Chapitre 1

Internet et Web

1.1 Internet

1.1.1 Définition

L'Internet est un ensemble de réseaux plus ou moins hétérogènes, disséminés aux quatre coins du monde. Tous ces réseaux sont interconnectés à l'aide de liaisons multiples : RTC (Réseau Téléphonique Commuté), lignes spécialisées, lignes hauts débits, etc. D'un point de vue pratique, cet ensemble de réseaux apparaît aux yeux de l'utilisateur comme un seul et même réseau.

1.1.2 Histoire de l'Internet

Internet est né aux Etats-Unis dans les années 70, lorsque l'ARPA (Advanced Research Project Agency) décida de créer le réseau ARPAnet. Le but de ce réseau était de relier efficacement les centres de recherches entre eux : d'une part pour partager les ressources informatiques et d'autre part pour échanger du courrier électronique. Du fait de son application militaire, son principe fondamental était de pouvoir résister à toutes sortes d'agressions (attaques militaires, bombardements, catastrophes naturelles...). Ses concepteurs eurent une idée révolutionnaire pour l'époque : le réseau ne devait avoir aucun point névralgique (central ou déporté) dont l'arrêt ou le dysfonctionnement aurait pour conséquence le blocage total du réseau. Les données transitant par le réseau devaient pouvoir être re-routées automatiquement vers un autre canal en cas de problème sur une partie du réseau. Le Département de la Défense (le DoD) fut secondé pour son élaboration par les grandes universités américaines. Ces dernières imaginèrent et développèrent les protocoles de transmission. D'ARPAnet le militaire à Internet le civil, il n'y avait qu'un pas qui fut vite franchi. Timidement dans un premier temps,

le réseau commença réellement à prendre son essor dans les années 80 avec le début de la micro informatique. A cette époque, les micro-ordinateurs commençaient à remplacer les terminaux passifs dans les grosses entreprises et dans les universités. Mais le coût encore relativement élevé des moyens à mettre en oeuvre pour établir une liaison, les réservaient aux universités et aux grosses entreprises. Dans les années 90, l'arrivée massive dans les foyers américains de la micro informatique familiale fut le véritable démarrage de l'accès grand public.

1.1.3 Les différents services

Voici une liste des différents services les plus utilisés sur Internet :

Courrier électronique

C'est le premier service qui a été disponible sur Internet. Les premiers utilisateurs avaient déjà besoin de communiquer entre eux. Le principe est basé sur le courrier postal, sauf que ce dernier est beaucoup plus lent que les e-mail, lorsqu'il faut un à deux jours pour recevoir un courrier postal, il faut moins de deux secondes pour recevoir un courrier électronique.

FTP (File Transfer protocol)

Après s'être mis en contact, les premiers utilisateurs voulaient ensuite être capable d'échanger des documents et des programmes. FTP permet de se connecter à un serveur à distance pour déposer ou prendre des fichiers. Il est encore très utilisé aujourd'hui pour déposer des pages Web, des images et autres documents sur des serveurs pour les rendre disponible sur le Word Wide Web.

Le chat

Le chat permet de discuter en temps réel. Il suffit soit d'avoir un logiciel spécialisé soit de vous connecter à un serveur hébergeant un chat, que vous trouverez avec un moteur de recherche sur le Web.

Le Word Wide Web

Ce qui a rendu Internet si populaire est le Web. Il ne faut pas confondre entre Internet et le Web, le Web est l'un des service disponible grâce à Internet.

1.2 Le WORLD WIDE WEB

Le World Wide Web (WWW) peut être vu comme un énorme système distribué composé de millions de clients et de serveurs pour accéder à des documents liés. Les serveurs gèrent des collections de documents, tandis que les clients fournissent aux utilisateurs une interface facile à utiliser pour présenter et accéder à ces documents.

1.2.1 Évolution du Web

Le web 1.0, encore appelé web traditionnel, est avant tout un web statique, centré sur la distribution d'informations. Il se caractérise par des sites orientés produits, qui sollicitent peu l'intervention des utilisateurs. Les premiers sites d'e-commerce datent de cette époque. Longtemps considéré comme le « Web statique » (voir passif), l'Internet des années 1990 était vu comme un réseau de bibliothèques où l'on pouvait simplement accéder à l'information et non interagir avec. En effet celui-ci avait un fonctionnement linéaire, un contenu était proposé en ligne et l'utilisateur se contentait de le consulter sans avoir la possibilité de le modifier. De plus une fois le contenu mis en ligne, il n'est que peu mis à jour.

Le web 2.0, ou web social, change totalement de perspective. Il privilégie la dimension de partage et d'échange d'informations et de contenus (textes, vidéos, images ou autres). Il voit l'émergence des réseaux sociaux, des smartphones et des blogs. Le web se démocratise et se dynamise. L'avis du consommateur est sollicité en permanence et il prend goût à cette socialisation virtuelle. On l'appelle le « Web participatif » ou encore le « Web Social », car c'est un système complètement interactif, accessible depuis les quatre coins de la planète et dans lequel l'utilisateur est actif. C'est une évolution voyant la complexité des technologies augmenter au profit de la simplicité d'utilisation, permettant à une personne n'ayant que peu de connaissances informatiques de maitriser les fonctionnalités du Web. Les internautes sont les acteurs principaux du Web 2.0, ils partagent de l'information, ajoutent du contenu (images, vidéos, texte...) et ont la possibilité de partager de l'information. Les exemples de sites Web 2.0 sont les réseaux sociaux, les blogs, les sites de e-commerce et tout ce qui est services hébergés. Même s'il a initialement été conçu pour être une nouvelle version du World Wide Web, il a fini par changer la façon dont les utilisateurs finaux et les développeurs utilisent le Web.

Le web 3.0, aussi nommé web sémantique, vise à organiser la masse d'informations disponibles en fonction du contexte et des besoins de chaque utilisateur, en tenant compte de sa localisation, de ses préférences, etc. C'est un

FIGURE 1.1 – Schéma comparatif du Web 1.0 et Web 2.0.

web qui tente de donner sens aux données. C'est aussi un web plus portable et qui fait de plus en plus le lien entre monde réel et monde virtuel. Il répond aux besoins d'utilisateurs mobiles, toujours connectés à travers une multitude de supports et d'applications malines ou ludiques. Le Web 3.0 représente la liberté, il n'est plus seulement accessible depuis un ordinateur mais aussi depuis les smartphones, tablettes et autres objets connectés. Son apparition introduit la notion de « cloud computing » (ou de nuage), les informations et données personnelles sont de plus en plus stockées sur le net, et accessible à tout moment depuis n'importe quelle plateforme. On trouve de nombreuses applications directement en ligne, ce qui fait du web un écosystème à part entière. Les informations de l'utilisateur sont stockées lorsqu'il navigue sur internet et réutilisées afin de lui proposer un contenu adapté à son profil. Internet est partout, et on voit émerger l'avènement des technologies mobiles. De nos jours tous les objets de la vie quotidienne, comme la voiture ou même les chaussures, sont connectés au web et communiquent ensemble afin d'échanger automatiquement des données et ainsi « facilité » la vie des

utilisateurs.

1.2.2 Vue d'ensemble de WWW

Le WWW est essentiellement un énorme système client-serveur avec des millions de serveurs distribués dans le monde entier. Chaque serveur conserve une collection de documents ; chaque document est stocké sous forme de fichier (bien que des documents puissent également être générés sur demande). Un serveur accepte les demandes d'extraction d'un document et le transfère au client. En outre, il peut également accepter les demandes de stockage de nouveaux documents.

Le moyen le plus simple de se référer à un document est d'utiliser une référence appelée URL (Uniform Resource Locator). Il spécifie où se trouve un document, souvent en intégrant le nom DNS de son serveur associé avec un nom de fichier par lequel le serveur peut rechercher le document dans son système de fichiers local. En outre, une URL spécifie le protocole au niveau de l'application pour le transfert du document sur le réseau. Différents protocoles sont disponibles (HTTP, FTP, SMTP...).

Un client interagit avec les serveurs Web via une application spéciale appelée navigateur. Il existe plusieurs navigateurs Internet dont les plus connus sont : Internet Explorer, Safari, Opera, Firefox et Google Chrome. Un navigateur est responsable de l'affichage correct d'un document. En outre, un navigateur accepte l'entrée d'un utilisateur principalement en permettant à l'utilisateur de sélectionner une référence à un autre document, qu'il récupère ensuite et affiche. Cela conduit à l'organisation globale représentée sur la Figure 1.2.2.

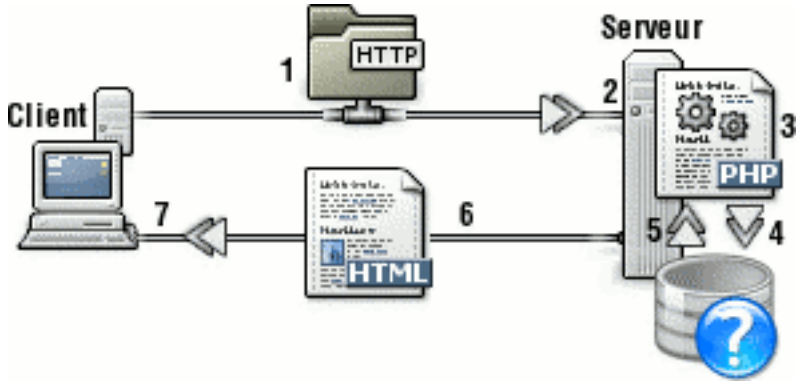


FIGURE 1.2 – L'organisation globale du Web.

1.3 Niveaux d'abstraction d'une application

En règle générale, une application informatique peut être découpée en trois niveaux d'abstraction distincts :

- **La couche présentation :** Aussi appelée IHM (Interface Homme/Machine), elle permet l'interaction de l'utilisateur avec l'application.
- **La couche traitements :** Elle décrit les travaux à réaliser par l'application. Ils peuvent être découpés en deux familles :
  — **Les traitements locaux**, regroupant les contrôles effectués au niveau du dialogue avec l'IHM, visant essentiellement le contrôle et l'aide à la saisie ;
  — **Les traitements globaux**, constituant l'application elle-même. Cette couche, appelée Business Logic ou couche métier, contient les règles internes qui régissent un système donné.
- **La couche données :** Elle gère le stockage des données et leur accès.

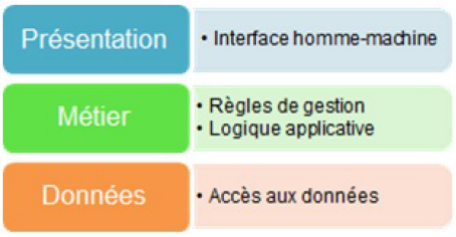


FIGURE 1.3 – Les couches applicatives.

1.4 Architecture client/serveur

Qu'est-ce qu'un serveur ?

On appelle logiciel serveur un programme qui offre un service sur le réseau. Le serveur accepte des requêtes, les traite et renvoie le résultat au demandeur. Le terme serveur s'applique à la machine sur lequel s'exécute le logiciel serveur.

Qu'est-ce qu'un client ?

On appelle logiciel client un programme qui utilise le service offert par un serveur. Le client envoie une requête et reçoit la réponse.

Qu'appelle-t-on architecture client/serveur ?

C'est la description du fonctionnement coopératif entre le serveur et le client. Les services Internet sont conçus selon cette architecture. Ainsi, chaque application est composée de logiciel serveur et logiciel client. A un logiciel serveur, peut correspondre plusieurs logiciels clients développés dans différents environnements : Unix, Mac, PC... ; la seule obligation est le respect du protocole entre les deux processus communicants.

L'organisation d'un environnement client-serveur diffère selon le type d'architecture du réseau et le type de client :

- **Architecture pair à pair :** Une architecture pair à pair (peer-to-peer ou P2P en anglais) est un environnement client-serveur où chaque programme connecté est susceptible de jouer tour à tour le rôle de client et celui de serveur.
- **Architecture à deux niveaux :** Une architecture à deux niveaux ou une architecture deux tiers (two-tier architecture en anglais) est un environnement client-serveur où le client demande une ressource au serveur qui la fournit à partir de ses propres ressources.
- **Architecture à trois niveaux :** Une architecture à trois niveaux ou une architecture trois tiers (three-tier architecture en anglais) ajoute un niveau supplémentaire à l'architecture à 2 niveaux, permettant de spécialiser les serveurs dans une tâche précise, ce qui donne un avantage de flexibilité, de sécurité et de performance :
  — un client qui demande une ressource via une interface utilisateur (généralement un navigateur web) chargée de la présentation de la ressource ;
  — un serveur d'application (appelé middleware) qui fournit la ressource, mais en faisant appel aux ressources d'un autre serveur ;
  — un serveur de données qui fournit au serveur d'application les ressources requises pour répondre au client.
- **Architecture à N niveaux :** Une architecture à N niveaux ou architecture N tiers (N-tier architecture en anglais) ajoute encore des niveaux supplémentaires à l'architecture à 3 niveaux, permettant de spécialiser les serveurs davantage.

Chapitre 2

Structure d'un document HTML

2.1 Introduction

Une page web créée avec HTML doit être pensée en distinguant deux parties.

— Un contenu, structuré au moyen des éléments HTML (grandes divisions, titres, paragraphes, tableaux, images et liens, etc.). À ce stade, et même s'il en a déjà une idée, le créateur ne doit pas nécessairement avoir une vue définitive de la présentation finale. Il lui faut maîtriser principalement l'organisation des informations à fournir à un utilisateur.

— Une feuille de style CSS, définissant la mise en page de ces éléments en fonction du média qui va opérer le rendu du contenu (polices et tailles de caractères, bordures, marges, couleurs, positionnement dans la page, etc.). Les médias se diversifiant en effet de plus en plus en devenant des éléments portables dotés de petits écrans, le traditionnel écran d'ordinateur n'est plus le principal vecteur d'affichage d'une page web.

2.2 Généralités

2.2.1 Définition de site web

Comparable à un livre, un site web est un ensembles de pages, composés de texte et de médias, relier entre elles par des hyperliens, défini et accessible par une adresse web. Un site web est hébergé sur un serveur web accessible via le réseau mondial internet ou un intranet local. L'ensemble des sites web constituent le World Wide Web.

2.2.2 Etapes de création d'un site

Une fois le projet est clairement définit, la création passe par 4 étapes :

1. Définition du visuelle (Web design).
2. Développement web (Création du programme).
3. Intégration de son contenu (Texte, images, vidéos...).
4. Mise en production (Mise en ligne du site).

2.2.3 Acteurs du développement web

- Webdesigner : chargé du visuelle.
- Développer web : informaticien.
- Administrateur système et BDD : il s'occupe de l'hébergement du site web, pour accès rapide.
- Référenceur/rédacteur : ils s'occupe du positionnement sur Google.
- Chef de projet qui coordonne le tout.

2.3 HTML 5 (HyperText Markup Language)

HTML 5 (HyperText Markup Language) est un langage de balisage (dit aussi langage de marquage) qui permet de structurer le contenu des pages web dans différents éléments.

2.3.1 Les balises et attributs

En HTML, tout contenu, qu'il s'agisse de texte, d'images ou d'éléments multimédias les plus divers, doit être enfermé dans *une balise*. Chaque balise possède un nom déterminé, en générale, une balise a la structure suivante :

$< nom\_balise > Contenu < /nom\_balise >$

Son contenu est précédé par une balise d'ouverture $< nom\_balise >$ et suivi par une balise de fermeture $< /nom\_balise >$. Toutes les balises d'ouverture (ou marqueur) commencent par le signe "<" et se terminent par le signe ">". La balise de fermeture suit la même règle mais le nom de la balise est précédé d'un *slash (/)*. Les navigateurs interprètent donc les contenus en fonction du nom de la balise et de ses attribuent.

Les caractéristiques de chaque balise peuvent être précisées par des informations complémentaires que l'on désigne en tant *qu'attributs* de la balise. Il peut s'agir par exemple de la définition de la largeur, de la hauteur ou de l'adresse du contenu. La valeur de tous les attributs doit être définie entre

guillemets. La syntaxe conforme d'une balise ayant des attributs est donc la suivante :

$< nom\_balise\ attributl = "valeurl"\ attribut2 = "valeur2" > Contenu < /nom\_balise >$

2.3.2 Structure d'un document HTML 5

Le langage HTML 5 est une amélioration du langage HTML 4, avec des simplifications par rapport à la version XHTML qui était de mise avant lui. Tout document peut donc débuter de la même manière par la déclaration suivante :

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<ti11e> HTML 5 et CSS 3 </title>
</head>
<body>
<!- - Tout le contenu de la page - ->
</body>
</html>
```

La déclaration DOCTYPE, obligatoire dans tout document, précise le type de document qui va être créé. La balise <html > est la balise racine du document, son contenu est constitué de l'en-tête du document, introduit par la balise <head> et terminé par la balise </head>, puis par le corps du document introduit par <body> et terminé par </body>.

2.3.3 Les principales balises HTML 5

Les commentaires

Il est toujours utile de commenter votre code HTML 5, pour en permettre une meilleure compréhension, en particulier pour le relire un certain temps après l'avoir écrit. Tous les commentaires que vous écrirez seront ignorés par le navigateur, et vous pouvez donc vous exprimer librement. Pour placer un commentaire, vous devez l'ouvrir avec les symboles $<! - -$, et le fermer avec les symboles $- - >$

La balise

La balise peut être directement inclus dans le corps du document <body>. Elle crée une division de la page. Ce type de division permet de grouper dans un seul bloc un ensemble composé soit de textes, soit de balises inclus, auxquels on pourra appliquer globalement des styles particuliers. Une division créée avec permet d'inclure une très grande variété de balises HTML. C'est donc un élément très riche qui se prêtait bien à la création de la structure entière d'une page avant HTML 5, et auquel il est possible d'appliquer par la suite des styles propres et des positions précises.

Les balise header, footer, aside et nav

HTML 5 a introduit un ensemble de nouveaux éléments ayant chacun une fonction sémantique précise. Ceci a pour but principal de faciliter l'indexation du contenu des pages par les moteurs de recherche. Leur rôle est de structurer la page en plusieurs grandes zones dont le simple nom de la balise est évocateur de son utilité. Il s'agit des éléments <header>, <footer>, <aside> et <nav>.

La balise <header> comme son nom l'indique constitue l'en-tête de la page. Il va pouvoir contenir un bandeau et/ou une liste de liens organisés en menus. Dans la continuité du précédent, la balise <footer> a pour rôle de contenir le pied de page qui peut être commun à toutes les pages du même site et inclure des informations comme une adresse de contact avec le webmestre ou les coordonnées de l'entreprise par exemple.

La balise <section> sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page. La balise <aside> porte également un nom explicite. C'est en effet le conteneur d'informations ou de liens connexes avec le contenu principal. Il pourrait être utilisé par exemple pour faire référence à des sujets complémentaires ou pour définir des termes spécifiques employés dans le contenu principal. En termes de positionnement, il est logiquement placé sur un coté de la page, bien que ce ne soit pas une obligation.

Il peut y avoir plusieurs blocs <aside> dans la page. Sur Wikipédia, par exemple, il est courant de voir à droite un bloc d'informations complémentaires à l'article que l'on visualise.

La balise <nav> contient la liste des liens utiles à la navigation vers les différentes pages du site ou vers des pages externes en rapport avec lui. Il peut être inclus aussi bien dans l'en-tête <header> que directement dans <body> ou même dans <footer> en guise de rappel des liens pour éviter de remonter en haut de page. La balise <nav> doit regrouper tous les principaux liens de

navigation du site. Vous y placerez par exemple le menu principal de votre site. Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise <nav> :

```
    <nav>
<ul>
<li><a href="index.html">Accueil</a></li>
<li><a href="forum.html">Forum</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
</nav>
```

La balise <article> sert à englober une portion généralement autonome de la page. C'est une partie de la page qui pourrait ainsi être reprise sur un autre site. C'est le cas par exemple des actualités (articles de journaux ou de blogs).

Les blocs de citations : la balise <blockquote>

La balise <blockquote> sert à créer une division de petite taille dans le corps d'une page. A l'origine, il était destiné à contenir des blocs de citations, comme un petit poème ou une note. Son contenu est par défaut mis en évidence au moyen d'un affichage en retrait par rapport aux bords gauche et droit de la page. Chaque élément est suivi d'un saut de ligne, tout comme pour les paragraphes.

Les titres et groupes de titres

Dans une page web, c'est en priorité les titres qui identifient les grandes sections de texte. Les titres sont contenus dans les balises $< h1 > ... < /h1 >$, pour les titres de premier niveau, à $< h6 > ... < /h6 >$, pour les titres de plus bas niveau. Entre ces extrêmes, nous pouvons utiliser les titres $< h2 >$, $< h3 >, < h4 >$ *et* $< h5 >$. Si aucun style personnalisé ne leur est donné, les navigateurs affichent les titres dans des polices de tailles dégressives pour les balises $< h1 >$ (le plus grand) à $< h6 >$ (le plus petit). Les balises de titre sont automatiquement suivis d'un saut de ligne, après la balise de fin.

Les paragraphes : la balise <p>

Comme dans un traitement de texte, le contenu d'une page peut être divisé en différents paragraphes. Chaque paragraphe sera par défaut précédé et suivi d'un saut de ligne pour marquer la séparation avec le contenu précé-

dent et suivant. Chaque paragraphe doit être contenu dans l'élément , et donc délimité par les balises et .

Les styles physiques

Parmi les éléments en ligne utilisables pour contenir du texte, certains permettent de créer des styles physiques pour leur contenu. Une partie d'entre eux correspondent aux modifications courantes que chacun peut effectuer dans son traitement de texte, comme mettre un texte en gras, en italique, ou certains caractères en indice ou en exposant. Ce type de marquage est indépendant de la taille et de la police de caractères. D'autres éléments agissent de manière relative sur leur contenu en permettant d'afficher dans une police plus grande ou plus petite que la police utilisée dans le texte qui précède sans préjuger de cette taille. A chacun de ces éléments correspond un style par défaut qui peut donner satisfaction ; dans le cas contraire, ce style pourra être personnalisé à loisir avec CSS.

Mettre un texte en gras

Pour mettre en gras une partie de texte, il faut l'inclure dans la balise (donc entre et). Nous pouvons également utiliser la balise   qui, visuellement, donne le même effet. Cependant le vrai rôle de cette balise est d'exprimer à la machine l'importance de la phrase. En effet, de nombreux programmes analysent le code source des pages web, à commencer par les robots de moteurs de recherche. Ces robots parcourent le Web en lisant le code HTML de tous les sites. C'est le cas des robots de Google et de Bing, par exemple. Les mots-clés « importants » ont tendance à avoir plus de valeur à leurs yeux, donc si quelqu'un fait une recherche sur ces mots, il a plus de chances de tomber sur votre site.

Mettre un texte en italique

il existe deux balises qui affichent par défaut leur conteneur en italique. Il s'agit des balises <i> et . L'élément <i> a clairement pour destination une mise en italique et plutôt une mise en évidence d'un passage, comme pour . Il est possible d'imbriquer des balises dans les balises <i> et pour que le texte apparaisse à la fois en italique et en caractères gras.

Modifier la taille du texte

À l'intérieur d'une division de la page, nous pouvons modifier la taille relative du texte en utilisant la balise <small></small>. Il permet d'afficher son contenu avec une taille de police plus petite que celle du contexte quel que soit celui-ci. En imbriquant les balises <small> les unes dans les autres, on peut obtenir des tailles de polices de plus en plus petites. Le nombre d'imbrications possibles dépend de la taille de la police du conteneur.

Créer des exposants et des indices

Les traitements de texte offrent aussi la possibilité de mettre des caractères en exposant ou en indice. Cette opération est également possible en HTML. Pour mettre un texte en exposant, il faut l'inclure dans la balise <sup> (entre ^{et}), et pour écrire un texte en indice, il faut l'inclure dans la balise <sub> (entre _{et}).

Créer un retour à la ligne

La balise
 permet de créer un retour à la ligne. C'est une balise orpheline, d'où l'utilisation du caractère antislash (/) en guise de signe de fermeture.

Surligner un texte

Afin de pouvoir surligner un texte, HTML 5 a introduit la balise <mark> dont le contenu est mis en évidence dans tous les navigateurs par un fond jaune vif comme on le fait couramment avec un surligneur sur du papier.

Les listes

La présentation sous forme de liste permet une structuration de l'information telle qu'elle peut apparaître dans une table des matières. On peut également mettre en évidence les points importants. Les utilisateurs de traitement de texte sont familiarisés avec cette façon de procéder. Elle implique qu'une série d'informations aient un rapport entre elles, par exemple sous forme d'énumération d'une liste de tâches à réaliser. Ces listes d'informations peuvent être numérotées ou marquées par une puce graphique. De la même façon, avec HTML 5 on peut créer des listes d'items numérotés, nommées listes ordonnées, ou de listes à puces, nommées listes non ordonnées. Un troisième type de listes permet également d'énumérer des termes et d'en donner les définitions.

* Les listes ordonnées

Pour créer une liste dans laquelle la notion d'ordre a une importance, nous pouvons utiliser une liste ordonnée dont chaque item sera numéroté par défaut à l'aide d'entiers incrémentés de 1 à N, suivis d'un point puis du contenu de chaque item. Une liste ordonnée doit commencer par la balise <ol> (pour Ordered List) qui doit obligatoirement contenir au moins une balise <li> qui elle-même renferme le contenu visible de chaque item. Il faut donc que <ol> contienne autant de balises <li> qu'il y a d'items dans la liste désirée.

En plus des attributs globaux, cet balise possède l'attribut start dont la valeur est un nombre pour que la numérotation ne commence pas à 1, comme c'est le cas par défaut, mais à un nombre ou une lettre précisés et l'attribut type permet de choisir le style de la numérotation ; nous avons les choix suivants :

- type="1" : numérotation décimale : 1, 2, 3...
- type="a" : numérotation alphabétique minuscule : a, b, c, d...
- type="A" : numérotation alphabétique majuscule : A, B, C, D...
- type="i " : numérotation en chiffres romains minuscules : i, ii, iii, iv...
- type="I" : numérotation en chiffres romains majuscules : I, II, III, IV...

* Les listes non ordonnées

Les listes non ordonnées (Unordered List) fournissent un outil de structuration similaire au précédent mais sans la notion de numérotation. Elles sont également appelées listes à puces car chaque item est précédé d'une puce graphique qui, par défaut, est un disque plein de la même couleur que le texte qui la suit.

Une liste à puces est introduite par la balise <ul> et fermée par . Comme la balise <ol>, son seul contenu direct est un ou plusieurs éléments <li>.

* Les listes de définitions

Une liste de définitions permet de créer une liste de termes, chacun d'entre eux étant suivi de sa définition. Le conteneur de l'ensemble de la liste est la balise <dl> qui ne peut contenir que les balises <dt>, <dd>, ou la balise <dl> elle-même et rien d'autre.

Le plus souvent, la balise <dt> contient le terme et <dd> en renferme la définition.

Les liens

La balise HTML 5 primordial pour la création de liens est <a>, dont le contenu, situé entre les balises <a> et , est la partie visible, texte ou image, sensible au clic. Par défaut, le lien s'affiche en bleu souligné.

* Les liens externes (lien vers un autre site)

Pour faire un lien vers un site externe, il faut ajouter à la balise <a> un attribut, href, pour indiquer vers quelle page le lien doit conduire.

Pour plus d'informations

Remarque : si vous faites un lien vers un site qui comporte une adresse avec des &, comme : http ://www.site.com/ ?data=15&name=george, vous devrez remplacer tous les « & » par « &amp ; »

* Un lien vers une autre page de son site

Dans le cas où, les deux pages sont situées dans un même dossier, il faudrait écrire le nom du fichier html comme valeur de l'attribut href : <a href="page2.html">. On dit que c'est un lien relatif. Dans le cas où les deux pages se situes dans des dossiers différents, le lien doit être rédigé comme ceci : <a href="contenu/page2.html">

* Un lien vers une ancre

Quand le contenu d'une page est volumineux, l'utilisateur ne peut pas en avoir une vision globale. Il est alors souhaitable de lui proposer une table des matières ou un menu composé de liens internes vers les différentes sections de la page. Il pourra ainsi accéder directement au point de son choix sans faire défiler la page. Chaque point cible de la page doit être signalé par un lien particulier, appelé *ancre*. Une ancre est une sorte de point de repère que vous pouvez mettre dans vos pages HTML lorsqu'elles sont très longues.

Pour créer une ancre, il suffit de rajouter l'attribut id à une balise qui va alors servir de repère. Ce peut être n'importe quelle balise, un titre par exemple, cela nous servira ensuite pour faire un lien vers cette ancre. Par exemple :

<h2 id="mon_ancre">Titre</h2>

Ensuite, il suffit de créer un lien, où l'attribut href contiendra un dièse (#) suivi du nom de l'ancre. Exemple :

Aller vers l'ancre

Remarque : L'attribut id sert à donner un nom « unique » à une balise, pour s'en servir de repère. Ici, on s'en sert pour faire un lien vers une ancre mais, en CSS, il nous sera très utile pour « repérer » une balise précise. Pour que la valeur de id soit reconnue par tous les navigateurs, il faut évitez des espaces ou des caractères spéciaux, on utilise que des lettres et chiffres.

Pour afficher une infobulle lorsqu'on pointe sur le lien on utilise l'attribut title. Il est possible de « forcer » l'ouverture d'un lien dans une nouvelle fenêtre. Pour cela, on rajoutera target="_blank" à la balise <a>. Selon la configuration du navigateur, la page s'affichera dans une nouvelle fenêtre ou un nouvel onglet. Vous ne pouvez pas choisir entre l'ouverture d'une nouvelle fenêtre ou d'un nouvel onglet.

* Lien déclenchant l'envoi d'un e-mail

Pour permettre aux visiteurs d'entrer en contact avec le webmaster, afin qu'ils envoient leurs observations ou questions, on utilise des liens de type **mailto**. Rien ne change au niveau de la balise, on doit simplement modifier la valeur de l'attribut href comme ceci :

```
<p><a href="mailto :votreadresse@gmail.com">Envoyez-moi un e-mail !</a></p>
```

* Lien déclenchant le téléchargement d'un fichier

Pour permettre aux visiteurs de télécharger des documents sur votre site, on procède de la même manière que pour les liens, mais en indiquant, pour l'attribut href, le nom du fichier à télécharger après l'avoir placé dans le même dossier que la page html.

Par exemple, supposez que vous vouliez faire télécharger coursHTML.zip. Placez simplement ce fichier dans le même dossier que votre page web (ou dans un sous-dossier) et faites un lien vers ce fichier :

```
<p><a href="coursHTML.zip">Télécharger le cours</a></p>
```

Insertion des images

Les sites grand public se doivent d'égayer leurs pages avec des illustrations photographiques ou simplement graphiques. L'abus d'images pouvant se révéler aussi nocif que leur absence en termes d'attractivité du site pour le visiteur, il appartient au concepteur d'effectuer un choix judicieux de ses illustrations. De plus, il faut encore tenir compte du poids des contenus multimédias en kilo-octets. Le poids des images est sans commune mesure avec celui du code d'une page HTML 5 et vient rapidement allonger le temps d'affichage complet d'une page si elle en contient beaucoup.

* Les différents formats d'une image

Voici quel format adopter en fonction de l'image que vous avez : Une photo : utilisez un JPEG. N'importe quel graphique avec peu de couleurs (moins de 256) : utilisez un PNG 8 bits ou éventuellement un GIF. N'importe quel graphique avec beaucoup de couleurs : utilisez un PNG 24 bits. Une image animée : utilisez un GIF animé.

* L'insertion d'images

La balise <img /> permet l'insertion des images cette balise doit être accompagnée de deux attributs obligatoires.

- **src** : il permet d'indiquer où se trouve l'image que l'on veut insérer, soit en indiquant un chemin absolu (ex. : http ://www.site.com/fleur.png), soit mettre le chemin en relatif (ce qu'on fait le plus souvent). Ainsi, si l'image est dans un sous-dossier images, on doit taper : src="images/fleur.png".
- **alt :** cela signifie « texte alternatif ». On doit toujours indiquer un texte alternatif à l'image, un court texte qui décrit ce que contient l'image. Ce texte sera affiché à la place de l'image si celle-ci ne peut pas être téléchargée. Cela aide aussi les robots des moteurs de recherche pour les recherches d'images.

* Titre d'une image

Pour améliorer la présentation d'une image, nous disposons d'une balise <figure> qui permet de titrer correctement une image, dans lequel sont insérés un élément <img/> qui permet l'affichage de l'image et un élément <figcaption> qui crée un titre (au-dessus ou en dessous, par exemple, mais pas les deux à la fois). La syntaxe générale de l'ensemble est la suivante :

```
<figure>
<img src="image.jpg" alt="commentaire" />
<figcaption>Légende</figcaption>
</figure>
```

La balise <figure> a un rôle avant tout sémantique. Cela veut dire qu'elle indique à l'ordinateur que l'image a du sens et qu'elle est importante pour la bonne compréhension du texte. Cela peut permettre à un programme de récupérer toutes les figures du texte et de les référencer dans une table des figures, par exemple.

Insertion d'un élément audio

Une autre nouveauté dans le domaine du multimédia est la balise <audio> qui permet de lire des fichiers son dans un navigateur, sa syntaxe est la suivante :

```
<audio src="musique.mp3"></audio>
```

On peut compléter la balise avec les attributs suivants :

- **controls** : pour ajouter les boutons *Lecture*, *Pause* et la barre de défilement.
- **width** : pour modifier la largeur de l'outil de lecture audio.
- **loop** : la musique sera jouée en boucle.
- **autoplay** : la musique sera jouée dès le chargement de la page.
- **preload** : indique si la musique peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - auto (par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
  - metadata : charge uniquement les métadonnées (durée, etc.).
  - none : pas de préchargement.

2.3.4 Insertion d'une vidéo

En HTML 5 pour insérer une vidéo dans la page web, on utilise la balise <video>. La syntaxe de base de <video> est la suivante :

```
<video src="nomvideo">...</video>
```

On peut lui ajouter des particularités avec les attributs supplémentaires :

- **poster** : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo.
- **controls** : pour ajouter les boutons *Lecture*, *Pause* et la barre de défilement.
- **width** : pour modifier la largeur de la vidéo.
- **height** : pour modifier la hauteur de la vidéo.
- **loop** : la vidéo sera jouée en boucle.
- **autoplay** : la vidéo sera jouée dès le chargement de la page.
- **preload** : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :

— auto (par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.
— metadata : charge uniquement les métadonnées (durée, dimensions, etc.).
— none : pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.

2.3.5 Les tableaux

Dans une page HTML 5, il est possible de réaliser une présentation d'informations de type textuelle ou graphique sous la forme de tableau. La balise de base est <table> </table>, elle permet d'indiquer le début et la fin d'un tableau. En elle-même, elle ne fournit aucun résultat visuel et ne prend d'importance que par ses éléments enfants :

- <tr> </tr> : indique le début et la fin d'une ligne du tableau ;
- <td> </td> : indique le début et la fin du contenu d'une cellule.

Pour les cellules qui jouent le rôle d'en-tête de colonne ou de ligne, on utilise <th> (pour table head) au lieu de <td>. Pour terminer la présentation d'un tableau, nous pouvons lui attribuer un titre général qui doit être contenu dans la balise <caption>, lui-même inclus dans <table>. Cette balise doit être la première à apparaître dans <table>. Cette position peut être modifiée en recourant à un style CSS.

```
<table border="1">
<caption>Un tableau élémentaire</caption>
<tr>
<td> Ligne 1 Colonne 1 </td>
<td> Ligne 1 Colonne 2 </td>
<td> Ligne 1 Colonne 3 </td>
</tr>
<tr>
<td> Ligne 2 Colonne 1 </td>
<td> Ligne 2 Colonne 2 </td>
<td> Ligne 2 Colonne 3 </td>
</tr>
<tr>
<td> Ligne 3 Colonne 1 </td>
<td> Ligne 3 Colonne 2 </td>
<td> Ligne 3 Colonne 3 </td>
</tr>
</table>
```

Diviser un gros tableau

Si un tableau est assez gros, on peut le structurer au le découpant en plusieurs parties. Pour cela, il existe des balises HTML qui permettent de définir les trois *zones* du tableau :
l'en-tête (en haut) : il se définit avec les balises <thead></thead> ;
le corps (au centre) : il se définit avec les balises <tbody></tbody> ;
le pied du tableau (en bas) : il se définit avec les balises <tfoot></tfoot>.

Fusion

Pour créer des tableaux à structure irrégulière, il faut fusionner des cellules voisines comme dans un tableur. On procède à cette fusion en utilisant les attributs colspan et rowspan des balises <td> et <th>.

Il existe deux types de fusion :

* **La fusion de colonnes** : On procède à la fusion de cellules d'une même ligne en définissant la valeur de l'attribut colspan d'une balise <td> ou <th> avec un entier, lequel indique le nombre de cellules à fusionner en partant de la gauche. La syntaxe à suivre pour fusionner N cellules est la suivante :
<td colspan="N"> Contenu de la cellule</td>

* **La fusion de lignes** : La fusion de cellules situées dans les lignes adjacentes peut être définie à l'aide de l'attribut rowspan des balises <td> et <th>. *rowspan* a pour valeur le nombre de cellules à fusionner. La fusion doit être déclarée dans la cellule la plus haute. Sa syntaxe est donc la suivante :

<td rowspan="N"> Contenu de la cellule</td>

Les formulaire

Les éléments constitutifs d'un formulaire doivent être contenus entre les balises <form> et </form>. C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

Pour récupérer les données saisies, on doit ajouter deux attributs à la balise <form> :

- **method** : cet attribut indique par quel moyen les données vont être envoyées. Il existe deux solutions pour envoyer des données sur le Web :
  * method="get" : c'est une méthode en général assez peu adaptée car elle est limitée à 255 caractères. La particularité vient du

fait que les informations seront envoyées dans l'adresse de la page (http ://...). Il est recommandé d'utiliser la méthode : post.
    * method="post" : c'est la méthode la plus utilisée pour les formulaires car elle permet d'envoyer un grand nombre d'informations. Les données saisies dans le formulaire ne transitent pas par la barre d'adresse.
- **action** : c'est l'adresse de la page ou du programme qui va traiter les informations. Cela ne peut pas se faire en HTML et CSS, on utilisera en général un autre langage, dans ce cours nous utiliserons le PHP.

```
<form method="post" action="traitement.php">
<p>Texte à l'intérieur du formulaire</p>
</form>
```

En HTML, il existe deux zones de texte différentes permettant de saisir du texte dans un formulaire :

- **La zone de texte monoligne** : comme son nom l'indique, on ne peut y écrire qu'une seule ligne. Elle sert à saisir des textes courts, par exemple un pseudo.
- **La zone de texte multiligne** : cette zone de texte permet d'écrire une quantité importante de texte sur plusieurs lignes.

Un formulaire permet le plus souvent la saisie de texte, par exemple pour indiquer son nom ou son adresse. En HTML, un grand nombre de champs de saisie sont créés avec l'élément <input />. C'est son attribut type qui détermine la catégorie de champ qui est obtenue.

Pour insérer une zone de texte dans une ligne, on va utiliser la balise <input />. On peut enrichir cette balise avec les attributs suivants :

— name : il attribue un nom à la zone de saisie, ce qui permet de récupérer dans une variable la valeur saisie sur le serveur.
— size="N" permet de fixer la longueur visible de la zone de texte à N caractères, ce qui n'empêche pas des saisies plus longues.
— maxlength="N" permet de limiter le texte saisi à N caractères. Au-delà de ce nombre, les frappes effectuées au clavier sont inopérantes. Cet attribut peut servir à mettre en adéquation la longueur d'une donnée avec celle du champ d'une base de données dans laquelle elle doit être enregistrée.
— value : il définit un texte par défaut qui est affiché dans la zone de texte tant que l'utilisateur n'en a pas saisi un autre. C'est cette valeur qui est transmise au serveur si l'internaute ne modifie rien dans le champ texte. Comme dans l'exemple ci-après :
<input type="text" name="pays" maxlenght="25" value="France" />

— placeholder : il joue en partie le même rôle que value dans la mesure où son contenu est affiché dans la zone de saisie de texte, mais il a l'avantage de s'effacer tout seul sans script si l'utilisateur clique sur la zone. De plus, en cas d'absence de saisie, son contenu n'est pas transmis au serveur ; il ne faut donc pas s'en servir comme valeur par défaut.

* La saisie de texte

Pour saisir un texte, on utilise la balise suivante :
<input type="text" />

Ce n'est pas encore suffisant, il faut donner un nom à votre zone de texte. Ce nom n'apparaît pas sur la page mais il vous sera indispensable, en PHP, pour reconnaître d'où viennent les informations.

Pour donner un nom à un élément de formulaire, on utilise l'attribut name :

<input type="text" name="pseudo" />

Pour donner un label à un champ, on utilise la balise <label>, qu'il faut lier à la zone de texte. Pour ce faire, on doit donner un nom à la zone de texte, non pas avec l'attribut *name* mais avec l'attribut id (que l'on peut utiliser sur toutes les balises).

Pour lier le label au champ, il faut ajouter à la balise *label* un attribut for qui a la même valeur que l'id du champ.

Pour créer une zone de texte multiligne, on utilise la balise <textarea> </textarea>. Comme pour tout autre balise du formulaire, il faut lui donner un nom avec name et utiliser un label qui explique de quoi il s'agit.

HTML5 apporte de nombreuses fonctionnalités nouvelles relatives aux formulaires. De nouveaux types de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut type de la balise <input /> l'une des nouvelles valeurs disponibles.

- **type="password" :** la zone de texte va se comporter comme une zone de mot de passe où on ne voit pas à l'écran les caractères saisis.
- **type="email" :** la valeur *email*, permet d'indiquer au navigateur que l'utilisateur doit saisir une adresse e-mail. Il peut afficher une indication si l'adresse n'est pas un e-mail.
- **type="url" :** comme pour *email*, la valeur *url*, permet d'indiquer au navigateur que l'utilisateur doit saisir une adresse adresse absolue (commençant généralement par http ://).
- **type="tel" :** ce champ est dédié à la saisie de numéros de téléphone.

- **type="number" :** ce champ permet de saisir un nombre entier et on peut ajouter les attributs suivants :
  - min : valeur minimale autorisée.
  - max : valeur maximale autorisée.
  - step : c'est le pas de déplacement. Si vous indiquez un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6...).

- **type="search" :** la valeur *search*, permet d'indiquer au navigateur que le champ de saisie est un champ de recherche. Le navigateur décide ensuite comment afficher le champ de recherche. Ainsi, il peut ajouter une petite loupe au champ pour signifier que c'est un champ de recherche et éventuellement mémoriser les dernières recherches effectuées par le visiteur.

* Les balises d'options

Pour créer une zone de case à cocher dans votre formulaire qui demandent au visiteur de faire un choix parmi une liste de possibilités, il faut utiliser un input de type *checkbox* : <input type="checkbox" name="choix" checked />

L'attribut *checked* permet de faire en sorte qu'une case soit cochée par défaut.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Les formulaires</title>
</head>
<body>
  <form method="POST" action="traitement.php">
    <p>
      Choisissez votre genre : <br/><br/>
      <input type="checkbox" name="homme"
id="homme"/><label for="homme">Masculin</label><br/>
      <input type="checkbox" name="femme"
id="femme"/><label for="femme">Féminin</label><br/>
    </p>
  </form>
</body>
</html>
```

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités. Elles ressemblent un peu aux cases à cocher mais elles doivent être organisées en groupes. Les options d'un même groupe possèdent le **même** nom (name), mais chaque option doit avoir une valeur (value) différente.

```
<p>
Type de carte bancaire :</p><br />
<input type="radio" name="carte" value="visa" id="visa"/> <label
for="visa">VISA</label><br />
<input type="radio" name="carte" value="amex" id="amex" />
<label for="amex">AmEx</label><br /> <input type="radio"
name="carte" value="mastercard" id="mastercard" /> <label
for="mastercard">Mastercard</label><br />
```

Les liste déroulante est un autre moyen de faire un choix parmi plusieurs possibilités. Pour les mettre en oeuvre, on va utiliser la balise <select> </select> qui indique le début et la fin de la liste déroulante. On ajoute l'attribut name à la balise pour donner un nom à la liste. Puis, à l'intérieur du <select> </select>, nous allons placer plusieurs balises <option> </option> (une par choix possible). On ajoute à chacune d'elles un attribut value pour pouvoir identifier ce que le visiteur a choisi.

Il est possible de regrouper plusieurs champs dans un seul cadre, et ça en utilisant la balise <fieldset> </fieldset> qui peut contenir une légende avec la balise <legend> </legend>. Vous pouvez aussi, faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut required.

* Finaliser et envoyer le formulaire

Enfin, pour la soumission du formulaire on utilise toujours la balise <input /> qui existe en quatre versions :
type="submit" : le principal bouton d'envoi de formulaire et le plus utilisé.
type="reset" : remise à zéro du formulaire.
type="image" : équivalent du bouton submit, présenté cette fois sous forme d'image. Rajoutez l'attribut src pour indiquer l'URL de l'image.
type="button" : bouton générique, qui n'aura (par défaut) aucun effet. En général, ce bouton est géré en JavaScript pour exécuter des actions sur la page.
On peut changer le texte affiché à l'intérieur des boutons avec l'attribut value.

Exemple : <input type="submit" value="Envoyer" />

2.3.6 Les styles CSS

La création de styles CSS (Cascading Style Sheets ou feuilles de style en cascade) est le complément indispensable du langage HTML 5. D'une

part, cette séparation permet d'alléger les pages en centralisant les définitions des styles en un point unique, une seule définition pouvant s'appliquer à un grand nombre d'éléments. D'autre part, elle facilite également la maintenance et l'évolution des sites par voie de conséquence. Elle apporte aussi une plus grande rigueur dans la conception des pages et peut permettre un travail collaboratif entre plusieurs programmeurs opérant en parallèle, d'où une réduction des délais de fabrication.

On peut écrire du code en langage CSS à trois endroits différents :
- dans un fichier .css (méthode la plus recommandée) ;
- dans l'en-tête <head> du fichier HTML ;
- directement dans les balises du fichier HTML via un attribut style (méthode la moins recommandée).

1. Dans un fichier .css

Le plus souvent, on écrit le code CSS dans un fichier spécial ayant l'extension **.css** (contrairement aux fichiers HTML qui ont l'extension .html). C'est la méthode la plus pratique et la plus souple. Cela nous évite de tout mélanger dans un même fichier.

Pour indiquer au fichier HTML quel fichier CSS lui est associé pour la mise en forme on rajoute la ligne suivante dans la balise <head> du fichier HTML : <link rel="stylesheet" href="nomfichier.css" />

2. Dans l'en-tête <head> du fichier HTML

Il existe une autre méthode pour utiliser du CSS dans ses fichiers HTML : cela consiste à insérer le code CSS directement dans une balise <style> à l'intérieur de l'en-tête <head>.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<style>
p
{
color : blue ;
}
</style>
```

3. Directement dans les balises

Dernière méthode, vous pouvez ajouter un attribut style à n'importe quelle balise. Vous insérerez votre code CSS directement dans cet attribut :

```
<body>
<p style="color : blue ;">Bonjour et bienvenue sur mon site !</p>
</body>
```

* Des commentaires dans du CSS

Pour faire un commentaire dans du CSS, on utilise /*, suivi de votre commentaire, puis */ pour terminer votre commentaire. Vos commentaires peuvent être écrits sur une ou plusieurs lignes.

* Appliquer un style : sélectionner une balise

Dans un code CSS on trouve trois éléments différents :

- Un sélecteur qui va déterminer à quel balise et éventuellement dans quelles conditions va s'appliquer le style.
- La déclaration des propriétés que l'on veut voir appliquées à la balise sélectionné. Elle doit être incluse entre des accolades ouvrante ({) et fermante (}).
- Dans ces accolades doivent apparaître une ou plusieurs propriétés, déterminées chacune par un mot-clé propre à CSS suivi du caractère deux-points (:), puis de la valeur attribuée à cette propriété. Si nous définissons plusieurs propriétés dans le même style, il faut séparer chaque déclaration de la précédente par le caractère point-virgule (;).

Schématiquement, une feuille de style CSS ressemble donc à cela :

```
balise1
{
propriete1 : valeur1 ;
propriete2 : valeur2 ;
propriete3 : valeur3 ;
}
balise2
{
propriete1 : valeur1 ;
propriete2 : valeur2 ;
propriete3 : valeur3 ;
propriete4 : valeur4 ;
}
balise3
{
propriete1 : valeur1 ;
}
```

on écrit le nom de la balise (par exemple h1) et on ouvre des accolades pour, à l'intérieur, mettre les propriétés et valeurs que l'on souhaite. On peut mettre autant de propriétés que l'on veut à l'intérieur des accolades. Chaque propriété est suivie du symbole « deux-points » (:) puis de la valeur correspondante. Enfin, chaque ligne se termine par un point-virgule (;).

Appliquer un style à plusieurs balises

Nous pouvons très facilement appliquer le même style à plusieurs éléments différents en les énumérant et en les séparant par une virgule dans le sélecteur. Plutôt que de multiplier les définitions :

```
h1 {color : black ; background-color : red ;}
div {color : black ; background-color : red ;}
p {color : black ; background-color : red ;}
```

Nous pouvons écrire, directement, le style suivant :

```
h1, div, p
{
color : black ;
background-color : red ;
}
```

Les sélecteurs avancés

Avant toutes mise en forme, il faut sélectionner la partie correctement.

*** : sélecteur universel**

Pour appliquer un style à tous les éléments, nous utiliserons le sélecteur universel * avant la définition d'une ou plusieurs propriétés. Par exemple, pour que la couleur de fond de tous les éléments soit jaune, nous écrirons :

```
*
{
background-color : yellow ;
}
```

Cela n'empêche pas de modifier cette couleur de fond pour une balise particulière, en la redéfinissant uniquement pour celui-ci, par exemple :

```
*{
background-color : yellow ;
}
p{
background-color : gray ;
}
```

Dans ce cas, toutes les balises ont un fond jaune, sauf <p> qui a un fond gris redéfini spécialement.

A B : une balise contenue dans une autre

```
h3 em
{ }
```

Sélectionne toutes les balises situées à l'intérieur d'une balise <h3>. Notez qu'il n'y a **pas de virgule entre les deux noms de balises.**

A + B : une balise qui en suit une autre

```
h3 + p
{ }
```

Sélectionne la première balise <p> située après un titre <h3>.

A[attribut] : une balise qui possède un attribut

```
a[title]
{ }
```

Sélectionne tous les liens <a> qui possèdent un attribut title.

A[attribut="Valeur"] : une balise, un attribut et une valeur exacte

a[title="Cliquez ici"]
{ }

Idem, mais l'attribut doit en plus avoir exactement pour valeur « Cliquez ici ».

A[attribut*="Valeur"] : une balise, un attribut et une valeur

a[title*="ici"]
{ }

Idem, l'attribut doit cette fois contenir dans sa valeur le mot « ici » (peu importe sa position).

Appliquer un style : class et id

Pour appliquer un style défini à une balise particulière, on peut utiliser des attributs spéciaux qui fonctionnent sur toutes les balises : l'attribut *class* et l'attribut *id*. C'est des attributs que l'on peut mettre sur n'importe quelle balise. Ces deux attributs sont quasiment identiques, la seule différence entre les deux est que la valeur de *id* doit être unique dans tout le fichier HTML contrairement à l'attribut *class* où on peut avoir plusieurs balises qui ont la même valeur de l'attribut *class*.

Une fois la balise identifiée, nous pourrons utiliser le nom de la balise dans le fichier CSS précédé d'un point (.).

Le nom de la classe peut être un mot quelconque, en évitant quand même les noms des propriétés CSS et des balises HTML 5 car cela occasionnerait des confusions. Nous pouvons par exemple définir la classe nommée *evidence* en écrivant le code HTML :

```
<p class="evidence">Texte contenu du paragraphe</p>
```

Pour rendre ce paragraphe, par exemple en rouge, le CSS doit contenir le code suivant :

```
.evidence
{
color : red ;
}
```

Si nous utilisons *id* en lieu de *class*, lorsque nous définirons leurs propriétés dans le fichier CSS, il faudra faire précéder le nom de l'id par un dièse (#).

Balise universelle

Il arrivera parfois que vous ayez besoin d'appliquer un attribut *class* (ou un id) à certains mots qui, à l'origine, ne sont pas entourés par des balises. Pour cela, on utilise des balises dites universelles, qui n'ont aucune signification particulière. Ces deux balises sont :

—   : c'est une balise de type inline, c'est-à-dire une balise que l'on place au sein d'un paragraphe de texte, pour sélectionner certains mots uniquement.

—   : c'est une balise de type block, qui entoure un bloc de texte, elle est une balise fréquemment utilisée dans la construction d'un design.

2.3.7 Formatage du texte

La taille

Pour modifier la taille du texte, on utilise la propriété CSS font-size. La taille peut être définie de manière absolue ou relative.

1. **Indiquer une taille absolue** : en pixels (px), en centimètres (cm) ou millimètres (mm). Cette méthode est très précise mais il est conseillé de ne l'utiliser que si c'est absolument nécessaire, car on risque d'indiquer une taille trop petite pour certains lecteurs. exp : font-size : 16px ;

2. **Indiquer une taille relative** : en pourcentage, « em » ou « ex », cette technique a l'avantage d'être plus souple. Elle s'adapte plus facilement aux préférences de taille des visiteurs.

   Il y a plusieurs moyens d'indiquer une valeur relative.

   — La première méthode est d'utiliser les sept tailles disponibles suivante :
   - xx-small : minuscule ;
   - x-small : très petit ;
   - small : petit ;
   - medium : moyen ;
   - large : grand ;
   - x-large : très grand ;
   - xx-large : très très grand.

   — Cependant, cette technique nous nous offre que sept tailles disponibles. Un autre moyen d'indiquer la taille d'un texte est l'utilisation de "em" :
   - Si vous écrivez 1em, le texte a une taille normale.

- Si vous voulez grossir le texte, vous pouvez inscrire une valeur supérieure à 1, comme 1.3em.
- Si vous voulez réduire le texte, inscrivez une valeur inférieure à 1, comme 0.8em.

— D'autres unités sont disponibles. Vous pouvez essayer le "ex", qui fonctionne sur le même principe que le "em" mais qui est plus petit de base, et le pourcentage (80%, 130%...).

La police

La propriété CSS qui permet d'indiquer la police à utiliser est font-family. Vous devez écrire le nom de la police comme ceci :

```
balise
{
font-family : police ;
}
```

Seulement, pour éviter les problèmes si l'internaute n'a pas la même police que vous, on précise en général plusieurs noms de police, séparés par des virgules :

```
balise
{
font-family : police1, police2, police3, police4 ;
}
```

Le navigateur essaiera d'abord d'utiliser la police1. S'il ne l'a pas, il essaiera la police2. S'il ne l'a pas, il passera à la police3, et ainsi de suite. En général, on indique en tout dernier serif, ce qui correspond à une police par défaut (qui ne s'applique que si aucune autre police n'a été trouvée).

Voici une liste de polices qui fonctionnent bien sur la plupart des navigateurs :

— Arial ;
— Arial Black ;
— Comic Sans MS ;
— Courier New ;
— Georgia ;
— Impact ;
— Times New Roman ;

— Trebuchet MS ;
— Verdana.

Utiliser une police personnalisée avec @font-face

CSS 3 offre un moyen d'utiliser n'importe quelle police sur son site. Cela fonctionne bien avec la plupart des navigateurs, cependant : il faudra que le navigateur de vos visiteurs télécharge automatiquement le fichier de la police, dont le poids peut atteindre, voire dépasser 1 Mo et il existe plusieurs formats de fichiers de polices et ceux-ci ne fonctionnent pas sur tous les navigateurs.

Voici les différents formats de fichiers de polices qui existent et qu'il faut connaître :

— .ttf : TrueType Font. Fonctionne sur IE9 et tous les autres navigateurs.
— .eot : Embedded OpenType. Fonctionne sur Internet Explorer uniquement, toutes versions. Ce format est propriétaire, produit par Microsoft.
— .otf : OpenType Font. Ne fonctionne pas sur Internet Explorer.
— .svg : SVG Font. Le seul format reconnu sur les iPhones et iPads pour le moment.
— .woff : Web Open Font Format. Nouveau format conçu pour le Web, qui fonctionne sur IE9 et tous les autres navigateurs.

En CSS, pour définir une nouvelle police, vous devez la déclarer comme ceci :

```
@font-face
{
font-family : 'MaSuperPolice' ;
src : url('MaSuperPolice.eot') ;
}
```

Le fichier de police (ici MaSuperPolice.eot) doit ici être situé dans le même dossier que le fichier CSS (ou dans un sous-dossier, si vous utilisez un chemin relatif). L'idéal est de proposer plusieurs formats pour la police : le navigateur téléchargera celui qu'il sait lire. Voici comment indiquer plusieurs formats :

```
@font-face {
font-family : 'MaSuperPolice' ;
src : url('MaSuperPolice.eot') format('eot'),
url('MaSuperPolice.woff') format('woff'),
url('MaSuperPolice.ttf') format('truetype'),
url('MaSuperPolice.svg') format('svg') ; }
```

*** Mettre en italique**

En CSS, pour mettre un texte en italique, on utilise font-style qui peut prendre trois valeurs :

— italic : le texte sera mis en italique.
— oblique : le résultat est légèrement différent de l'italique proprement dit.
— normal : le texte sera normal. Cela vous permet d'annuler une mise en italique. Par exemple, si vous voulez que les textes entre ne soient plus en italique.

*** Mettre en gras**

La propriété CSS pour mettre en gras est font-weight et prend les valeurs suivantes :

— bold : le texte sera en gras ;
— normal : le texte sera écrit normalement. Par exemple, si vous voulez que les textes entre ne soient plus en gras.

*** Soulignement et autres décorations**

La propriété CSS text-decoration permet, entre autres, de souligner le texte, mais pas seulement. Voici les différentes valeurs qu'elle peut prendre :

— underline : souligné.
— line-through : barré.
— overline : ligne au-dessus.
— blink : clignotant. Ne fonctionne pas sur tous les navigateurs (Internet Explorer et Google Chrome, notamment).
— none : normal (par défaut).

Nous pouvons également enrichir la présentation du texte à l'aide de la propriété font-variant qui peut prendre les valeurs suivantes :

— small-caps : cette valeur permet de transformer toutes les minuscules d'un texte en petites majuscules, les majuscules conservant leur taille normale.
— normal : permet de conserver pour le texte par défaut.

Grâce à la propriété text-transform, il est possible d'agir sur la casse du texte d'un élément. Elle est héritée par tous les éléments enfants et elle peut prendre les valeurs suivantes :

— uppercase : le texte est mis en majuscules.
— lowercase : le texte est mis en minuscules.
— capitalize : seule la lettre initiale de chaque mot est mise en majuscules.

*** Ombrage de texte en CSS 3**

Dans le domaine des effets graphiques, CSS 3 permet de créer une ombre pour un texte quelconque inclus dans un élément, grâce à la propriété text-shadow dont la syntaxe est :

text-shadow : X Y Z couleur ;

dans laquelle X et Y sont des nombres entiers qui désignent le décalage respectivement horizontal et vertical de l'ombre (à droite et en bas si les nombres sont positifs, l'inverse sinon) et Z le flou de l'ombre (valeur 0 pour une ombre aussi nette que le texte). Voici un exemple :

```
h1
{
text-shadow : 4px -4px 2px #F00 ;
}
```

*** L'alignement du texte**

Le langage CSS nous permet de faire tous les alignements connus : à gauche, centré, à droite et justifié. Pour cela on utilise la propriété text-align et on indique l'alignement désiré :

- left : le texte sera aligné à gauche (c'est le cas par défaut).
- right : le texte sera aligné à droite.
- center : le texte sera centré.
- justify : le texte sera justifié.

Faire flotter une image

Pour faire flotter une image dans du texte, on utilise la propriété float. Cette propriété peut prendre deux valeurs :

- left : l'élément flottera à gauche.
- right : l'élément flottera à droite

Couleurs et images de fond

Pour modifier la couleur du texte en CSS, on utilise la propriété color. pour définir la valeur de *color*, nous pourrons utilisé l'une des trois méthodes suivantes :

- Indiquer la couleur directement en anglais ; exemple : color : red | blue | green...
- Utiliser une valeur hexadécimale à trois ou six positions représentant les composantes RGB de la couleur précédée du caractère (#) (comme

#F3C ou #FA3258).
— Enfin à l'aide de la fonction rgb( ) qui doit avoir trois paramètres entiers variant de 0 à 255, représentant également les composantes RGB en notation décimale (par exemple rgb(56,250,20)).

Nom de la couleur	Triplet RGB	en français
aqua	#00FFFF	Vert d'eau
black	#FFFFFF	Noir
blue	#0000FF	Bleu
fuchsia	#FF00FF	Fuchsia
gray	#808080	Gris
green	#008000	Vert
lime	#00FF00	Ciron vert
maroon	#800000	Marron
navy	#000080	Bleu marine
olive	#808000	Olive
purple	#800080	Pourpre
red	#FF0000	
silver	#C0C0C0	Argent
teal	#008080	Sarcelle
white	#FFFFFF	Blanc
yellow	#FFFF00	Jaune

FIGURE 2.1 – Les 16 couleur du CSS

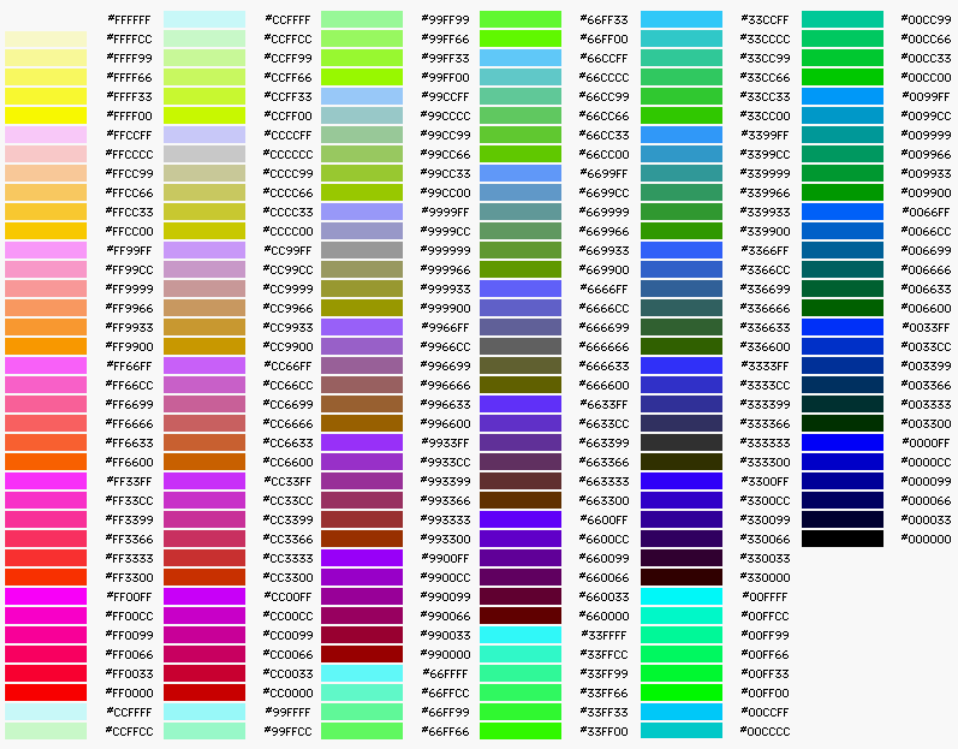


FIGURE 2.2 – Couleurs en hexadécimale

La couleur de fond d'un élément est définie par la propriété background-color que nous pouvons appliquer à tous les éléments HTML 5.

background-color :couleur ;

Par défaut, l'arrière-plan est transparent et laisse donc apparaître la couleur de fond de la balise- parent.

*** Couleurs de fond en dégradé**

Pour obtenir des dégradés linéaires, la syntaxe à utiliser est la suivante :

background :linear-gradient(parametre1, couleurl X%,couleur2 Y%,...couleur N1%) ;

parametre1 peut prendre les valeurs suivantes :

— to top : vers le haut
— to bottom : vers le bas (par défaut)
— to right : vers la droite
— to left : vers la gauche
— to top right : vers en haut à droite
— to bottom left : vers en bas à gauche
— etc.

La couleur donnée en premier sera placée à gauche ou en haut, selon l'orientation. Les valeurs X et Y représentent la répartition du dégradé dans le conteneur et sont facultatives.

Exemple : background : linear-gradient(to top, black, #660066, white) ; ce qui donne le résultat suivant :

Image de fond : Pour appliquer une image comme un fond, on utilise la propriété background-image. Comme valeur, on doit renseigner : url("nomimage.png").

Par exemple :

```
    body
{
background-image : url("neige.png") ;
}
```

On peut compléter la propriété background-image par plusieurs autres propriétés qui permettent de changer le comportement de l'image de fond.

Par défaut, l'image de fond est répétée horizontalement et verticalement, et occupe tout l'espace disponible de l'élément. Pour modifier cette caractéristique, nous pouvons limiter le type de répétition à l'aide de la propriété background-repeat qui peut prendre l'une des valeur suivante :

- no-repeat : le fond ne sera pas répété.
- repeat-x : le fond sera répété uniquement sur la première ligne, horizontalement.
- repeat-y : le fond sera répété uniquement sur la première colonne, verticalement.
- repeat : le fond sera répété en mosaïque (par défaut).

En utilisant la valeur no-repeat, l'image de fond sera placée dans l'angle supérieur gauche. Si cette position par défaut ne convient pas, il est possible d'en définir une autre explicitement à l'aide de la propriété background-position qui peut prendre les valeurs suivants :

- Deux valeurs en pixels pour indiquer la position du fond par rapport au coin supérieur gauche de la page, par exemple : background-position : 30px 50px ;, le fond sera placé à 30 pixels de la gauche et à 50 pixels du haut.
- Utiliser, directement, des valeurs en anglais :
  - top : en haut ;
  - bottom : en bas ;
  - left : à gauche ;
  - center : centré ;
  - right : à droite.

  Il est possible de combiner ces valeurs. Par exemple, pour aligner une image en haut à droite, on tape :
  background-position : top right ;

CSS nous offre aussi la possibilité d'autoriser ou non le défilement de l'image en utilisant la propriété background-attachement qui peut avoir soit la valeur *scroll* qui correspond au comportement par défaut qui implique le défilement de l'image avec le contenu, la valeur *fixed* permet de conserver l'image à sa position initiale, définie par background-position.

Une autre possibilité intéressante ajoutée dans CSS 3 est la propriété background-size qui permet de dimensionner une ou plusieurs images de fond indépendamment des dimensions réelles des images. Sa syntaxe est la suivante :

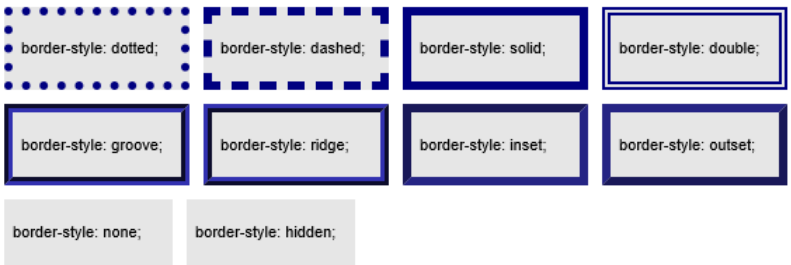
background-size : Xl px Y1 px,.....Xn px Yn px ;

Les valeurs de X et Y désignant les dimensions horizontales et verticales désirées de chacune des images et pouvant aussi être données en pourcentage. En donnant la valeur *cover* à la propriété background-size pour un élément ayant une image de fond unique, l'image de fond choisie est étendue pour occuper tout l'espace disponible dans le conteneur de l'élément.

*** Les bordures et les ombres**

CSS 3 propose plusieurs propriétés pour donner de nouveaux aspects aux bordures. L'une d'elles est la propriété border, qui peut prendre trois valeurs différente pour modifier les bordures, à savoir, la largeur de la bordure (en pixel), la couleur et la type de la bordure qui peut prendre l'une des valeurs suivante :

— none : pas de bordure (par défaut) ;
— solid : un trait simple ;
— dotted : pointillés ;
— dashed : tirets ;
— double : bordure double ;
— groove : en relief ;
— ridge : autre effet relief ;
— inset : effet 3D global enfoncé ;
— outset : effet 3D global surélevé.



Ainsi, pour modifier le cadre de fieldset, par exemple, pour avoir une bordure grise, en tirets, épaisse de 3 pixels, nous allons écrire :

```
    fieldset
{

    border : 3px gray dashed ;
}
```

Dans le cas où vous voulez mettre des bordures différentes en fonction du côté (haut, bas, gauche ou droite), vous devrez utiliser les propriétés CSS suivantes :

— border-top : bordure du haut ;
— border-bottom : bordure du bas ;
— border-left : bordure de gauche ;
— border-right : bordure de droite.

CSS 3 nous offre aussi la possibilité d'arrondir les bordures, à l'aide de la propriété border-radius

```
    fieldset
{

    border-radius : 10px ;
}
```

On peut aussi préciser la forme de l'arrondi pour chaque coin. Dans ce cas, indiquez quatre valeurs :

```
    fieldset
{

    border-radius : 20px 5px 10px 15px ;
}
```

Les valeurs correspondent aux angles suivants dans cet ordre :

1. en haut à gauche ;
2. en haut à droite ;
3. en bas à droite ;
4. en bas à gauche.

Comme nous l'avons déjà vu pour le texte, CSS nous offre la possibilité de créer des ombres. On peut créer aussi des ombres de boite en utilisant la propriété box-shadow qui s'applique à tout le bloc et prend quatre valeurs dans l'ordre suivant :

1. le décalage horizontal de l'ombre ;
2. le décalage vertical de l'ombre ;
3. l'adoucissement du dégradé ;
4. la couleur de l'ombre.

fieldset
{

box-shadow : 6px 6px 0px black ;
}

*** Création d'apparences dynamiques**

CSS nous permet aussi de modifier l'apparence des éléments de façon dynamique, l'une des propriété CSS qui nous permet cela est :hover qui permet d'apporter des modification lors du survole de l'élément au quel elle est appliquée. Contrairement aux autres propriété CSS, on rajoute *:hover* après le nom de la balise ou de la classe dans le CSS, comme ceci :

a :hover
{
color : red ;
}

Cet exemple permet d'afficher les liens en rouge lorsque l'utilisateur pose la souris sur le lien.

CSS 3, nous offre aussi la propriété :active qui permet d'appliquer un style particulier au moment du clic. En pratique, il n'est utilisé que sur les liens. Par exemple :

a :active /* Quand le visiteur clique sur le lien la couleur du texte deviendra jaune */
{
color : yellow ;
}

La propriété :focus quand à elle apporte des modification lorsque l'élément est sélectionné. Vous pouvez aussi appliquer un style à un lien vers une page qui a déjà été vue. Par défaut, le navigateur colore le lien en un violet pour changer cela on utilise :visited. Par exemple :

a :visited
{
color : #00FFFF ; }

Les dimensions des block

Pour modifier les dimensions de n'importe quel block (div, footer, header, aside...), CSS 3 mit à notre disposition deux propriétés :

- width : c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).
- height : c'est la hauteur du bloc. On l'exprime soit en pixels (px), soit en pourcentage (%).

*** Les marges**

Afin d'aérer le contenu d'une page et en particulier l'espace entre le rendu d'un élément et ses voisins dans la page, nous pouvons définir une marge autour de chaque élément. Pour définir la largeur des marges d'un élément, nous disposons de la propriété margin qui indique la taille de la marge extérieure qu'on indique généralement en pixels (px) mais elle peut être aussi exprimé en utilisant d'autre unités (ex, em, %, mm, cm,). Les marges peuvent être négatives, et dans ce cas, la boîte d'un élément sort de celle de son parent. Par exemple :

```
div
{
margin :  2px 0 3px 1px ; }
```

Cela signifie 2 px de marge en haut, 0 px à droite, 3 px en bas et 1 px à gauche. On peut agir dynamiquement sur une seule des marges :

- margin-top : définit la marge haute ;
- margin-right : définit la marge droite ;
- margin-bottom : définit la marge basse ;
- margin-left : définit la marge gauche.

*** Les espacements**

Il est possible de définir une zone située entre la boîte de contenu d'un élément et sa bordure. Cette zone qui est nommée l'espacement (padding) permet, comme la propriété margin , d'aérer la présentation mais cette fois non pas entre deux éléments voisins, mais directement autour du contenu. Cet espacement est créé par la propriété padding qui s'applique à tous les éléments HTML 5, excepté ceux qui sont inclus dans la balise <table>.

```
div
{
padding :  12px 15px 3px 1px ; }
```

Cela signifie 12 px de marge en haut, 15 px à droite, 3 px en bas et 1 px à gauche. Comme pour les propriétés de bordure ou de marge exposées précédemment, il est encore possible de définir individuellement chacun des

espacements d'un élément au moyen des propriétés suivantes :

- padding-top : définit l'espacement haut.
- padding-right : définit l'espacement droit.
- padding-bottom : définit l'espacement bas.
- padding-left : définit l'espacement gauche.

Si vous voulez que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété overflow. Voici les valeurs qu'elle peut accepter :

- visible (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
- hidden : si le texte dépasse les limites, il sera coupé. On ne pourra pas voir tout le texte.
- scroll : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte.
- auto : c'est le mode pilote automatique. En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire).

*** Centrer des blocs**

Pour centrer un bloc, il faut respecter les règles suivantes :
— donnez une largeur au bloc (avec la propriété width) ;
— indiquez que vous voulez des marges extérieures automatiques, comme ceci : margin : auto ;.

*** La propriété *display***

La propriété CSS display est la plus importante pour contrôler la mise en page. Chaque élément à une valeur display par défaut dépendante du type de l'élément. Principalement nous avons deux types de balise HTML :

- **inline :** comme par exemple les balise <a>, , , ces éléments se placent les uns à côté des autres.
- **block :** Les balises <p>, , <section>, sont des balises de type block.

La propriété permet de transformer n'importe quel élément de votre page d'un type vers un autre. Par exemple les liens sont de type *inline*, donc ils sont affiché par défaut les uns à coté des autres pour changer ce comportement on utilise *display*, comme ceci :

```
a
{
display : block ;
}
```

Grâce à ça, les liens vont se positionner les uns en-dessous des autres et il devient possible de modifier leurs dimensions.

Voici quelques-unes des principales valeurs que peut prendre la propriété display en CSS :

- **inline :** Eléments d'une ligne. Se placent les uns à côté des autres.
- **block :** Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
- **inline-block :** Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
- **none :** Eléments non affichés.

On peut donc décider de masquer complètement un élément de la page avec cette propriété en lui donnant la valeur *none*. Pour faire apparaître ces éléments par la suite, vous devrez faire appel à JavaScript. La valeur *inline-block* permet de faire une combinaison des inlines et des blocs. Les éléments s'affichent côte à côte et peuvent être redimensionnés.

Dans ce cas il serait intéressant d'utiliser la propriété vertical-align qui permet de modifier l'alignement vertical des éléments. Voici quelques-unes des valeurs possibles pour cette propriété :

- **baseline :** aligne de la base de l'élément avec celle de l'élément parent (par défaut) ;
- **top :** aligne en haut ;
- **middle :** centre verticalement ;
- **bottom :** aligne en bas ;
- **(valeur en px ou %) :** aligne à une certaine distance de la ligne de base (baseline).

*** Style des tableaux**

La balise <caption>, contient le titre du tableau. Par défaut, en HTML 5 pur, il est placé au-dessus du tableau. Nous pouvons déterminer explicitement sa position à l'aide de la propriété caption-side dont la syntaxe est la suivante :

```
caption-side : top | bottom ;
```

Les différentes propriétés de création des bordures exposées, sont applicables aux cellules des tableaux. Pour améliorer la gestion des bordures et en particulier celle des cellules contiguës, CSS fournit la propriété border-collapse qui permet de fusionner ou de séparer ces bordures voisines. Sa syntaxe est la suivante :

```
border-collapse : collapse | separate ;
```

A la valeur *collapse* correspond la fusion des bordures, et à la valeur *separate* leur séparation.

Chapitre 3

Les langages de script côté client

3.1 Javascript

Le JavaScript est un langage de programmation exécuté côté client. Il vient s'insérer dans une page HTML et permet de faire interagir le visiteur avec la page, en réagissant différemment selon les actions de celui-ci. Le JavaScript est un langage orienté objet, cela signifie que quand on code en JavaScript, on se base sur des objets.

L'intérêt du JavaScript est qu'il n'est pas obligatoirement exécutés au chargement de la page. Il peut être lancé lorsqu'un événement spécifique se produit. Généralement le code Javascript dans une page Web sert à :

- Faire bouger, apparaitre ou disparaitre des éléments de la page (un titre, un menu, un paragraphe, une image...).
- Mettre à jour des éléments de la page sans recharger la page (changer le texte, recalculer un nombre, etc).
- Attendre que l'utilisateur face quelque chose (cliquer, taper au clavier, bouger la souris...) et réagir (faire une opération) suite à cette action.

Javascript permet aujourd'hui, avec le support d'autres technologies (Flash, HTML5, canvas, CSS3, WebGL...), de faire des choses très évoluées comme de la 3D, de la manipulation d'images, de sons et de videos.

Remarque : attention à ne pas confondre JavaScript et Java. Ces deux langages, bien que syntaxiquement assez proches à la base, reposent sur des concepts fondamentaux complètement différents et servent à effectuer des tâches totalement différentes. Il faut aussi faire attention, comme l'internaute a le contrôle de sa machine, il peut choisir de désactiver le support de Javascript sur son navigateur. Dans ce cas, ce dernier ignorera le code Javascript et fera comme s'il n'était pas là et tous les éléments qui fonctionnent avec Javascript ne marcheront pas.

3.1.1 Insérer du JavaScript dans une page HTML

Pour coder en JavaScript, nous n'allons avoir besoin que d'un éditeur de texte, par exemple on peut utilisé *Bloc note* ou *Notepad++*.

Il y a deux manières d'ajouter du code JavaScript dans une page :

1. en liant depuis la page HTML un fichier externe, dans lequel sont placées les instructions JavaScript, l'extension du fichier externe contenant du code JavaScript est **.js**. On va indiquer aux balises le nom et l'emplacement du fichier contenant notre script, grâce à la propriété *src* :
   <script type="text/javascript" src="monscript.js"></script>
2. en ajoutant le code JavaScript à l'intérieur de la balise *<script> et </script>*, on peut les placer soit dans l'en-tête (<head> ... </head>), soit dans le corps (<body> ... </body>) de la page HTML :

   <script type="text/javascript">
   // Mon code Javascript
   ...
   </script>

3.1.2 Les éléments du langage

Les commentaires

Il y a deux manières d'écrire un commentaire sous JavaScript.

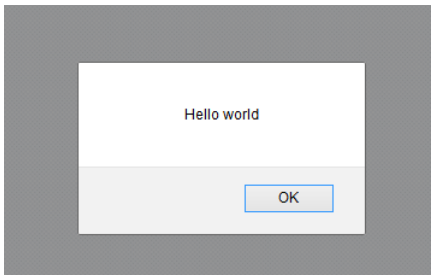
- Les commentaires sur une seule ligne : ils se placent sur la fin de la ligne, après // (deux slashs).

- Les commentaires sur plusieurs lignes : ils se placent entre /* et */ il n'y a ici aucune limitation sur la longueur du commentaire.

Afficher une boîte de dialogue

Le code est celui-ci :
alert('Hello world') ;

"Hello world" est le texte à afficher, vous pouvez le remplacer par le texte de votre choix, en laissant les apostrophes de part et d'autre de votre message. Le résultat obtenu est :



Les variables

En informatique, une variable permet de stocker une valeur, une information, qu'elle soit numérique, sous forme de texte ou autre. En JavaScript on a 4 types de base :

— **entier :** 127 (base 10), 0755 (base 8), 0xFA15 (base 16)
— **flottant :** 0.123, -0.4
— **booléen :** true, false
— **chaîne de caractères :** "chaine" ou 'chaine'

Afin d'utiliser une variable il faut d'abord la déclarer, c'est à dire lui réserver un espace de stockage en mémoire. Pour déclarer une variable, il suffit d'écrire :

```
var nomvariable ;
```

Il est à noter qu'après chaque instruction, il faut mettre un point-virgule (**;**) et que le JavaScript est sensible à la casse (différence entre lettre majuscule et minuscule). Pour le nom des variables il faudra faire attention à ne pas choisir des mots réservés.

Exemple :

```
var nbr = 10 ;
var fl = 3.141 ;
var str1 = "L'étoile" ;
var str2 = 'brille' ;
var lien = '<a href="index.htm">Home</a>' ;
```

En JavaScript, contrairement à de nombreux langages de programmation, lors de la déclaration d'une variable on ne précise pas le type de cette dernière on dit que le langage est *typé dynamiquement*. Cela veut aussi dire que l'on peut y mettre du texte en premier lieu puis l'effacer et y mettre un nombre quel qu'il soit, et ce, sans contraintes.

Les Opérateurs

Opérateur	Signification
==	égal à
!=	différant de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à
&&	ET logique
||	OU logique
!	NON logique
/	division
-	soustraction
	multiplication
+	addition pour les nombres et concaténation pour les chaînes de caractères

Les conditions

Les conditions permettent d'exécuter une action en fonction d'une condition qui sera vraie ou fausse en fonction du résultat d'une comparaison ou d'un test logique. La syntaxe se présente sous la forme suivante :

```
if (test)
{
action;
}
else
{
autreAction;
}
```

Exemple :

```
var moy = 12;
if(moy >= 10){
alert('Vous avez la moyenne');
}else{ alert("Il faut faire plus d'effort"); }
```

Les boucles

Les boucles permettent de répéter une action plusieurs fois tant qu'une condition est respectée. JavaScript nous offre trois types de boucle :

La boucle while

```
while (condition) {
action1 ;
action2 ;
}
```

La boucle *do while* : Cette boucle ressemble très fortement à la boucle while, sauf que dans ce cas **la boucle est toujours exécutée au moins une fois**. Sa syntaxe est :

```
do {
action ;
} while (condition) ;
```

La boucle for

Cette boucle fonctionne de la même manière que *while* mais l'initialisation, la condition et l'incrémentation se déclare sur une seule ligne, sa syntaxe est la suivante :

```
for (initialisation ; condition ; incrémentation) {
action ;
}
```

Les tableaux

Un tableau est une variable qui contient plusieurs valeurs, appelées items. Chaque item est accessible au moyen d'un indice et dont la numérotation commence à partir de 0.

Pour créer un tableau et l'enregistrer dans une variable, on utilise la syntaxe suivante :

var nomTableau = new Array() ;

Il est possible de créer un tableau contenant, initialement, certaines valeurs. Pour cela, on indique dans l'ordre les valeurs de notre tableau, en les

séparant par des virgules, à l'intérieur des parenthèses après Array.

var noms = new Array("Pierre", "Paul", "Jacques") ;/ On déclare un tableau de chaines de caractères qui a initialement trois valeurs */*
var bestScores = new Array(2.5, 4.7, 3.14) ; / On déclare un tableau de trois réels */*

Pour accéder à un élément d'un tableau , on utilise la syntaxe suivante : nomTableau[indice]. Pour ajouter un nouvel élément, on modifie simplement sa valeur, comme s'il existait déjà.

Exemple :

```
var table = new Array("Pierre", "Paul", "Jacques") ;
alert("La seconde case vaut : " + table[1]) ; // on lit l'élément d'indice 1
table[1] = "Tom" ; // on le modifie
table[3] = "Dupont" ;
```

JavaScript nous offre quelques fonctions prédéfinit pour nous faciliter la programmation. Pour un tableau nommé monTableau, on peut avoir sa longueur grâce à la fonction *length*, comme ceci : monTableau.length. Il existe aussi la fonction *sort()*, qui permet de trier un tableau (par ordre croissant) : monTableau.sort().

Pour parcourir un tableau numéroté on peut utiliser la boucle for : en effet, on veut accéder à tous les tableau[i], avec i allant de 0 à tableau.length -1.

Voici donc une fonction qui retourne, sous forme de chaîne de caractères, le contenu du tableau :

```
function lire1(tab)
{
var chaine = "Le tableau contient :"
for(var i=0 ; i<tab.length ; i++)
chaine += i + " -> " + tab[i] ;
return chaine ;
}
```

Les fonctions

Pour éviter d'exécuter plusieurs fois le même code, on utilise les fonctions. Une fonction est un bout de code qui effectue une tâche ou un calcul relativement indépendant du reste du programme. Une fonction se déclare

comme ceci :

function nomDeLaFonction(argument1, argument2,...) {
CODE
}
Une fonction contient les éléments suivants :

- Un nom
- Une liste d'arguments. Ils permettent de donner des paramètres, des données en entrée à la fonction. Ils ne sont utilisables que dans le corps de la fonction.
- Un corps.
- Eventuellement une valeur de retour. La fonction peut exécuter que du code, mais elle peut également retourner un résultat.

Pour faire appel à une fonction, il suffit de citer le nom de la fonction suivi de la liste des arguments.

3.1.3 Les objets

Comme pour C++ et Java, le JavaScript est aussi un langage orienté objets. Pour rappel, un objet représente une entité du monde réel (ou du monde virtuel pour les objets immatériels) qui se caractérise par un ensemble de propriétés (attributs) et d'un comportement (opérations).

Pour créer un objet en JavaScript, on utilise le mot-clé **new**, comme ceci :

var etudiant1 = new Etudiant("Azzar", "Kenza", "15st1678"); // etudiant1 est le nom de l'objet

Pour accéder à un attribut d'un objet (d'une instance), on utilise le nom de l'objet, suivi d'un point, puis du nom de l'attribut, comme ceci : objet.attribut.

etudiant1.prenom = "Marie";

Pour exécuter une méthode d'un objet (d'une instance), on utilise la même syntaxe que pour accéder à un attribut, sans oublier les parenthèses (puisqu'une méthode est une fonction). Avec notre classe "Etudiant", on va pouvoir créer une méthode qui retournera une description sous forme de caractère. On l'utilisera ainsi :

var message = etudiant1.description();
alert(message);

Destruction d'un objet : Pour détruire un objet, il suffit de lui affilier la valeur *null*.

etudiant1 = null;// nous avons détruit l'objet etudiant1

3.1.4 Les objets HTML

Pour JavaScript, tous les éléments HTML sont des objets (images, liens, etc.) : on va donc pouvoir s'en servir pour en connaître ou en modifier les caractéristiques (comme l'adresse de l'image ou ses dimensions).

L'objet window

L'objet *window* est l'objet de base de JavaScript, il représente la fenêtre du navigateur. En JavaScript, l'objet *window* est dit implicite, c'est-à-dire qu'il n'y a généralement pas besoin de le spécifier dans son code. L'objet *window* possède de nombreux sous-objets, par exemple les fonctions *alert()*, *prompt()* et *confirm()*, sont toutes les trois des fonctions de *window*.

L'objet document

L'objet document est un sous-objet de window. Cet objet représente la page HTML affichée dans le navigateur. C'est un objet assez important et c'est via lui que l'on va pouvoir accéder aux éléments HTML pour ensuite les modifier.

Nous allons passer par ses méthodes pour accéder aux éléments de notre page.

* Les objets du document

Il est possible d'accéder à tous les éléments de la page HTML avec deux méthodes de l'objet document :

* document.getElementById("id");

* document.getElementsByTagName("balise");

- getElementById("id")

Cette méthode permet d'accéder très facilement à l'élément dont l'id est *id*. Comme deux éléments différents doivent avoir des ids différents, cette fonction nous donne donc un seul élément.

- getElementsByTagName("balise") Cette méthode permet de récupérer, sous la forme d'un tableau, tous les éléments de la même famille. Par exemple, pour récupérer tous les * * de la page, il suffit de faire comme ceci :

var divs = document.getElementsByTagName('div') ;

document.getElementsByTagName('textarea') ;

Cet exemple va retourner un tableau contenant tous les éléments <textarea> de la page.

Puisque la méthode retourne un tableau, il est bien sûr possible d'accéder à chacun des éléments. Si on veut accéder au second <textarea> de la page, on procédera donc ainsi (rappelez-vous, on commence à compter à partir de 0) :

document.getElementsByTagName('textarea')[1] ;

Exemples :

monImage = document.getElementsByTagName("img")[0] ;
monImage.src = "banniere.jpg" ;
monImage.width = "800" ;
monImage.height = "200" ;

Par exemple pour afficher la destination d'un lien :

alert(document.getElementById("idLien").href) ;

Les évènements

Les évènements correspondent aux actions effectuées par l'utilisateur (la plus courante étant le clic de souris) qui permettent d'interagir avec le contenu parcouru. Chaque action est associée à un évènement : on associe ainsi au clic de la souris l'évènement "onclick", au double clic l'évènement "ondblclick".

Les évènements sont associés aux balises html, il est ainsi possible de déclencher des évènements sur la grande majorité des balises html.

Voici la liste des événements principaux, ainsi que les actions à effectuer pour qu'ils se déclenchent :

— **onClick et onDblClick :** lors d'un clic / double clic sur l'élément en question

— **onKeyPress :** lorsqu'on appuie sur une touche avec cet élément sélectionné
— **onKeyDown et onKeyUp :** lorsqu'une touche est enfoncée / relâchée avec cet élément sélectionné
— **onMouseOver et onMouseOut :** lorsque le pointeur de la souris arrive sur l'élément / sort de cet élément
— **onMouseMove :** lors d'un déplacement de la souris au-dessus de cet élément
— **onMouseDown et onMouseUp :** lorsque le bouton de la souris est enfoncé / relâché sur cet élément

Exemple

```
<img src="azerty.png" alt="" onclick="alert('Ceci est une image ?');"
/>
```

Chapitre 4

Les langages de script côté serveur

4.1 Environnement de travail

Pour qu'on puisse exécuter un langage serveur (dans ce cours nous allons nous baser sur PHP) sur notre ordinateur, il faudrait que celui-ci se comporte comme un serveur. Pour cela il faudra installer les mêmes programmes que ceux que l'on trouve sur les serveurs qui délivrent les sites web aux internautes :

* Apache : c'est un serveur Web. Il s'agit du plus important de tous les programmes, son rôle est d'écouter les requêtes émises par les navigateurs (qui demandent des pages Web), de chercher la page demandée et de la renvoyer. Cependant, Apache ne peut traiter que des pages HTML. Il faut donc le compléter avec d'autres programmes.

* PHP : c'est un langage de script, il permet de décrire dans une page Web, un affichage dynamique d'information. Les instruction PHP sont généralement contenues dans des fichiers d'extension **.php**.

En clair, en combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.

* MySQL : c'est un système de gestion de bases de données. Il permet d'enregistrer des données de manière organisée sous forme de table et de permettre la manipulation de ces données à travers le langage de requête SQL.

La combinaison Apache, PHP et MySQL est la plus courante sur les serveurs web. Il est possible de les installer un à un mais il existe ce qu'on appelle

des packs qui contiennent tous ces éléments. Il existe plusieurs paquetages pour différents systèmes d'exploitation, sous Windows le paquetage le plus utilisé est **WAMP Server** qui a l'avantage d'être régulièrement mis à jour et disponible en français.

WAMP (Windows Apache MySQL Php) est une plate-forme de développement Web pour Windows exploitant le serveur Web Apache, le langage de script PHP et le SGBD MySQL. Elle intègre également PHPMyAdmin pour gérer facilement les bases de données.

Voici les liens de téléchargement pour les trois systèmes d'exploitation :

MAC OS : https ://www.mamp.info/en/downloads/
Linux : https ://www.apachefriends.org/fr/index.html
Windows : http ://www.wampserver.com/

Une fois téléchargé, installez-le en laissant toutes les options par défaut. Il devrait s'installer dans un répertoire comme C :\wamp et créer un raccourci sur votre bureau.

Lorsque vous lancez WAMP, une icône doit apparaître en bas à droite de la barre des tâches, à côté de l'horloge, comme sur la figure suivante.

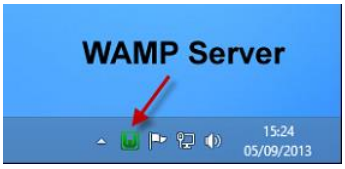


FIGURE 4.1 – Icône de WAMP

Si une fenêtre apparaît pour vous indiquer que le pare-feu bloque Apache, cliquez sur Autoriser l'accès. Vérifiez que vous n'avez pas Skype ouvert en même temps. Les deux programmes ne peuvent pas tourner en parallèle. Dans ce cas, fermez Skype pendant que vous utilisez WAMP.

Une fois WAMP bien installé, vous pouvez lancer la page d'accueil de WAMP. Faites un clic gauche sur l'icône de WAMP puis cliquez sur Localhost, comme le montre la figure 4.2.

Une page web similaire à la capture de la figure 4.3 devrait s'ouvrir dans votre navigateur. Si la page s'affiche chez vous, cela signifie qu'Apache fonctionne.

Comme vous pouvez le voir sur la figure 4.3 la section « Vos projets » est vide. Considérez que chaque site web que vous entreprenez de faire est un nouveau projet.

Nous allons créer un projet de test que nous appellerons *tests*. Pour ce

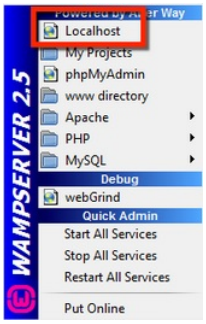


FIGURE 4.2 – Menu localhost de WAMP

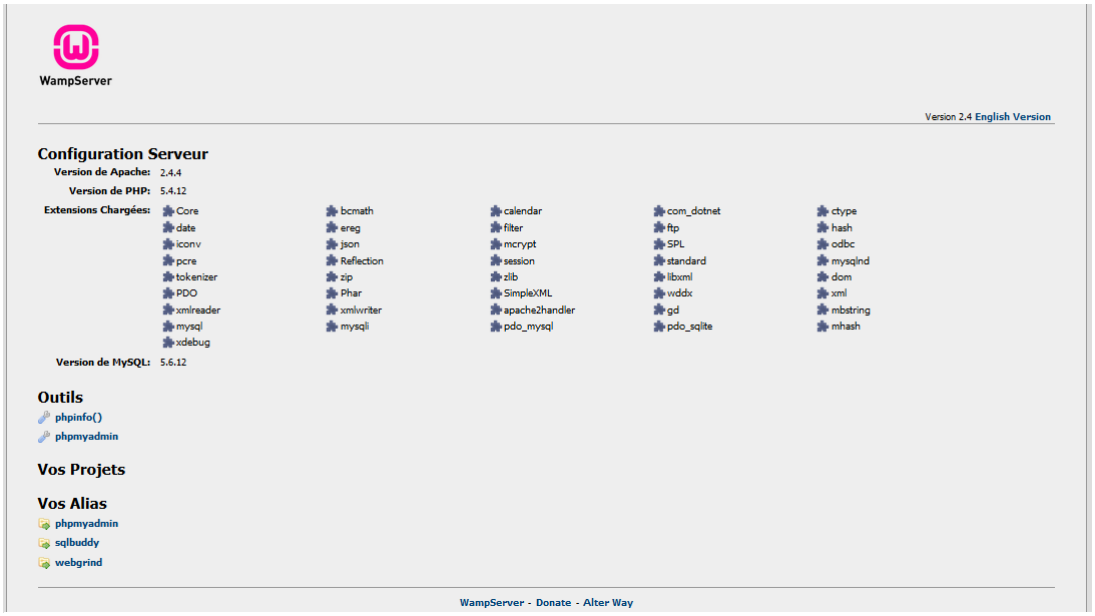


FIGURE 4.3 – Page d'accueil de WAMP

faire, allez dans le dossier où WAMP a été installé, puis dans le sous-dossier intitulé *www*, généralement on le trouve dans : C :\wamp\www. Une fois dans ce dossier, créez un nouveau sous-dossier que vous appellerez *tests* où vous allez créer vos pages Web en PHP. Retournez sur la page d'accueil de WAMP et actualisez-la. La section *Vos projets* devrait maintenant afficher *tests* car WAMP a détecté que vous avez créé un nouveau dossier.

Pour accéder à ce dossier il faudra taper cette URL dans la barre d'adresse : http ://localhost/tests/

4.1.1 Enregistrer une page PhP et l'afficher

Dès que vous ajoutez ne serait-ce qu'une seule ligne de PhP dans votre page HTML, vous devez changer son extension de **.html** à **.php** afin que celle-ci fonctionne correctement.

Ensuite, vous devez absolument enregistrer votre fichier dans le dossier prévu à cet effet de WAMP, MAMP ou XAMPP selon votre installation. Pour WAMP, par exemple, vous devez absolument enregistrer votre fichier dans le dossier **www** (lui-même contenu dans le dossier wamp) tandis qu'avec MAMP le dossier sera le dossier htdocs.

4.2 Connexion et manipulation des bases de données

4.2.1 Concepts sur les bases de données

Base de données (BDD)

Une base de données est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par un groupe d'utilisateurs.

Système de gestion de base de données (SGBD)

Un SGBD est un ensemble de programmes permettant la gestion et l'accès à une BDD. C'est l'interface entre la base de données et l'utilisateur.

Exemple des SGBD : Oracle, SQL-serveur, MySQL, ACCESS.

Définition de table

Une table est un tableau à deux dimensions dans lequel chaque colonne, appelée *attribut* porte un nom différent et où les données figurent en ligne. Aussi bien pour les colonnes que pour les lignes, une table peut en contenir un nombre quelconque et leur ordre est indifférent.

On trouve aussi les vocabulaire d'*enregistrement* ou *tuple* pour désigner une ligne et de *champ* pour une colonne.

Clé primaire

Une clé primaire est un attribut particulier et indispensable pour pouvoir identifier chaque ligne d'une manière unique.

4.2.2 Les fonctions du langage SQL

SELECT, FROM

SELECT est l'opérateur de projection permettant de choisir les attributs à afficher, il est suivi de noms d'attributs ou de "*" pour sélectionner tous les attributs.

FROM est suivi d'un ou de plusieurs noms de table

WHERE

La commande WHERE dans une requête SQL permet d'extraire les lignes d'une base de données qui respectent une condition. Cela permet d'obtenir uniquement les informations désirées.

LIKE

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiple.

INSERT

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO.

4.2.3 Opération sur les bases de données

Pour commencer il faut créer une base de données. Une base de données est constituée d'un ensemble de tables, les tables contenant les données. La base de données correspond donc en quelque sorte à un dossier qui contient des fichier (les tables).

Création d'une base de donnée qui a comme nom exempleBDD

```
CREATE DATABASE exempleBDD ;
```

Suppression d'une base de donnée existante

```
DROP DATABASE exempleBDD IF EXISTS ;
```

L'option IF EXISTS pour savoir s'il y a une présence ou non d'une base de donnée qui a le nom *exempleBDD* pour éviter d'avoir une erreur si elle n'existe pas.

4.2.4 Opération sur les tables

Une fois que la base de données a été définie, il faut ensuite définir les tables qui la composeront avec la liste des champs qui la composent.

Création d'une table

CREATE TABLE etudiant (idEtudiant int NOT NULL auto_increment PRIMARY KEY, nom VARCHAR, prenom VARCHAR) ;

Suppression d'une table

DROP TABLE etudiant ;

4.2.5 Opération sur les enregistrements

Insertion d'enregistrement dans une table

INSERT INTO etudiant (nom, prenom) VALUES ('Aitmokrane','Hind') ;

La commande INSERT ajoute un nouvel enregistrement dans la table *etudiant*. La valeur *idEtudiant* n'ai pas mentionnée car nous l'avons déclaré *auto_ increment* elle sera automatiquement insérer.

Recherche d'enregistrements dans une table

SELECT colonne1, colonne2, colonne3,... FROM nomDeTable WHERE conditions ;

La commande SELECT permet de sélectionner des données afin de les afficher. Par exemple pour afficher tous les étudiants :

SELECT * FROM etudiant ;

Pour afficher les étudiants dont le nom commence par la lettre A :

SELECT nom, prenom FROM etudiant WHERE nom LIKE 'A%' ;

Effacer des données

DELETE FROM nomTable WHERE conditions ;

Par exemple, pour effacer le premier étudiant de la table :

DELETE FROM etudiant WHERE idEtudiant = 1 ;

Modification des données

UPDATE nomTable SET colonne1 = 'nouvelleValeur', colonne2 = 'nouvelleValeur'.... WHERE conditions ;

Par exemple pour modifier la nom du deuxième étudiant :

UPDATE etudiant SET nom = 'Nabet' WHERE idEtudiant = 2 ;

4.3 PHP

PHP (Php Hypertext Preprocessor) est un langage de programmation qui s'exécute sur un serveur et permet de créer des pages Web dynamiques et interactives. Il vient s'insérer au milieu du code HTML. Ces bouts de code PHP seront les parties dynamiques de la page, c'est-à-dire les parties qui peuvent changer toutes seules. Une page interactive permet à un visiteur de saisir des données personnelles. Ces dernières sont ensuite transmises au serveur, où elles peuvent rester stockées dans une base de données pour être diffusées vers d'autres utilisateurs. Un visiteur peut, par exemple, s'enregistrer et retrouver une page adaptée à ses besoins lors d'une visite ultérieure.

Il faut aussi savoir qu'un client (un navigateur) ne peut lire que du HTML et du CSS. Il ne comprend donc pas le PHP. Le rôle principal du PHP va donc être de procéder aux opérations demandées puis de générer et de renvoyer du HTML (ou du CSS) pour chaque visiteur. L'intérêt du PHP reste donc de renvoyer un code HTML (ou CSS) potentiellement différent pour chaque visiteur demandant à voir une page.

4.3.1 La forme d'une balise PHP

Les éléments **<?php** et **?>** marquent respectivement le début et la fin de tout script PHP, qu'il soit inclus dans du code HTML ou isolé dans un fichier ne contenant que du code PHP. Vous pouvez inclure autant de blocs

de code PHP que vous le désirez dans un document HTML, à condition que chacun d'eux soit délimité par ces marqueurs.

4.3.2 L'instruction *echo*

L'une des instruction PHP simple et la plus basique est l'instruction *echo*. Cette instruction permet d'insérer du texte ou le résultat retourné par une fonction dans la page web.
Exemple :
< ?php echo "Ceci est du texte" ; ?>

A l'intérieur de la balise PHP on écrit l'instruction *echo* suivie du texte à afficher entre guillemets. Les guillemets permettent de délimiter le début et la fin du texte, ce qui aide l'ordinateur à se repérer. Enfin, l'instruction se termine par un point-virgule pour annoncer la fin de l'instruction. On peut aussi afficher des balises. Par exemple :

< ?php echo "Ceci est du texte" ; ?>

Le mot **texte** sera affiché en gras grâce à la présence des baliseset.

Remarque : Si votre texte comporte des guillemets, il faudra précéder le guillemet d'un antislash :

< ?php echo "Cette ligne a été écrite \" uniquement \" en PHP." ; ?>

4.3.3 Les commentaires

Il est toujours utile de commenter les scripts que vous écrivez. Les commentaires ne sont pas pris en compte par l'analyseur PHP.

PHP supporte les trois syntaxes de commentaires suivantes :

- commentaires sur une seule ligne introduits par les caractères //.
- commentaires sur plusieurs lignes introduits par les caractères /* et fermés par les caractères */.
- commentaires de type UNIX, ne comportant qu'une seule ligne introduite par le caractère #.

4.3.4 Les variables

Comme tout langage, PHP manipule des données. Pour un site dynamique, ces données sont variables. De plus, elles peuvent être de types différents, tel du texte sous forme de chaîne de caractères, sous forme de nombres entiers ou décimaux ou encore sous forme de valeurs booléennes vrai ou faux. Ces types de base sont les plus employés, mais il en existe d'autres, qui peuvent être des types composés, comme les tableaux et les objets.

En PHP, la variable existe tant que la page est en cours de génération. Dès que la page PHP est générée, toutes les variables sont supprimées de la mémoire car elles ne servent plus à rien. C'est donc une petite information temporaire présente en mémoire vive.

Chaque variable possède un identifiant particulier, **qui commence toujours par le caractère dollar ($)** suivi du nom de la variable. Vous reconnaîtrez toujours qu'il y a une variable par la présence du symbole « dollar » ($).

Les règles de création des noms de variable sont les suivantes :

- Le nom commence par un caractère alphabétique, pris dans les ensembles [a-z], [A-Z] ou par le caractère de soulignement (_).
- Les caractères suivants peuvent être les mêmes plus des chiffres.
- La longueur du nom n'est pas limitée, mais il convient d'être raisonnable sous peine de confusion dans la saisie du code. Il est conseillé de créer des noms de variable le plus « parlant » possible.
- Les noms des variables sont sensibles à la casse (majuscules et minuscules). $mavar et $MaVar ne désignent donc pas la même variable.

L'affectation de valeur aux variables se fait à l'aide de l'opérateur =, soit après la création de la variable, soit en même temps.

Exemple :
< ?php $age = 17 ; ?>

Dans cet exemple on affecte la valeur 17 à notre variable $age, et on n'oublie pas le point-virgule (;) qui permet de terminer l'instruction. D'une manière général l'affectation se fait comme ceci :

$mavar = expression ;

la variable *$mavar* prend la valeur de l'expression, qui peut être une valeur numérique, par exemple, une chaîne de caractères littérale, mais aussi une autre variable ou encore une expression PHP valide contenant des fonctions.

Exemple :

```
< ?php
$mavar =75 ;
$mavar ="Paris" ;
$mavar = 7*3+2/5-91 ; //PHP évalue l'expression puis affecte le résultat
echo $mavar ;
?>
```

La concaténation

Pour concaténer deux chaîne de caractère sous PHP, il va falloir séparer les éléments les uns des autres à l'aide d'un point (.), comme ceci :

```
< ?php

$age= 17 ;

echo 'Le visiteur a ' . $age .' ans' ;

?>
```

4.3.5 La structure de base : if... else

Comme tout langage, PHP dispose d'instructions conditionnelles qui permettent d'orienter le déroulement d'un script en fonction de la valeur de données. L'instruction *if* est la plus simple et la plus utilisée des instructions conditionnelles. Présente dans tous les langages de programmation, elle est essentielle vu qu'elle permet d'orienter l'exécution du script en fonction de la valeur booléenne d'une expression.

Sa syntaxe est la suivante :

```
< ?php
$age = 8 ;
if ($age <= 12)
{

echo "Salut gamin ! Bienvenue sur mon site !<br />" ;
$autorisation_entrer = "Oui" ;
}

else
```

```
{

echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer. Au revoir !<br />" ;
$autorisation_entrer = "Non" ;
}

?>
```

4.3.6 switch

switch...case permet de comparer la valeur d'une expression avec une liste de valeurs prédéterminées par le programmeur et d'orienter le script en fonction de la valeur de cette expression.

Voici un exemple avec switch :

```
< ?php
$note = 10 ;
switch ($note) // on indique sur quelle variable on travaille
{
case 0 : // dans le cas où $note vaut 0
echo " nul ! ! !" ;
break ;
case 5 : // dans le cas où $note vaut 5
echo "très mauvais" ;
break ;
case 7 : // dans le cas où $note vaut 7
echo " mauvais" ;
break ;
case 10 :
echo "Tu as pile poil la moyenne, c'est un peu juste" ;
break ;
}
```

4.3.7 Les boucles

Les boucles permettent de répéter des opérations élémentaires un grand nombre de fois sans avoir à réécrire le même code. Selon l'instruction de boucle utilisée, le nombre d'itérations peut être défini à l'avance ou être déterminé par une condition particulière.

La boucle for

La boucle *for* permet d'exécuter plusieurs fois la même instruction ou le même bloc sans avoir à réécrire les mêmes instructions. Sa syntaxe est la suivante :

```
for(expression1 ; expression2 ; expression3)
{
//instruction ou bloc ;
}
```

expression1 est toujours évaluée. Il s'agit généralement de l'initialisation d'une ou plusieurs variables servant de compteur pour la boucle. *expression2* est ensuite évaluée avec une valeur booléenne : si elle vaut TRUE, la boucle continue et les instructions comprises dans le bloc sont exécutées, sinon la boucle s'arrête. Si elle est toujours vraie on obtient une boucle infinie, vérifiez donc qu'elle peut être fausse. *expression3* n'est exécutée qu'à la fin de chaque itération. Il s'agit le plus souvent d'une instruction d'incrémentation de la variable compteur.

La boucle while

La boucle for oblige à préciser les valeurs limites, à connaître à l'avance pour lesquelles la boucle va s'arrêter. La boucle *while* permet d'affiner ce comportement en réalisant une action de manière répétitive tant qu'une condition est vérifiée ou qu'une expression quelconque est évaluée à *TRUE* et donc de l'arrêter quand elle n'est plus vérifiée (évaluée à *FALSE*). La syntaxe de *while* est la suivante :

```
while(expression) {
//Bloc d'instructions à répéter ;
}
```

La boucle do...while

La boucle *do...while* apporte une précision à la boucle *while*. Dans celle-ci, en effet, si l'expression booléenne est évaluée à *FALSE*, les instructions qu'elle contient ne sont jamais exécutées. Avec l'instruction *do...while*, au contraire, la condition n'est évaluée qu'après une première exécution des instructions du bloc compris entre do et while. La syntaxe de la boucle *do... while* est la suivante :

```
do {
//bloc d'instructions
}while(expression) ;
```

4.3.8 Les tableaux

Les tableaux représentent un type composé car ils permettent de stocker sous un même nom de variable plusieurs valeurs indépendantes. Les éléments de ces tableaux peuvent être de type integer, double, boolean, string ou même array, ce qui permet de créer des tableaux de tableaux, c'est-à-dire des tableaux multidimensionnels. Chaque élément du tableau, est repéré par un indice numérique (le premier ayant par défaut la valeur 0 et non 1). D'où l'expression de tableau numéroté. Chaque élément peut aussi être identifié par une étiquette, qui est une chaîne de caractères, nommée clé, associée à l'élément du tableau. Ce type de tableau est appelé tableau associatif.

On distingue donc deux types de tableaux :

1. Les tableaux numérotés ;
2. Les tableaux associatifs.

les tableaux numérotés

Pour créer un tableau numéroté en PHP, on utilise généralement la fonction **array**

```
< ?php

$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît') ;

?>
```

Il est à noter que l'ordre a beaucoup d'importance. Vous pouvez aussi créer manuellement le tableau case par case :

```
< ?php

$prenoms[0] = 'François' ;

$prenoms[1] = 'Michel' ;

$prenoms[2] = 'Nicole' ;

?>
```

Les tableaux associatifs

Pour en créer un tableau associatif, on utilisera la fonction **array**, mais on va mettre l'étiquette devant chaque information :

```
< ?php

$coordonnees = array ( 'prenom' => 'François', 'nom' => 'Dupont',
'adresse' => '3 Rue du Paradis', 'ville' => 'Marseille') ;

?>
```

Rechercher dans un tableau

PHP nous offre trois fonctions pour effectuer une recherche dans un tableau :

* **array_key_exists** : pour vérifier si une clé existe dans un tableau ;
* **in_array** : pour vérifier si une valeur existe dans un tableau ;
* **array_search** : pour récupérer la clé d'une valeur dans un tableau.

Exemple :

Vérifier si une clé existe dans un tableau.

```
< ?php
$coordonnees = array ( 'prenom' => 'François',
'nom' => 'Dupont',
'adresse' => '3 Rue du Paradis',
'ville' => 'Marseille') ;
if (array_key_exists('nom', $coordonnees))
echo 'La clé "nom" se trouve dans les coordonnées !' ;
?>
```

4.3.9 Les fonction

Une fonction est un bloc de code qui n'est pas exécuté de manière linéaire dans un script. Ce code ne le sera que lors de l'appel explicite de la fonction. Écrit une seule fois, ce code peut être exécuté aussi souvent que nécessaire. Cela allège d'autant l'ensemble du code.

La procédure de déclaration d'une fonction doit suivre les règles de syntaxe suivantes :

— Commencez par définir l'en-tête de la fonction à l'aide du mot-clé *function* suivi du nom de la fonction et de parenthèses. Les noms de fonctions suivent les règles énoncées pour les variables : caractères alphabétiques et signe « _ » pour le premier caractère puis caractères alphanumériques pour la suite.
— Les parenthèses qui suivent le nom de la fonction peuvent contenir différents noms de variables comme paramètres de la fonction.
— Ouvrez un bloc de code limité par des accolades contenant l'ensemble du code de définition de la fonction.
— Si la fonction doit retourner une valeur, ce qui n'est pas obligatoire, il faut faire précéder la variable ou l'expression qui contient cette valeur du mot-clé *return*.

Vous obtenez la structure générale suivante :

```
function mafonction($x,$y,...)
{
//code de définition de la fonction
return $var;
}
```

Pour appeler la fonction, vous écrivez : mafonction(4,5,...)

La position de la déclaration d'une fonction dans le script n'a pas d'importance. Vous pouvez ainsi appeler une fonction au début du script alors qu'elle n'est définie qu'en fin de script. Il est toutefois préférable de définir les fonctions en début de script, car cela améliore la présentation et la lisibilité du code.

4.3.10 Gestion des exceptions

Une exception est un mécanisme qui permet d'intercepter une erreur générée par un script et de déclencher une action en réponse à cette erreur. Certaines erreurs sont inévitables comme celles qui sont dues à des problèmes de connexion défaillante ou de bug sur le serveur. Nous nous intéressons ici davantage aux erreurs qui peuvent être provoquées par des actions de l'utilisateur comme des saisies erronées qui provoquent l'arrêt du script ou encore à celles qui peuvent survenir lors de la tentative d'accès à une ressource inexistante.

La gestion des erreurs a pour but de signaler « proprement » les problèmes au visiteur et d'éviter l'affichage des messages d'erreur bruts tels que PHP les envoie au navigateur. Par exemple si on exécute le code suivant :

```
< ?php
$a=10;
$b=0;
```

echo $a/$b ;
?>

Nous aurons alors l'erreur suivante qui correspond à la division par 0 :
Warning : Division by zero in *c* : *\wamp5\www\test.php*
PHP 5 introduit la classe prédéfinie *Exception* qui offre une gestion évoluée des exceptions. Gérer une exception consiste à délimiter un ou des blocs de code et à prévoir une action particulière qui doit être réalisée dans le cas où l'erreur prévue se produit. Ces blocs constituent les gestionnaires d'exception. Pour les créer, procédez comme suits :

- Créez un bloc à l'aide de l'instruction *try* qui délimite le code dans lequel peut survenir une erreur. Il peut s'agir, par exemple, du code qui gère les saisies faites par des utilisateurs dans un formulaire. Ce bloc contient une instruction *throw* pour déclencher l'exception en créant un objet de type *Exception* à l'aide du mot clé *new*.
- Créez un bloc à l'aide de l'instruction *catch* associée au bloc *try* précédent. Il comporte le code qui va gérer l'erreur si elle se produit. C'est ce bloc qui utilise l'objet *Exception* créé par l'instruction *throw*. Si aucune erreur ne se produit dans le bloc *try*, l'objet *Exception* n'est pas créé, et le code du bloc *catch* est ignoré. L'exécution se poursuit dans tous les cas après le bloc *catch*.

Un gestionnaire d'exception a donc la structure suivante :

```
try
{
  //Code à surveiller
  if(erreur prévue){throw new Exception();}
  else{// Résultat;}
}
catch(Exception $except)
{
  //Gestion de l'erreur
}
```

Le nom de l'objet utilisé dans l'instruction catch est sans importance. Un même script peut comporter autant de bloc *try... catch* que vous voulez. Il est également possible d'imbriquer des blocs *try* les uns dans les autres, à condition que chacun ait un bloc *catch* associé.

Donc pour le code précédent on introduit l'exception comme suit :

< ?php

```
$a=10;
$b=0;
try
{ if($b==0) {throw new Exception("Division par 0",7);}
else{echo "Résultat de : a / b = ",$a/$b;}
}
catch(Exception $except) {
echo $except ->getMessage();
?>
```

4.3.11 Récupération des données du formulaire

Dans un précédent chapitre (chapitre 2) nous avons vu comment créer des formulaires, dans cette section, nous allons voir comment récupérer les données entrées par l'utilisateur dans les différents champs du formulaire.

Lorsque l'utilisateur clique sur le bouton d'envoi, une requête HTTP est envoyée au serveur à destination du script désigné par l'attribut *action* de l'élément <form>. La requête contient toutes les associations entre les noms des champs et leur valeur. Ces associations se trouvent dans l'en-tête HTTP si la méthode *POST* est utilisée et dans l'URL s'il s'agit de la méthode *GET*.

Via la méthode GET

Si vous indiquez la valeur *GET* à l'attribut *method* de la balise <*form*> alors vos données serrons transmis via l'URL qui aura une forme qui ressemblera à :

http ://www.google.fr/search ?nom=OUSSID&prenom=FAFA

Le point d'interrogation sépare le nom de la page PHP des paramètres. Ensuite, ces derniers s'enchaînent selon la forme *nom=valeur* et sont séparés les uns des autres par le symbole **&**.

Donc lorsqu'on utilise la méthode *GET*, la page PHP cible (la valeur de l'attribut *action*) va automatiquement créer un tableau associatif qui porte le nom ***$_GET*** dont les clés correspondent aux noms des paramètres envoyés en URL. On peut donc récupérer ces informations pour les manipuler, par exemple pour les afficher on utilise le code suivant au sein d'une page HTML :

```
<p>Bonjour < ?php echo $_GET['prenom'] . ' ' . $_GET['nom'] ;?> !</p>
```

Il faudra cependant faire des vérifications si les paramètres ne sont pas vides. Pour cela on utilise la fonction prédéfinie *isset()*. Cette fonction teste si une variable existe.

```
< ?php if (isset($_GET['prenom']) AND isset($_GET['nom'])) // On a le nom et le prénom
{
echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . '!';
}
else // Il manque des paramètres, on avertit le visiteur
{
echo 'Il faut renseigner un nom et un prénom !';
}
?>
```

Via la méthode POST

Dans ce cas les données ne transiteront pas par l'URL, l'utilisateur ne les verra pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut. Pour récupérer les données saisies la page cible va créer un tableau associatif qui porte le nom ***$_POST*** dont les clés sont les noms associés aux champs par l'attribut *name*.

Comme pour la méthode *GET*, il faudra faire attention aux valeurs saisie par l'utilisateur dans les champs. Ils peuvent par exemple saisir une valeur non conforme à un format imposé (par exemple une adresse email ou une date), envoyer un champ vide ou injecter une faille XSS qui est une technique qui consiste à injecter du code HTML contenant du JavaScript dans vos pages pour le faire exécuter. Par exemple si l'un des visiteur saisie la ligne suivante dans l'un des champ : Coucou alors la page cible va afficher les résultat suivant : Bonjour **Coucou**. Comme vous pouvez le voir la page cible à exécuter le code html et elle a affiché le mot Coucou en gras. Le visiteur pourrait même injecter du JavaScript dans les champs avec la balise *<script>* ce qui pourrais causer d'important dégâts.

Pour éviter cela PHP offre la fonction *htmlspecialchars* qui va transformer les chevrons des balises HTML <> en **&lt ;** et **&gt ;** respectivement. Cela provoquera l'affichage de la balise plutôt que son exécution. Il faut penser à utiliser cette fonction sur tous les textes envoyés par l'utilisateur qui sont susceptibles d'être affichés sur une page web. Il existe aussi la fonction *strip_ tags* qui sert à retirer les balises HTML que le visiteur a tenté d'envoyer plutôt que de les afficher.