Developpement Web

D.Moreaux

7 octobre 2018

Table des matières

Première partie

HTML

Chapitre 1

Template du document

```
<!doctype␣html>
<html␣lang="fr">
<head>
<meta␣charset="UTF-8">
<title>Titre␣page</title>
<link␣rel="stylesheet"␣href="styles.css">
<script␣src="script.js"></script>
</head>
<body>

</body>
</html>
```

La langue précisée dans `lang="fr"` doit correspondre à la langue dans laquelle le document est écrit

La balise `<title>` contient le titre qui doit se trouver dans la barre de titre du navigateur.

On peut utiliser une ou plusieurs lignes `<link rel="stylesheet">` pour lier un ou plusieurs fichiers de styles à la page web

On peut utiliser une ou plusieurs lignes `<script>` pour attacher un ou plusieurs fichiers Javascript

Le contenu de la page se place dans la balise `<body>`

Presque toutes les balises dans le corps de la page accepteront deux attributs (optionnels) : `id` et `class`. Le premier permet de nommer de manière unique un élément, le second permet de spécifier un ensemble de classes CSS (noms séparés par des espaces) à appliquer à l'élément en question.

Ainsi, `<div id="zone1">...</div>` *nommera* la division *zone1*

et `<span class="format1 format2">...</span>` associera les classes CSS *format1* et *format2* au span.

Chapitre 2

Mise en page

2.1 Titres

Les balises `<h1>...</h1>`, `<h2>...</h2>`,..., `<h6>...</h6>` permettent de définir des titres, des sous-titres, ...

Ces balises se chargent d'isoler le titre sur une ligne et ne doivent être utilisées que pour les titres [1] et dans l'ordre.

2.2 Paragraphes

Les paragraphes doivent être entourés de balises `<p>...</p>`.

Ces balises se chargent des retraits pour les premières lignes de paragraphe, des espacements (plus grand) entre deux paragraphes, des retours à la ligne,...

2.3 Les listes

Il existe principalement deux sortes de listes : les listes ordonnées (les items sont numérotés 1), 2), 3),...ou repris avec des lettres a), b), c), ...) et les listes non ordonnées (liste à puces)

La totalité de la liste doit être placée dans une balise `<ul>...</ul>` ou `<ol>...</ol>` selon que la liste est non ordonnée ou ordonnée. Cette balise permet de signaler la liste comme unique.

En plus de cela, chaque item de la liste doit être mis dans une balise `<li>...</li>`. Cette balise permet de gérer les puces et numéros, de générer les textes adéquats, ...

1. les moteurs de recherche se basent sur ce fait lors de leur indexation des pages web

Les listes ne peuvent pas se trouver dans des paragraphes ou des titres.

```
<ul>
  <li> ... </li>
  <li> ... </li>
  ...
</ul>
```

2.4 balise *générique*

Le HTML prévoit deux balises de mise en page génériques. Ces balises n'ont pas d'apparence propre. On les utilisera soit pour associer un style CSS à une partie du document, soit conjointement à du Javascript pour marquer des zones du document qui peuvent être modifiées.

La première de ces balises est `<span>`. Cette balise sert à marquer une zone dans un texte continu. On s'en sert par exemple pour mettre en évidence quelques mots dans un texte.

La seconde de ces balises des `<div>`. Cette balise sert à marquer un bloc de texte. On peut l'utiliser pour une zone allant d'un paragraphe devant être mis en évidence à une page complète sur laquelle on veut appliquer un style.

Chapitre 3

Liens

Au niveau HTML, les liens seront fait entre les pages soit à l'aide d'une balise spécifique, soit au travers du Javascript.

Ces liens permettent d'une part de naviguer d'une page du site à une autre mais également d'envoyer le visiteur sur un autre site ou de lancer des programmes externes tels que le client mail.

3.1 URL et chemin relatif

Pour identifier une ressource, on peut utiliser un URL (Universal Ressource Locator) ou indiquer où elle se trouve par rapport à la page dans laquelle on se trouve (chemin relatif).

Dans les deux cas, cet identifiant peut pointer vers une page web, une image, un fichier à télécharger,...

3.1.1 chemin relatif

Le web provenant du monde UNIX, le séparateur entre les noms de répertoires et les noms de fichiers est le `/` et non le `\` comme c'est le cas sous MS Windows [1].

Pour désigner un fichier dans le même répertoire, on se contentera d'utiliser son nom. Attention, les noms sont *case sensitive* ce qui signifie que les majuscules et les minuscules sont des caractères différents [2]. Il est fortement conseillé de conserver les noms de fichiers en minuscules.

1. à noter que la plupart des environnements autres que MS Windows utilisent le / comme séparateur
2. quand on utilise un serveur web sous MS Windows, l'operating system est incapable de faire la différence mais dès que le site sera mis en ligne, cela ne sera plus le cas

De même, afin d'éviter les problèmes, on n'utilisera que les lettres non accentuées, les chiffres, la ligne de soulignement `_`, le point et le tiret `-` (pas d'espaces, d'apostrophes, de parenthèses, ...)

Si le fichier se trouve dans un sous-répertoire, on utilisera le nom du répertoire, un slash (`/`) et le nom du fichier. On peut utiliser la séquence `..` pour remonter d'un niveau de répertoire.

3.1.2 URL

Une URL est composée de plusieurs parties, certaines étant optionnelles :

1. Le protocole
2. L'utilisateur
3. Le mot de passe
4. Le serveur
5. Le port
6. La ressource sur le serveur

Le protocole indiquera la nature de l'URL. Il peut s'agir de `http:` ou `https:` pour un site web (non sécurisé ou sécurisé) mais aussi de `ftp:` pour un serveur de fichiers FTP, `mailto:` pour une adresse E-Mail, `telnet:` pour une connexion à une machine,...

Dans le cas où le protocole n'est pas géré par le navigateur, ce dernier peut transmettre le traitement à une application externe telle que le client mail.

Pour un site web, la forme générique d'une URL est une des suivantes :

```
http://adresse
http://adresse/page
http://adresse/page#ancre
http://adresse:port
http://adresse:port/page
http://adresse:port/page#ancre
```

Pour un serveur FTP on peut trouver une URL du genre
`ftp://user:pass@serveur:port/fichier`

3.2 Placer un lien dans une page

Pour placer un lien, on utilise la balise `<a>`. Le texte qui permettra d'accéder à la destination sera entre la balise et sa fermeture et la destination sera fournie en attribut dans la balise.

```
<a␣href="contact.html">Contactez-moi</a>
<a␣href="http://www.perdu.com">Perdu␣?</a>
```

3.3 Les ancres

Il est possible de placer des ancres dans un texte. Ces ancres permettent d'accéder à un endroit précis d'une (longue) page lorsque l'on suit un lien.

On placera l'ancre sur un mot en utilisant la balise `<a␣name="nomancre">...</a>`. Lorsque quelqu'un suivra un lien vers la page dans lequel l'ancre sera précisée, il se retrouvera directement à cet endroit dans la page.

A noter qu'il est également possible d'utiliser un lien qui ne fourni que l'ancre, pour permettre de sauter à cet endroit dans la page.

```
<a␣href="#nomancre">sauter␣plus␣loin</a>
<a␣href="page2.html#nomancre2">Plus␣d'info</a>
```

Chapitre 4

Images

Lorsque l'on place des images dans une page web, il faut être prudent face à la taille de ces dernières. La taille totale de la page et des ressources qu'elle charge (CSS, Images, Javascript,...) doit en effet rester limitée afin de permettre un chargement rapide et d'éviter la charge sur le serveur.

Pour insérer une image, on utilisera la balise

```
<img␣src="fichier.jpg"␣alt="descriptif"␣width="300"␣height="200">
```

Le descriptif sera affiché dans le cas d'un navigateur texte par exemple (mais il sera aussi utilisé pour l'indexage par les moteurs de recherche ou par les systèmes pour malvoyants).

Il est nécessaire de fournir les dimensions de l'image afin que la place soit réservée au niveau du rendu de la page avant que l'image n'ait fini d'être chargée. Si la taille fournie est différente de celle de l'image, l'image sera redimensionnée (mise à l'échelle) à la dimension indiquée.

Les formats d'image supportés sont les suivants :

Extension	Couleurs	Transparent	Animé
.jpg	16M		
.png	16M	X	
.png (APNG)	16M	X	X
.gif	256	X	X

On peut aussi utiliser des fichiers SVG qui sont des fichiers vectoriels

Chapitre 5

tables

Lorsque l'on désire insérer une table en HTML, on utilise la balise `<table>`. Pendant des années, les tables ont été utilisées pour la mise en page graphique, entre autres suite à l'impulsion donnée par des logiciels tels que Dreamweaver.

Cette façon de faire est désormais déconseillée et les tables ne doivent plus être utilisées que pour présenter des informations sous forme de tableau [1]

De la même manière, la mise en page de la table est sensée se trouver complètement au niveau des CSS et la plupart des attributs de mise en page ne sont tout simplement plus valides en HTML5.

5.1 table simple

Une table doit être perçue comme un ensemble de ligne, chaque ligne contenant le même nombre de colonnes. Il est important que le nombre de colonnes soit constant d'une ligne de la table à l'autre.

Les lignes seront marquées par une balise `<tr>` et la fin de la ligne par la fermeture de la balise `</tr>`.

Les cases contenues dans une ligne peuvent contenir soit des données, soit des en-têtes. On placera les données dans des balises `<td>` et les en-têtes dans des balises `<th>`.

Une table simple ressemblera donc à

```
<table>
  <tr>
    <th>Nom</th>
    <th>Prenom</th>
```

1. une exception peut parfois être faite pour les formulaires

```
  </tr>
  <tr>
    <td>Beeblebrox</td>
    <td>Zaphod</td>
  </tr>
  <tr>
    <td>Dent</td>
    <td>Arthur</td>
  </tr>
  <tr>
    <td>Perfect</td>
    <td>Ford</td>
  </tr>
</table>
```

5.2 Table complexe

Dans certains cas, on désirera qu'une case s'étende sur plusieurs colonnes de la table ou sur plusieurs lignes (ou les deux).

Pour faire cela, on utilisera l'attribut `colspan="n"` ou `rowspan="n"` où n est le nombre de colonnes/lignes sur lesquelles s'étaler.

Quand on utilisera `colspan`, la cellule (`<th>` ou `<td>`) couvrira le nombre de colonnes indiqué et on ne devra pas préciser les colonnes en question.

Ainsi, dans une table de 6 colonnes, si on place un `colspan="2"` sur une colonne, on aura un total de 5 `<td>` (le sixième étant compris dans la colonne étendue par `colspan`).

De même, lorsque l'on utilise un `rowspan`, il faudra omettre la colonne correspondant sur les lignes qui suivent.

Chapitre 6

Les formulaires

Lorsque l'on désire transmettre des informations au serveur, cela peut se faire au travers de requêtes GET et de requêtes POST. Ces requêtes peuvent être faites au travers de formulaire (GET et POST) ou au travers d'URL [1] (seulement GET).

6.1 Requetes GET et POST

6.1.1 fonctionnement

La requête GET envoie les paramètres au niveau de l'URL. Ces paramètres sont encodés sous la forme `nom=valeur` où le nom ne peut contenir que des lettres et des chiffres et où la valeur doit être encodée [2].

Les lettres non accentuées et les chiffres sont conservés dans l'encodage URL et on peut donc les utiliser sans se soucier de l'encodage. Par contre, les espaces sont encodés.

Comme les paramètres sont passés au niveau de l'URL, la taille totale envoyée en paramètres dans une requête GET est limitée.

La requête POST envoie les données dans la requête au serveur. Plusieurs encodages sont possibles (par défaut, l'encodage est semblable aux requêtes GET mais il peut être changé, par exemple lorsque l'on désire envoyer un fichier). Cela retire la limitation de taille au niveau du protocole. Par contre, le serveur et le moteur utilisé derrière (ici le PHP) peuvent imposer des limites.

1. Cette méthode est généralement utilisée au niveau de liens ou d'images
2. On parle d'URL-Encoding, les caractères spéciaux sont remplacés par des %xx où xx est le code ASCII en hexadécimal

	GET	POST
Avantages	Simple utilisable sans formulaire cache-friendly	pas de limite de taille permet l'envoi de fichiers masque les paramètres des logs (cache et serveur)
Inconvénients	taille limitée valeurs passées visible dans les logs	plus lourd au niveau communication

6.1.2 Choix entre GET et POST

En général, on utilisera des requêtes GET sauf dans les cas suivants :
- L'envoi d'informations confidentielles (carte de crédit, mot de passe,...)
- L'envoi de données potentiellement volumineuses (long texte par exemple)
- Pour forcer l'utilisateur à confirmer le réenvoi des données lorsqu'il rafraichi la page
- L'envoi de fichiers
- si on désire éviter de faire apparaître les paramètres dans l'historique (pour éviter d'accumuler un grand nombre de données différentes dans ce dernier)

6.1.3 Requête GET dans une URL

Il est possible de générer une requête GET au niveau d'une URL. Cette URL pourra soit être présente dans un lien, auquel cas, cliquer sur le lien appellera la page avec les paramètres passés, ou dans une image (ou un fichier CSS, une vidéo, un son, ...) quand ces derniers sont générés par un programme (script PHP ou autre) et que l'on désire fournir des options [3]

Pour former une telle URL, on fera suive le nom du fichier par un point d'interrogation puis les paires `nom=valeur` séparées par des caractères & .

L'URL faisant une requête GET aura donc la forme suivante :

```
http://www.example.com/page.php?nom1=val1&nom2=val2&nom3=val3
```

3. Par exemple, Mimetex permet de générer une image contenant une formule mathématique passée en paramètre. On peut aussi trouver des images *compteur de visite* où le paramètre peut donner un identifiant (pour que le programme puisse gérer plusieurs compteurs) ou dans des *webbugs*, images invisibles (1x1 transparente) utilisée pour tracer le navigateur

6.2 Formulaire

Un formulaire est composé d'une balise `<form>` dans laquelle se trouveront des zones d'entrée. Chaque zone d'entrée aura un attribut `name` qui précisera le nom sous lequel la valeur entrée sera transmise. A noter que l'attribut `id` peut également être présent lorsque l'on désire pointer sur l'élément au niveau CSS ou Javascript mais cet attribut ne sera pas utilisé lors de l'envoi des données.

6.2.1 La balise FORM

La balise FORM contiendra au minimum les deux attributs suivants :

action contient l'URL de la page qui devra traiter les données envoyées

method contiendra soit GET soit POST pour indiquer comment les données seront transmises.

```
<form␣action="process.php"␣method="get">
Nom␣:␣<input␣type="text"␣name="nom">
<input␣type="submit"␣value="Envoyer">
</form>
```

6.2.2 La balise INPUT

La balise INPUT permet de créer différents types d'entrées. L'attribut `type` permettra de préciser le type d'entrée concerné.

A noter que dans le cas des radio et des checkbox, seule la case est fournie par l'élément input, le texte doit être ajouté à côté.

6.2.3 SELECT et TEXTAREA

Deux autres types de zones d'entrée sont disponibles :

La `<textarea>` permet de définir une grande zone d'entrée texte (par exemple pour entrer un message). La valeur par défaut sera placée entre les balises ouvrante et fermante. Le texte sera utilisé tel quel (s'il contient de l'HTML, il ne sera pas pris en compte et sera affiché comme du texte). De nouveau, l'attribut `name` permettra de fournir le nom sous lequel le texte sera envoyé.

Avant, la taille de la zone était définie au travers d'attributs. De nos jours, on préfère la contrôler à l'aide des CSS.

type	utilité
text	zone d'entrée de texte. L'attribut `value` fourni le texte par défaut.
password	zone d'entrée de texte caché (le texte apparaît sous forme d'étoiles ou autres semblables)
submit	bouton déclenchant l'envoi des données. l'attribut `value` fourni le texte du bouton. L'attribut `name` peut être omis mais s'il est présent, le paramètre contiendra le texte de `value`
checkbox	case à cocher. Si la case est cochée, le paramètre sera présent. Si l'attribut `value` est présent, son contenu sera envoyé quand la case est cochée. Pour que la case soit cochée par défaut, il est nécessaire de définir l'attribut `checked`
radio	Radiobox (case pour sélectionner un élément parmi plusieurs). Les différentes radioboxes d'un même groupe doivent avoir le même `name` et chacun avoir l'attribut `value`. Lorsque le formulaire est envoyé, le paramètre contiendra la valeur de l'attribut `value` de la radiobox sélectionnée. L'attribut `checked` permet d'indiquer la radiobox sélectionnée par défaut.
color	si le navigateur le supporte, fourni un *color picker* sinon se comporte comme un champ texte. La couleur sera envoyée sous la forme #RRGGBB
date	si le navigateur le supporte, fourni une zone d'entrée munie d'un calendrier pour sélectionner une date. La date sera au format année-mois-jour
email	champ texte spécialisé dans l'entrée d'E-Mail
url	champ texte spécialisé dans l'entrée d'une URL
tel	champ texte spécialisé dans l'entrée d'un numéro de téléphone
number	Permet l'encodage d'un nombre. Les attributs `min` et `max` permettent de définir l'intervalle
range	fourni un *slider* permettant de sélectionner un nombre. les attributs `min` et `max` permettent de spécifier l'intervalle (par défaut de 0 à 100)

L'élément `<select>` permet de créer une liste déroulante ou une liste d'items parmi lesquels choisir. L'attribut `size` permettra de choisir le nombre de lignes affichées (une valeur de 1 sélectionnant la liste déroulante).

Il utilise une série d'éléments `option` qui définissent les choix possibles

```
<select␣name="couleur"␣size="1">
␣␣<option␣value="red">Rouge</option>
␣␣<option␣value="green">Vert</option>
␣␣<option␣value="blue">Bleu</option>
␣␣<option␣value="black"␣selected>Noir</option>
</select>
```

Lorsque le formulaire est envoyé, la valeur de l'option choisie sera utilisée. L'attribut `selected` permet d'indiquer la valeur par défaut.

Deuxième partie

CSS

Introduction

Les CSS ou feuilles de style en cascade [4] sont des fichiers contenant des règles sur l'apparence de la page web.

Chaque navigateur dispose en interne d'un ensemble de feuilles de style qui décrivent les apparences de base des différentes balises reconnues. Les balises `<div>` et `<span>` sont spéciales en ce sens que seul le style `display:` indiquant comment la balise s'intègre dans le flux de texte est présent, en faisant des balises neutres.

On peut fournir des feuilles de style de plusieurs manières : directement dans la balise, dans une balise spéciale au niveau du `<head>` ou au travers d'un fichier extérieur [5].

On peut attacher plusieurs fichiers de styles à une page Web. Sur un même site, on essaiera généralement d'utiliser les mêmes feuilles de styles sur toutes les pages afin de garder de l'uniformité.

4. Cascading Style Sheet
5. On peut également modifier les styles au travers du javascript

Chapitre 7

Sélecteurs

Pour décider quelles règles doivent être appliquées sur une balise, on utilise des sélecteurs.

7.1 Sélecteurs simples

Les sélecteurs simples sont de trois sortes :

balise Le sélecteur s'applique aux textes contenus dans la balise indiquée `h1 { ... }` s'appliquera aux textes contenus dans les balises `<h1>`.

classe Le sélecteur s'applique aux textes contenus dans une balise référençant la classe correspondante `.listing { ...}` sera référencé par les balises contenant l'attribut `class="listing"`.

On peut utiliser les classes au niveau de toutes les balises même si elles sont principalement utilisées avec les balises `<div>` et `<span>`.

ID Le sélecteur s'applique à la balise **unique** qui possède l'ID indiqué `#idelem { ... }`.

Dans un document HTML, on peut fournir un ID à une balise tout en respectant les deux points suivants : une balise ne peut posséder qu'un seul ID et un ID ne peut être associé qu'à une seule balise. `<div id="menu"`

7.2 Selecteurs composites

Il est possible de combiner plusieurs sélecteurs afin de pointer des éléments plus spécifiques.

Ainsi, il est possible de combiner une balise avec une classe en collant les deux parties. `div.titre` désigne une `<div>` qui possède la classe `.titre`

Il n'est généralement pas utile de combiner une balise avec un ID dans la mesure où un ID est sensé être unique dans le fichier HTML.

Il est également possible de définir une balise par rapport aux balises qui l'entourent. L'élément référencé sera le dernier de la liste d'éléments décrits.

A l'intérieur de Pour parler d'un sélecteur validé par un élément se trouvant à l'intérieur d'un autre élément décrit par un second sélecteur (par exemple, un `span` situé dans une `div`), il suffit de préciser les deux sélecteurs séparés par un espace `div␣span`. Ce selecteur pointera sur le `span`.

Fils de Pour parler d'un sélecteur validé par un élément se trouvant directement à l'intérieur d'un autre élément (et pas dans un élément se trouvant dans le second), on séparera les deux éléments par un `>`. Ainsi, pour désigner un `span` situé directement dans un `div`, on utilisera la notation `div>span`.

Suivant Pour parler d'un sélecteur validé par un élément qui suit directement un autre élément, on utilisera un `+` pour les séparer. Ainsi, pour indiquer un `p` qui en suit directement un autre, on utilisera `p+p`.

Il est également possible de fournir plusieurs sélecteurs pour un ensemble de propriétés en les séparant par des virgules. Ainsi, `ul,ol` indiquera des propriétés appliquées aux deux types de listes.

7.3 Ordre de priorité

Lorsque plusieurs règles CSS définissent la même propriété sur un élément, une de ces règles devra être choisie. Pour ce faire, on calculera la priorité sous la forme de 4 nombres :

1. Le nombre de `style="..."` définissant la propritété (0 ou 1)
2. Le nombre d'ID impliqués dans le sélecteur
3. Le nombre de classes (et de pseudo-classes) impliqués dans le sélecteur
4. Le nombre d'éléments cités dans le sélecteur

Une règle telle que `#title div div:hover .left span` aura une priorité de 0 (déclaré au niveau CSS), 1 (un ID, `#title`), 2 classes CSS (`:hover` est une peudo classe) et 3 éléments (deux `div` et un `span`). 0 1 2 3

Les priorités des deux règles en conflit seront comparées de gauche à droite et celle avec la plus haute valeur rencontrée sera prise en compte. En cas de priorités égales, la seconde règle prendra le dessus.

7.4 Pseudo-classes

Les pseudo-classes CSS sont des classes gérées par le navigateur. Elles permettent de localiser de façon plus précise un élément ou de réagir à certains événements gérés par le navigateur.

Ces pseudo-classes sont désignées en utilisant un `:` au lieu d'un point.

:hover Cette pseudo-classe est active lorsque le pointeur de la souris se trouve sur l'élément et inactive sinon. `span:hover` indiquera un span au dessus duquel se trouve le pointeur de souris par exemple.

:visited Utilisé sur un lien (élément `a`), il sera actif si le lien pointe vers une page déjà visitée (présente dans l'historique du navigateur).

:first-child L'élément est le premier fils de l'élément parent (attention, du texte compte comme un *noeud texte*)

:nth-child(n) L'élément est le *nième* fils de l'élément parent. n peut être une valeur numérique, une formule du genre $an + b$ où a et b sont des entiers ou encore un des mots clés `odd` ou `even` (impair ou pair).

Chapitre 8

Les propriétés

Les propriétés CSS sont de la forme `propriete: valeur;` où le `;` final est nécessaire.

8.1 Police de caractères

Plusieurs propriétés contrôlent l'aspect des caractères.

font-size	taille des caractères ex : 12pt
font-family	type de caractères : sans sérif, avec sérif, ...
font-style	normal, italic ou oblique
font-weight	normal, bold (gras), bolder (plus gras) ou lighter (plus léger) ou une valeur de 100 à 900
font-variant	normal ou small-caps (petites majuscules au lieu de minuscules)

Pour les familles de fonte, on utilisera généralement une des lignes suivantes :

— Arial,Helvetica,sans-serif
— Verdana,Geneva,sans-serif
— "Times New Roman",Times,serif
— "Courier New",Courier,monospace

8.2 Couleurs

8.2.1 Codes de couleurs

Dans les CSS, on travaille principalement sur le système de couleurs RCG [1]. Chaque composante sera représentée par une valeur de 0 à 255 (en décimal) ou de 00 à FF (en hexadécimal) correspondant à de 0 à 100% de la composante de couleur en question.

Les mélanges de base de ces 3 couleurs donnent les couleurs qui suivent

Couleur	R	G	B	hex
Noir	0	0	0	`#000000`
Rouge	255	0	0	`#FF0000`
Vert	0	255	0	`#00FF00`
Bleu	0	0	255	`#0000FF`
Jaune	255	255	0	`#FFFF00`
Magenta	255	0	255	`#FF00FF`
Cyan	0	255	255	`#00FFFF`
Blanc	255	255	255	`#FFFFFF`

On évitera d'utiliser les noms de couleurs tant que possible pour privilégier les code hexadécimaux.

A partir des couleurs de base, on peut tendre vers le blanc en augmentant les composantes qui sont à 0 de façon uniforme ou tendre vers le noir en diminuant les composantes à 255 de façon uniforme.

Plus particulièrement, les valeurs `#101010`, `#202020`,. . .,`#909090`, `#A0A0A0`, `#B0B0B0`, . . .donnent des tons de gris allant du gris très foncé au gris très clair.

8.2.2 Propriétés liées aux couleurs

Pour changer la couleur du texte, on utilisera la propriété `color:`.

La règle suivante permettra de mettre le texte en rouge

```
color:␣#FF0000;
```

Pour changer la couleur de l'arrière plan, on utilisera la propriété `background-color:` ou la propriété raccourci `background:` [2]

Pour utiliser un arrière plan gris, on utilisera

1. Red Green Blue, les trois couleurs de base en synthèse additive (quand on mélange des lumières)
2. Les propriétés raccourci permettent de fixer plusieurs propriétés en une seule ligne

```
background-color: #C0C0C0;
```

On peut également utiliser des couleurs sur les bordures, les décoration de texte, le curseur dans les zones d'entrée, ...

On utilise souvent les codes couleurs en hexadécimal afin qu'ils soient plus repérables dans un fichier CSS. La plupart des logiciels graphiques fournissent le code dans ce format.

8.3 Le modèle de boîte

Les éléments peuvent être de deux types principaux : *block* ou *inline*.

Il est possible de définir le type d'élément à l'aide de la propriété `display` qui peut prendre comme valeurs block, inline ou none (pour ne pas afficher l'élément).

Les éléments *inline* correspondent à un morceau de texte et peuvent commencer et terminer n'importe où au niveau d'une ligne. On les utilisera pour changer la fonte, la couleur ou autres d'un texte.

Les éléments de type *block* sont des zones rectangulaires qui s'empilent normalement en dessous l'un de l'autre.

Les éléments de type *block* sont constitués de plusieurs parties.

Au centre, la zone de contenu dont la taille peut être précisée à l'aide des propriétés `width` et `height`.

Autour de la zone contenu se trouve un espace vide appelé *padding*. La taille de cet espace est définie par les propriétés `padding`, `padding-left`, `padding-right`, `padding-top` et `padding-bottom`. La directive `padding` permet de fixer les 4 valeurs en une seule propriété.

Autour du padding se trouve la bordure. La bordure sera définie par la propriété `border` ainsi que les propriétés liées (un ensemble par côté de la boîte). Les propriétés synthétiques telles que `border` prendront 3 valeurs dans l'ordre : la largeur du trait, le type de trait et la couleur du trait.

```
border-left: 2px solid #000000;
```

Pour terminer, autour de la bordure se trouve une marge. Cette marge se défini par la propriété `margin` et les propriétés propres à chaque côté. Quand deux blocks disposant de marges se trouvent côte à côte, seule la marge la plus grande est prise en compte (les marges se superposent).

8.4 Le Positionnement des blocs

Troisième partie

Javascript

Introduction

Le HTML permet de présenter des informations à l'écran mais de manière statique. Si on désire que la page puisse être plus dynamique, il est nécessaire d'utiliser un langage de programmation au niveau du navigateur.

Ce rôle peut être rempli par le Javascript ou ECMAScript. Malgré le nom de ce langage, il n'a rien à voir avec le langage JAVA.

Au début limité à manipuler la page HTML et intercepter les événements (clic de souris, déplacement de souris,...), le Javascript a peu à peu vu de nouvelles fonctionnalités apparaître : dessin 2D et 3D dans un élement de type canvas, contrôle d'un élément audio pour jouer des sons à la demande, requêtes faites au serveur associé à la page en arrière plan, connexion en continu à un serveur [3], connexion à des serveurs différents grâce à un mécanisme ajouté au niveau du HTTP pour autoriser de telles connexions,....

Il devient maintenant possible de créer une application s'exécutant principalement côté navigateur à l'aide du Javascript. C'est ainsi que l'on peut trouver des traitement de texte et tableurs, des logiciels de dessin 3D ou même des jeux programmés en Javascript où le serveur ne sert souvent plus qu'à sauver les données.

3. les HTML Sockets

Chapitre 9

utiliser un script en Javascript

9.1 Insérer du Javascript dans une page HTML

Le Javascript peut être inséré dans la page HTML de plusieurs façons.

— Directement dans une balise : on peut insérer du javascript dans des attributs d'une balise. Cette façon de faire ne permet que des codes très courts et est très limitatif quand aux caractères qui peuvent être employés. Elle rend également le code Javascript difficile à repérer vu qu'il est éparpillé dans la page.

— Dans une balise `<script>` au niveau du `<body>` : cette solution est souvent utilisée par facilité (on place souvent le script en fin de page pour que la page soit chargée quand il s'exécute), a cependant les défauts de rendre le Javascript plus difficile à repérer (il peut se trouver n'importe où dans la page) et de ne pas séparer présentation (HTML/CSS) et programme (le Javascript).

— Dans une balise `<script>` au niveau du `<head>` : Si le script est plus repérable que dans le cas de la balise body, le problème de manque de séparation reste présent.

— Dans un fichier extérieur à la page web déclaré au niveau du `<head>`. Cette méthode est la méthode à privilégier. D'une part, cela permet de séparer HTML et Javascript [1], d'autre part, cela permettra au script d'être mis en cache séparément de la page HTML (utile par exemple si la page est générée en PHP) et d'être chargé en même temps que la page HTML (le navigateur faisant plusieurs connexions pour accélérer le chargement).

1. cela évite les problèmes avec certains validateurs HTML et évite de devoir utiliser des mises en commentaire cryptiques dans le cas de XHTML

On utilisera donc quasi exclusivement [2] des Javascript dans des fichiers séparés.

Pour accéder à ces fichiers, on utilisera la balise :

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

Il faut faire particulièrement attention à fournir la balise de fin faute de quoi la totalité du texte suivant la balise script sera considérée comme faisant partie du script et ne sera pas prise en compte par le moteur HTML [3]

Le script s'exécutera dès qu'il est chargé, ce qui peut demander quelques précautions (par exemple, pour s'assurer que les ressources que l'on désire manipuler sont déjà présentes).

2. certains codes comme le code de Google Analytics sont fait pour être copiés dans la page HTML

3. A noter que cela est également vrai pour la balise <style>

Chapitre 10

Le langage Javascript

La structure du langage Javascript ressemble fort aux langages C et PHP.

- Les boucles `for` utilisent la même syntaxe (`for(a=0;a<5;a++) { ... }`)
- Les tests conditionnels `if (...) { ... } else { ... }` utilisent la même syntaxe
- Comme en PHP et contrairement au C, la commande `switch` accepte de travailler avec des chaînes de caractères.
- L'assignation d'une valeur à une variable se fait de la même manière : `a=5;`
- Les lignes demandent également à être terminées par un `;`

On pourra donc écrire

```
a=4;
b=0;
for(i=0;i<10;i++) {
  if (i>a) {
    b=b+i;
  } else {
    b=b+2*i;
  }
}
```

A noter que comme en PHP, il n'est pas nécessaire de déclarer les variables avant de les utiliser.

10.1 Les variables

En Javascript, les variables sont stockées dans des tables de symboles. Ces tables reprennent les paires nom de variable et référence vers le contenu.

Les différentes données qui peuvent être sauvées dans les variables sont réservées en mémoire et les variables contiendront des pointeurs ou références vers les données en question.

On trouvera deux types de tables de symboles : celles associées à un objet et celles associées à l'exécution d'une fonction.

La table de symboles associée à un objet permettra de stocker les propriétés de ce dernier. On accédera aux dites propriétés par la notation `obj.prop` où obj est l'objet dont il est question et prop le nom de la propriété.

La table d'objet d'une exécution de fonction est créée lorsque l'on exécute la fonction et sert à stocker les variables locales. Les paramètres de la fonction sont des variables locales.

Comme tous les autres éléments (chaîne, objet, ...) en Javascript, les tables de symboles seront effacées lorsqu'un processus appelé *garbage collect* sera déclenché, à condition qu'il n'y ai plus aucune référence qui pointe dessus.

Un cas particulier est l'objet `window` qui désigne la fenêtre de navigateur. Cet objet est utilisé quand aucun autre objet ne l'est. Ainsi, lorsque l'on parle d'une variable globale (non locale à une fonction) `a`, cette variable est sauvée dans l'objet `window` et on peut indifféremment l'appeler par `a` ou `window.a`.

Il est toujours possible d'accéder aux variables globales en passant par l'objet `window`, même si une variable du même nom a été créée dans le contexte de la fonction en cours.

Les variables peuvent contenir n'importe quel type de données : nombres, chaînes de caractères, objets, fonction,...

Une valeur spéciale appelée `null` peut être utilisée pour vider une variable [1]

10.2 Fonctions

Pour déclarer une fonction en Javascript, on utilisera une des formes suivantes :

```
function test(param1,param2) {
  ...
  return res;
}

test2 = function (param1, param2) {
```

1. Afin de permettre au garbage collect d'effacer une objet, il est parfois nécessaire de sauver null dans la variable qui pointe sur cet objet

```
  ...
  return res;
};
```

La première forme va créer une fonction et la sauver directement dans la variable test.

La seconde forme va créer une fonction sans nom et la sauver dans une variable test2 (mais elle pourrait être également passée en argument d'une autre fonction, être utilisée dans un return, ...). On parle de fonction anonyme.

Entre parenthèses, on précise la liste des paramètres de la fonction. Les paramètres transmis à la fonction lorsqu'elle sera appelée seront sauvés dans des variables locales à la fonction.

Pour exécuter une fonction, il suffit de la faire suivre de parenthèses contenant les paramètres.

```
x=test(2,4);
y=function (i) { return i*i; } (5);
```

Dans le second cas, on exécute directement la fonction anonyme que l'on a créée et la variable y contiendra 25 ($5 * 5$).

Si une fonction n'a pas de paramètres, il suffit de mettre les parenthèses sans rien à l'intérieur. Si on appelle une fonction avec moins de paramètres que ce qu'elle demande, les derniers paramètres seront *undefined*[2]

Il est possible de créer une fonction acceptant un nombre quelconque de paramètres. Pour cela, on utilisera l'objet `arguments` qui peut être accédé comme un tableau et possède la propriété `length` pour déterminer le nombre de paramètres.

A l'intérieur d'une fonction, on peut déclarer une variable comme locale à l'aide du mot clé `var`. Lorsqu'une variable est déclarée comme locale, elle est créée dans la table de symboles de l'exécution de fonction en cours et elle masquera une éventuelle variable globale du même nom jusqu'à la sortie de la fonction.

Si la fonction doit retourner une valeur, elle le fait à l'aide de l'instruction `return` suivie de la valeur à retourner. La valeur retournée par une fonction peut être n'importe quoi : null, nombre, chaîne de caractères, tableau, objet, fonction,...

2. on peut les tester avec if (param === undefined)

10.3 Tableaux

En javascript, une variable peut contenir un tableau. Un tableau est en fait un objet disposant de certaines propriétés [3], ce qui fait que pas mal d'objets peuvent être accédés comme s'ils étaient des tableaux [4].

Pour accéder à un élément d'un tableau, il suffit de mettre son indice entre crochets rectangulaires : `a[5]`. Comme en C et en PHP, les tableaux commencent à l'indice 0.

Pour déterminer le nombre d'éléments dans un tableau, on utilisera la propriété `length` de la manière suivante : `a.length`.

Si on demande à écrire un élément qui n'existe pas, le tableau sera étendu à cet élément, comme en PHP.

On peut assigner directement un tableau à une variable à l'aide de la syntaxe :

```
var a= [1, 4, 2, 8, 5, 7];
var b= [ ];
```

La deuxième ligne permet de créer un tableau vide. A noter que la notation `new Array()` n'est plus conseillée.

Les éléments présents dans un tableau peuvent être quelconques et on peut mélanger des éléments de type différent. Il est donc possible de créer des tableaux de tableaux, des tableaux mélangeant des chaînes et des entiers, ...

Les indices d'un tableau sont des nombres comme en C. Si on essaye d'utiliser un indice chaîne (tableaux associatifs comme en PHP), le tableau sera automatiquement transformé en un objet et les fonctions standard du tableau (dont `length`) ne seront plus disponibles.

Les tableaux disposent de nombreuses instructions permettant de les découper, d'ajouter ou d'enlever des éléments, de les trier,....

10.4 Les chaînes de caractères

En Javascript, les chaînes de caractères sont des objets. Ils disposent dès lors d'une série de propriétés qui permettent d'agir dessus.

---

3. Les propriétés en question permettent d'énumérer les éléments, de retourner le nombre d'élément, de lire ou modifier les éléments, ...

4. c'est par exemple le cas de la variable arguments qui contient les paramètres d'une fonction

Tout comme les tableaux, la propriété `length` permettra de connaître la taille de la chaîne. A noter que l'on peut l'appeler à partir d'une variable ou à partir d'une chaîne littérale :

```
str.length
"test".length
```

On trouve également les fonctions suivantes (qui peuvent aussi être appelées à partir d'une chaîne littérale) :

s.charAt(n)	retourne le nième caractère de la chaîne s (le premier est le caractère 0)
s.substr(start,len)	retourne une sous-chaîne de s commençant au caractère start et de taille len. si start est négatif, compte à partir de la fin de la chaîne
s.substring(start,end)	retourne une sous-chaîne commençant au caractère start et se terminant au caractère end.
s1.indexOf(s2)	retourne -1 si la chaîne s2 n'est pas trouvée dans s1 sinon retourne la position de la première occurence
s1.replace(srch,rep)	cherche la chaîne srch dans s1 et la remplace par la chaîne rep
s1.split(s2)	trouve le caractère s2 dans la chaîne et découpe la chaîne en morceaux séparés par s2 (qui est retiré). Retourne un tableau avec les différents morceaux. Si s2 est une chaîne vide, retourne un tableau où chaque élément est une chaîne de 1 caractère
s.trim()	retire les caractères blancs en début et en fin de chaîne
s1.concat(s2,s3,...)	retourne une chaîne formée des chaîne s1, s2, s3, ...bout à bout

La fonction `parseInt(s)` va transformer la chaîne en nombre entier. Si la chaîne commence par 0x, la valeur est supposée en hexadécimal. Si la valeur commence par 0, certains navigateurs considéreront qu'elle est en octal (base 8, chiffres de 0 à 7, fonction dépréciée).

De la même façon, `parseFloat(s)` permettra de convertir une chaîne en nombre à virgule flottante.

La fonction `String(x)` par contre convertira la valeur x en chaîne de caractères.

10.5 Les Regexp

Les expressions régulières ou regexp permettent de décrire un *pattern* pour ensuite le rechercher dans une (ou des) chaîne de caractères.

Ce pattern est composé de caractères (ou ensembles de caractères) et de modificateurs. Les modificateurs permettent de préciser le nombre de fois que le terme qu'ils modifient doit apparaître.

10.5.1 Les modificateurs

Les modificateurs sont les suivants :

?	0 ou 1 fois
$*$	0 fois ou plus
$+$	1 fois ou plus
{4,}	4 fois ou plus
{4,9}	Entre 4 et 9 fois

Le modificateur se place après le caractère qu'il modifie. Ainsi, `A*` indique la lettre A répétée 0, 1, 2 fois ou plus (la chaîne vide, A, AA, AAA, AAAA, ...)

Pour répéter un groupe de caractères, on utilisera les parenthèses. Ainsi, `(AB)+` correspondra aux possibilités (AB, ABAB, ABABAB, ABABABAB, ...)

10.6 Les caractères

La plupart des caractères ont leur propre signification. Pour les caractères spéciaux, il faudra les échapper avec un \ pour les insérer.

Dans les ensembles, un ^ placé en début d'ensemble permet de préciser l'ensemble des caractères différents de et un - placé en fin de mettre un caractère -. A noter que les autres caractères spéciaux ($+$, $*$, ., ...) perdent leur signification spéciale dans un ensemble. `[+*. ]` indiquera un des quatre caractères $+$, $*$, . et espace.

Lorsque l'on utilise un modificateur après un ensemble de caractères, il n'est pas nécessaire que ce soit à chaque fois le même caractère de l'ensemble

.	un caractère quelconque
^	le début de la chaîne
$	la fin de la chaîne
[abc]	un des caractères a, b ou c
[a-e]	un des caractères entre a et e (a,b,c,d,e)
[a-zA-Z]	un des caractères entre a et z ou entre A et Z
[a-z-]	un des caractères entre a et z ou le caractère -
[^a-z]	un caractère qui n'est pas entre a et z.

qui soit repris. Par exemple, `[abc]*` acceptera les chaînes aaa, bbb, ccc mais aussi aab, bac, cca, ...

On peut regrouper plusieurs caractères à l'aide de parenthèses. Ces parenthèses feront que l'ensemble des caractères situés à l'intérieur seront traités comme un caractère unique [5].

La barre verticale permet de séparer différents mots possibles. Il n'est pas nécessaire de placer des parenthèses entre les mots en questions ce qui fait qu'on utilise souvent la forme `(aaa|bbb|ccc)` pour préciser que l'on a une des trois chaînes aaa, bbb ou ccc.

`^\+?[0-9/- ]+$` permettra de reconnaître un numéro de téléphone, de la forme +32 12 42 54 23, +32-12-425423, 0032-12-425423, 012/425423,...(le + ne peut se trouver qu'en début de chaîne, pour le reste, on accepte uniquement des chiffres, des /, des - et des espaces). A noter que cela accepterait également des numéros incorrects tels que +- ou ///

`^(http|https|ftp)://[a-z0-9-]+(\.[a-z0-9-]+)+$` permet de reconnaître une URL de type http ://www.example.com ou ftp ://ftp.belnet.be (mais rien ne peut suivre le nom du serveur)

10.6.1 Utilisation à partir du Javascript

Pour utiliser une regexp dans le javascript, on se servira des fonctions test ou match. La première s'exécute sur la regexp et prend une chaîne en paramètre, la seconde s'exécute sur une chaîne et prend la regexp en paramètre. Si la regexp n'est pas trouvée, elles retournent toutes les deux une valeur fausse (null pour match par exemple)

Pour préciser une regexp, on la placera entre slashes. On peut éventuellement la terminer avec la lettre i (après le 2eme slash) pour indiquer qu'elle est *case* **i***nsensitive* (pas de différence majuscules/minuscules).

5. Cela permet aussi de *capturer* le texte correspondant, qui pourra être récupéré au niveau javascript

On aura ainsi :

```
var re = /^Hello World$/i;
var re2 = /[0-9]+/;
if (re.test("Goodbye World")) {
  ...
}
if ("ABC 123 DEF".match(re2)) {
  ...
}
```

Les regexp peuvent également être utilisées avec d'autres fonctions mais cela sort du cadre de ce cours.

Chapitre 11

L'arbre DOM

Introduction

Lorsque la page Web est reçue, l'ensemble du texte et des balises HTML est converti en une structure en arbre appelée *arbre DOM*.

A la tête de cette structure se trouve la balise BODY. Tout ce qui est contenu directement dans le body se retrouvera comme noeuds enfants du noeud principal, soit sous forme de noeud texte, soit sous forme de noeud élément.

Si une balise est incluse dans une autre balise, elle sera enfant de la balise extérieure. Il en sera de même pour le texte contenu dans une balise.

Par exemple, si on a

```
<body>
  <div>
    <p>
      Hello World
    <br>
      Goodbye World
    </p>
    <p>
      <span>
        Portez ce vieux whisky au juge blond qui fume
      </span>
    </p>
  </div>
</body>
```

Le noeud racine (body) contiendra un noeud div.

Le noeud div contiendra deux noeuds p.

Le premier noeud p contiendra 3 noeuds : un noeud texte, un noeud br et de nouveau un noeud texte

Le deuxième noeud p contiendra un noeud span

Le noeud span contiendra un noeud texte

Les différents noeuds de l'arbre DOM sont présents au niveau du Javascript et une série de fonctions et de propriétés permettent d'accéder à ces différents noeuds et de les manipuler. Lorsque l'on manipule l'arbre DOM, la page web se modifie en conséquence.

Lorsque l'utilisateur manipule des éléments de l'arbre DOM, il génèrera des *événements* auxquels il est possible d'associer des fonctions. Ainsi, l'utilisateur pourra démarrer du code Javascript qui au final affichera le résultat au travers de manipulations de l'arbre DOM.

11.1 Localiser un élément

Les différentes méthodes d'accès aux éléments utilisent un élément de référence. On peut donc utiliser ces fonctions et valeurs à partir de l'objet `document` qui est associé au body du document HTML mais également à partir de n'importe quel autre élément de l'arbre DOM.

Si on utilise ces fonctions à partir d'un autre élément que document, seule la partie de l'arbre DOM concernée (l'élément en question et les éléments qu'il contient (ses éléments descendant) seront concernés.

Pour retrouver un élément auquel un ID a été associé, on utilisera la fonction `getElementById()`. Cette fonction sera généralement utilisées à partir de l'objet document et elle prend un paramètre : la valeur de l'ID de l'élément à retrouver.

```
el=document.getElementById("Zone1");
```

Cette fonction retournera un et un seul élément ce qui ne pose normalement pas de problème étant donné que dans un document HTML, une valeur ne peut être utilisée que dans un seul ID (il est interdit de donner le même ID à deux éléments différents).

Si deux éléments venaient à avoir le même ID, le comportement dépendra du navigateur.

11.2 Modifier un élément

Les différentes caractéristiques d'un élément, ainsi que son contenu, peuvent être modifiés à partir du Javascript.

Le contenu d'un élément peut être modifié au travers de la propriété `innerHTML`. Cette propriété contient le contenu de l'élément formatté en HTML. Ce contenu peut être récupéré ou modifié à partir du Javascript.

Les éléments de type img disposent d'une propriété `src` qui contient l'URL de l'image. Il est également possible de modifier ou consulter les dimensions de l'image à l'aide des options `width` et `height`.

```
el.innerHTML='<span class="abc">Text</span>';
```

Les balises de type input disposent de propriétés `value` et `checked` qui permettent de connaître ou de modifier la valeur de la zone d'entrée (le texte entré dans le contrôle, le texte qui doit apparaître sur un bouton, la valeur qui sera envoyée lors d'un submit si la checkbox ou radiobox est cochée) ou l'état d'une checkbox ou radiobox.

```
<input type="text" id="nom">


function litNom() {
  var el=document.getElementById("nom");
  var n=el.value
  return n;
}
```

11.3 Les CSS

Les éléments possèdent une propriété `className` qui contient la liste des classes CSS associées par l'attribut class de l'élément. On peut préciser une ou plusieurs classe CSS, séparées par des espaces, tout comme dans l'attribut class.

La plupart des propriétés CSS sont également accessibles au travers de l'objet style associé à un élément. Ainsi, on peut accéder à la propriété `display` au travers de `el.style.display`.

Quand un nom de style est en plusieurs morceaux séparés par des tirets, on aura un nom composé des différents morceaux, chaque morceau sauf le premier ayant l'initiale en majuscule. Ainsi, `background-image` sera accessible au travers de `el.style.backgroundImage`.

Les propriétés CSS de base doivent être utilisées. On ne peut donc pas utiliser les propriétés *raccourcies* telles que `border`, `background`, . . .mais bien utiliser `borderTopWidth`, `backgroundColor`,. . . [1]

1. A noter que certains navigateurs pourraient accepter les propriétés raccourcies mais ce comportement n'est pas garanti sur tous les navigateurs

Les changements de style effectués au niveau des éléments passent en priorité sur ceux décrits dans les feuilles de styles.

Chapitre 12

Les événements

Introduction

En Javascript, le code ne sera pas exécuté de manière linéaire. Très peu de code sera exécuté lors du chargement de la page. La plus grande partie du code ne s'exécutera que suite à des événements.

Le premier des événements à traiter sera d'ailleurs celui indiquant que la page est chargée. Ce n'est qu'à ce moment là que l'on pourra commencer à agir sur l'arbre DOM.

Quand le javascript s'exécute, la page Web est d'ailleurs généralement figée. Ce n'est que quand le code Javascript aura fini de s'exécuter que l'utilisateur récupère le contrôle de la fenêtre de navigation.

12.1 Chargement de la page

Lorsqu'une page est chargée, cela se passe en plusieurs étapes :

— chargement du HEAD
— chargement et traitement des ressources précisées dans le HEAD : CSS, Javascript, ...
— chargement du BODY en parallèle avec le chargement des ressources du HEAD. Il est impossible de prévoir dans quel ordre ces tâches se termineront
— l'événement *load* est généré sur l'objet window

D'autres événements se déclencheront mais ne sont pas détaillés ici.

Pour être sûr que la page est chargée, il faudra donc attendre l'événement `window.onload` avant d'accéder à l'arbre DOM.

On utilisera donc généralement une fonction `init()` qui sera exécutée

une fois la page chargée [1] et le script se terminera par une ligne

```
window.onload=init;
```

12.2 les gestionnaires d'événement

Les différents éléments possèdent une série de points pour accrocher des gestionnaires d'événements. Ces points se nomment `on` suivi du nom de l'événement : `onclick`, `onmousemove`, ...

Lorsque l'événement se produit, le javascript appelle la fonction sauvée dans la variable correspondante en lui passant un paramètre : un objet `Event` qui contient les détails sur l'événement.

A noter qu'en Javascript, on n'est pas obligé de déclarer tous les paramètres dans la définition de fonction. Un gestionnaire d'événement qui n'a pas besoin de l'objet `Event` peut être défini comme une fonction sans paramètres (même si l'objet `Event` sera de toutes façons transmis).

Un gestionnaire d'événement peut également être défini comme une fonction anonyme

```
btn.onclick = function (ev) {
       ev.target.value="done";
   };
```

1. un peu l'équivalent du main() du langage C

Quatrième partie

PHP

Introduction

Le PHP est un langage interprété côté serveur qui permet de générer du HTML en temps réel.

Avant le PHP, on utilisait généralement des CGI [2] qui étaient des programmes exécutés par le serveur web et qui généraient le contenu que le serveur devait envoyer. La technologie CGI est relativement lente et difficile à programmer ce qui a entrainé le succès du PHP, nettement plus simple à mettre en place.

2. Common Gateway Interface

Chapitre 13

Les bases de PHP

Pour utiliser le PHP, il est nécessaire de stocker les pages sur un serveur Web qui supporte ce langage. Si on ouvre le fichier directement dans le navigateur (sans passer par un serveur web), le code PHP ne sera pas exécuté.

Le fichier aura pour extension `.php` (d'autres extensions sont possibles).

On pourra placer une ou plusieurs sections PHP dans ce fichier. Ces sections commenceront par `<?php` et se termineront par `?>`. On considérera que tout ce qui est en dehors de ces sections sera envoyé tel quel au navigateur [1]

L'ensemble du fichier sera considéré comme un programme unique. Une variable définie dans une section sera accessible dans une autre section, il est possible d'ouvrir un bloc après une condition dans une section PHP et le finir dans une section qui suit, ...

Les instructions PHP doivent toutes être terminées par un point-virgule (;).

13.1 echo et chaînes

La commande `echo` permet d'envoyer du texte au navigateur. L'ensemble des morceaux de texte présents hors des sections PHP et envoyés par des `echo` formera un tout continu qui sera interprété comme du HTML [2].

On peut utiliser la commande `echo` avec un nombre ou avec une chaîne de caractères. Dans le premier cas, le nombre sera converti en chaîne avant d'être envoyé.

Il existe deux types de chaînes de caractères [3] : les chaînes entre guillemets

1. On peut imaginer que les textes en dehors de ces sections PHP sont en fait envoyées par une instruction PHP echo
2. sauf si on précise un autre type de données
3. en fait, il y en a une troisième, les documents HERE, qui ne sera pas vue en première

"texte" et les chaînes entre apostrophes 'texte'. Dans le premier cas, les variables présentes dans la chaîne seront interprétées, dans le second cas, le texte qui correspond au nom de variable sera affiché tel quel.

Les chaînes entre guillemets permettent également d'utiliser des séquences d'échappement semblable à celles rencontrées en C comme \n pour insérer un retour à la ligne.

```
<?php
echo␣"Hello␣World\n";
?>
```

13.2 Les variables

En PHP, toutes les variables sont précédées du caractère $. Ce caractère permet de faire la différence entre variable et texte, entre autres dans les chaînes de caractères.

On peut utiliser les opérateurs classiques du C : +, -, *, / pour l'addition, la soustraction, la multiplication et la division. On dispose également des opérateurs % (modulo), ! pour inverser une valeur logique (négation), << et >> pour les décalages, ...

A tout ces opérateurs s'ajoute l'opérateur . utilisé entre deux chaînes de caractères pour les concaténer.

L'assignation se fait également comme dans le langage C à l'aide de l'opérateur =.

```
<?php
$a = 10;
$b = 20;
$c = $a + $b;
$d = "Hello " . 'world';
?>
```

Tout comme dans le langage C, il est possible d'utiliser les opérateurs d'assignation +=, -=, ...mais aussi .= par contre, il est conseillé de n'utiliser les opérateurs ++ et -- que dans leur forme postfixe ($a++).

13.3 Les conditions

les structures conditionnelles du C sont également disponibles en PHP :

```
if ($a==1) {
   ...
} else {
   ...
}

for($a=0;$a<5;$a++} {
   ...
}

while($a<10) {
   ...
}
```

La structure `switch` a été étendue pour permettre non seulement une variable numérique mais aussi une variable chaîne en paramètre :

```
switch($a) {
  case "abc":
     ...
  break;
  case "def":
  case "ghi":
     ...
  break;
}
```

13.4 Les tableaux

Un tableau en PHP est un objet capable de stocker plusieurs éléments, soit numérotés, soit étiquetés (on parle alors de tableau associatif ou de hash).

La fonction `array()` permet de créer un tableau. Cette fonction retourne un tableau dont les éléments sont ceux précisés en paramètres. Si aucun élément n'est précisé, il s'agira d'un tableau vide.

Ce tableau pourra être stocké dans une variable comme on stockerait un nombre ou une chaîne.

Pour accéder à un élément du tableau, on utilisera les crochets rectangulaires autour de la valeur sélectionnant l'élément (soit un entier, soit une chaîne de caractères).

```
<?php
```

```
$a=array();
$b=array(123, 456, "abc", "def");
$c=array();
$a[0]=42;
$a[1]=$b[2];
$c["xyz"]=98;
$c[0]="mno";
echo $b[0];
?>
```

A noter que les éléments d'un tableau peuvent être de type différent et qu'il est possible de mélanger les clés numériques et chaînes.

Dans le cas d'un tableau numérique où tous les éléments ont été créés, la fonction `count()` retourne le nombre d'éléments du tableau.

```
<?php
$a=array("que","j","aime","a","faire","apprendre","un","nombre",
         "utile","aux","sages");

for($i=0;$i<count($a);$i++) {
  echo "<div>" . $a[$i] . "</div>";
}
?>
```

Il est possible de sauver des tableaux dans un tableau, ce qui permet de créer un tableau à deux dimensions. Il suffira d'utiliser la notation `$a[$l][$c]` pour accéder à la ligne `$l` et à la colonne `$c`. A noter qu'il est nécessaire d'initialiser les lignes avec des tableaux (même vides).

```
<?php
$a=array(array(),array(),array());
$a[1][2]=42;
$a[3]=array();
$a[3][5]="abc";
?>
```

13.5 La session

Afin de conserver des informations concernant un utilisateur entre deux pages, PHP dispose d'un mécanisme de session.

Ce mécanisme est activé par une instruction `session_start()` qui doit être envoyée avant que le premier caractère ne soit envoyé au navigateur (même un retour de chariot est de trop).

Une fois cette instruction utilisée, le tableau `$_SESSION` est rétabli à sa valeur à la fin de la page précédente (utilisant la session) et un mécanisme est mis en place pour sauver le contenu de ce tableau une fois que la page a terminé de s'exécuter.

A noter que les variables de session ne sont accessibles qu'à partir du PHP sur le serveur même si le navigateur dispose de la clé qui permet d'y accéder sous la forme du cookie de session.

Si le navigateur est fermé ou si l'on attend suffisemment longtemps avant que la session ne soit réouverte, la session est perdue.

S'il est possible de vider complètement la session, il s'agit généralement d'une mauvaise idée. En effet, la session peut être utilisée pour sauver différentes informations sans aucun lien entre elles (comme la langue dans laquelle la page doit être affichée, le fait qu'un avertissement concernant les cookies ait été affiché, les informations d'authentification, ...) et effacer la session effacerait toutes ces informations en masse.

13.6 La gestion des formulaires

Au niveau de la communication entre le navigateur et le serveur, il est possible de transmettre des informations à deux endroits différents.

D'une part, on peut les communiquer dans l'URL, sous la forme d'une série de paires clé/valeur, on parlera de *paramètres GET*. D'autre part on peut les transmettre dans le contenu de la requête, on parlera alors de *paramètres POST*.

Les paramètres GET sont encodés sous la forme d'une série d'éléments `cle=valeur` séparés par des `&`. Ces paramètres sont séparés de l'URL par un caractère `?`.

```
http://www.exemple.com/index.php?param1=val1&param2=val2
```

On peut envoyer des paramètres GET directement dans une URL ou à l'aide d'un FORM dont la méthode est GET [4].

Il existe plusieurs façons pour encoder des paramètres POST. Les plus courantes sont `application/x-www-form-urlencoded` qui est l'encodage par défaut et `multipart/form-data`. Il existe d'autres encodages mais la plupart ne sont simplement pas gérés par PHP.

4. A noter que l'on ne peut pas utiliser de paramètres GET dans l'URL de la balise FORM dans ce cas

On utilisera généralement l'encodage par défaut, le second étant principalement utilisé lorsque l'on envoie des fichiers à l'aide d'un formulaire.

Au niveau PHP, les paramètres GET seront automatiquement sauvés dans le tableau associatif `$_GET` et les paramètres POST dans le tableau `$_POST`. On pourra donc utiliser `$_GET["param1"]` pour obtenir la valeur de `param1` et ainsi de suite.

Les éléments du formulaire seront transmis comme suit :

- Les éléments *disabled* ne sont pas transmis
- pour les checkboxes, seules les checkboxes cochées sont envoyées
- Pour les radioboxes, si aucune option n'a été sélectionnée, le groupe de radioboxes n'est pas envoyé
- chaque contrôle INPUT génère une paire clé/valeur composée de son NAME et de sa VALUE [5]
- Les SELECT seront envoyés avec le NAME du SELECT et la VALUE de l'option choisie.
- Le TEXTAREA enverra le contenu de la balise sous le NAME précisé dans la balise.
- Si un NAME apparaît plusieurs fois, il sera envoyé plusieurs fois (à chaque fois avec la valeur correspondante). Cela peut être exploité en PHP en utilisant un NAME du genre `abc[]` qui fournira au final un tableau contenant les différentes VALUE.
- Le bouton SUBMIT utilisé pour envoyer le FORM sera envoyé avec son NAME et sa VALUE. Cela permet d'avoir plusieurs boutons SUBMIT.

Pour les checkboxes, il faudra donc tester si le valeur est définie pour savoir si la case est cochée. Pour les autres contrôles, il suffira de lire la valeur [6]

5. Les checkboxes non cochées ne sont pas envoyées

6. il est préférable de vérifier qu'elle existe avant pour éviter des erreurs potentiellement exploitables en cas d'attaque

Chapitre 14

Les Bases de données

Introduction

Lorsque l'on désire un stockage persistant de données, il est généralement fait appel à des bases de données.

PHP supporte depuis longtemps de nombreuses bases de données. Cependant, ce support passait par de nombreux ensembles d'instructions, chacun spécifique à une seule base de données.

Afin de simplifier la programmation, une nouvelle interface a vu le jour nommée PDO. Les anciennes interfaces sont peu à peu dépréciées et pourraient disparaître avec le temps.

14.1 PDO

Pour créer une connexion sur une DB à l'aide de PDO, on utilise une des syntaxes :

```
$dbh = new PDO(connectstr);
$dbh = new PDO(connectstr,user,pass)
```

Le premier paramètre est la chaîne de connexion. Cette chaîne dépendra du moteur de base de données utilisé.

La seconde forme permet de fournir un nom d'utilisateur et un mot de passe pour la connexion.

La valeur retournée est la ressource de connexion de la base de données concernée (on peut effectuer des connexions vers plusieurs DB différentes dans le même script). En cas de problème, une valeur null est retournée.

La chaîne de connexion commence par le protocole de la base de données utilisée. Cela permettra à PDO de choisir le driver correspondant. Dans la

documentation, cette chaîne est souvent désignée par DSN et son format exact dépend de la base de données choisie.

Avant d'effectuer des requêtes, il faut les préparer. On peut disposer de plusieurs requêtes préparées en même temps et on peut réutiliser la même requete préparée plusieurs fois.

```
$stm = $dbh->prepare(requeteSQL);
```

Lorsque l'on prépare une requête, on remplace les valeurs variables par des ?. C'est ainsi que l'on transformera les requêtes

```
SELECT nom,prenom FROM utilisateurs WHERE prenom='Albert'
INSERT INTO produit(nom,prix) VALUES('clé USB',15)
UPDATE score SET highscore=1000 WHERE idusr=15
```

en des requêtes du genre :

```
SELECT nom,prenom FROM utilisateurs WHERE prenom= ?
INSERT INTO produit(nom,prix) VALUES(?,?)
UPDATE score SET highscore=? WHERE idusr=?
```

Outre le fait que cela permet de réutiliser la même requête plusieurs fois, ce remplacement des paramètres permet d'éviter les problèmes de type injection SQL vu que les termes variables (potentiellement venant du navigateur) seront transmis par la suite, sans être interprétés.

Pour exécuter une requête préparée, on utilisera la syntaxe suivante :

```
$stm->execute();
$stm->execute(array(valeur1,valeur2,...));
```

La seconde forme permet de fournir des valeurs qui remplaceront les points d'interrogation (?) dans l'ordre où ils apparaissent dans la requête SQL.

L'utilisation de requêtes préparées permet de ne pas devoir s'inquiéter de la présence de caractères spéciaux dans les valeurs transmises.

Une fois la requête exécutée, le résultat sera sauvé dans la *prepared statement* et devra être récupéré avant de réexécuter la requête (une nouvelle exécution effaçant les anciens résultats).

La manière classique pour lire les résultats est la boucle

```
while ($row = $stm->fetch()) {
... $row["colonne1"] ...
... $row["colonne2"] ...
}
```

cette boucle permet de récupérer dans la variable *row* chacune des lignes résultat de la requête, une par une. Les résultats seront accessibles au travers des noms des colones ou des numéros des colonnes (commençant par la colone 0)

Un cas particulier est celui où la ligne ne peut exister que zéro ou une seule fois. Dans ce cas, il n'est pas nécessaire d'utiliser une boucle, on peut utiliser directement

```
if ($row=$stm->fetch()) {
... [trouvé] ...
} else {
        ... [non trouvé] ...
}
```

14.2 SQLite 3

Dans les versions récentes de PHP, la librairie SQLite3 est généralement pésente et on peut utiliser les bases de données SQLite.

Une base de données SQLite 3 est contenue dans un seul fichier. Le fichier et le répertoire dans lequel il se trouve doivent pouvoir être modifiés par le PHP (création du fichier verrou, écriture dans la base de données,...)

Le DSN PDO pour SQLite 3 a la forme suivante : `sqlite:nomfichier` où nomfichier est le nom du fichier SQLite relativement au répertoire où se trouve le script PHP appelé.

Si on tente d'ouvrir un fichier qui n'existe pas, un nouveau fichier sera créé.

Un outil ligne de commande appelé SQLITE3.EXE [1] permet d'accéder à la base de donnée en mode console.

Sous SQLite3, la commande `.help` affiche un résumé des commandes propres à SQLite (non-SQL), telles que `.dump` qui permet d'afficher la totalité de la DB ou `.open` qui permet d'ouvrir un fichier de base de données.

SQLite 3, contrairement à de nombreux moteurs SQL, ne gère en interne que quatre types de données : `INTEGER`, `TEXT`, `REAL` et `NULL`. Tous les autres types SQL seront convertis dans ces 4 types.

De même, le support SQL de SQLite 3 est limité. Certaines commandes ou fonctionnalités n'existent pas.

Une commande particulièrement intéressante est la commande `.read` qui permet de lire un fichier externe contenant des commandes SQL à exécuter.

1. sqlite3 sous linux

Généralement, on placera les commandes servant à initialiser la base de données dans un fichier .SQL que l'on chargera ainsi dans la DB fraîchement créée.

14.3 SQL de base pour SQLite3

14.3.1 Créer une table

On créera une table à l'aide de la commande

```
CREATE TABLE nom_table (
     champ_1  type_1,
     champ_2  type_2,
       ...
     champ_n  type_n);
```

Le type pourra être n'importe quel type SQL mais sera au final transformé en INTEGER, TEXT ou REAL. Il est donc conseillé de se limiter à ces 3 types de données dans SQLite 3.

Pour créer un champ numérique auto-incrémenté, on utilisera la chaîne suivante pour le type :

```
INTEGER PRIMARY KEY AUTOINCREMENT
```

Cette définition est propre à SQLite 3 et devra être transformée si on désire utiliser une autre base de données.

14.3.2 Effacer une table

On effacera une table à l'aide de la commande

```
DROP TABLE nom_table;
```

Cette commande effacera immédiatement la table. Pas de *undo* possible ni ne demande de confirmation.

14.3.3 Ajouter une entrée

Pour ajouter une entrée, on utilisera la syntaxe suivante

```
INSERT INTO nom_table(champ_1, champ_2, ... , champ_n)
    VALUES (val_1, val_2, val_3, ... , val_n);
```

Si on désire insérer des données multiples, on pourra placer plusieurs ensembles de valeurs séparés par des virgules après le mot VALUES. Chaque ensemble de valeurs se trouve entre parenthèses.

On ne doit pas insérer de valeur dans le champ *auto-increment*.

14.3.4 Afficher les données

Pour afficher le contenu d'une table, on utilisera

```
SELECT champ_1, champ_2, ... , champ_n FROM  nom_table ;
```

Il est possible de remplacer la liste des champs par une étoile (∗) mais cela peut être source d'erreurs et n'est généralement pas conseillé.

Cette commande retournera tous les enregistrements de la table. Si on veut se limiter à certains enregistrements, il suffit d'ajouter une clause WHERE en fin de requête (avant le ;)

14.3.5 Mise à jour des données

Pour mettre à jour les données d'une table, on utilisera

```
UPDATE nom_table SET champ_1=valeur_1, champ_2=valeur_2, ... ;
```

De nouveau, si on ne désire pas modifier toutes les entrées de la table, il suffira d'ajouter une clause WHERE.

14.3.6 Effacement des données

Pour effacer les données d'une table (sans effacer la table), on utilisera

```
DELETE FROM nom_table ;
```

Cette commande vide complètement la table. Si on ne veut effacer que certaines entrées, on utilisera de nouveau une clause WHERE.

14.3.7 Choisir les enregistrements

Pour limiter une requête à certains enregistrements, on terminera la requête par une clause WHERE.

Cette clause commence par le mot clé `WHERE` et sera suivi d'une série de conditions séparées par des mots clés `AND` et `OR`. En cas de mélange de AND et de OR, penser à utiliser des parenthèses.

Les conditions seront exprimées en spécifiant un champ, un comparateur et une valeur (ou un autre champ) `age>18`. L'égalité sera testée à l'aide d'un simple = (contrairement au langage PHP par exemple) et l'inégalité sera testée par <>.

On peut également utiliser l'expression `champ IS NULL` pour détecter un champ dans lequel la valeur spéciale `NULL` a été écrite (ou un champ non initialisé) ou `champ IS NOT NULL` pour s'assurer qu'un champ n'est pas NULL.