu'il s'agit bien d'une chaîne, le script ne fonctionnera pas.

4 Utilisation très partielle du DOM (en seconde lecture)

Le **DOM HTML**, on l'a vu, est une API (Interface de programmation d'application) qui permet aux scripts d'accéder aux contenus, balises, attributs et styles d'un document HTML.

Il a pour ambition que chaque balise, chaque attribut, chaque élément (identifié ou nommé) contenu dans un document puisse être accessible (pour éventuellement être modifié). Au delà et parallèlement aux longues références pointées vers les objets et leurs propriétés, il propose un autre mode d'accès par les nœuds.

Pour accéder à un nœud disponible dans le document, sont utilisées les méthodes de l'objet document : accès par l'identifiant (attribut id), par le nom, (attribut name), par le nom de balise, pour un texte sélectionné. Ce sont les méthodes : **getElementById**, **getElementsByName**, **getElementsByTagName**, **getSelection()**.

Partant de là, vous pouvez adresser les nœuds attribut, les nœuds texte et les autres nœuds enfant d'un élément.

Exemple qui va afficher "Où est ma place ?" comme on le souhaite.

```
<html><head>  <script>
function aligner(comment)
{document.getElementById("indecis").align =comment;}
</script></head>
<body>
<h1 id="indecis">Où est donc ma place?</h1>
<a href="javascript:aligner('left')"> à gauche? </a><br />
<a href="javascript:aligner('center')"> au centre? </a><br />
<a href="javascript:aligner('right')"> à droite?</a> <br />
</body></html>
```

V.10 Javascript et les formulaires

Objectif : savoir manipuler les formulaires avec Javascript

1 L'objet correspondant à la balise <form>

C'est un sous objet de l'objet document qui comprend à son tour d'autres objets. Si le formulaire a un nom (fo par exemple), on peut y accéder en écrivant **document.fo** ; s'il n'en a pas, on y accède en écrivant **document.forms[0]** s'il s'agit du premier formulaire du document.

*Quand un formulaire est chargé avec la page web, ce n'est pas un objet seul qui est instancié, mais un tableau **forms** qui est le tableau des formulaires du document.*

Remarque : il est préférable d'utiliser le nom des éléments, plutôt que les indices : les noms sont indépendants de l'organisation du formulaire.

2 Les méthodes et événements applicables aux formulaires

Deux méthodes : **submit()** qui provoque l'envoi du formulaire vers l'url indiquée dans l'attribut action et **reset()** qui ré-initialise les données du formulaire avec les valeurs par défaut.

Deux événements correspondants : **onreset** qui détecte la ré-initialisation et **onsubmit** qui détecte l'envoi du formulaire. Ce dernier peut permettre, par exemple, la soumission d'un formulaire après l'avoir passé au crible de la validation.

3 Les sous-objets de l'objet form provenant d'une balise input

Les principales propriétés associées à l'objet "input" sont :

name	Nom du champ
type	type du champ <i>text, button, radio, checkbox, submit, reset</i>

value	Libellé texte
defaultvalue	Valeur par défaut du champ (utile avec reset)
size	Taille du champ
maxlength	Taille maximale du champ de type texte
checked	Case à cocher ou radio bouton coché ou non
disabled	Grisé (modification impossible par le visiteur)
readOnly	Lecture seule
class	Nom de la classe de style
style	Chaîne de caractères pour le style

La liste ci-dessus est générique : certaines particularités apparaissent suivant le "type".

Exemples

document.fo.champ1.value="nouveau!" pour un champ text

document.fo.rad[2].checked=true pour un type radio ou checkbox

Les méthodes acceptées sont focus() et blur() : l'élément peut être rendu actif ou non.

Exemple : **document.fo.champ1.focus()** : le champ1 prend le focus, devient actif.

Les évènements acceptés : onfocus, onblur, onclick.

Exemple pour faire apparaître un message dans la barre d'état :

```
<html><head><title>Test formulaire</title> </head><body>
<form name="focus">
<input type="text" name="champ1" value="contenu" onfocus="window.status = this.name"
onBlur="window.status=''">
<p />Le nom du champ texte apparaît dans la barre d'état quand il est actif
</form> </body></html>
```

*Remarque : pour éviter d'écrire un "long chemin" pour accéder à un élément, il est préférable d'utiliser l'objet prédéfini **this** qui représente l'objet javascript courant.*

Exemple avec le gestionnaire onclick : la gestion du tableau des boutons radio :

```
<html><head><title>Test formulaire</title>
<script>
function testerRadio(OperSys)
{for (var i=0; i<OperSys.length;i++)
{if (OperSys[i].checked) alert("Système = "+OperSys[i].value)}}
</script> </head>
<body> <form name="fo">
<input type="radio" name="os" value="Windows NT" />Windows NT
<input type="radio" name="os" value="Linux" />Linux
<input type="radio" name="os" value="Autre" />Autre
<input type="button" value="Tester" onClick="testerRadio(this.form.os)"> </form>
</body></html>
```

Le chargement du formulaire crée un tableau de radio-boutons liés de nom os. Une boucle parcourt la liste des boutons du groupe : on repère le bouton qui a la propriété checked à true et on affiche alors la valeur correspondante.

Utilisation "particulière" de blur() pour empêcher la modification d'un champ

```
<input type="text" value="Non modifiable" name="champ" onFocus="this.blur()" />
```

4 Les sous-objets de l'objet form provenant d'une balise <select>

Ce sont des listes qu'il convient de nommer pour pouvoir l'utiliser avec javascript. La création d'une liste donne naissance à un tableau **options** alimentant les lignes de la liste.

Les propriétés de l'élément select sont :

name	Nom de la liste
size	Nombre de lignes à afficher
multiple	Sélection multiple autorisée
disabled	Grisage de la liste
class	Classe de feuille de style
style	Style de la liste
selectedIndex	indice de la ligne courante

Exemples :

- pour récupérer l'indice la ligne sélectionnée :

```
this.form.liste1.selectedIndex
```

- pour créer une scrolled-list de nom choix et de 8 lignes

```
<select name="choix" size="8">
```

Les propriétés des éléments option sont :

defaultSelected	option choisie par défaut	
selected	option sélectionnée	
length	Nombre de lignes	
value	Valeur d'une ligne	
text	Libellé d'une ligne	

Exemples :

- pour récupérer le nombre de lignes :

```
this.form.liste1.options.length
```

- pour récupérer la valeur de la ligne sélectionnée :

```
this.form.liste1.options[this.form.liste1.selectedIndex].value
```

Les méthodes sont add() pour ajouter une ligne, remove() pour supprimer une ligne, focus() pour donner le focus à une ligne et blur()

Les évènements les plus courants : onchange (qui détecte la sélection d'une nouvelle ligne), onfocus, onblur.

Prenons un exemple d'ajout dynamique d'éléments dans une liste :

```
<html><head> <script>
function ajouter()
{
  // création d'une nouvelle valeur d'option
  nouvel_element = new Option(document.formulaire.nouveau.value);
  // Positionnement de la nouvelle option à la fin de la liste
  document.formulaire.choix.options[document.formulaire.choix.length]
    = nouvel_element ;
  // Réinitialisation du champ texte
  document.formulaire.nouveau.value = "";
}
```

```

</script></head>
<body>
<form name="formulaire" >
<select name="choix" size="8"> <option>Un élément</option> </select><p />
Entrer le nouvel élément à ajouter à la liste : <input type="text"
name="nouveau"><br />
<input type="button" value="Ajouter" onClick="ajouter()">
</form> </body></html>

```

Dans cet exemple est défini un formulaire contenant une liste avec un élément initial, un champ de saisie et un bouton. Dans le champ de saisie, l'utilisateur peut entrer les éléments qu'il désire ajouter à la fin de la liste. En cliquant sur le bouton, la valeur saisie est prise dans la liste comme élément supplémentaire. Le champ nouveau est vidé.

Les instructions de la fonction `ajouter()`, successivement, créent un nouvel objet `option` avec pour valeur le contenu du champ nouveau, placent la nouvelle option en fin de liste, remettent le champ texte à zéro.

*Attention : un **objet** `option` à deux arguments : text d'abord (qui correspond au décor, à ce qui se voit et qui est obligatoire) et value ensuite (qui est optionnel).*

Quand on lit dans la première ligne de la fonctionvalue, il s'agit de mettre en "text" de l'option la value du champ texte nouveau !!

5 Les zones multilignes (textarea)

Elles sont essentiellement utilisées pour permettre au visiteur de saisir un texte assez long.

Ses propriétés sont `name`, `disabled`, `readonly`.

Ses méthodes sont `focus()` et `blur()`.

Ses événements sont `onchange`, `onfocus`, `onblur` et `onscroll` quand on provoque le défilement de la zone

6 Exemple détaillé

Ce script permet de sélectionner dans une liste une couleur de fond qui se chargera dynamiquement

```

<html> <head> <title>fond.html</title>

<script>
// définition d'une fonction nbChange recevant en paramètre un objet
function bgChange(selObj)
{
  newColor = selObj.options[selObj.selectedIndex].text ;
  document.bgColor = newColor ; // le fond de la page prend la couleur new color
}
</script> </head>
<body> <b>Modification des couleurs de fond</b> <br>
<form>
<select size="10" onChange="bgChange(this) ;">
  /* ici on appelle la fonction bgChange avec en paramètre this
  (cette liste, select) comprenant les couleurs ci dessous.
  A chaque sélection cette fonction est exécutée. */
  <option>Teal <option>Orange <option>Yellow <option>Green <option>Navy
  <option>Blue <option>Indigo <option>Violet <option>White <option>Red
</select>
</form> </body> </html>

```

7 Exercices immédiats

Pour éviter de recevoir plusieurs fois un même formulaire, on veut empêcher le visiteur de s'exciter sur le bouton en lui envoyant un message *"Le formulaire est en cours de traitement... Patience !!"* Construire une fonction qui gère ceci.

VI Concevoir, publier et référencer un site web

Le web est **universel** : le site à construire devra être accessible de partout pour un maximum de gens et dans les meilleures conditions. Aussi faudra-t-il prendre soin

- de sa **portabilité** sur tous les réseaux (débits), sur toutes les machines (systèmes d'exploitation) et sur tous les navigateurs.

- de son **ergonomie**, c'est-à-dire du mariage du **confort** de l'utilisateur et de l'**efficacité** de son interaction avec l'information présentée

VI.1 Conception globale d'un site web

Objectif : savoir unifier la présentation (**charte graphique**), le contenu (**charte éditoriale**) et l'organisation (**navigation aisée**) d'un site web pour assurer la cohésion des pages ... dans un esprit maison.

1 L'internaute est pressé

Le visiteur d'un site web est **pressé** (79% des visiteurs parcourent très rapidement la page). Il doit arriver au but rapidement : on doit lui-mettre sous le nez ce qu'il est censé chercher à l'aide du graphisme et des liens

2 La transition d'une page à une autre

Les pages doivent arborer un **aspect unifié** (charte graphique) de façon à ce que la transition d'une page à une autre se fasse en douceur (par exemple le sommaire toujours en colonne à gauche, une navigation par menu horizontal, le logo toujours en haut et à droite) et en interne (il faut que le visiteur reste sur le site).

Il peut être pratique, dans des cas bien particuliers (un texte complémentaire par exemple, une aide), d'ouvrir une fenêtre indépendante de la fenêtre principale.

3 La grille de structure visuelle

La structure d'une page, c'est le choix d'un modèle de page, d'une grille si on organise la page en colonnes et en lignes. Les éléments de la structure doivent être agencés de façon à ce qu'ils ne se mélangent pas et c'est pour cette raison qu'il faut ménager des espaces libres "**actifs**" qui appuient la structuration en évitant toute confusion. Le site français de Google, très aéré et très sobre, axe bien sur la recherche. Le site français de Yahoo donne l'impression de trop plein d'informations ... mais n'est-ce pas la marque d'un annuaire bien fourni ?





4 Exercices immédiats

Dans quelle mesure une mise en page au moyen d'une grille améliore-t-elle la présentation d'un site web ?

Quelles techniques utiliser pour structurer une page web ?

Expliquer la différence entre espace libre actif et espace libre passif.

Quelle est l'erreur courante en matière de test de site ?

Pour quelle raison un visiteur quitte-t-il un site ?

VI.2 Conception adaptée au public

Objectif : connaître son public et savoir se mettre à sa disposition.

1 Etre en phase avec le visiteur

Le travail conceptuel doit avant tout reposer sur le visiteur : qui est-il, qu'attend-il de votre site ?

Combien y aura-t-il de visiteurs ? Connaître son public, c'est résoudre une partie des problèmes que peut poser un site.

Quand on connaît le profil des visiteurs, quand on peut répondre à la question "le visiteur viendra-t-il une seule fois sur le site ou y reviendra-t-il plusieurs fois ?", on va pouvoir anticiper l'interaction avec le visiteur, lui fournir rapidement l'information qu'il recherche (pas de pages superflues dès l'accueil). A partir de ceci, on va pouvoir décrire les pages web spécifiques à ses besoins en proposant un jeu pertinent de rubriques, des hyperliens efficaces et bien en évidence.

2 Les différents types de sites (liste non exhaustive)

Les sites institutionnels sont avant tout informatifs.

Objectifs : développement de l'image de l'institution, communication, recrutement de talents.

Les sites marchands sont avant tout transactionnels.

Objectifs : vendre, mais aussi fidéliser.

Les sites communautaires proposent contenus et services créés par leurs membres.

Objectifs : fédérer des ressources humaines autour de thèmes spécifiques, générer des échanges.

Les intranets offrent en interne des fonctionnalités de travail collaboratif.

Objectifs : augmenter la compétitivité, accélérer la circulation de l'information, personnaliser l'accès aux informations

3 Ecrire pour le web

Le type d'expression n'est pas uniforme . Si on parle à un teenager ou à un professionnel expérimenté, la charte éditoriale (le vocabulaire, le contenu) n'est pas la même comme pour un magazine.

Le web ne se prête pas aussi bien à la lecture que le papier.

Ecrire pour le web est donc de faire en sorte que le visiteur "tombe" rapidement sur le contenu qui l'intéresse et qu'il comprenne très vite qu'il l'a effectivement trouvé ! D'où l'importance d'un bon titrage / sous-titrage (et de l'utilisation des balises <h>) et d'un bon texte introductif avec dès le début des mots-clés bien en évidence (de plus les mots-clés augmentent la pertinence du site dans le classement des moteurs de recherche).

Voici quelques axes essentiels d'écriture sur le web :

- faire ressortir les mots-clés (gras, couleur, lien)
- rédiger de manière factuelle (on va directement au fait : pas de littérature, de l'information !)
- choisir de bons titres
- mettre une introduction identifiant le contenu
- utiliser des listes à puces ou des tableaux

Ce qui implique :

- un titre pertinent pour l'article bien lisible et dégagé et des sous-titres aidant à la compréhension du contenu
- **une introduction / conclusion** pour captiver l'attention du visiteur dès le départ (technique de la pyramide inversée : **conclusion, détail et information de fond (sic!)**).
- un texte plutôt court (2 fois plus que sur papier) sans oublier une version imprimable plus consistante
- des paragraphes courts, séparés par des sous-titres, ne contenant chacun qu'une seule idée
- une mise en page évitant fonds sombres et textes clairs avec une police lisible (arial, verdana, pas trop d'italiques).
- de bons liens, en cours de texte, ou mieux à la fin de la partie concernée

[CF www.e-gineer.com, www.contentious.com,
www.prezenz.com/f/web/reportz_web_webwriting_f.html]

VI.3 Conception adaptée à l'écran

Objectif : savoir organiser ses écrans

1 Un écran n'est pas une page papier

L'espace d'affichage d'un écran d'ordinateur est très différent d'un support imprimé. Il faut bien garder ceci à l'esprit quand la métaphore de la page imprimée est omniprésente.

Il est d'abord, comme les écrans de télévision, **au format "paysage"** et non au format "portrait" comme sur papier et votre écriture devra s'y adapter. Mais surtout, le papier reflète la lumière tandis qu'un écran d'ordinateur est **rétro-éclairé**. Ceci a une forte influence sur les couleurs et les contrastes où les possibilités à l'écran sont bien plus nombreuses que sur papier. Se rappeler qu'une partie importante des visiteurs a des difficultés visuelles plus ou moins marquées : le texte devra bien se détacher du fond.

Pour finir, la résolution d'un écran est bien inférieure à celle d'une page imprimée. Des graphismes peuvent en ressortir "granuleux" (un tirage laser se fait à 600 ppi, une résolution moyenne d'ordinateur est 72ppi !).

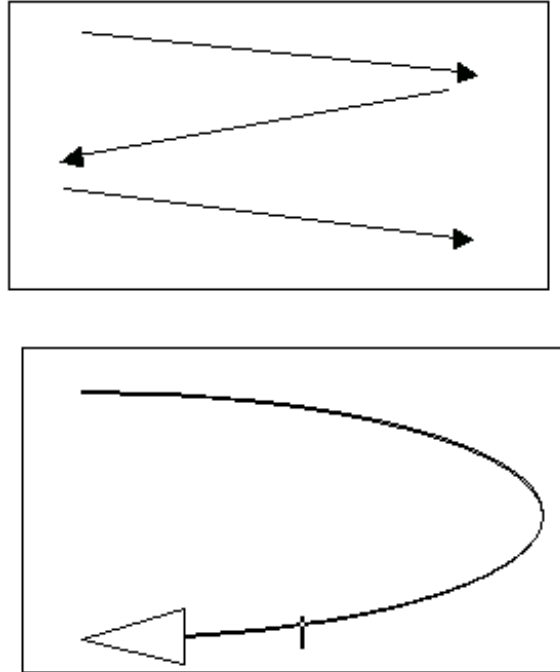
2 Les zones de l'écran

S'il n'est pas toujours facile de prévoir le parcours du visiteur, il existe un consensus quant à l'importance relative des différentes zones de l'écran (le haut, le bas, les colonnes droite et gauche et la partie centrale).

L'usage naturel est de lire de gauche à droite avant de revenir à gauche, mais ce qui attire l'œil, c'est avant tout la partie centrale surtout si elle est bien dégagée. Ensuite et par ordre d'importance décroissante, viennent la partie haute, la partie droite, celle du bas et celle de gauche.

Un autre usage courant est de placer le sommaire dans la colonne de gauche et un (sous-) menu horizontal en haut.

Des statistiques sur les mouvements oculaires ont fait apparaître deux habitudes de lecture en ligne :



Approche d'une lecture 4/3 comme à la télévision

3 Les possibilités de déplacement

Les visiteurs doivent pour accéder à ce qu'ils cherchent facilement et rapidement : ils ne doivent pas se perdre dans le dédale des pages

Aussi, pour accélérer leur quête, le concepteur

- doit adopter une structure de pages organisée en **rubriques** avec une navigation claire
- faire un **plan** du site (affichant toutes les pages et les rubriques qui les contiennent), surtout si le site est complexe,
- afficher une même barre ou colonne de navigation sur toutes les pages sans offrir trop de rubriques,
- utiliser des liens hypertexte pour une approche plus sémantique, moins linéaire.
- idéalement, pouvoir atteindre chacune des pages en 2 ou 3 clics.

Plus il y a de liens hypertexte, plus la navigation sera rapide. Les placer dans le corps du texte est très efficace, mais s'il y en a beaucoup vaut mieux en faire une table.

4 Le contenu d'une page

Une page web, autre qu'un annuaire, ne doit pas contenir trop d'informations. Un visiteur a une capacité cognitive qu'il ne peut dépasser sans trop de perte d'information. L'habileté consiste à le guider sémantiquement (par des indices) dans un contenu bien structuré.

5 Exercices immédiats

Citer des avantages liés à un bon positionnement des éléments de navigation.

Expliquer pourquoi "Cliquez" ici est un lien textuel peu efficace.

Sur le web , choisir un site et

- 1) indiquer les éléments qui unifient les pages en parcourant le site
- 2) montrer les espaces libres, actifs et passifs
- 3) indiquer des éléments qui vous semblent incohérents et émettez des recommandations
- 4) Imprimer une page et faites vos commentaires sur l'impression

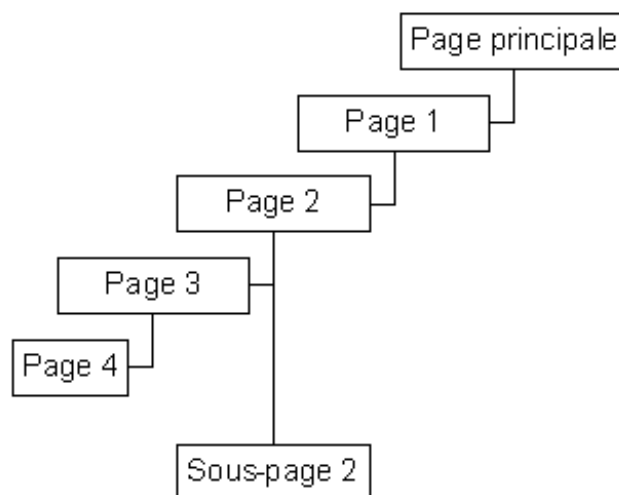
VI.4 La structure du site web

Objectif : savoir créer un organigramme de la structure et de la logique qui sous-tendent la présentation d'un contenu web.

1 Les écrans

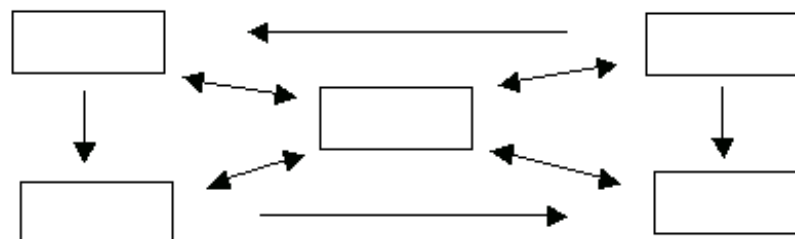
Pour créer votre site web, vous allez avoir à en dessiner les principaux écrans, puis en construire l'enchaînement (expliquer comment on passe d'un écran à l'autre), c'est-à-dire définir une navigation (détaillée plus loin). Ceci peut se représenter par une structure. Nous en présentons quelques modèles dans ce qui suit.

2 Structure linéaire



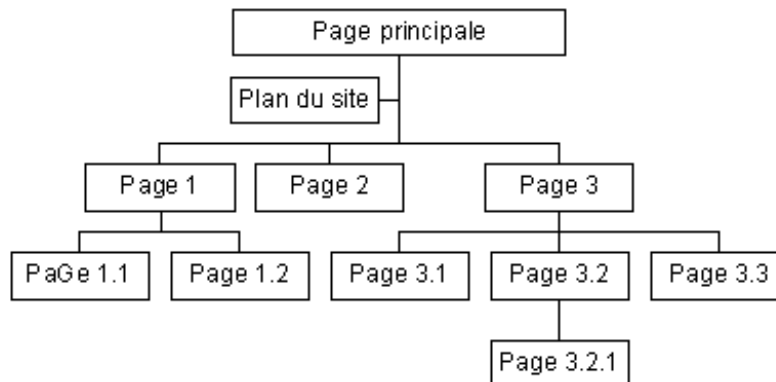
Chaque page peut aller à la suivante et revenir vers la précédente. Elle peut disposer en plus d'un lien direct vers la page principale ainsi que de liens vers des sous-pages.

3 Structure « web »



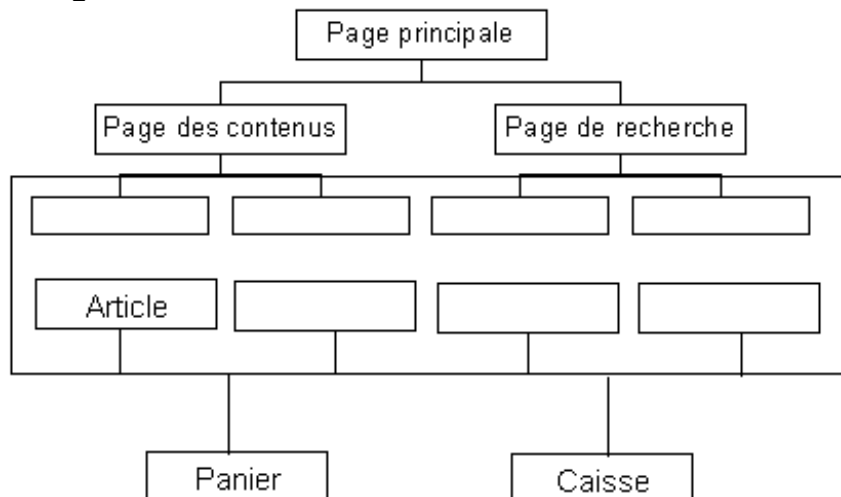
Le visiteur peut passer librement d'une page à une autre. Cette structure est valable pour des petits sites ou des portions de sites si on indique clairement au visiteur sa position courante et ses possibilités de déplacement.

4 Structure hiérarchique



C'est la structure la plus répandue. Dans les sections verticales (Page 1 et descendants, ...), la navigation est linéaire. Chaque page comporte une barre de navigation donnant accès aux autres pages de la section, à la page principale ou encore au plan du site.

5 Structure en catalogue



Cette structure convient aux sites de commerces électroniques. Le visiteur navigue dans les articles. A mesure qu'il achète les articles sont rajoutés au panier, puis passe à la caisse. Cette structure implique un traitement transactionnel d'arrière plan.

VI.5 La navigation sur le site

Objectif : savoir prévoir les options de navigation au sein de la structure informationnelle retenue.

1 Une navigation exploitable

Les hyperliens offrent d'énormes possibilités de navigation ... mais aussi de se perdre !

Si l'internaute doit accéder à un site extérieur au votre, il doit pouvoir le faire sans perturber son sens de l'orientation au sein du votre (*se souvenir qu'une url sur deux n'existe plus au bout de 27 mois*).

Au sein de votre site, l'internaute doit pouvoir se situer facilement. Il doit pouvoir répondre aux questions suivantes très facilement :

- Où suis-je ?
- Où puis-aller ? Comment
- Comment revenir au point de départ
- Savoir quelle est la page courante et avoir une idée claire de son contenu.

Il doit donc disposer à la fois de liens explicites et cohérents et de solution de substitution aux boutons du navigateur. Ces liens peuvent être textuels ou graphiques.

2 Une navigation textuelle

Les liens textuels offrent un support de navigation efficace s'il n'y a pas surcharges d'informations. Ils peuvent être positionnés dans le corps du texte (**lien contextuel** pour une explication ou une association d'idée) ou encore dans des **menus** (type sommaire à gauche ou barre horizontale). Il semble, pour un menu, que trois niveaux ne soient pas à dépasser.

Dans une barre de navigation ou un sommaire, la page courante doit être indiquée (surlignage).

Si le site est vaste, pourquoi ne pas faire une page web contenant la "table des matières" de façon à se rendre directement à l'information désirée.

3 Une navigation graphique

En plus des liens textuels, on peut ajouter dans ses pages web des signaux graphiques de navigation (des **flèches** par exemple), mais alors toujours les mêmes (par cohérence et pour limiter le temps de chargement).

On peut utiliser des images plus élaborées (des cartes par exemple) que l'on découpe (à la façon d'un puzzle) pour en faire des zones cliquables.

Yahoo (voir ci-dessous) utilise même du texte graphique ! Ce peut être plus explicite qu'un graphique seul (icône)



Comment organiser simplement la navigation de la structure hiérarchique proposée au point 4 ci-dessus ?

VI.6 Conventions pratiques : noms de fichiers, url, arborescence des répertoires

Objectif : savoir choisir, de façon appropriée, une convention de dénomination.

1 Nom de fichiers

On développe son site sur son propre poste, « **en local** », dans un répertoire de travail.

Quand on le publie, c'est sur un serveur web qui n'aura pas forcément le même système d'exploitation. Dans ce cas, le transfert de fichiers peut générer des erreurs de liens et de noms de fichiers.

D'un système d'exploitation à l'autre, les conventions de dénomination de fichiers changent. Par exemple, Linux respecte la casse mais pas Windows ou MacIntosh. Ce qu'on fait en général, c'est utiliser systématiquement des minuscules dans les noms de fichiers.

Certains caractères présentent des incompatibilités : Windows n'accepte pas \, < et >, MacIntosh n'accepte pas le point-virgule et Linux le slash et l'espace. Vaut mieux donc ne pas utiliser tous ces caractères.

Chaque site web dispose d'une page d'accueil : elle est repérée par un slash final dans l'url. Si certains serveurs acceptent d'autres noms pour cette page, vaut mieux la nommer systématiquement index.htm ou index.html .

2 url

L'utilisation des url peut s'avérer délicates. Elles peuvent être comme on l'a vu **absolues**, c'est-à-dire avec le protocole utilisé, le domaine, le chemin et le nom du fichier ou **relatives**, c'est-à-dire faisant l'impasse sur le protocole et le nom de domaine. Par exemple, un fichier enregistré dans le même répertoire que le fichier appelant n'est appelé que par son nom seul.

Remarque : lorsqu'on navigue sur le web, la saisie du protocole n'est pas nécessaire (il y en a un par défaut). Dans du code, les url absolues doivent systématiquement l'intégrer.

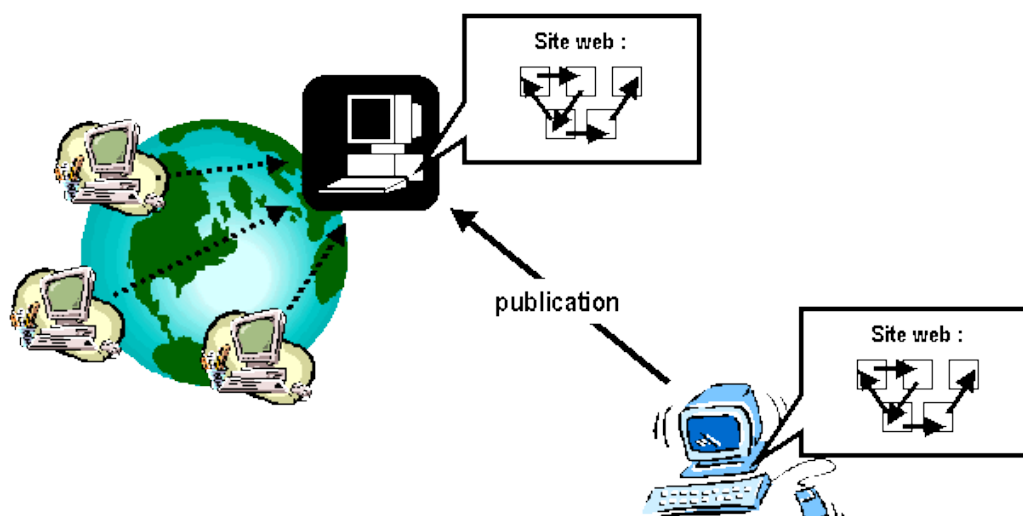
3 Un bon nom de domaine

Le serveur web qui va recevoir les fichiers de votre site va avoir à les ranger dans ses répertoires utilisateurs et vous aurez comme adresse `www/serveur/utilisateurlambda`, ce qui n'est pas très « alléchant » pour un site commercial. Le mieux alors est créer directement un alias ou acheter un nom de domaine qui va pointer vers votre emplacement réel sur le serveur web.

4 Contraintes professionnelles, éthique

Les pages que vous publiez vous représentent. Non seulement, elles doivent respecter tous les dispositifs légaux de publication, mais aussi votre statut du moment, à savoir étudiant. Si vous publiez sur le serveur de l'UFR, vous ne pouvez pas faire du commerce ou de la publicité.

Le non respect de ces principes de base peut entraîner le retrait sans préavis de vos pages du serveur.



VI.7 Publication d'un site web

Objectif : savoir publier un site web, c'est-à-dire « l'installer » sur un serveur web ... et le maintenir.

1 Publier un site

Publier son site web, c'est le mettre à la portée de tous les internautes sur un serveur du réseau internet, c'est-à-dire trouver un fournisseur d'accès qui assurera le service d'hébergement. Ce sera soit l'UFR Mathématiques et Informatique elle-même, soit un des hébergeurs gratuits du marché (voila, free, ...)

2 Publier à partir de son dossier universitaire « public_html »

L'UFR Mathématiques et Informatique a son propre serveur web. Pour publier via ce serveur, il suffit :

- de créer un répertoire `public_html` dans son compte personnel,

- de lui donner de bons droits d'accès pour que tout un chacun puisse y accéder via l'internet (sous Linux, on se met en mode console et on écrit `chmod 755 public_html` ; sous windows, on ouvre une fenêtre de commandes et on lance la même commande.
- d'y déposer les fichiers web produits à l'aide d'un copier/coller.

Remarque : les fichiers web ne sont pas déposés sur le serveur web de l'UFR, mais par un dispositif maison on peut y accéder via l'internet : www.ens.math-info.univ-paris5.fr/~hx00000.

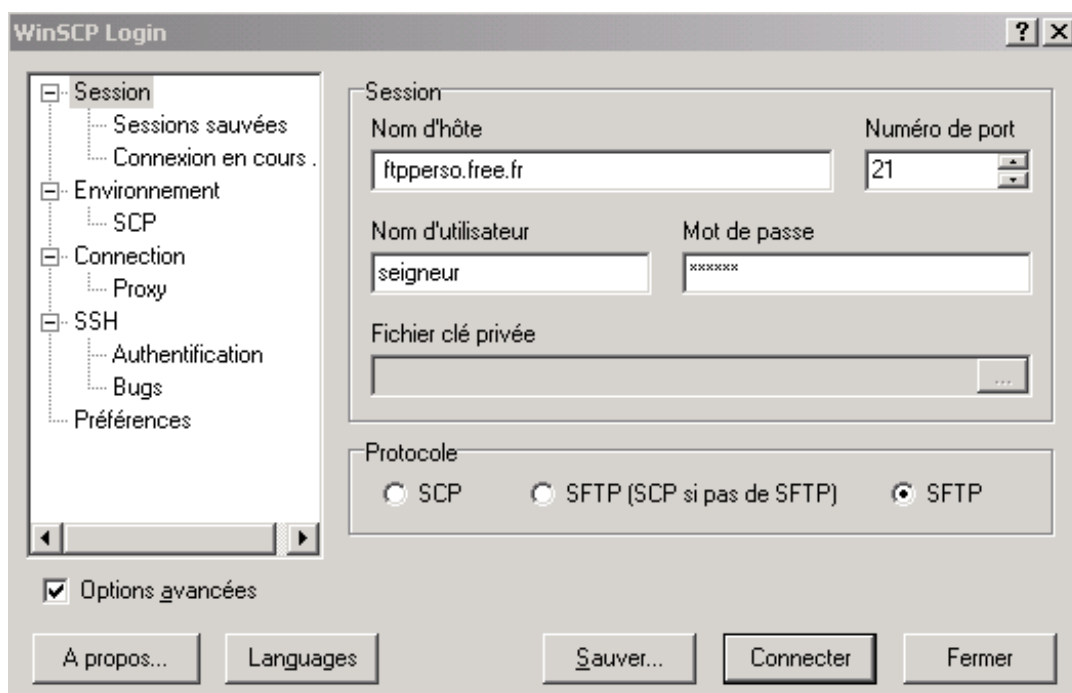
3 Publier chez un hébergeur

On décrit ici la démarche pour ouvrir un espace d'hébergement gratuit sur le web. Il existe différentes solutions et des acteurs offrant des services similaires (100 Mo pour des pages personnelles).

La première étape consiste à vous créer un compte chez l'hébergeur : il vous faut une adresse email et saisir de nouveaux login et mot de passe. Ceci va permettre de créer un accès FTP et un répertoire sur le serveur web. En général, ceci nécessite 24 heures.

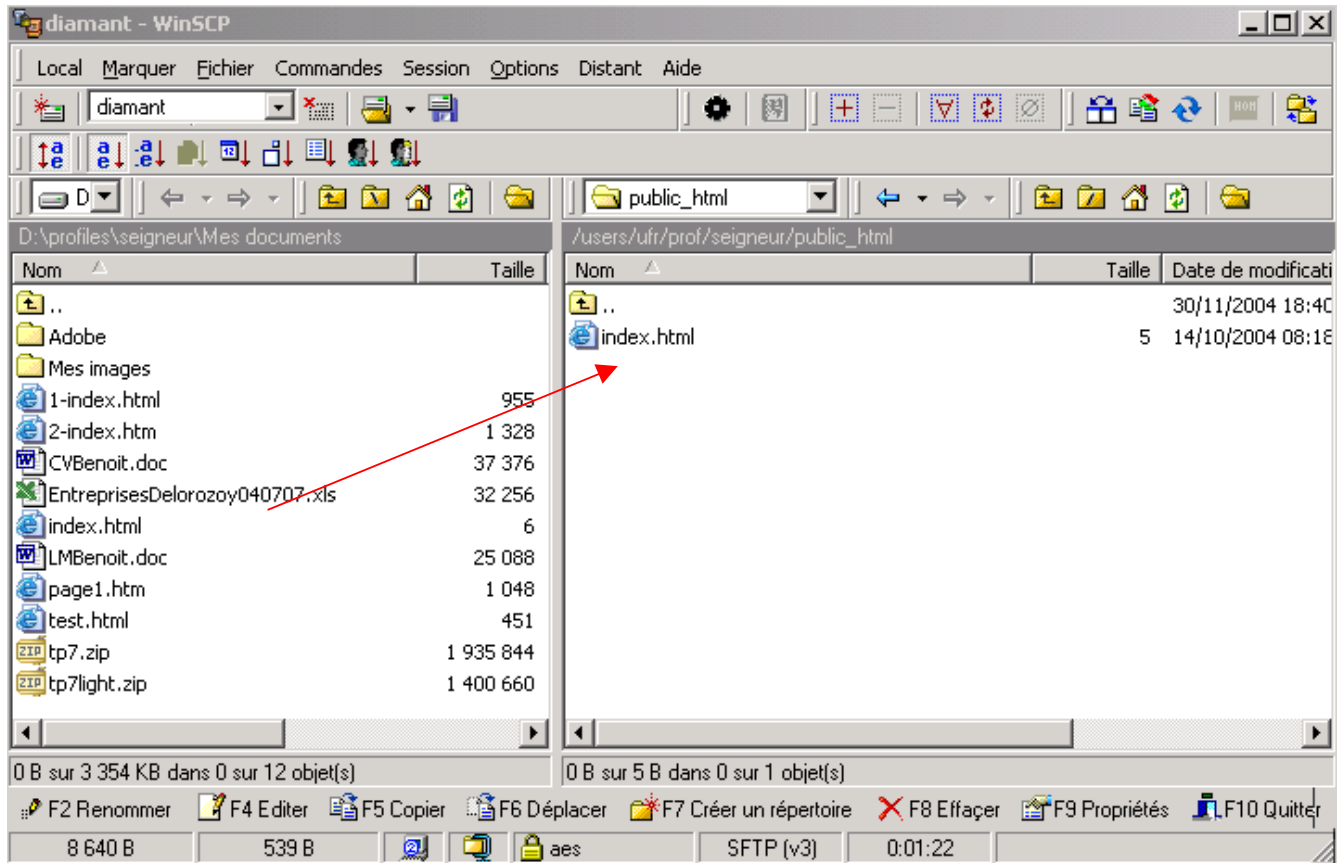
Par la suite, pour déposer vos pages web, vous allez utiliser un logiciel FTP (Filezilla, WinSCP3 sous windows ou gftp sous Linux).

Exemple avec WinSCP3, mais ceci se présente de la même façon avec les autres logiciels de transfert de fichier



On indique le nom d'hôte distant, le numéro de port (donné en général, 21 ou 22), votre nom d'utilisateur (login) et votre mot de passe. Puis on clique sur le bouton Connecter

Une fois connecté en FTP, vous devez simplement faire glisser vos pages dans la partie de droite qui est l'arborescence de votre espace web.



Après cette opération vous pourrez visiter votre site à l'adresse : http://login*.hebergeur.fr

4 Tests

Les tests effectués tout au long du développement doivent se poursuivre en vraie grandeur après la mise sur le serveur web : ils sont à faire de préférence avant de faire connaître votre url !

Ils doivent être faits dans différents environnements :

- sur plusieurs navigateurs
- sur les systèmes d'exploitation principaux : Linux, Windows et Mac
- à différents débits de connexion (une bonne partie des internautes quittent un site web si la première page n'est pas téléchargée dans les 8 - 10 secondes !)
- sur les résolutions d'écran et les profondeurs chromatiques les plus répandues (8, 16 ou 24 bits)

5 Edition du site web

Mettre un site web à disposition, c'est offrir des informations régulièrement tenues à jour et optimisées. Quand on fait des modifications, comment les répercuter ?

Le problème est qu'un logiciel FTP est incapable de comparer le contenu du répertoire local qui contient les fichiers modifiés et que l'on désire publier, et le répertoire distant, sur le serveur, qui n'est pas à jour, pour en déduire quels fichiers doivent être transférés, en écrasant l'ancienne version, et quels sont ceux qui doivent éventuellement être effacés.

Il faut donc se souvenir de ce que l'on vient de modifier, le sélectionner à la main, se connecter sur le serveur, naviguer dans ses fichiers pour mettre son travail au bon endroit, ou tout transférer à chaque fois, ce qui va consommer de la bande passante.

VI.8 Référencement

Objectif : améliorer la visibilité de vos pages grâce aux techniques de référencement.

1 Rappel sur les moteurs de recherche et annuaires

Quand on recherche des informations sur Internet, on a recours à des sites spécialisés qui permettent d'effectuer cette recherche. Il existe deux types de système de recherche sur Internet : les **moteurs de recherche** et les **annuaires**.

Ces deux systèmes sont tout à fait complémentaires pour leurs utilisateurs, mais sont dans le fond radicalement différents.

Un moteur de recherche est un système de recherche automatisé, c'est à dire sans aucune intervention humaine. Un "robot" se charge à longueur de journée de parcourir le web pour visiter des pages et les ajouter dans sa base de données. Le robot analyse le contenu des pages pour les répertorier et les classer par ordre d'intérêt dans sa base de données. Ce système permet de prendre en compte une grande part du contenu du web, mais le plus gros problème reste la pertinence souvent discutable des résultats obtenus et l'aspect démesuré des données à trier.

Un annuaire, par contre, comporte moins de sites. Ceux ci sont ajoutés "manuellement" par les responsables des différents sites et annuaires. Les annuaires répertorient donc beaucoup moins de pages, mais la qualité des requêtes est souvent meilleure. Les sites sont classés dans plusieurs catégories que l'utilisateur peut parcourir rapidement grâce à un système de sélection de catégories qui permet d'affiner la recherche.

Il existe enfin des méta-moteurs. Ce sont des moteurs de recherche qui génèrent des requêtes sur d'autres moteurs de recherche et rassemblent les réponses qui leur semblent les plus pertinentes. Les méta-moteurs ne possèdent pas de bases de données propres, il est donc impossible de référencer un site dans ce type d'outil de recherche.

2 Préparation préalable des pages web

Cette étape est la phase décisive du référencement d'un site : elle consiste à préparer la page par l'insertion dans l'en-tête (head) de balises "méta" qui ne seront pas vues par l'utilisateur et qui comportent les différentes propriétés de la page.

Elles comportent en général deux attributs "*name*", dont la valeur détermine le type de l'information et "*content*", dont la valeur correspond à cette information. Ces balises meta sont principalement :

- **Description** : Contient la description du contenu de la page. Cette description doit contenir moins de 200 caractères. C'est cette description qui sera généralement utilisée comme résumé dans les moteurs de recherche.
- **Keywords** : Ce sont les mots clés de la page. Ceux ci sont séparés par des virgules. Ils doivent contenir moins de 1000 caractères.
- **Author** : Auteur de la page.
- **Copyright** : Personne morale ou physique détentrice des droits de reproduction de la page.
- **Generator** : Logiciel qui a permis de générer la page.

Il existe d'autres types de balises méta qui permettent de donner des informations au navigateur de l'internaute. Ces balises peuvent tout de même être analysés par les robots. Elles n'ont pas d'attribut "*name*", celui ci est remplacé par l'attribut "*http-equiv*".

La principale est **Content-Type** qui spécifie le jeu de caractères utilisé.

Exemple d'en-tête d'une page avec des balises meta :

<head>

```
<meta http-equiv="content-type" content="text/html; charset=iso-8859-15" />
<title>Cours d'HTML</title>
<meta name="Description" content="Conception et publication de pages web />
<meta name="keywords" content="HTML, balises, frames, formulaires, tableaux, images, web,
référencement, hébergement, medias, sites, internet, web" />
```

</head>

Bien remplir ces balises, c'est faire que les mots clés se retrouvent dans la description et dans le titre des pages. Le mieux est de les choisir "efficaces" pour le référencement et d'en limiter le nombre pour qu'ils aient du poids

3 Le référencement du site

Une fois que les pages ont été réalisées et mises en ligne, il faut référencer le site. Pour ceci, deux solutions : soit référencer les sites en visitant chaque moteur de recherche, soit utiliser un site spécialisé qui permet de référencer automatiquement le site dans les principaux moteurs de recherche.

Le référencement moteur de recherche par moteur de recherche est beaucoup plus efficace que le référencement automatique. Cependant, il peut être intéressant, mais c'est payant, d'effectuer au moins un référencement automatique afin d'être référencé dans les moteurs de recherches les moins connus, mais peut-être plus spécialisés.

Remarques :

- pour référencer un site, il est préférable de soumettre l'adresse complète de la page <http://www.exemple.com/index.html> plutôt que <http://www.exemple.com/>.
- De plus, il est inutile de référencer toutes les pages. Seule la première page est à référencer : le moteur de recherche va suivre l'ensemble des liens à partir de l'URL soumise.
- Quand un site est référencé dans un moteur de recherche, il est inutile de référencer à nouveau le site lorsqu'il a subi un changement. En effet, les moteurs de recherche mettent régulièrement à jour leur base de données.

Pour le référencement dans les annuaires, il faut préparer au préalable une description du site ainsi qu'une liste de mots clés. Une adresse électronique valide peut également être demandée. Il faut noter que les annuaires étant normalement mis à jours par des personnes physiques, le temps avant que votre site ne soit testé et ajouté à la base peut être relativement long. Les annuaires peuvent également refuser votre site parce qu'ils jugent que votre site ne correspond pas à leur attentes, tant sur le plan qualitatif que sur la nature du contenu.

Exemple de référencement sur Yahoo.

Suggérer un site à Yahoo, c'est-à-dire l'insérer dans le guide web de Yahoo, est organisé en trois étapes.

1) vérifier que le site n'est pas déjà répertorié dans Yahoo France

2) trouver la rubrique la plus appropriée parmi les 14 rubriques génériques :

- Actualités et médias
- Art et culture
- Commerce et économie
- Divertissement
- Enseignement et formation
- Classement géographique
- Informatique et Internet
- Institutions et politique
- Références et annuaires
- Santé
- Sciences et technologies
- Sciences humaines
- Société
- Sports et loisirs
-

Lorsque vous suggérez un site, soyez aussi précis que possible. Parcourez la base en profondeur pour trouver une sous-rubrique appropriée. Vous ne pouvez soumettre votre site dans une rubrique principale. Si vous soumettez un lien dans une rubrique principale ou dans une rubrique trop générique, vous recevrez certainement un message d'erreur vous demandant de recommencer. Examinez donc la rubrique que vous avez choisie : y a-t-il des sites listés à ce niveau ? Cette rubrique

est-elle divisée en sous-rubriques dont l'une serait susceptible d'accueillir votre site ? Allez bien en profondeur

3) **Proposer** le site

Une fois choisie la rubrique qui vous semble la plus appropriée, cliquez sur le lien « **Proposer un site** » à droite de la page de cette rubrique. Rappelez-vous aussi qu'il s'agit d'une **suggestion** : l'emplacement final de votre site sera déterminé par les surfeurs de Yahoo! France

4 **Astuces**

Une astuce qui marche est de créer une communauté de sites. Le principe est de mettre un lien vers d'autres sites qui, en échange, mettent un lien vers le site. Ceci s'appelle faire des échanges de liens. Ainsi, quand un moteur de recherche va visiter l'un des sites de la communauté, il va, par la même occasion, et en suivant les liens, visiter et référencer les autres sites.

Cette dernière technique fonctionne relativement bien avec des moteurs de recherche de type "Google" qui évaluent la popularité d'un site en comptant le nombre de sites qui pointe vers lui.

VI.9 **Problème / atelier**

Le projet final de cette formation est de réaliser un site web "perso" en utilisant le plus de technologies apprises, de le publier dans le public_html de l'UFR.

L'étudiant, avant de commencer ce projet (qui fait fonction d'examen final), devra rédiger le cahier de charges en plusieurs parties comme décrit en annexe et le faire parvenir aux chargés de travaux dirigés.

Ce site web est **personnel**, ce n'est pas un travail d'équipe. Il doit refléter votre personnalité, votre position dans la société et informer sur votre projet professionnel (vous pouvez proposer votre CV sous forme européenne). Il doit être bien structuré, clair et agréable à regarder (le graphisme et plus généralement l'esthétique seront appréciés).

Une version bilingue est la bienvenue.

Annexes

1 Repères chronologiques

1945 Vannevar Bush décrit un appareil, le MEMEX, capable de restituer toutes sortes d'informations stockées sur des micro-films. Plus fort est le fait que le MEMEX devait être capable de créer des **liens** entre les différents documents.

1968 C'est la guerre froide : les années 60 ont vu les crises de Cuba, de Prague, le processus de décolonisation, l'arrivée de nouvelles puissances nucléaires. Le département de la défense américain cherche à améliorer la fiabilité des transmissions militaires. En cas d'attaque nucléaire, il ne peut accepter que la destruction d'une partie de son réseau bloque complètement ses transmissions. Aussi réagit-il : une structure, l'ARPA met en place le premier réseau de transmission par paquets (Arpanet) avec quatre nœuds situés dans de grandes universités américaines. En même temps sont développés des protocoles pour l'établissement de connexions logiques avec des machines distantes (Host to Host protocol) par l'intermédiaire de routeurs.

1972 Les premiers services disponibles sur ce réseau : Telnet et FTP.

1974 Les bases de l'architecture TCP / IP avec comme objectif l'indépendance des liaisons par rapport aux lignes physiques, aux réseaux eux-mêmes, aux postes hôtes

Fin des années 1970 Les premiers micro-ordinateurs ne sont que l'assemblage d'un micro-processeur à quelques MHz, d'une RAM de quelques Koctets et d'une faible mémoire de stockage.

1980 Le projet Xanadu (Nelson) se propose de créer un environnement pour stocker toute la production mondiale (de tous les médias). Tous ces documents sont reliés entre eux par des liens : Nelson est considéré comme l'inventeur de mots hypertexte et hypermédia.

1982 Les premiers PC : informatique personnelle et clones.

1983 Séparation du sous réseau militaire : (Milnet). Naissance d'internet
Naissance des grands « domaines » américains : edu, org, net, com, gov.

1984 Fédération des réseaux avec les premières versions du protocole IP : l'internet est accessible en France avec la connexion du réseau Transpac.

1986 La vague Minitel ... et les premières messageries.
Création du métalangage SGML

1988 Les premiers réseaux numériques à intégration de service (RNIS, Numeris en France)

1989 *L'idée du World Wide Web est lancée par Tim Berners Lee du CERN. Il s'agit de mettre en consultation sur un ordinateur distant des fichiers textuels à l'aide d'un navigateur. Le projet WWW doit son succès à trois technologies : HTTP (asynchrone), les langages de balisage (les "...ML" à vocation d'universalité), l'architecture client-serveur.*

1990 Le langage de requête SQL : démarrage du client serveur
Premier micro-processeur Intel pour les portables : les tout premiers pas du nomadisme.
Naissance de Linux.

1992 Début de la guerre du logiciel : l'empire Microsoft attaque ! (Win 95, Win NT).

1995 Le réseau sans fil naît en France dès la libération de la bande hertzienne des 2,4 GHz.

1996 Java arrive avec sa drôle de machine virtuelle : unification des postes de travail autour du couple machine virtuelle / OS.

1998 Ibm passe à l'Open source
Première recommandation du W3C pour XML.

1999 Les start-up essaient. La netéconomie arrive.

Le potentiel technologique de la fin des années 70 nous fait sourire aujourd'hui surtout quand on entend un futurologue dire : « En 2030, une capacité de calcul d'un dollar aura la même performance que celle d'un cerveau humain » ! (.... le sourire disparaît).

2 Modèle européen de Curriculum Vitae

Informations personnelles

Nom (noms et prénoms)
 Adresse (No, rue, Code postal, Ville, Pays)
 Téléphone
 Télécopie
 Courrier électronique
 Nationalité
 Date de naissance (jour, mois , année)

Informations professionnelles

*Décrire successivement chaque expérience professionnelle
 en commençant par la plus récente (Date de ... à ...)*

Nom et adresse de l'employeur
 Type ou secteur d'activité
 Fonction ou poste occupé
 Principales activités et responsabilités

Education et formation

*Décrire séparément chaque programme d'enseignement ou de formation
 en commençant par le plus récent (Date de ... à ...)*

Nom et type de l'établissement dispensant l'enseignement ou la formation
 Principales matières ou compétences professionnelles
 Intitulé du certificat ou du diplôme délivré
 Niveau dans la classification nationale (le cas échéant)

Aptitudes et compétences personnelles

acquises au cours de votre vie et de votre carrière mais pas nécessairement validées par des certificats et diplômes officiels.

Langue maternelle

Précisez votre langue maternelle.

Autres langues

Précisez la langue
 Lecture > Indiquez votre niveau: excellent, bon, élémentaire.
 Ecriture > Indiquez votre niveau: excellent, bon, élémentaire.
 Expression orale > Indiquez votre niveau: excellent, bon, élémentaire.

Aptitudes et compétences sociales

Vivre et travailler avec d'autres personnes, dans des environnements multiculturels, à des postes où la communication est importante et dans des situations où le travail d'équipe est essentiel (activités culturelles et sportives par exemple), etc.
 Décrivez ces compétences et indiquez dans quel contexte vous les avez acquises.

Aptitudes et compétences organisationnelles

Coordination et gestion de personnes, de projets, de budgets; au travail, en bénévolat (activités culturelles et sportives par exemple) et à la maison, etc.
 > Décrivez ces compétences et indiquez dans quel contexte vous les avez acquises.

Aptitudes et compétences techniques

Liées à l'informatique, à des types spécifiques d'équipement, de machines, etc.
 Décrivez ces compétences et indiquez dans quel contexte vous les avez acquises.

Aptitudes et compétences artistiques

Musique, écriture, dessin, etc.

Décrivez ces compétences et indiquez dans quel contexte vous les avez acquises.

Autres aptitudes et compétences

Non mentionnées précédemment.

Décrivez ces compétences et indiquez dans quel contexte vous les avez acquises.

Permis de conduireInformation complémentaire

Indiquez ici toute autre information utile, par exemple nom de personnes de contact, références, etc.

Annexes

Enumérez les pièces jointes au CV le cas échéant.

En anglais

PERSONAL INFORMATION

Name : SURNAME, other name(s)

Address : House number, street name, postcode, city, country

Telephone

Fax

E-mail

Nationality

Date of birth : Day, month, year

WORK EXPERIENCE

Dates (from - to)

> Add separate entries for each relevant post occupied, starting with the most recent.

Name and address of employer

Type of business or sector

Occupation or position held

Main activities and responsibilities

EDUCATION AND TRAINING

Dates (from - to)

> Add separate entries for each relevant course you have completed, starting with the most recent.

Name and type of organization providing education and training

Principal subjects/occupational skills covered

Title of qualification awarded

Level in national classification (if appropriate)

PERSONAL SKILLS AND COMPETENCES

Acquired in the course of life and career but not necessarily covered by formal certificates and diplomas.

MOTHER TONGUE

> Specify mother tongue

OTHER LANGUAGES

> Specify language

Reading skills > Indicate level: excellent, good, basic.

Writing skills > Indicate level: excellent, good, basic.

Verbal skills > Indicate level: excellent, good, basic.

SOCIAL SKILLS AND COMPETENCES

Living and working with other people, in multicultural environments, in positions where communication is important and situations where teamwork is essential (for example culture and sports), etc.

> Describe these competences and indicate where they were acquired.

ORGANISATIONAL SKILLS AND COMPETENCES

Coordination and administration of people, projects and budgets; at work, in voluntary work (for example culture and sports) and at home, etc.

> Describe these competences and indicate where they were acquired.

TECHNICAL SKILLS AND COMPETENCES

With computers, specific kinds of equipment, machinery, etc.

> Describe these competences and indicate where they were acquired.

ARTISTIC SKILLS AND COMPETENCES

Music, writing, design, etc.

> Describe these competences and indicate where they were acquired.

OTHER SKILLS AND COMPETENCES

Competences not mentioned above.

> Describe these competences and indicate where they were acquired.

DRIVING LICENCE(S)

ADDITIONAL INFORMATION

> Include here any other information that may be relevant, for example contact persons, references, etc.

ANNEXES

> List any attached annexes.

3 Comment réaliser succinctement un cahier des charges pour un site web

*Le cahier de charges est l'élément déclencheur pour commencer la conception d'un site **en se posant de la façon la plus exhaustive les bonnes questions***

- *pour analyser les **besoins** et les **contraintes**,*
- *pour envisager les différents aspects : l'arborescence, la navigation, la clarté des pages, la lisibilité typologique, l'image que veut refléter le site, le choix de la charte graphique ...*

A CONTEXTE

Description globale du projet

Qui est le client ?

Cadre de développement de votre travail (pour telle entreprise,)

Qui êtes-vous ?

Cadre de votre intervention

Périmètre du projet : ce qu'il touche, ce qu'il ne touche pas.

Projet à part entière ou sous-projet ? Dans le second cas, comment va se faire l'intégration ?

Existant

Sites web, hébergement

Documents papier ou informatique

Personnes ayant déjà travaillé sur le sujet

Qui est la maîtrise d'ouvrage, la maîtrise d'oeuvre ?

Quels sont les autres ressources d'expertises ?

Fournir les contacts de tous les intervenants

B OBJECTIF PRINCIPAL (et éventuellement secondaires)

Pourquoi un nouveau site ? *en une ou deux lignes*

Objectifs secondaires à donner si très différenciés de l'objectif principal

C PUBLIC VISE

Cette partie est primordiale : le site est fait pour lui !

En l'occurrence, étudiants, candidats, parents ?

Où se trouve ce public ? Important en matière de formation (rôle du présentiel)

Message à faire passer ? Perspectives ?

D SPECIFICATIONS FONCTIONNELLES

Qu'est-ce que le site va offrir (fonctionnalités, services, informations, ...) ?

C'est ce qu'explique un document : la maîtrise d'ouvrage exprime son besoin pour le projet formulé en termes de fonctionnalités et de contraintes.

Relatant les besoins exacts des utilisateurs, **il doit être rédigé indépendamment des concepts de solutions envisageables** afin de laisser le plus grand éventail de solutions possibles.

Méthodologie de rédaction des SF

Orienter l'étude

*Du général au spécifique. Le premier point de la démarche va donc consister à **regarder le projet d'un oeil extérieur**, à prendre du recul, à se poser les bonnes questions :*

Rechercher l'information

*La recherche de l'information doit être canalisée et formalisée. C'est un processus qui doit être mené rigoureusement dès le début du projet **afin d'appréhender l'essentiel du besoin**.*

Traduire le besoin en fonctions

*Le **passage du besoin en fonction** s'effectue au travers de l'analyse fonctionnelle qui recense, caractérise, ordonne, hiérarchise et valorise les fonctions.*

Formaliser les travaux

Cette formalisation consiste à développer l'analyse fonctionnelle.

Contrôler les SF

*Le contrôle du document est très important. En effet, on remarque que cette étape n'est généralement pas effectuée de façon optimale alors qu'elle est un frein aux dysfonctionnements qui peuvent apparaître beaucoup plus tard dans le projet. **(par une autre équipe par exemple)***

Valider les SF

Il s'agit de s'assurer que le passage du besoin exprimé au besoin fonctionnel est conforme aux objectifs visés. C'est un travail qui peut s'avérer fastidieux et risqué si le volume d'information est important. L'objectif est donc ici de rendre efficace la validation en réduisant son domaine d'action et tout en conservant sa représentativité.

Contraintes (techniques ou non)

Présentation (animation ?)

Charte graphique

Langues affichées

Mise en page
Interactivité (forum, FAQ, formulaires, ..)

Calendrier (livrables)

sécurité

aspects juridiques

publicité acceptable ?

E Spécifications techniques

Conventions de développement

Enregistrer les pages avec des noms parlant pour les URL en ligne
Nommer par des titres parlant chaque page dans la balise <TITLE> ;
Remplir les balises <META> pour les descriptions et les mots clés de chaque page ;
Préciser la date de création et celle des dernières modifications sur la page d'accueil, voir toutes les pages

Choisir tableaux feuilles de style

Choisir les logiciels à utiliser

Graphisme

Appliquer les styles css pour les mises en page et présentations ;
Mise en page en 800x600 s'adaptant aux autres résolutions ;

Élaborer **une navigation logique**, simple et intuitive
Pouvoir revenir sur l'accueil en cliquant sur le logo de n'importe quelle page du site ;
Choisir une structure linéaire, rayonnante, arborescente
Principes d'accès par rubriques
Arborescence à fournir (rubriques significatives et claires)

Contraintes techniques

Construire des pages par rapport à quel débit ?.