Cours

Développement WEB

BTS Systèmes Numériques
Informatique & Réseaux

FÉLIX
LE DANTEC
du secondaire au supérieur

<HTML>

Chapitre 1 – Généralités
Chapitre 2 – Liens
Chapitre 3 – Contenus

BTS Systèmes Numériques
Informatique & Réseaux

FÉLIX
LE DANTEC
du secondaire au supérieur

Chapitre 1
HTML & Généralités

1.Préambule

La diapo 1 présentée précédemment fait état de différentes versions d'HTML et en particulier de la version 4. Aujourd'hui on parle de la version 5.

Quelles sont les différences entre ces deux versions ?

HTML5 apporte des évolutions majeures et sera la plateforme de développement principale dans un proche avenir.
Le balisage est la partie visible de l'iceberg de HTML5 avec entre autre le support natif des lecteurs vidéo/audio, la spécialisation des blocs conteneurs (div & span) etc.
Mais il faut ajouter à cela la naissance d'une véritable ***API*** (Application Programming Interface) pour étendre le dynamisme et l'interaction avec le ***DOM*** (Document Object Model – déjà existant depuis HTML4).

Liste non exhaustive des nouvelles fonctionnalités :

- Stockage des données coté client
- dessin 2D & 3D avec les canvas
- microformats
- lecture des flux audio/vidéo
- communication bidirectionnelle (websockets, serveur push etc.)
- géolocalisation
- applications web hors ligne
- lecture et écriture des fichiers locaux (client)

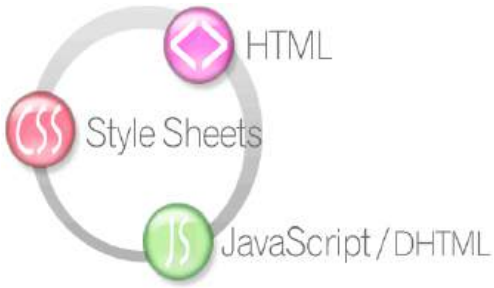
Peut-on considérer que HTML5 est de même nature que HTML4 ?

Le volet balisage d'HTML5 est semblable dans le principe que HTML4, en revanche sous le nom HTML5 on désigne une API de développement ce qui n'est pas le cas de la version 4.
Dans l'absolu :
HTML 5 = Balisage + Style + API
Le style permet, à la manière des traitements de texte, de mettre en forme les documents HTML. Ce sont des instructions spécifiques regroupées sous la norme ***CSS*** (Current Style Sheet).
L'API repose sur le langage « ***JavaScript*** » qui apporte le dynamisme, l'ergonomie et l'interaction pour développer de véritables applications exécutées par le navigateur.

Pour résumer :

HTML5 = HTML + CSS3 + JS6

HTML 4 + CSS2 + JS



Remarque :
*La version 5 n'est toujours pas validée par le consortium **W3C** (chargé de mettre en place les recommandations) mais déjà supportée par la plupart des navigateurs.*

2.Introduction HTML

HTML (HyperText Markup Language) est un langage de description de documents. Quelques autres langages, ayant la même vocation existaient au démarrage du World Wide Web, mais le besoin de définir un nouveau langage s'est imposé ; il fallait fournir une fonctionnalité supplémentaire : la ***navigation hypertexte***

> ***La navigation hypertexte** consiste à fournir la possibilité de cliquer sur certaines zones du document (lien hypertexte) pour accéder une autre partie du site, voire un autre site Web.*

Peut-on parler de programmation HTML ?

Comme écrit ci-dessus, HTML est un langage de description de documents mais en aucun cas un langage de programmation. La distinction est importante car elle permet de dissocier les fonctions de l'API dans sa globalité. La programmation sera faite en JavaScript.

Est-il nécessaire de programmer en JavaScript pour faire une page web ?

Non, HTML décrit la façon dont le navigateur interprète la page web, elle sera chargée une fois et ne bougera plus. On parle alors de page « statique ».

Qu'est-ce que cela signifie « interprète » ?

La navigateur lit le contenu du fichier et fait la distinction entre le contenu et la mise en page. Les informations sont lues à la volées (au moment du chargement) et interprétée en même temps quelle qu'en soit la source (fichier sur disque dur ou via la connexion réseau). Ce mode de lecture « instantanée » remonte à l'époque où les connexions étaient très lentes (via modem) et pour éviter d'avoir une page blanche pendant de très longues secondes, le navigateur affichait progressivement la page web en fonction de la vitesse de téléchargement.

3. Balisage d'une page HTML

Rien n'est plus facile que d'écrire un document HTML : il suffit d'écrire le texte dans un fichier (d'extension .htm ou .html).
Cependant, il existe des « instructions » HTML permettant de rendre un document plus complet, plus agréable à lire.

Qu'est qu'une instruction HTML ?

Toute instruction HTML, pour être reconnue du texte proprement dit, doit être entourée par les caractères **<** et **>** par exemple **`<hr>`** est donc une instruction du langage HTML (elle permet de placer un ligne horizontale de séparation dans le document). **On parle alors de balise HTML**.

3.1-Structure d'une page HTML

Elle doit débutée par une balise particulière notée **<html>** qui indique au navigateur qu'il va recevoir une page HTML. Cette balise indiquera aussi la fin du document.

Exemple :

```
<html>      ← début du document HTML
… document web
</html>     ← fin du document HTML
```

Tout ce qui est situé entre `<html>` et `</html>` est considéré par le navigateur comme étant du HTML.

Remarque :
Le caractère « / » permet de créer une balise dite « antagoniste » (on parle aussi de balise fermante) qui délimite la zone d'influence. Certaines balises ne nécessitent pas d'antagoniste .

On distingue deux parties dans une page HTML :

- **L'en-tête** du fichier HTML sert à définir diverses choses propres à la page HTML, mais qui n'ont pas forcément d'incidence sur le contenu de la page, comme par exemple le titre apparaissant dans la barre de titre du navigateur WEB.
- **Le corps** du fichier HTML qui sera la partie visible du document

a. L'en-tête

Elle est définie à l'aide des balises **<head>** et **</head>**.

Exemple :

```
<html>
      <head>
      … définition de l'en-tête
      </head>
      … suite du document HTML
</html>
```

On remarque que l'en-tête est intégrée au document html (située dans la zone délimitée par `<html>`).
Elle est **toujours** placée au début du document.

Quel est son rôle ?

L'en-tête contient des informations qui ne seront pas affichées dans la page web mais qui ont une influence sur celle-ci.
Elle permet :

- de titrer le document
- de définir des méta-données qui serviront à l'affichage, aux moteurs de recherches etc.
- d'inclure des scripts javascript, des instructions CSS
- de faire le lien entre le document HTML et des fichiers externes

Cela signifie que l'en-tête va contenir des balises qui lui sont propres et qui doivent être interprétées au début du chargement de la page web.

Exemple :

```
1.  <!DOCTYPE html>
2.  <html>
3.          <head>
4.                  <meta http-equiv="content-type" content="text/html; charset=utf-8">
5.                  <meta name="keywords" content="Lycée félix le dantec, lycée,BTS,licence pro">
6.                  <meta name="description" content="Lycée Félix le Dantec,  4 BTS et 3 licences Pro.">
7.                  <title>Lycée Félix le Dantec V2014</title>
8.                  <link rel="stylesheet" href="/media/system/css/modal.css" type="text/css">
9.                  <style type="text/css">
10.                 #jsn-page {
11.                         min-width: 1100px;
12.                 }
13.                 .boxtitle {
                                font-family: Verdana;
14.                             font-size: 12px;
15.                             font-weight: bold;
16.                             text-align: left;
17.                             color: #E9E9E9;}
18.                 </style>
19.                 <script src="/media/jui/js/jquery.min.js" type="text/javascript"></script>
20.                 <script type="text/javascript">
21.                         window.addEvent('domready', function() {
22.                                 SqueezeBox.initialize({});
23.                                 SqueezeBox.assign($$('a.modal'), {
24.                                   parse: 'rel'
25.                                 });
26.                         });
27.                         jQuery(window).on('load',  function() {
28.                                 new JCaption('img.caption');
29.                         });
30.                          (function($){
31.                                 $('.dropdown-toggle').dropdown();
32.                                 })(jQuery);
33.                 </script>
34.         </head>
    …Reste de la page HTML
    </html>
```

Exercice

A partir de l'exemple ci-dessus :

1-Donnez la zone d'influence de l'en-tête (numéros de ligne)

Lignes 3 à 34

2-Donnez les différentes balises incluses dans l'en-tête ainsi que leur rôle respectif.

<meta> , balise qui fournit des informations relatives à la page, mots clés pour les moteurs de recherche, la table d'encodage de caractères
<link>, associe un fichier externe de style modal.css
<style>, balise qui délimite du code CSS
<script> ligne 19, associe un fichier js externe
<script> ligne 20, délimite du code js

3-Que constatez-vous pour certaines d'entre elles - ligne 4 par exemple en comparaison avec la ligne 7 ?

<meta> ne délimite pas de zone d'influence, pas de </meta> d'ou le / en fin de balise

4-Que fait la ligne 8 ?

inclus le fichier externe modal.css pour la style du document

5-Qu'est ce qui différencie les lignes 19 et 20 ?

l'attribut src ligne 19 qui inclus le fichier jquery.min.js

b. Le corps

Le corps d'un fichier HTML débute par la balise <body> et se termine par </body>. Ce couple de balises succède à l'en-tête .
Elle peut prendre un certain nombre de paramètres qui permettent de spécifier les caractères généraux de la page comme les différentes couleurs dans la page (texte, fond, lien etc.), les polices et les tailles de caractères par défaut etc.
On utilise de moins en moins ces paramètres, la présentation et le style sont désormais décrits dans les feuilles de styles CSS.

Exemple :

```
<html>
      <head>
      ...zone en-tête
      </head>
      <body>
      ...zone du corps du document HTML
      </body>
</html>
```

Quel est son rôle ?

Cette balise permet de définir la partie du document html qui sera affichée dans la fenêtre du navigateur. Elle contient le texte, les images, vidéos etc. qui sont intégrés dans la page web.

Exemple :

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>TP 8.1 - Manipulation des Canvas</title>
5.          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6.          <link rel="stylesheet" type="text/css" href="style.css">
7.          <!--TP gestion graphique par canvas-->
8.      </head>
9.      <body>
10.         <canvas id="dessin" width="900" height="600"></canvas><span id="trait">Epaisseur:<br><input
    type="radio" name="epaisseur" onclick="chgEpaisseur(1);">1px<br>
11.             <input type="radio" name="epaisseur" onclick="chgEpaisseur(2);" checked="checked"
    >2px<br/><input type="radio" name="epaisseur" onclick="chgEpaisseur(4);" />4px<br>
12.             <input type="radio" name="epaisseur" onclick="chgEpaisseur(8);" />8px<br><br>
13.             <span id="gomme" onclick="gomme();">Gomme</span>
14.         </span>
15.         <div class="palette">Couleur active :<input type="text" size="6" maxlength="7" id="coulActive"
    ><span id="affCoul"></span><br>
16.
17.             <center><p id="bouton" onclick="genererImage();">Générer Image</p></center><br>
18.
19.             <p id="coords"></p><br>
20.         </div>
21.             <script type="text/javascript" src="script.js"></script>
22.     </body>
23. </html>
```

Remarque :
<!doctype html> ligne 1 de l'exemple précédent permet de signifier que la page web répond à la recommandation HTML5. Ce n'est pas une balise au sens strict, mais c'est obligatoire et cela indique au navigateur que ce qui suit est écrit suivant la recommandation HTML5.

3.2-les attributs

Les attributs sont des éléments associés aux balises parentes afin d'améliorer les propriétés de celle-ci. Certaines balises ne nécessitent pas d'attribut.

Exemple

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

http-equiv et content sont des attributs et leurs valeurs respectives sont placées entre ".

Exercice
D'après l'exemple page précédente :
1-Donnez les attributs et leurs valeurs respectives de la balise ligne6. Que signifient-elles ?

```
rel="stylesheet" type="text/css" href="style.css"
```

rel = désigne le type associé à <link>
type= le type pris en compte par le navigateur pour l'intégration dans la page
href = désigne le chemin et le fichier à intégrer

2-Donnez une balise qui renferme de attributs numériques ainsi que leur signification.

```
<canvas id="dessin" width="900" height="600">
```
width et height désigne la taille du canvas longueur et hauteur en pixels.

Remarque
<body> ***ne doit pas avoir d'attributs en HTML5*** (style imposé en CSS) en revanche avec **HTML4**, elle peut prendre un certain nombre de paramètres qui permettent de spécifier les caractères généraux de la page comme les différentes couleurs dans la page (texte, fond, lien etc.), les polices et les tailles de caractères par défaut etc.
On utilise de moins en moins ces paramètres, la présentation et le style sont désormais décrits dans les feuilles de styles CSS.
Malgré cela, voici quelques paramètres employés :

- ***background***, ce paramètre permet de spécifier le motif pour le fond la page. Il est important de noter que les dimensions du motif (au format GIF ou JPEG) ne sont pas importantes. En effet, si le motif est plus grand que la page alors il sera tronqué. Dans le cas inverse où le motif est plus petit, alors il sera répété autant de fois qu'il sera nécessaire pour couvrir la totalité de la page.
- ***bgcolor***, cet autre paramètre est utilisé pour spécifier la Couleur de fond de la page. En présence d'un paramètre 'background' c'est l'image de fond qui prévaut sur la couleur de fond. Malgré tout, il peut être intéressant de spécifier les deux paramètres. En effet, si la personne qui visualise une page est distante, ou si le réseau est encombré, les images vont arriver avec un certain décalage par rapport au code HTML : il serait donc être judicieux d'avoir, durant ce laps de temps, une Couleur rappelant les nuances du motif de fond, afin d'assurer une certaine cohésion pour le document.
- ***text***, permet de fixer la Couleur par défaut du texte de la page HTML. Il est bien entendu possible d'écrire des pages avec des textes de différentes Couleurs, mais en l'absence de spécification, celle définie par défaut sera prise. Si ce paramètre n'est pas initialisé, c'est la Couleur par défaut du navigateur Web qui sera prise le cas échéant.
- ***link***, de la même manière, ce paramètre permet de définir la Couleur des liens non encore explorés.
- ***vlink***, spécifie la Couleur des liens déjà visités.
- ***alink***, ce dernier paramètre permet donc définir la Couleur que prendrons les liens lorsqu'ils seront sélectionnés par un click souris ou bien le clavier (un instant relativement bref).

3.3-Commentaires

Comme pour la plupart des langages, HTML permet de faire des commentaires dans le code HTML. Le commentaire n'est pas destiné à être affiché et il doit être placé entre `<!-- commentaire -->`.

Recherchez dans l'exemple précédent un commentaire, indiquez la ligne et son contenu.

Ligne 7 - `<!--TP gestion graphique par canvas-->`

3.4-Les caractères spéciaux

L'intérêt du langage HTML est de pouvoir écrire des documents visualisables quelle que soit la plateforme utilisée (Système Unix, Windows, ...).

Un problème se pose toutefois . La table de caractères ASCII souvent utilisée, ne permet que de coder 128 caractères. Les caractères accentués ne font pas partie de cette table de caractères.

Cependant, chaque plate-forme a étendu, à sa guise, le jeu de caractères ASCII à 256 caractères, bien entendu, simple e accentué (é ou è) n'est pas codé de la même manière. A cela il faut ajouter la langue dans laquelle le document a été écrit (Les caractères accentués n'existent pas en anglais).

Pour pallier à ce petit problème, des séquences de caractères ont été définies pour pouvoir spécifier un caractère non ASCII. Le tableau qui suit, donne les correspondances entre les séquences de caractères et le caractère sous-jacent (notez que les signes < et >, utilisés par le langage, sont donc eux aussi utilisables dans vos documents par le biais d'une séquence particulière).

3.5-Les couleurs

En HTML, il y a deux façons de spécifier une couleur.

La première consiste à donner les trois composantes RGB de la couleur désirée sous le format "#RRGGBB", où RR représente l'intensité de rouge entre 00 et FF (en hexadécimal), GG l'intensité de vert et BB celle du bleu : il est donc possible de définir jusqu'à 16 581 375 couleurs.

La seconde méthode, plus simple, consiste à simplement nommer la couleur désirée. Cette méthode est peu précise, car elle implémente un nombre restreint de couleurs.

Exemples :
Rouge = #FF0000 Vert=#00FF00 Bleu=#0000FF Blanc=#FFFFFF Noir=#000000

Exercice

Écrire la ligne permettant de définir les couleurs par défaut dans le corps du document HTML.
Couleur du fond = gris Lien non actif, déjà visité en jaune
Texte = bleu

```
<body bgcolor="#CCCCCC" text="#0000FF" link="#FFFF00" vlink="#FFFF00">
```

Séquences	Caractères	Séquences	Caractères
&lt;	<	&eacute;	é
&gt;	>	&egrave ;	è
&amp;	&	&ecirc ;	ê
&quot;	"	&ocirc ;	ô
&agrave;	à	&ugrave ;	ù
&acirc;	â	&ucirc ;	û
&icirc;	Î	&uuml ;	ü
&ccedil;	ç		

3.6-Table d'encodage

Les caractères transmis entre client et serveur sont définis dans une table d'encodage. A la base on se réfère toujours à une table dite ASCII (American Standard Code for Information Interchange).

Cette table regroupe 128 caractères :
- du 0 au 31, servent au contrôle
- du 32 au 126, ponctuation et alphanumérique
- 127ème, caractère d'effacement

Ne sont pas pris en compte entre autre les caractères accentués.

Les tables d'encodage permettent donc palier ce défaut et d'ajouter les caractères manquants suivant la langue employée. UTF8 est un encodage qui est réalisé sur 4 octets maximum contrairement à ASCII qui est effectué sur 7 bits (1 octet).
Le système unicode s'appuyant sur la table d'encodage UTF-8 doit prendre en compte toutes les langues.

4.Règles d'écriture en HTML

Le navigateur semble extrêmement tolérant en terme d'interprétation des instructions HTML. Le W3C a toujours souhaité que la rétro-compatibilité soit effective, les différentes versions d'HTML doivent être prises en compte. Cela se traduit par un traitement complexe et un mode de fonctionnement particulier du navigateur – le mode « QUIRK ».

Pour éviter cela, il existe une syntaxe HTML5 (issue de la recommandation XHTML) qui consiste à :

-Toutes les balises encadrantes doivent être fermées sauf les balises `<img>`, `<hr>`, `<br>` par exemple. En XHTML il y a obligation de fermer toutes les balises en ajoutant un / à la fin (`<br />`)
-le nom des balises et attributs écrit en minuscules
-toutes les valeurs des attributs sont toujours placées entre apostrophes ou guillemets
-chaque attribut doit avoir une valeur y compris ceux qui étaient considérés comme vides, par exemple `checked` s'écrit `checked="checked"`
-tout comme HTML, les balises doivent être correctement imbriquées , la première ouverte sera la dernière fermée. Si cette propriété n'est pas respectée, la/les balise(s) impliquée(s) ne sera(ont) pas interprétée(s)

Il est fondamental de les respecter.

Chapitre 2
HTML & Liens

Les liens Hypertextes ou sur Objets (images, fichiers etc.) sont l'essence même du Web. Ils permettent de naviguer de page sur une même page, de page en page ou même de site en site, ils assurent aussi la possibilité d'échanger des fichiers de tout type tant que le navigateur n'est pas capable de l'interpréter.

1.Les liens hypertextes

Un lien hypertexte est considéré comme un élément actif d'une page Web, en cliquant sur un mot ou un ensemble de mot, on peut « surfer » au grès de nos envies. Comme son nom l'indique, le lien hypertexte n'est réalisé que sur des caractères de type Texte.

Il existe 3 types de liens hypertextes :

- Ancre dans une même page, qui permet quand la page est beaucoup plus grande que la taille verticale de l'écran, d'accéder directement à l'information à partir d'un menu.
- Lien interne de page en page sur un même site
- Lien externe inter-sites

D'une manière générale, c'est toujours la même balise qui est employée `<a ...> … </a>`.

1.1-Ancre

Une Ancre permet de se déplacer directement sur un emplacement donné dans une même page sans utiliser « l'ascenseur vertical » du navigateur.

Exemple :

Menu en haut de page

```
Introduction
Chapitre 1- Les Régions Françaises
Chapitre 2 – Les Départements
Chapitre 3 – Les préfectures
```

En supposant que l'introduction et les 3 chapitres font parties intégrantes de la même page, en cliquant sur l'un des chapitres, on accède immédiatement à la zone de texte correspondante.

Syntaxe :

```
<a href="#nom_de_lancre">Lien hypertexte</a>
```

`href`, attribut qui désigne la cible vers laquelle pointe le lien hypertexte placé entre les balises `<a>` et `</a>`. On remarque aussi que le lien peut comprendre plusieurs mots séparés par des espaces et la ponctuation. **La valeur de href doit obligatoirement** commencer par un `#` pour désigner une ancre dans la page.

L'ancre est crée de la façon suivante :

```
<a name="nom_de_lancre"></a>
```

C'est l'attribut `name` qui distingue le lien de l'ancre. Cette fois on utilise name sans `#` devant le nom de l'ancre. Il n'est pas nécessaire de placer du texte entre `<a>` et `</a>`.

Exercice :
En reprenant l'exemple précédent et en supposant que l'on nomme l'ancre 'dpt' pour accéder au chapitre départements :

1- Écrire le lien du menu en haut de page pointant sur l'ancre dpt.

```
<a href="#dpt" >Les départements</a>
```

2- Écrire l'ancre qui caractérise le chapitre 'Les départements', on supposera que , d'un point de vue HTML, c'est un titre de type h3.

```
<a name="dpt"></a><h3>Les Départements</h3>
```

Remarque :
Une ancre peut être désignée par l'attribut 'id' dans n'importe quelle balise. Cet attribut est employé dans d'autres contextes qui seront vus par la suite.
`<p id="article1">bla bla bla … </p>` → *ici le paragraphe a une ancre nommée article1 et qui correspond aussi à un id.*

1.2-Lien interne

L'ancrage est considéré comme un lien interne mais pour passer de page en page on associe en paramètre l'URL de celle-ci de deux façons :

- Soit en **absolu** en tenant compte de la totalité de l'URL
- Soit en **relatif** en prenant en compte la racine du site ou de la page active

Exemples :
Sur le site 'monsite.fr', lien entre pageA.html et pageB.html (lien sur la pageA.html).

En absolu :

```
<a href="http://www.monsite.fr/pageB.html"> Voir la page B </a>
```

En relatif :

Cas où pageA et pageB sont dans le même répertoire

```
<a href="pageB.html"> Voir la page B </a>
```

Cas où page B est située dans un sous répertoire 'chap1' (racine_du site/chap1) pageA étant dans sous la racine du site

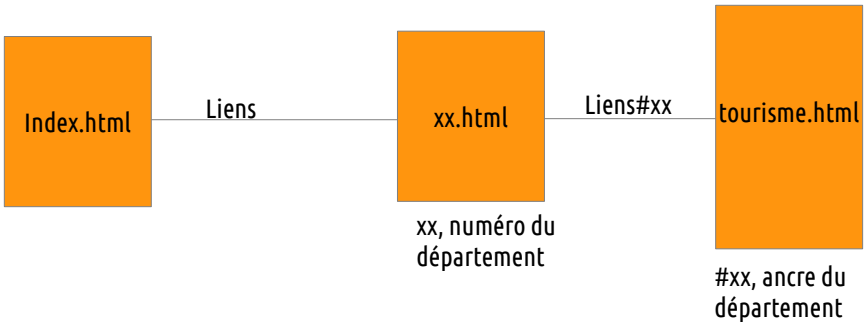
```
<a href="chap1/pageB.html"> Voir la page B</a>
```

Le navigateur prend en compte la première requête de connexion pour compléter dynamiquement l'URL.

Exercice

Soit le site suivant « bretagne.bzh » qui comprend 6 pages html :
-index.html, accès principal au site qui renferme un menu pour accéder aux pages de présentation des départements bretons
-22.html, 29.html, 35.html et 56.html
-tourisme.html, page de présentation du tourisme

La carte du site est la suivante :



Les visiteurs accèdent aux différents pages des départements à partir d'index.html. Puis peuvent accéder **directement** à la page tourisme.html **et** au chapitre correspondant au département qu'il souhaite visiter via une ancre.

L'arborescence du site :

www.bretagne.bzh/index.html
```
        +departements/xx.html
        +articles/tourisme.html
```

1-Que caractérise « www.bretagne.bzh » sur le serveur web ? Pourquoi utilise t-on un nom de domaine ?

Une URL est caractérisée par un chemin et un fichier car un serveur web ou http est avant tout un serveur de fichier. Elle masque l'arborescence réelle sur le serveur par un nom de domaine pour des raisons de sécurité.

2-Quelle URL permet d'accéder directement à index.html ? Est-ce un accès en absolu ou relatif ?

www.bretagne.bzh/index.html
Accès en absolu obligatoirement pour la racine du site.

3-Si on suppose que www.bretagne.bzh correspond réellement à /var/www sur le serveur, quel est le chemin complet pour accéder à index.html ? Même question pour accéder à la page des Côtes d'Armor.

/var/www/index.html

/var/www/departements/22.html

4-On souhaite créer le menu d'index.html. On vous demande d'écrire le lien qui permet d'accéder à la page de présentation des Côtes d'Armor.
En absolu :

```
<a href="http://www.bretagne.bzh/departements/22.html">Côtes d'Armor</a>
```

En relatif :

```
<a href="departements/22html">Côtes d'Armor</a>
```

5-On vous demande d'écrire l'ancre du département des Côtes d'Armor en respectant la carte du site présentée page précédente.

```
<a name="22"></a>
```

6-Écrire le lien qui pointe sur la page tourisme au chapitre des Côtes d'Armor à partir de la page du même département.
En absolu :

```
<a href="http://www.bretagne.bzh/articles/tourisme.html#22">Tourisme en Côtes d'Armor</a>
```

En relatif :

```
<a href="../articles/tourisme.html#22">Tourisme en Côtes d'Armor</a>
```

1.3-Lien externe

Le lien externe permet d'accéder à un autre site. Compte tenu que les noms de domaines ne sont plus les mêmes, le lien ne pourra être écrit qu'en absolu.

2.Les attributs de <a>

En règle générale on utilise les paramètres suivant pour définir un lien :

- href , désigne l'URL du lien et/ou ancre
- name, désigne l'ancre dans une page
- id, identificateur à utiliser pour Javascript ou CSS
- title, affiche un texte flottant quand la souris passe sur le lien
- target, désigne vers quelle fenêtre va s'ouvrir le lien

« target » prend les valeurs suivantes :

Valeur	Signification
`_blank`	Ouvre une nouvelle fenêtre pour afficher la page visée par le lien
`_self`	Affiche la page ciblée dans la fenêtre active (valeur par défaut si target n'est pas spécifiée)
`_parent`	Ouvre le lien sur la page hiérarchiquement supérieure
`_top`	Ouvre la page au-delà du « FRAMESET »
`Nom du cadre`	Cas particulier (voir les iFrames/Frameset)

Exercice

Écrire le lien « Bretagne » permettant d'ouvrir une nouvelle fenêtre pour accéder sur le site « bretagne.bzh », le lien sera marqué par un texte flottant au passage de la souris indiquant « Site internet des départements bretons ».

```
<a href="http://www.bretagne.bzh" target="_blank" title="Site internet des départements bretons">Bretagne</a>
```

3.Liens sur objets

Le lien sur objet correspond à un lien sur un élément de la page Web qui n'est pas du texte. La plupart du temps ce sont des liens sur images mais on peut aussi réaliser sur les fichiers, des documents multimédia (rich média) etc.

3.1-Liens sur images

Avant même de faire un lien sur une image, il faut l'intégrer dans la page web. Pour cela on utilise la balise `<img>`.
Cette balise dispose d'un certains nombre de paramètres :

- **`src`**, paramètres qui contient l'URL de l'image à afficher (même méthode que le paramètre HREF pour <a>)
- **`width`** et **`height`** , qui définissent la taille à l'affichage , valeur exprimée soit en pixels soit en % de la taille d'origine.
- **`alt`**, affiche une courte description quand la souris passe sur l'image
- **`border`**, affiche un cadre autour de l'image, la valeur de celui-ci en pixels définit l'épaisseur du cadre
- **`hspace`** et **`vspace`**, attributs fixant un espace vertical et horizontal autour de l'image
- **`align`**, permet de positionner l'image par rapport à un texte. Cet attribut peut prendre 9 valeurs

<img> associée à <a> permet de créer un lien à condition que cette dernière inclus la première.

Exercice
1- En vous référant à l'exercice de page 2, on vous demande d'insérer dans la page `index.html` l'image « `carte.png` », celle-ci est stockée dans le répertoire images situé sous la racine du site. L'image insérée doit avoir une taille de 640 x 480 pixels.
En relatif :

```
<img src="images/carte.png" width="640" height="480">
```

2- Ajoutez un lien sur cette image pour que l'on accède au site de la région Bretagne « `www.bretagne.fr` » dans une nouvelle fenêtre.

```
<a href="http://www.bretagne.fr"><img src="images/carte.png" width="640" height="480"></a>
```

3.2-Taille et format des images

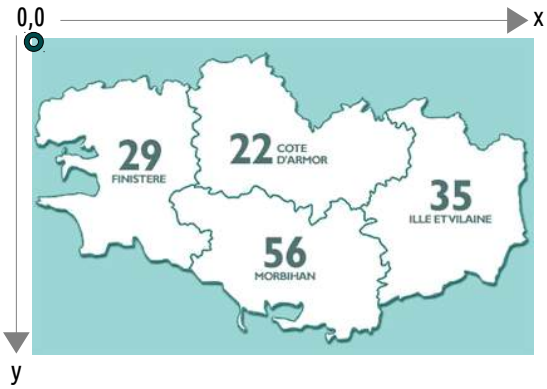
Un navigateur est susceptible de lire n'importe quelle image toutefois il est fortement conseillé d'en utiliser certains particulièrement bien adaptés pour le web comme **`png`**, **`jpg`** ou **`gif`**.

Le format png est d'autant plus intéressant qu'il gère nativement la transparence comme le gif et une grande palette de couleur comme le jpg, de plus le format est libre.

Dans tous les cas, ces formats permettent de compresser une image tout en conservant le meilleur rendu possible. Au final, la taille de l'image (en octets) doit être minimisée pour garantir un téléchargement rapide surtout si la page html en contient plusieurs.

Les paramètres width et height réduisent la taille à l'écran mais en aucun cas son « poids » en octets. Le navigateur mettra autant de temps à télécharger l'image. Donc il est conseillé d'utiliser un logiciel de traitement d'image avant publication.

Remarque - Coordonnées d'une image



La référence d'une image est toujours située en haut à gauche, ce point à pour coordonnées 0 en x et 0 en y.

L'axe horizontal est défini en x et le vertical en y. L'unité est le pixel, point relatif et dépendant de la résolution de l'écran.

3.3-Mapping

Méthode qui permet, à partir d'une seule image, de créer plusieurs liens placés à différents endroits sur cette même image. Pour cela on utilise la balise `<map>`, on lui associe un paramètre `name`. Il ne faut surtout pas oublier de nommer cette balise, cela permet de faire le lien avec l'image qui va servir au mapping.

Principe

Le navigateur a besoin de calculer les formes géométriques associées à des liens puis de les associer à une image. Naturellement, il faut que le mapping soit cohérent avec la taille de l'image.
Pour « mapper » une image, il faut définir les formes, 3 possibles :

- rectangulaire avec deux sommets (haut à gauche et bas à droite)
- circulaire avec les coordonnées du centre et le rayon
- polygonale avec tous les points de la forme

Mise en place du mapping en HTML

La balise `<map>` permet de créer le mapping, il faut que cette balise soit située au-dessus de l'image associée au mapping :

```
<map name="bzh">
      <area shape="rect"   coords="x1,y1,x2,y2" href="url1">
      <area shape="circle" coords="x,y,r" href="url2">
      <area shape="poly"   coords="x1,y1,x2,y2,x3,y3,...,xn,yn" href="url3"
</map>
<img src="carteBzh.png" usemap="#bzh">
```

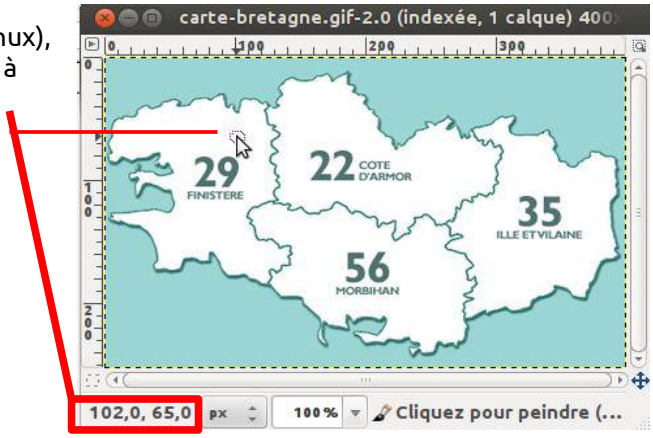
Les attributs :

- `shape` désigne la forme découpée
- `coords` définit les points caractéristiques de la forme
- `href` définit le lien associé à la forme

Le paramètre `usemap` de la balise img associe l'image au découpage.

Détermination des valeurs de coords

Avec GIMP (logiciel de retouche d'image sous Linux), les coordonnées du curseur sont affichés en bas à gauche de la fenêtre. C'est un moyen simple de trouver les bonnes valeurs.



3.4-Liens sur fichier

Les liens sur fichier peuvent être interprétés différemment suivant les extensions prises en charge par le navigateur. En effet, si celui-ci fait le lien avec une application installée sur le poste client, il peut exécuter un « plug-in » pour l'afficher directement (Fichier PDF, certains fichiers texte etc.).

En revanche, si une extension est non reconnue, le navigateur procède au téléchargement du fichier. A vrai dire, toutes les informations sont téléchargées, certaines seront directement interprétées par le navigateur ou un logiciel externe d'autres seront enregistrées localement.

Chapitre 3
HTML & Cadres – Contenus

Il arrive parfois que la mise en page soit complètement déstructurée en changeant de résolution d'écran. L'utilisation des cadres et tableaux permet de maintenir une cohésion dans l'affichage de la page HTML grâce à un astucieux système d'auto-adaptation.

1.Tableau

L'insertion d'un tableau se fait par le biais de la balise `<table>...</table>`. On lui adjoint ensuite des balises qui permettent de créer des lignes et enfin des cellules. Les colonnes ne sont pas déclarées afin de faciliter la fusion des cellules.

1.1- Exemple d'un Tableau simple

Tableau contenant 1 ligne et 2 colonnes :

```
<table>
      <tr>
            <td>Cellule 1</td><td>Cellule 2</td>
      <tr>
</table>
```

Un tableau est défini en fonction du nombre de lignes (axe vertical) et du nombre de cellules par ligne (axe horizontal). Ces cellules renferment le contenu du tableau à mettre en page. On ne parle pas des colonnes car elles ne disposent pas de balise HTML dédiée.

Exercice

1- D'après vous à quoi correspondent les balises <tr> et <td> ? Quels sont leurs rôles respectifs et quelles en sont les contraintes?

<tr> permet de définir une ligne et contient des cellules. Elle doit être fermantes.
<td> caractérise une cellule et son contenu doit être encadré, autre remarque une cellule est incluse dans une ligne et pas l'inverse.

2-On vous demande de créer un tableau simple contenant 2 lignes et 3 colonnes. Ci-dessous le contenu de celui-ci :

Ville	Département	Population
Lannion	22	19 920 (2011)

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <table>
        <tr><td>Ville</td><td>Département</td><td>Population</td></tr>
        <tr><td>Lannion</td><td>22</td><td>19 920 (2011)</td></tr>
    </table>
</html>
```

3-On souhaite que sur la première ligne, le texte des cellules soit en gras . Modifiez le tableau en conséquence.

```
<tr><td><b>Ville</b></td><td><b>Département</b></td><td><b>Population</b></td></tr>
```

Remarques
Par défaut le tableau fait toute la largeur de l'écran, le texte est systématiquement justifié à gauche et les bordures des cellules sont fixées. Il existe des paramètres qui permettent de personnaliser le tableau.

1.2- Attributs de `<table>`

En HTML5, cette balise (comme beaucoup) n'ont pas d'attribut, le style et la mise en forme doit être faite avec CSS.
Toutefois avec HTML4 il est possible de définir certains attributs :

- **`border`**, définit l'épaisseur des traits de bordure en pixels
- **`width`**, définit la largeur du tableau soit en pixels soit en % par rapport à la taille du cadre qui contient le tableau
- **`summary`**, permet d'afficher une description courte du tableau (en règle général en haut de tableau)
- **`bgcolor`**, permet de colorer le fond du tableau suivant le codage HTML RGB en code Hexadécimal (voir cours chapitre 1)
- **`frame`** et **`rules`**, permettent de modifier les lignes séparatrices des cellules, FRAME définit l'apparence des bordures du tableau et RULES définit quels lignes sont tracées (verticales, horizontales etc.)
- **`cellpadding`** & **`cellspacing`**, permettent de définir l'espacement soit du contenu de la cellule avec sa bordure soit entre cellules

1.3-Lignes et cellules

En HTML5, `<tr>` et `<td>` sont soumises aux mêmes règles que `<table>`. Avec HTML4, il y a certains attributs autorisés.

Pour les lignes :

- **`align`**, permet de justifier le contenu de toutes les cellules d'une même ligne (center, right ou left)
- **`bgcolor`**, permet de colorer le fond de toutes les cellules d'une même ligne

Pour les cellules :

- **`bgcolor`**, affecte une couleur en fond de cellule
- **`width`**, fixe la largeur de la cellule en pixels ou en % (Attention à la taille du tableau)
- **`align`**, permet de justifier le contenu dans la cellule (center, right ou left)
- **`rowspan`** et **`colspan`**, permettent de créer des fusions de cellules respectivement en ligne et en colonne. La valeur associée correspond au nombre de ligne ou de colonne fusionnées

Exercice

On vous demande de coder le tableau fourni en capture ci-dessous, la taille de celui-ci sera fixée à 500 pixels, la couleur de fond de la première ligne sera grise (`#c0c0c0`). Le titre du tableau est associé à la balise **`<caption>`*`titre`*`</caption>`** située en début de tableau et avant la définition de la première ligne.

Tableau et fusion de cellule

Départements	Villes	
22	Lannion	19 920 (2011)
	Saint Brieuc	46 173 (2011)
	Dinan	10 851 (2011)
	Guingamp	7 276 (2011)

```
<table border="1" width="500">
        <caption>Tableau et fusion de cellule</caption>
        <tr align="center" bgcolor="#c0c0c0"><td><b>Départements</b></td><td
        colspan="2"><b>Villes</b></td></tr>
        <tr><td rowspan="4">22</td><td>Lannion</td><td>19 920 (2011)</td></tr>
        <tr><td>Saint Brieuc</td><td>46 173 (2011)</td></tr>
        <tr><td>Dinan</td><td>10 851 (2011)</td></tr>
        <tr><td>Guingamp</td><td>7 276 (2011)</td></tr>
 </table>
```

1.4-Balises complémentaires

Pour faciliter la gestion des styles et affichages avec CSS, HTML5 fournit des balises « enfants » dissociant le tableau en 3 parties distinctes :

- En-tête du tableau qui correspond à la (aux) première(s) ligne(s) avec la balise <thead>...</thead>
- Le pied du tableau qui correspond à la (aux) dernière(s) ligne(s) avec la balise <tfoot>...</tfoot>
- Le corps du tableau avec <tbody>...</tbody>

Description d'un tableau avec <thead>, <tbody> et <tfoot> :

thead, tfoot et tbody

Villes	Départements
Lannion	22
Rennes	35
Brest	29
Cours Tableaux en HTML5	

Code HTML du tableau

```
<table border="1">
  <caption>thead, tfoot et tbody</caption>
  <thead><tr><th>Villes</th><th>Départements</th></tr></thead>
  <tfoot><tr><td colspan="2">Cours Tableaux en
   HTML5</td></tr></tfoot>
  <tbody>
          <tr><td>Lannion</td><td>22</td></tr>
          <tr><td>Rennes</td><td>35</td></tr>
          <tr><td>Brest</td><td>29</td></tr>
  </tbody>
 </table>
```

Constatations :

- <tfoot> et <thead> sont déclarées avant <tbody>, cela permet au navigateur d'anticiper le calcul du rendu
- <thead> dispose d'une balise spécifique pour les cellules <th>
- <tfoot> et <tbody> conservent la balise <td>
- <th> formate le texte en gras et centré automatiquement
- Comme ces balises délimitent des zones elles doivent être fermées systématiquement

Remarque
<th> dispose d'attributs particuliers, le cours n'étant pas exhaustif, vous trouverez des informations complémentaires sur le site du W3C (http://www.w3.org/community/webed/wiki/HTML/Elements/th).

2.Les cadres

Ce chapitre est divisé en 2 parties :

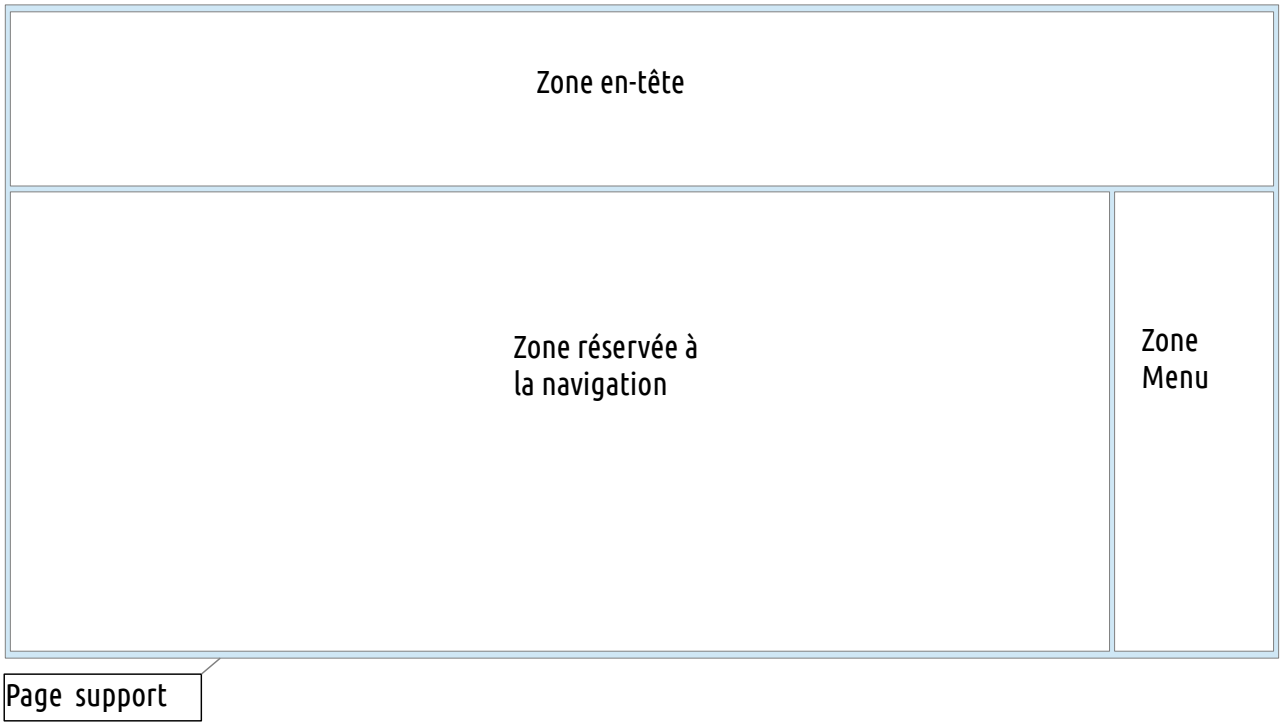
- Les FRAMES qui sont utilisés en HTML4 (ne sont plus supportées avec HTML5)
- Les iFRAMES susceptibles de remplacer les premiers avec HTML5

Nous aborderons les deux parties car il existe encore de nombreuses applications web qui reposent sur les FRAMES (interface de configuration des switchs par exemple).

2.1-Structure type avec cadres

Les cadres permettent de découper la page support en parties complètement indépendantes. Chacune d'entre elles affiche un contenu différent (une autre page HTML). Cela permet de mettre en œuvre une zone réservée à la navigation (contenu **dynamique** ou changeant) et des zones **statiques** comme une en-tête ou un menu.

Exemple de structure à base de cadres :



Il est possible de créer autant de cadre que l'on souhaite, toutefois 3 ou 4 cadres maximums suffisent, il faut assurer la lisibilité du site.

2.2-Page support

C'est la première page HTML à créer car elle permet de définir la position, la taille de chaque cadre et leur contenu par défaut. C'est cette même page que l'on chargera avec le navigateur.

Structure de la page support

```
1.  <!doctype html>
2.  <html>
3.      <head>
4.          <title>Exercices Cadres</title>
5.      </head>
6.      <frameset  ...>
7.          <frame ...>
8.          <noframes>
9.               … Partie affichée quand le navigateur ne prend pas en charge les cadres
10.         </noframes>
11.     </frameset>
12. </html>
```

Que remarque t'on à propos du contenu de la page support ?

Elle n'a pas de balise `<body>`, ce qui signifie que rien dans cette balise sera affiché.

Quelle est la balise de déclaration de cadres ?

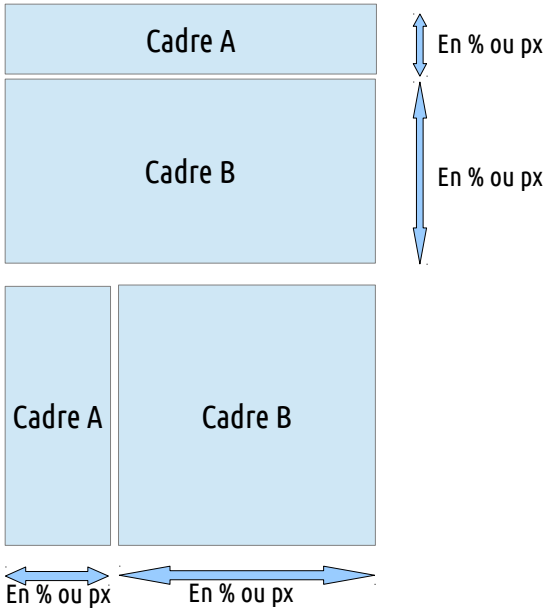
C'est la balise `<frameset>` car elle signale au navigateur qu'il va devoir insérer des cadres à la place d'un contenu normalement définit par `<body>`

Comment sont déclarés et configurés les cadres ?

Grâce à la balise `<frame>`, il doit y en avoir autant que de cadre à mettre en place.

2.3-Configuration des cadres

<frameset> définit la taille des cadres (par rapport à la fenêtre) et leur orientation (en lignes ou colonnes).



Ci-contre cadre A et cadre B sont positionnés verticalement donc définis en nombre de lignes. Ces valeurs doivent être en % ou en pixels. Dans tous les cas la référence correspond à la taille de fenêtre :

```
<frameset rows="taille CadreA,taille Cadre B">
```

Cadre A et cadre B sont positionnés horizontalement donc définis en nombre de colonnes. Ces valeurs doivent être en % ou en pixels. Dans tous les cas la référence correspond à la taille de fenêtre :

```
<frameset cols="taille CadreA,taille Cadre B">
```

Exercice
On veut 3 cadres superposés les uns aux autres, le premier fera 20 % de la fenêtre et le second 50 %. On vous demande d'écrire la déclaration des cadres à l'aide de frameset.

```
<frameset rows="20 %,50 %,30 %">
```

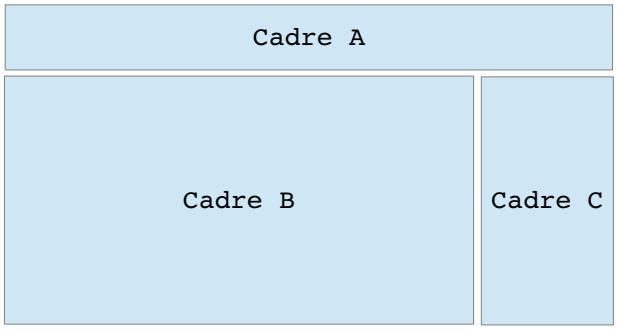
Remarques:
*Il n'est pas possible avec **une même balise** frameset de déclarer des cadres verticaux et horizontaux. En revanche l'imbrication des framesets est possible (voir plus bas).*
Seconde remarque la définition de la taille d'un cadre en pixel permet de le figer quelque soit la taille de la fenêtre :

<frameset cols="200,*">

Dans ce cas, le premier cadre fera 200 pixel de large et le second occupera le reste de la fenêtre.

2.4-Cadres imbriqués

On souhaite mettre en place la structure ci-dessous :

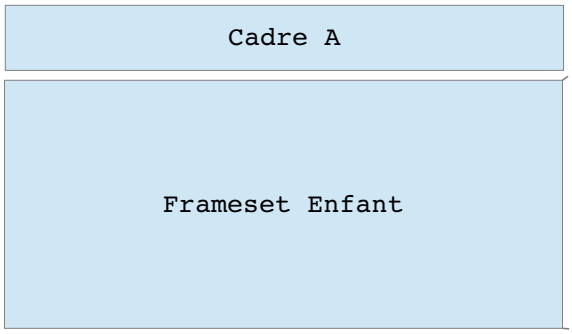


La page doit contenir 3 cadres.
Les cadres B & C définis horizontalement et le cadre A verticalement or nous avons vu précédemment qu'il n'était pas possible d'utiliser cols et rows pour un même `<frameset>`.

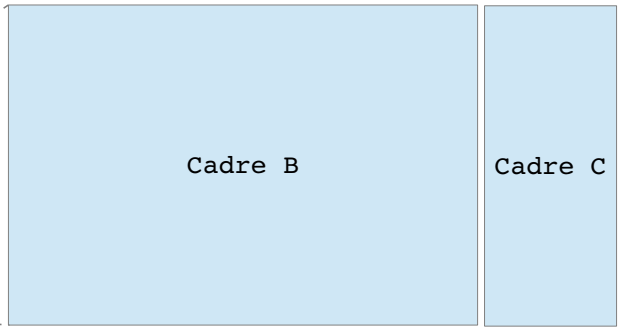
La solution consiste à créer deux framesets, dont un sera l'enfant du second.

En effet en observant attentivement la structure, nous pouvons la décomposer comme suit :

D'abord Frameset parent



Puis Frameset Enfant



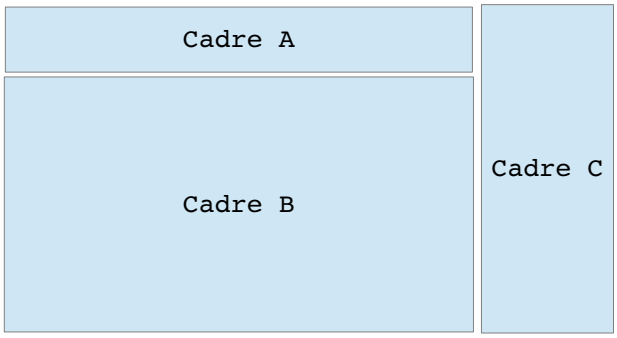
Ce qui donne avec HTML:

```
<frameset rows="30%,70%">
       … définition du cadre A
        <frameset cols="80%,20%">
            … cadre B
            … cadre C
        </frameset>
</frameset>
```

Frameset Enfant

Frameset parent

Exercice
Créer la structure ci-dessous en HTML, vous indiquerez les lignes de définition des cadres à la façon de l'exemple précédent. La largeur du cadre C devra être fixer à 300 pixels, les hauteurs de A et B seront relatives – respectivement 30 et 70 %.

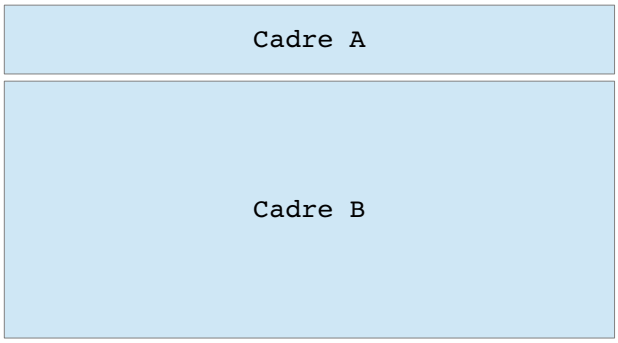


code HTML

```
<frameset cols="*,300">
      <frameset rows="30%,70%">
            … cadre A
            … cadre B
      </frameset>
      … cadre C
</frameset>
```

2.5-Définition des cadres et de leur contenu respectif

Les cadres vont contenir des pages HTML indépendantes les unes des autres, bien souvent on retrouve un bandeau horizontal qui caractérise le site, une barre de menu verticale et un cadre dit de navigation qui affiche les pages HTML désirées. Prenons une structure simple :



Le cadre A contient une page HTML nommée `bandeau.html`. Cette page contient une image caractérisant le site (titre en particulier) et quelques liens qui pointent vers différentes pages internes.
Ces pages sont alors affichées dans le cadre B. Par défaut, c'est à dire au premier chargement de la page support (`index.html`), cadre B affiche `accueil.html`.

Résumé de la Structure HTML

```
Racine du site → index.html          page support
                 + → bandeau.html    contenue dans le cadre A
                 + → accueil.html    contenue dans le cadre B
```

L'insertion des pages HTML est réalisée avec la balise <frame>. Elle spécifie la page à afficher par défaut ainsi que le nom du cadre déclaré.

Structure ci-dessus :

```
<frameset rows="200,*">
      <frame src="bandeau.html"     name="cadreA">
      <frame src="accueil.html"     name="cabreB">
</frameset>
```

L'attribut src indique la page contenue dans le cadre par défaut et name définit un nom pour le cadre. Ce nom est associé ensuite avec l'attribut **`target`**[1] d'un lien.

1 Voir chapitre 2 - §2 page 4

Exercice

1-Reprendre l'exercice de la page précédente en faisant cette fois la déclaration des cadres sachant que :

- cadre A contiendra la page « `titre.html` »
- cadre B contiendra « `accueil.html` »
- cadre C contiendra « `menu.html` »

L'arborescence du site est la suivante :

racine du site → `index.html`

```
+ accueil.html
+ annexes/titre.html
+ javaScript/menu.html
```

```
<frameset cols="*,300">
      <frameset rows="30%,70%">
            <frame src="annexes/titre.html"     name="cadreA">
            <frame src="accueil.html"           name="cadreB">
      </frameset>
      <frame src="javaScript/menu.html"   name="cadreC">
</frameset>
```

2-Imaginons que le menu contient les liens suivants:

- « Retour page d'Accueil » qui pointe vers `accueil.html`
- « Carte du site » qui pointent vers `annexes/carte.html`

Écrire le code HTML de ces deux liens, **intégrés dans la page menu.html**, qui permettent de modifier le contenu du cadre B.

```
<a href="accueil.html"        target="cadreB">Retour page d'Accueil</a>
<a href="annexes/carte.html"  target="cadreB">Carte du site</a>
```

2.6-Paramètres associés à <frame>

- `src`, désigne le chemin et/ou la page à afficher dans le cadre
- `name`, donne un nom au cadre
- `marginwidth`, en pixels précise la taille de la marge en largeur d'un cadre
- `marginheight`, en pixels précise la taille de la marge en hauteur d'un cadre
- `scrolling`, trois valeurs possibles YES – NO – AUTO, spécifie si un cadre dispose ou non d'une barre de défilement
- `noresize`, signale que le cadre est figé et que l'internaute ne peut le modifier à la souris

3.Les balises conteneurs (dit de blocs)- HTML5

Ce sont ces balises qu'il faut désormais utiliser pour la recommandation HTML5 (ce que nous ferons après le TP relatif à ce cours).

3.1- <iframe>

`iframe` est appelée à remplacer `frameset` et `frame` (balises non implémentées en HTML5). Cette balise remplit les mêmes fonctions tout en donnant beaucoup plus de souplesse dans la mise en page (qui se fera principalement avec CSS).

Les attributs de <iframe>

Attributs	Valeurs	Rôle
src	URL	Adresse et nom du document à afficher dans l'iframe
width	Pixel ou %	Largeur
height	Pixel ou %	hauteur
name	Nom	Nom unique d'une iframe dans la page support
sandbox	allow-forms allow-same-origin allow-scripts allow-top-navigation	Attribut de définition des autorisations et exceptions de navigation afin de sécuriser l'accès aux pages html (autoriser ou non des formulaires dans l'iframe par exemple)
seamless	seamless ou rien	Le rendu graphique de l'iframe fait que celle-ci fait partie intégrante de la page support
srcdoc	HTML	Contenu HTML par défaut, si le navigateur ne supporte pas cet attribut il s'appuiera sur src

Exercice
Créer une iframe dans la page support, d'une longueur de 1000px, hauteur de 500px, contenant par défaut le fichier iframe1.html, portant le nom cadre et positionner par défaut à 200px du bord gauche de l'écran.

```
<iframe width="1000" height="500" src="iframe1.html" name="cadre" style="margin-left: 200px;"></iframe>
```

3.2-nav, footer, header, aside et article

Il existe des balises « container » spécialisées. Elles sont issues d'une balise généraliste ** **[2]. La mise en page avec ces balises doit se faire avec CSS afin de définir la position de chacune par rapport aux autres (en relatif) ou par rapport à la page (absolu).
HTML5.0 introduit la notion de balises spécialisées dont le seul but est de rendre plus visible le code de la page HTML.

Remarque
Il y a d'autres balises avec un rôle similaire en particulier avec le formatage des titres et texte comme , <pre> etc. Ces balises seront utilisées en TP après avoir vu CSS et JavaScript.

2 Balise très importante qui sera vue avec JavaScript et le DOM

.CSS{ }

Chapitre 4 – Styles
Chapitre 5 – Techniques Avancées

BTS Systèmes Numériques
Informatique & Réseaux

FÉLIX
LE DANTEC
du secondaire au supérieur

Chapitre 4
CSS & Notions de style

1.Introduction

Quand on développe un site Web, on est rapidement confronté au problème du formalisme de la mise en page et du temps consacré à cela. En effet, pour chaque objet contenu dans les pages, il faut définir la couleur, la taille et la police de caractère si c'est du texte, les bordures et les couleurs de fond si c'est un tableau etc. .
La tache devient vite monstrueusement longue. CSS a été mis au point pour palier ce problème.

Le langage **CSS** (***C**ascading **S**tyle **S**heet*) est un complément d'HTML. Il définit, d'une manière globale, les propriétés de format des balises HTML employées dans une page ou sur la totalité d'un site Web.

2.Présentation du langage CSS

Dans la pratique, il est possible d'intégrer le langage CSS dans une page HTML mais pour un site complet, il est plus utile d'écrire un fichier externe qui sera appelé pour chaque page HTML visitée.
Il n'y a pas de règle précise mais par principe nous lui donnerons comme extension « **`.css`** » .

Avantages :

- La structure et la présentation sont gérées séparément
- La conception se fait sans se soucier de la présentation
- Le code HTML est réduit en taille et en complexité.
- CSS permet de contrôler le style de tous les éléments d'une page web

Exemple comparaison entre déclaration <HTML> et {CSS} de la couleur de fond de la page HTML :

```
<!--en HTML -->
<body bgcolor="#c0c0c0">
```

Serait équivalent :

```
/* en CSS */
body  {
            background-color :#c0c0c0 ;
      }
```

A première vue cela semble plus long à écrire et pourtant ce n'est pas le cas. En effet, en développement web, on privilégie la modularisation du code. Cela revient à dire que chaque partie du site est dissociée en plusieurs fichiers dont les rôles sont distincts.
Dans l'exemple ci-dessus, on peut imaginer des fichiers HTML et un fichier CSS. On obtient ainsi un style commun à tout le site sans avoir de doublon. Ici, la couleur du fond de chaque page serait définie de la même façon. Il en serait de même pour tout ce qui est présent dans le fichier de style.

3.Intégration du code CSS

Le code CSS peut être placé à 3 endroits différents :

- Dans la balise HTML à l'intérieur de la page
- Dans l'entête de la page HTML ou dans la page
- Dans un fichier Externe

En reprenant l'exemple précédent :

→ Dans la balise <body>

```
<body style="background-color:#c0c0c0 ;">
```

→ Dans l'en-tête de la page HTML

```
<head>
      …
      <style>
            body {
                  background-color:#c0c0c0 ;
            }
      </style>

</head>
```

→ Lien avec un fichier externe qui contient l'instruction CSS

```
<head>
      …
      <link rel="stylesheet" type="text/css" href="fichierDeStyle.css">


</head>
```

On remarque que l'instruction CSS est écrite de la même façon mais la structuration peut changer (entre les 2 premiers exemples). Dans un fichier css, aucune balise HTML n'est tolérée, tout doit être écrit en CSS.

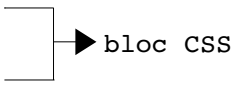
Il est fortement conseillé de placer le code CSS ou le lien vers un fichier externe dans l'en-tête. Le navigateur interprète la page HTML dès le début du téléchargement et calcule la mise en page à partir de ce qu'il connaît à cet instant précis.

4.Syntaxe, Héritage et combinaisons de sélecteurs

4.1-Syntaxe

La syntaxe du langage CSS est simple, elle repose sur le concept de définition de balise et de propriété associée. Toutefois les propriétés CSS ne portent pas nécessairement le même nom qu'en HTML, même si elles ont une action identique.

```
sélecteur {
            propriété: valeur;
          }
```



- sélecteur, ce peut être une balise mais pas seulement, en général il désigne un « objet » de la page HTML sur lequel on applique le style décrit dans le bloc
- propriété, c'est l'élément pour lequel on associe une valeur. Elle doit être compatible avec le type du sélecteur

Règles d'écriture :

- Un sélecteur peut se voir appliquer un bloc CSS comprenant plusieurs propriétés
- Plusieurs sélecteurs peuvent avoir un bloc CSS commun
- L'affection de valeur à une propriété est réalisée par « : »
- Chaque ligne du bloc CSS doit se terminer par « ; »
- Le bloc CSS est délimité par « { } »

Exemple :

```
body {
      background-color:#c0c0c0 ;
      }
```

Le sélecteur `body` caractérise l'objet <body> de la page HTML.
Le bloc CSS comprend une seule déclaration pour la propriété background-color à laquelle on applique la valeur #c0c0c0.

Remarque :
Le navigateur n'interprète pas les erreurs de syntaxe et surtout ne les signale pas.

Exercice :

On souhaite que le texte dans un paragraphe – balise HTML <p> - soit bleu, taille 12 pixels, justifié et police « ubuntu ».
1-Codez en CSS le style de cette balise.
Propriétés nécessaires « `font-size` », « `font-family` », « `text-align` » dont les valeurs attendues sont « *`right`*,*`left`*,*`center`* ou *`justify`* » et « `color` ».

```
p {
      font-size:12px ;
      font-family:ubuntu ;
      text-align:justify ;
      color:#0000FF ;
}
```

2-Que remarque t'on au niveau des certaines propriétés ?

`font-size` et `font-family` sont issues de la même propriétés font. En effet le signe « - » permet de spécifier une sous-propriété. Il en serait de même pour `text-align` qui serait issue de `text`.

3-Peut-on écrire autrement le style ?

```
p {
      font : 12px ubuntu ;
      text : justify ;
      color:#00F ;
}
```

Remarques :
Certaines propriétés peuvent avoir plusieurs valeurs séparées par un espace. Attention, cette règle ne s'applique pas systématiquement (voir site W3C, W3School & co). La définition d'une couleur peut être réduite à la condition que les deux caractères d'un canal soient identiques.

4.2-Héritage

La notion d'héritage est très usitée dans de nombreux domaines de l'informatique. On en parle beaucoup en programmation orientée objet (C++, java, php etc.) mais aussi en HTML/CSS.

Ce modèle s'appuie sur la relation « parent » - « enfant » sachant que ce que l'on entend par parent/enfant correspond à l'imbrication des sélecteurs entre eux. Par exemple, en général la balise <body> contient d'innombrables autres balises, on considère que <body> est le parent de ce qu'elle contient.

Exercice
Soit le Code source d'une page HTML :

```
1.      <!doctype html>
2.      <html>
3.              <head>
4.                      <title>
5.                              Héritage en CSS
6.                      </title>
7.                      <meta charset="utf-8">
8.              </head>
9.              <body>
10.                     <div>
11.                             <p>....blablabla... <a href="http://www.blabla.bla">BLABLA</a>...</p>
12.                             <img src="blablabl.png">
13.                     </div>
14.                     <table>
15.                             <caption>Tableau Héritage</caption>
16.                             <thead><tr><th>Colonne 1</th><th>Colonne 2</th><tr>
17.                             <tfoot><tr><td colspan="2">blablabla</td></tr></tfoot>
18.                             <tbody><tr><td>blablala1</td><td>blablabla2</td></tr></tbody>
19.                     </table>
20.             </body>
21.     </html>
```

1-Donnez le ou les sélecteur(s) enfant(s) de `<tr>`

lignes 15/16 et 18 on a `th` et `td`

2-Donnez le ou les sélecteurs parent(s) de `<tr>`

Mêmes lignes que précédemment `thead, tfoot, tbody`

3-On parle aussi de sélecteurs ancêtres, quels sont-ils pour `<tr>` ?

Ce sont les sélecteurs situés dans la lignée de `<tr>` et au dessus de celui-ci.
En plus des précédents, on retrouve table, body et html

4-Quelle est la différence entre un ancêtre et un parent ?

La parenté est directe avec un parent.

5-Quels sont les sélecteurs qui n'ont pas d'enfant ?

Title, meta, p, img, caption, td et th

6-Quel(s) est/sont le(s) sélecteur(s) qui n'a/ont pas de parent ?

`<html>` est la seule.

4.3-Principe de l'héritage en CSS

L'héritage des CSS est fondé sur le modèle Parent-Enfant(s) ce qui signifie que chaque élément Enfant reçoit en héritage tous les styles de son élément Parent.

Cet héritage est très pratique et permet d'éviter beaucoup de répétitions inutiles : en effet, en attribuant une propriété à un parent (par exemple font-size: 15px), elle sera transmise à tous ses enfants, mais aussi aux enfants de ces enfants, etc.

Précision : l'élément enfant héritera de toutes les propriétés de l'élément parent uniquement si ces propriétés s'héritent car l'héritage ne fonctionne pas non plus sur toutes les propriétés css (margin, padding et autres propriétés de bloc) .

Exercice
A partir de la page HTML de la page suivante, on vous demande de définir le style par défaut du texte de la page (quelques soient les balises qui l'encadre) de la façon suivante :
- taille des caractères 10 pixels
- couleur #9c9c9c
- police de caractère Arial

```
body {
      font-size:10px ;
      font-family : Arial ;
      color:#9c9c9c ;
      }
```

4.4-Définition de la descendance

Nous venons de voir la notion d'héritage, c'est à dire l'application d'un style d'un sélecteur à toute sa descendance. Il arrive parfois où nous souhaitons que certains sélecteurs soient affectés par un style particulier.

Imaginons un instant que nous avons une page avec du texte décrit en paragraphe avec <p> et certains d'entre eux sont créés avec une autre balise : qui peut éventuellement afficher une remarque par exemple. Les deux sélecteurs sont systématiquement intégrés dans un :

```
<body>
      ….
      <div>
            <p> ...parapaghe1...</p>
            <p> ...paragraphe2...</p>
            <span> ...remarque1...</span>
            <p> ...paragraphe3 … </p>
      </div>
      …
</body>
```

Définition du style du texte par défaut :

```
      body {
            style du texte par défaut
      }
```

Tous le texte est affecté par ce style car on utilise la balise <body>.

On veut que les remarques dispose d'un style qui leur est propre.

1ère solution :

```
body {
            style du texte par défaut
      }
span {
            style des remarques
      }
```

Ici tous les spans seront affectés par le style. Mais on veut que seuls les spans inclus dans la page et les divs soient affectés par ce style.

2ème solution :

```
body        {
                  style du texte par défaut
            }
div span    {
                  style des remarques
            }
```

Remarques:
-La balise ancêtre ou parente est toujours signalée en premier.
-Un span qui n'est pas dans un div ne sera pas affecté par le style

Exercice :
A partir de la page HTML donnée page 4 et toujours dans la continuité de l'exercice précédent, donnez le style CSS pour le texte dans le corps du tableau :

-taille 8px
-couleur bleue
-police ubuntu

```
tbody td {
            font-size:8px ;
            font-family:ubuntu ;
            color:#0000ff ;
}
```

Remarque :
Si des règles, pour des sélecteurs différents mais affectés par la hiérarchie, ont des propriétés communes, ce seront les dernières définies qui seront prises en compte. Ici les propriétés de td seront prédominantes sur celle de body.

4.5-Priorité des sélecteurs

Lorsque deux règles différentes ciblent le même sélecteur, c'est la dernière déclarée qui est prise en compte.

Exemple :

```
sélecteur lambda {
                    règle 1 ;
                  }
sélecteur lambda {
                    règle 2 ;
                  }
```

Ici ce sera la règle 2 qui sera prise en compte.

4.6-Combinaisons de sélecteurs

Il arrive parfois que différentes balises ont une même définition de style. Il est possible de définir les sélecteurs correspondants de la même façon en les regroupant :

Exemple :

```
div,p,span,a,h1 {
                  color       : #ca45a8 ;
                  font-size   : 10px ;
                  font-weight : 100 ;
            }
```

4.7-Sélecteurs d'enfant et d'adjacence

Le sélecteur d'enfant s'applique, comme son nom l'indique, à l'enfant ou aux enfants d'un élément désigné :

```
E1 > E2 {
                        propriétés  : valeurs ;
      }
```

Le style s'applique à Élément 2 si celui-ci est l'**enfant direct** de Élément 1.

Remarque :
A la différence avec la descendance (§4.4 page 5) le sélecteur d'enfant n'a aucune action si un Élément existe hiérarchiquement entre E1 et E2.

L'adjacence s'applique aux sélecteurs d'éléments frères. Le style peut alors s'appliquer si et seulement si l'élément est précédé par celui désigné par l'adjacence :

```
E1 + E2 {
            propriétés : valeurs ;
      }
```

Ici E1 précède E2 .

Exemple :

HTML

```
<h1>Chapitre 1 – HTML</h1>
<p> Présentation et mise en forme avec HTML</p>
<p> Ce cours présente le formalisme, les balises ...</p>
```

CSS

```
h1 + p {
      font-style:       italic ;
      font-size :       8px ;
      color :           grey ;
      }
```

Seul le premier paragraphe (en gras) est défini par ce sélecteur.

Exercices

Ex1 :

1.1-A partir de la définition donnée ci-dessous, écrire le sélecteur permettant de définir le style du texte contenu dans un paragraphe tel que :

- la police soit Arial
- en italique
- de couleur grise
- de taille 10px

CSS à compléter :

```
body {
      text-align :      justify ;
      font-style :      normal ;
      font-size :       10px ;
      color:            black ;
      font-family :     Arial, Ubuntu ;
}
p     {
      color :           grey ;
      font-style :      italic ;
}
```

1.2-Quelles règles s'appliquent dans ce cas ?

L'héritage, car certaines propriétés CSS sont communes comme la taille et la police de caractère. Body étant la balise englobante, `p` hérite de ses propriétés y compris de la justification.
La priorité pour les propriétés qui diffèrent mais qui sont définies pour les deux sélecteurs comme `font-style` et `color`.

Ex2 :

Le code HTML ci-dessous représente (schématiquement) la présentation des articles sur la page d'accueil d'un site d'informations. Chaque article est résumé et pour en lire le contenu, il faut cliquer sur son titre. Chaque div contient un article tronqué, le titre est défini avec `p`, le résumé dans le paragraphe situé juste sous le titre et enfin un second paragraphe pour la date de publication et le nom de l'auteur.

```
<div>
      <p><a href="url de l'article complet">...titre...</a></p>
      <p>...résumé du texte...</p>
      <p>...date et nom de l'auteur de l'article...</p>
</div>
```

Écrire le code CSS en respectant les consignes ci-dessous :

- Le texte par défaut doit être en blanc, police Ubuntu, taille `10px` et justifié.
- Le fond de chaque bloc d'article doit avoir le code couleur `#121212` (gris foncé)
- Un bloc d'article doit avoir la taille `350x100` (en pixels), la bordure est en pointillés de couleur jaune
- Tous les liens des blocs d'articles (où qu'ils soient) seront en jaune et gras.
- Le titre sera en rouge et centré
- Le résumé sera de couleur grise (c0c0c0) et en italic
- La date de publication et le nom de l'auteur seront sur fond blanc, le texte en noir et d'une taille de 8 pixels.

```
div {
      color :                 white ;
      background-color :      #121212 ;
      font-family :           Ubuntu ;
      font-size :             10px ;
      font-style :            normal ;
      height :                100px ;
      width :                 350px ;
      border-style :          dotted ;
      border-color :          yellow ;
      text-align :            justify ;
}
div a {
      color :                 yellow ;
      font-weight :           bold ;
}

div > p {
      color :                 red ;
      text-align :            center ;
}

div > p + p {
      color :                 #c0c0c0 ;
      font-style :            italic ;
}

div> p + p + p {
      background-color :      white ;
      color :                 black ;
      font-size :             8px ;
}
```

5.Les classes, identificateurs et pseudo-classes/pseudo-éléments

5.1-Les classes et Identificateurs

Les classes permettent de créer des sélecteurs personnalisés et applicables à différentes balises à condition que les propriétés soient compatibles. Elles sont complètement indépendantes des balises sur la page HTML et peuvent être nommées comme on le souhaite (en respectant le formatage et la syntaxe css/html).

Formalisme

en CSS

```
.nom_de_la_classe {
                    propriétés : valeurs ;
                }
```

relation sur la page HTML

```
<balise class="nom_de_la_classe">
```

Exemple

Sur la page HTML :

```
<div class="classe1">
      …
</div>
<canvas class="classe2"></canvas>
<fieldset class="classe1"></fieldset>
<div>
      …
</div>
```

Dans le fichier CSS :

```
.classe1 {
            propriétés appliquées aux éléments désignés par l'attribut
« class », dans l'exemple ci- dessus cela concerne le premier div et fieldset.
}
.classe2 {
            …
}
div   {
            propriétés appliquées à tous les « div »
}
```

Exercice

En reprenant l'exercice 2 de la page 8, nous souhaitons mettre en avant le dernier article publier. Celui-ci sera le premier affiché sur la page d'accueil (ordre chronologique inverse). Il est constitué de la même façon que les autres d'un point de vue HTML avec des différences sur le style. Le bloc sera encadré par un trait avec effet gravure (« groove »), le trait de bordure aura une épaisseur de 4 pixels et la couleur du fond sera #787878). Écrire le code HTML du bloc ainsi que son style, on nommera la classe « dernierArticle ».

HTML

```
<div classe="dernierArticle"> ...</div>
```

CSS

```
.dernierArticle {
                  background-color :      #787878 ;
                  border-style :          groove ;
                  border-width :          4px ;
}
```

Les identificateurs reposent sur le même principe que les classes à la différences qu'il ne peut y avoir qu'un seul « ID » sur une page HTML, une classe pouvant être applicable autant de fois que l'on souhaite.

Formalisme

HTML
```
<balise id="nom_de_l'id">...</balise>
```

CSS
```
#nom_de_l'id {
            propriétés :      valeurs ;
}
```

Exemple

HTML
```
<img id="logo" src="image">
```

CSS
```
#logo {
            height :    200 ;
            witdh :     150 ;
}
```
Sur la page il n'y a qu'une seule image identifiée par « logo » qui sera affectée par ce style.

Exercice
A quoi correspondent les sélecteurs suivants :

Sélecteurs	Correspondances
`.grasEtRouge { ...}`	Classe grasEtRouge
`h1.titreDePage { … }`	Balises h1 de la classe titreDePage
`div .titre { … }`	Toutes les balises de la classe titre faisant partie d'un div
`span , #citation,p > h3.sousTitre { ...}`	Tous les spans, les h3 de la classe sousTitre étant enfant direct de p et l'objet dont l'id est citation
`fieldset p.saisie > label + input { ...}`	Tous les inputs associés à un label (frère) étant enfant direct de paragraphes de la classe saisie inclus dans un fieldset.

Synthèse :
Que ce soit pour les classes comme pour les identificateurs :

- ***On peut les nommer comme on le souhaite (en respectant les règles de syntaxe)***
- ***l'héritage s'applique***

5.2-Pseudo-classes et pseudo-éléments

Ils permettent d'affiner le style appliqué à un certain nombre de balises en définissant une réaction à un événement ou bien à la position relative de la balise au sein des autres balises. Contrairement aux classes, le nom est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes/éléments.

Il existe plusieurs types de pseudos-classes :

- Les pseudo-classes dynamiques,
- Les pseudo-classes de lien,
- Les pseudo-classes de langue,
- Les pseudo-classes first-child,
- Les pseudo-classes de page,
- Les pseudo-éléments `:first-letter`/`:first-line` `:before`/`:after`,

5.3-Les pseudos-classes dynamiques (et de liens)

Les pseudo-classes dynamiques permettent de modifier le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier).

Il en existe quatre :

- La pseudo-classe `:hover` permet d'affecter un style à la balise sélectionnée lors d'un survol par le curseur de la souris : `a:hover {font-decoration: underline;}`
- La pseudo-classe `:focus` permet de définir un style à la balise sélectionnée lorsque le focus (sélection) lui est donné (par exemple lors de la sélection dans un élément de formulaire) : `textarea:focus {color: #FF0000;}`
- La pseudo-classe `:active` permet de définir un style à la balise sélectionnée lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche) : `a:active {color: #FF0000;}`
- La pseudo-classe `:visited` permet de définir un style pour un objet déjà visité comme un lien par exemple : `a:visited {font-decoration : overline;}`

Remarque :
`:focus` *concerne principalement les balises acceptant l'attribut HTML «* `tabindex` *» (qui définit l'ordre de sélection avec la touche TABulation). Sont concernées :*
`<a>` , `<area>` , `<button>` , `<input>` , `<object>` , `<select>` et `<textarea>`

Exercices

1-On souhaite que les liens soient définis en CSS de la façon suivante :
-Couleur par défaut, qu'ils soient déjà visités ou non, actifs → #FFFF00
-En petite majuscule → attribut `font-variant` valeur `small-caps`
-Les liens ne sont pas soulignés par défaut
-Quand on passe la souris sur un lien, celui-ci est souligné et surligné et les caractères apparaissent en gras
-Au clic le lien devient blanc – reste en gras et le fond de celui-ci est jaune foncé (#444400)
-La police par défaut est « courrier »

```
a {
      color:                  #FFFF00 ;
      font-variant :          small-caps ;
      font-family :           courier ;
      text-decoration :       none ;
}
a:hover {
      font-weight:            bold ;
      text-decoration :       overline underline ;
}
a:active{
      color:                  #FFFFFF ;
      background-color :      #444400 ;
}
```

2-On souhaite que les liens pointant vers « wikipédia » soient affectés par le style proposé précédemment et que les autres disposent d'un autre style. Proposez une solution en réécrivant uniquement les sélecteurs.

```
a.wiki {
            …
}
a.wiki:hover{
            ...
}
a.wiki:active{
            ...
}
```

3-Quand on passe la souris sur une vignette (image de taille réduite) , on souhaite zoomer l'image. Ecrire le code CSS pour toutes les images en supposant que les vignettes aient une taille de 150x100 (largeur x hauteur) et le zoom 300x200.

```
img {
      height:           100px ;
      width :           150px ;
}
img:hover{
      height :          200px ;
      width :           300px ;
}
```

5.4-Pseudo-classe descendante `first-child`

Nous avons déjà vu la notion de descendance (§4.7 page 7) qui permet d'affecter un style à tous les descendants directs d'un sélecteur parent.
La pseudo-classe descendante `:first-child` diffère sur un point, elle concerne le premier enfant d'un sélecteur même si celui-ci dispose de sélecteurs frères.

En reprenant l'exercice 2 de la page 8, nous avions les articles d'une page HTML décrit de la façon suivante:

HTML

```
<div>
      <p>...Le titre...</p>
      <p>...le texte...</p>
      <p>...date et auteur</p>
</div>
```

Schématiquement

```
<div>   - - parent
    ├─ <p> ┐
    ├─ <p> ├─ Enfants directs
    └─ <p> ┘
```

Les paragraphes sont frères et la descendance s'applique.

Aux 3 :

```
div > p {
        …
}
```

Au second paragraphe :

```
div > p + p {
        …
}
```

Au 3ème :

```
div > p + p + p {
                ...
}
```

Comment appliquer un style au premier enfant uniquement ?
En appliquant la pseudo-classe descendante:

```
div > p:first-child {
                …
      }
```

Exercice

On souhaite formater la première cellule de la première ligne d'un tableau de la façon suivante :
- police de caractère « Arial », taille 18 pixels, couleur blanc et en gras
- le fond de cette cellule sera en noir

Les autres cellules de la première ligne seront définies comme suit :
- police de caractère « Arial », taille 16 pixels, couleur noir
- le fond des cellules en gris clair

Le reste du tableau en fond blanc, police de caractère « courier », taille 14px et couleur gris foncé
Le tableau compte 3 lignes et 4 colonnes avec fusion des cellules de la première colonne. La bordure des cellules du tableau sera en pointillés et de couleur rouge.

Matériel	Carte E/S	Carte Embarquée	Projet
	Arduino UNO	Raspberry PI B 512	OpenDomotique
	Arduino Mega 2560	Beagle Bone Black	StatMyCar

Capture du tableau

1- Codez le tableau en HTML

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Exercice tableau CSS</title>
		<style>
			Voir question 2
		</style>
	</head>
	<body>
		<table>
			<tr>
				<td rowspan="3">Matériel</td>
				<td>Carte E/S</td>
				<td>Carte Embarquée</td>
				<td>Projet</td>
			</tr>
			<tr>

				<td>Arduino UNO</td>
				<td>Raspberry PI B 512</td>
				<td>OpenDomotique</td>
			</tr>
			<tr>

				<td>Arduino Mega 2560</td>
				<td>Beagle Bone Black</td>
				<td>StatMyCar</td>
			</tr>
		</table>
	</body>
</html>
```

2-Créez les styles à appliquer au tableau, ***on n'utilisera pas les classes ni les identificateurs***.

```
		td {
			border-style:		dashed;
			border-color:		#ff0000;
			border-width:		1px;
			font-family:		courier;
			font-size:		14px;
			color:			#333333;
			background-color:	white;
		}
		table tr:first-child > td{
			background-color:	#cccccc;
			font-family:		arial;
			font-size:		16px;
			color:			black;
		}
		table tr:first-child > td:first-child{
			background-color:	black;
			font-family:		arial;
			font-size:		18px;
			color:			white;
			font-weight:		bold;
		}
```

3-Complétez le style précédent de façon à ce que le fond des cellules et la couleur du texte inclus changent que la souris les survole. Couleur du texte en rouge et le fond en jaune. Le curseur de la souris doit être une croix.

```
td:hover {
                color :           red;
                background-color: yellow;
                cursor:           crosshair;
            }
```

5.5-Les pseudo-éléments

5.5.a `:before` et `:after`

Ces pseudos éléments permettent de rajouter du texte, un image et des caractères avant et après un sélecteur donné. Ils sont associés à une propriété spécifique – « `content` » .

```
selecteur:before {
                content :   valeur ;
            }
selecteur:after {
                content :   valeur ;
            }
```

Exemple

On souhaite que chaque citation soit encadrée par « et » . On utilise <blockquote> pour le texte cité :

```
blockquote:before {
                content : "«" ;
}
blockquote:after {
                content :"»" ;
}
```

Exercice

A partir du code HTML ci-dessous (et de la capture) créez le style de ce tableau de façon à ce que le caractère « € » soit rajouté après le prix.

```
<table>
        <tr>
                <td>Menus</td><td>Prix</td>
        </tr>
        <tr class="choix">
                <td>Déjeuner</td><td>10</td>
        </tr>
        <tr class="choix">
                <td>Entrée+Déjeuner</td><td>13</td>
        </tr>
        <tr class="choix">
                <td>Déjeuner+dessert</td><td>13</td>
        </tr>
        <tr class="choix">
                <td>Entrée+Déjeuner+dessert</td><td>16</td>
        </tr>
</table>
```

Menus	Prix
Déjeuner	10€
Entrée+Déjeuner	13€
Déjeuner+dessert	13€
Entrée+Déjeuner+dessert	16€

Remarque :

Le style global du tableau correspond à celui de l'exercice précédent.

Code CSS :

```
tr.choix td+td:after{
                                content: "€";
                    }
```

5.5.b-Les pseudo-éléments de texte

Ils permettent d'appliquer un style à une partie du texte délimitée par les balises auxquelles ils s'appliquent. Ainsi les pseudo-classes de texte s'utilisent généralement conjointement avec la balise de paragraphe (<p>), de bloc etc. .

`:first-line`, agit sur la première ligne d'un paragraphe
`:first-letter` agit sur le premier caractère d'un paragraphe ou bloc

Exercice
Codez le style afin que le premier caractère de chaque paragraphe débute par une lettrine. La taille de celle-ci sera toujours 3 fois plus grande que celle du texte et elle sera incrustée dans le paragraphe. La police retenue sera « Arial », elle sera en gras et italique. Enfin pour marquer sa présence, on ajoutera un effet d'ombrage (CSS3).

La syntaxe du langage CSS est simple, elle repose sur le concept de définition de balise et de propriété associée. Toutefois les propriétés CSS ne portent pas nécessairement le même nom qu'en HTML, même si elles ont une action identique.

Capture d'un paragraphe avec lettrine

Syntaxe ombrage

```
.titre {
        text-shadow : 2px 3px 4px #404040 ;
}
```

La première valeur indique le décalage vers la droite de l'ombre (ici 2 pixels)
La deuxième valeur indique la décalage vers le bas (3 pixels)
La troisième valeur indique la « force » du dégradé d'ombre (ici 4 pixels entre la couleur du fond et la couleur de l'ombre)
La dernière valeur correspond à la couleur de l'ombre.

```
p:first-letter {
                                font-size:  300%;
                                float:          left;
                                font-weight:    bold;
                                font-style: italic;
                                font-family:    Arial;
                                text-shadow: 2px 2px 4px #999; /*CSS3*/
                    }
```

5.6-Les pseudo-classes de page @

Cette pseudo classe fait partie de ce que l'on appelle les règle -at (règle @). Elles sont nombreuses mais il en existe 3 qui se distinguent des autres à savoir :

- `-@import` qui permet d'inclure un fichier css à partir d'un autre
- `-@media` qui définit le périphérique pour lequel on affecte des rêgles CSS
- `-@page` qui permet de définir le style pour l'impression

5.6-a @import

Pour inclure un fichier CSS à partir d'un autre, on utilise `@import` en **la plaçant en tête de fichier**.

Exemple

```
@import url(style/mapage.css)
```

style → répertoire

5.6-b @media

On définit que des styles uniquement pour les écrans et les imprimantes par exemple :

```
@media screen{
            … style …
            }

@media print{
            ...style...
            }
```

Les différents types de média reconnus :

- `-screen`
- `-print`
- `-aural` et `speech` pour la synthèse vocale
- `-handheld` pour les mobiles
- `-projection` pour vidéo projecteur
- `-all` pour tous (valeurs par défaut)

5.6-c @page

Le sélecteur @page permet de modifier les définitions de mise en page d'une page HTML (taille, marge, etc.) à l'impression, telles que les marges (margin-left, margin-top, margin- right, margin-bottom), la taille (size). Il faut alors imaginer la page web comme un ensemble de pages constituant un ouvrage papier.
Les pseudo-classes de page permettent ainsi de sélectionner la page, les pages de gauche, de droite ou bien la première page d'un document.

Exemple avec 4 marges définies :

```
@page {
      margin : haut bas droite gauche ;
      }
```

Les valeurs attendues peuvent être exprimées en cm/mm.

6.Les unités

Grâce aux feuilles de style il est possible de définir des valeurs numériques pour les propriétés de style de plusieurs façons :

- de façon **absolue**, c'est-à-dire dans une unité indépendante du format de sortie (en centimètres par exemple)
- de façon **relative**, c'est-à-dire dans une unité relative à un élément

Les valeurs des feuilles de style sont soit des nombres entiers, soit des nombres réels, c'est-à-dire des chiffres ayant une partie entière et une partie décimale.
D'une manière générale il est à noter l'utilisation du point (« . ») dans les notations décimales en lieu et place de la virgule (8.5 cm et non 8,5 cm).

Les valeurs peuvent par ailleurs dans certains cas être négatives (précédées du signe « - »). Certaines propriétés peuvent néanmoins accepter un intervalle restreint de valeurs.

Unité	Description	Unité	Description
cm	centimètre	**em**	Unité relative à la taille de police de l'élément sélectionné.
mm	millimètre	**ex**	Unité relative à la hauteur de la minuscule de l'élément sélectionné.
in	Pouce (1 pouce = 2,54cm)	**px**	unité dont le rendu dépend de la résolution du périphérique d'affichage
pt	Point (unité de la police)	**%**	pourcentage
pc	Pica , 1pc = 12 pt		
Absolue		Relative	

7.Les couleurs

Les couleurs en CSS sont définissables suivant 3 méthodes :
- Code RVB en Hexadécimal (voir Cours HTML Chapitre 1)
- Code RVB en décimal ou relatif
- Appel d'une couleur par son nom

7.1-Code RVB Décimal ou relatif

La plage de valeurs des couleurs hexadécimales va de 00 à FF.
Par un changement de base la plage décimale va de 0 à 255.
Pour allouer une couleur à un élément, il est nécessaire d'utiliser l'attribut rgb :

```
h1 { color : rgb(255,0,0);} /*255 pour le rouge, 0 pour le vert et 0 pour le bleu*/
```

On procède de la même façon avec le codage relatif :

```
h1 { color : rgb(100%,50%,75%);}
```

7.2-les couleurs nommées

Cette solution est simple mais peu précise car on ne peut utiliser que 16 couleurs.

```
h1 { color : red ;}
```

Chapitre 5
CSS & Techniques avancées

Dans ce chapitre nous aborderons des parties importantes et qui sont souvent mal comprises. Cela va du positionnement à l'optimisation.

1.Dimensionnement en CSS

1.1-Notion de boite

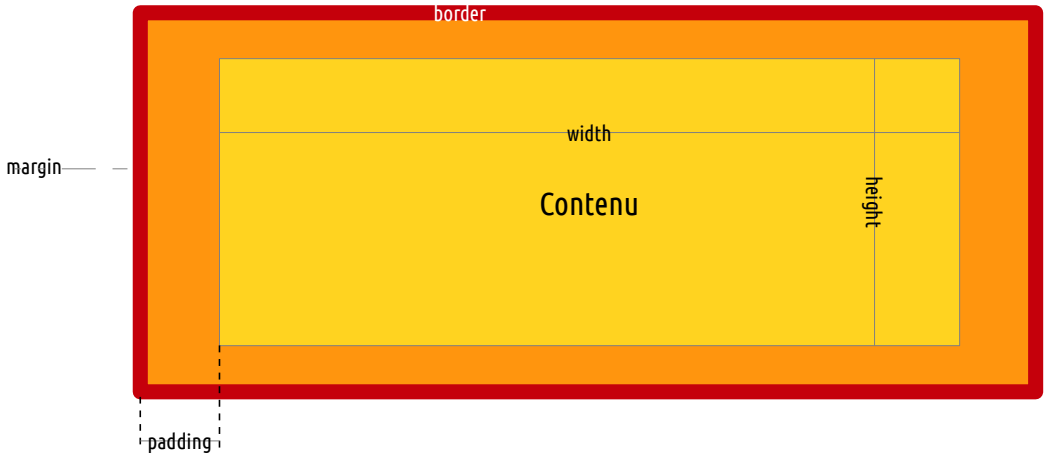
Selon le W3C est considéré comme une boite, tout élément structuré par une balise HTML et définit de la façon suivante :

- avec le contenu caractérisé par une largeur et une hauteur (`width` et `height`)
- avec une bordure, espace qui encadre la boite (`border`)
- avec des marges interne (`padding`) et externe (`margin`)

Marge interne :
Marge entre la bordure et le contenu de la boite

Marge externe :
Marge entre la boite et ses voisines, affecte le positionnement de la boite

En résumé



1.2-Dimensionnement

On imagine, à tort, que la dimension d'une boite est définie suivant les valeurs de width et height.

Prenons l'exemple suivant, un paragraphe définit comme suit :

```
p {
        width :         100px ;
        height :        50px ;
        padding :       10px ;
        border :        1px solid #C0C0C0 ;
        background :    #F2CA16 ;
}
```

Quelles sont les dimensions du paragraphe sur la page HTML?

Largeur = width + 2 x padding + 2 x border = 122 pixels

hauteur = height + 2 x padding + 2 x border = 72 pixels

Que représente la partie utile (en%) ?

Width/Largeurx100 = 81 %

height/hauteurx100= 70 %

Quelles sont les conséquences si on utilise des % pour définir `width` ou `height` ?

Par défaut le contenu occupe 100 % de leur espace, si on ajoute des marges internes, la longueur utile étant inchangée, la boite chevauche alors les voisines ou son contenu sort de celle-ci. En conclusion on ne définit pas la taille des boites en %.

1.3-Minima et maxima

Pour éviter les problèmes vus précédemment, il est d'usage de définir des variations de tailles, une forme de tolérance entre des valeurs minimum et maximum.

Propriétés

- la largeur `min-width` et `max-width`
- la hauteur `min-height` et `max-height`

Exercice
Définir le CSS de body si on veut que celui occupe toujours 80 % de la résolution de l'écran avec une largeur minimum de 800 pixels et maximum de 1200 pixels.

```
Body {
      width :           80 %;
      min-width : 800px ;
      max-width : 1200px ;
}
```

Quelle est la plage de résolution (minimum et maximum) avec ce réglage si on considère les écrans sont en formats paysage 16/9 ? En déduire la hauteur de body.

largeur écran mini= 800/.8 = 1000 maxi=1200/0.8 = 1500

hauteur = 9/16*largeur => hauteur mini = 9/16*1000 = 562 pixels
hauteur maxi= 9/16*1500 = 843 pixels

Si la résolution de l'écran est de 1920x1080, quelle sera la taille de body (hauteur et largeur) ?

Limité au maximum soit 1200 en largeur et 843 en hauteur.

1.4-Fusion des marges

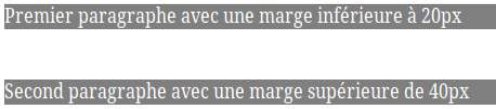
La fusion des marges est un mécanisme trop souvent méconnu et rarement prit en compte dans la mise en page verticale des éléments. Elle influe uniquement sur les marges externes `margin-top` et `margin-bottom`.

Les marges des éléments adjacents (frères ou imbriqués) se combinent pour en former qu'une (en règle générale, la plus grande des deux).

1er Exemple

CSS

```
p{
                        background-color:     grey;
                        margin-bottom: 20px;
                        height:        20px;
                        color:                white;
}
p+p{
                        margin-top:           40px;
}
```



HTML

```
<p>Premier paragraphe avec une marge inférieure à 20px</p>
<p>Second paragraphe avec une marge supérieure de 40px</p>
```

En s'appuyant sur le code et sur la capture, que peut-on constater ?

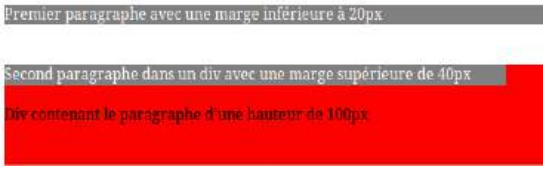
Seule la marge du 2nd paragraphe est prise en compte (40px). La fusion des marges correspond à la prise en compte de la plus grande des deux.

2ème Exemple

Cette fois on place le 2nd paragraphe dans un div dont on fixe la hauteur à 100 pixels.

CSS

```
p{
                background-color:     grey;
                margin-bottom:        20px;
                height:               20px;
                color:                white;

}
div {
                background-color:     red;
                color:                black;
                height:               100px;
}
div p{
                margin-top:           40px;
                width:                560px;
}
```



HTML

```
<p>Premier paragraphe avec une marge inférieure à 20px</p>

<div>
        <p>Second paragraphe dans un div avec une marge
supérieure de 40px</p>
        Div contenant le paragraphe d'une hauteur de 100px
</div>
```

Même question, que peut-on constater ?

Le div n'a pas de margin-top définit donc par défaut elle vaut 0, on pourrait donc imaginer la séparation entre celui-ci et le 1er paragraphe serait de 20px. Ce n'est pas le cas, la marge du 2nd paragraphe étant supérieure à celle du parent (div) est prise en compte et fusionnée avec celle du 1er paragraphe, on se retrouve alors dans le cas de l'exemple précédent.

***Pour résumer :* Seule la marge verticale la plus grande entre éléments adjacents (y compris entre parent/enfant) est prise en compte.**

2.Notions de rendu et de flux courant

2.1-Rendu - `display`

La représentation des éléments est assurée par le norme HTML ainsi que leur imbrication respective mais aussi avec CSS à travers la propriété « `display` » qui confère les caractéristiques suivantes :

- un rendu général sous forme de « boîte »
- une disposition par défaut (verticale, cote à cote etc.)
- marges interne et/ou externe
- dimensionnement

« `display` » accepte au total **17** valeurs :

Display (défaut)	Exemples	Spécificités
`block`	`<div>`, `<p>`, `<ul>`, `<h1>`	Les éléments de rendu CSS `display: block` se placent toujours l'un en dessous de l'autre par défaut (comme un retour chariot). Par exemple : une suite de paragraphes (`<p>`). Par ailleurs, un élément bloc occupe automatiquement, par défaut, toute la largeur disponible dans son conteneur et peut être dimensionné à l'aide des propriétés telles que `width`, `height`, `min-width`, ou `min-height`...
`inline`	`<a>`, `<em>`, `<span>`, `<strong>`	Les éléments de rendu `display: inline` se placent toujours l'un à côté de l'autre afin de rester dans le texte. Par défaut, il n'est pas prévu qu'ils puissent se positionner sur la page (même si cela est possible), ni de leur donner des dimensions (hauteur, largeur, profondeur) : leur taille va être déterminée par le texte ou les éléments qu'ils contiennent. Certaines propriétés de marges peuvent être appliquées aux éléments, comme les marges latérales (`margin-left` et `margin-right`).
`none`	`<head>`	L'élément est complètement retiré du flux, il n'est pas restitué par les agents utilisateurs.

Display (défaut)	Exemples	Spécificités
`list-item`	`<li>`	Les éléments de rendu `display: list-item` ont un rendu de type `block` mais peuvent bénéficier des propriétés CSS liées aux puces (`list-style`...), par exemple `<li>`.
`inline-block`	`<input>`, `<select>`	Les éléments de rendu `display: inline-block` conservent les mêmes caractéristiques que les `inline`, mais peuvent être dimensionnés, par exemple l'élément `<input>`.
`table`	`<table>`	Les éléments de rendu `display: table` sont similaires aux éléments de type `block` mais n'occupent par défaut que la largeur de leur contenu, par exemple `<table>`.
`table-cell`	`<td>`, `<th>`	Les éléments de rendu `display: table-cell` ont un rendu similaire aux éléments `inline-block` dans la mesure où ils s'affichent les uns à côté des autres et peuvent être dimensionnés. Leurs particularités supplémentaires est d'accepter la propriété `vertical-align` pour aligner leur contenu verticalement, d'avoir des hauteurs identiques entre éléments frères et de répartir automatiquement leur largeur au sein de leur parent.
`table-row`	`<tr>`	Les éléments de rendu `display: table-row` ont un rendu similaire aux éléments `block` dans la mesure où ils s'affichent les uns sous les autres. Leur particularité supplémentaire est de répartir automatiquement leur hauteur au sein de leur parent.

Nous verrons dans la suite du cours, l'influence de display dans le mode d'affichage d'un élément HTML.

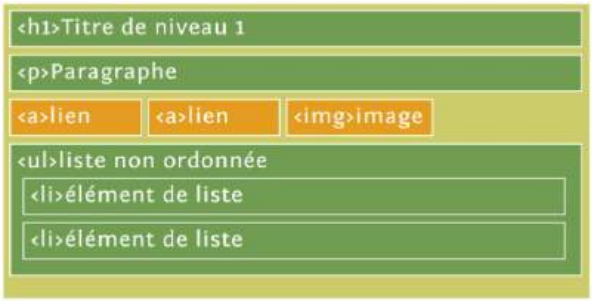
2.2-Le flux

L'ordre dans lequel apparaissent les balises (code HTML) est celui dans lequel les boîtes sont affichées et se positionnent dans le document, on parle alors de flux courant.
On dit alors que la mise en page des éléments s'effectue par défaut selon le flux courant.

Règles de positionnement dans le flux . Chaque élément :

- est situé dans le même plan que les autres
- se place le plus haut possible et le plus à gauche possible au sein de son parent
- est dépendant de l'élément frère précédent (deux éléments de type `block` se positionnent verticalement alors que de type `inline` ils se suivent sur la même ligne)

Exercice



Organisation des éléments

En vous appuyant sur la capture ci-contre, donnez, pour chaque balise le type de rendu par défaut :

h1, p, ul, li → block

a et img → inline

3.Positionnements

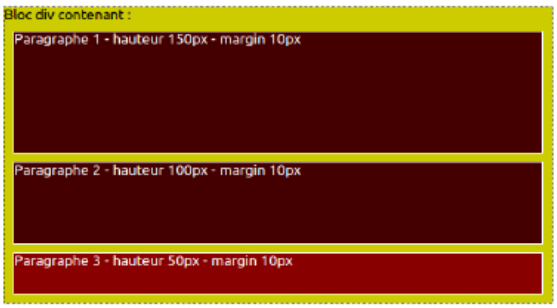
3.1-Positionnement Absolu
Définition d'un élément en position absolue :

```
#element {
            position :  absolute ;
}
```

Caractéristiques du positionnement absolu :

- L'élément se retrouve sur un autre plan que le flux courant, autrement dit il n'est plus dépendant de son environnement direct, un peu à la manière d'un calque dans un logiciel de retouche d'image
- Les éléments restants se réorganisent entre eux dans le flux courant sans prendre compte celui en positionnement absolu

Exemple avec les 2 captures ci-dessous :



Capture 1

Capture 2

Code HTML :

```
<div>
   Bloc div contenant :
        <p>Paragraphe 1 -
        hauteur 150px - margin 10px</p>
        <p>Paragraphe 2 -
        hauteur 100px - margin 10px</p>
        <p id="p3">Paragraphe 3 -
        hauteur 50px - margin 10px</p>
</div>
```

Code CSS - « capture 2 »:

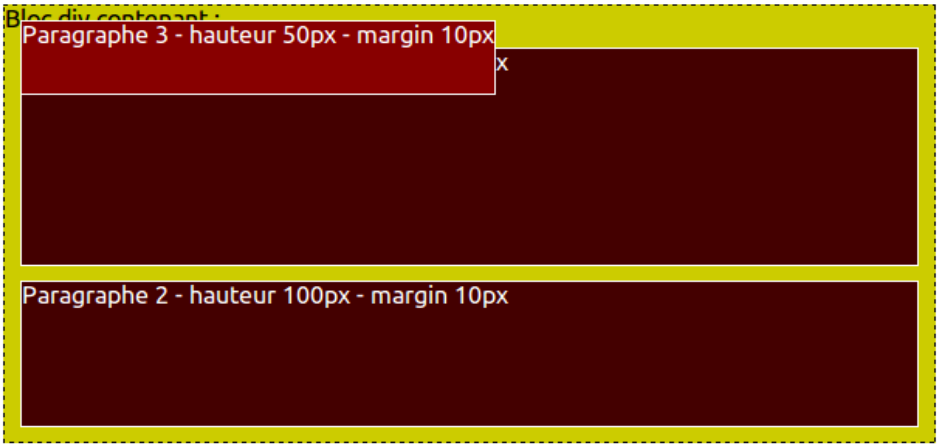
```
body {
        font-family:            ubuntu;
        font-size:              16px;
}

div {
        border:                 1px     dashed;
        background-color:       #CCCC00;
        width:                  600px;
        margin:                 50px;
}

p {
        border:         thin    solid;
        background-color:       #440000;
        color:                  white;
        margin:         10px;
        height:         150px
}

p+p{
        height:         100px;
}

#p3{
        height:         50px;
        background-color:       #880000;
        position:               absolute;
        top:                    0;
        left:                   0;
}
```

Que constatez-vous entre les deux captures ?

#p3 est en position absolu, il sort du flux (dans le flux capture 1) et se positionne dès qu'il retrouve un parent positionné (avec attribut position). Dans le cas présent, il remonte jusqu'à body, top et left sont définis par rapport à body.

Comment le repositionner par rapport à div (exemple capture 3)?

On positionne div en relatif « position : relative ». Il est toujours hors du flux car il n'est pas positionné après le paragraphe 2.



Capture 3

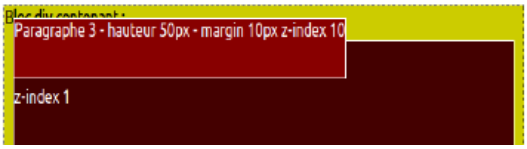
Règle à retenir :
Un élément positionné en absolu se place par rapport à son premier ancêtre positionné.

Positionnement absolu : mode de rendu

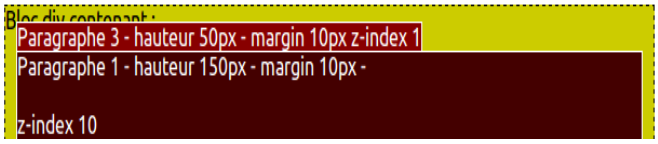
- La boite est dimensionnable quel que soit le rendu initial. Cela signifie qu'un élément HTML de rendu « inline » peut bénéficier des propriétés width,height etc. ce qui n'est pas possible dans le flux
- L'élément occupe exactement la largeur de son contenu (taille en fonction du nombre de mots par exemple)
- Les marges externes ne sont plus sujettes à la fusion

La superposition des calques

En position absolue, les éléments sont supposés se superposer aux autres et par conséquents masquer leur contenu. Il est possible de définir un ordre de superposition avec la propriété « z-index » :



p3 -> z-index:10 et p1 -> z-index:1



p3 -> z-index:1 et p1 -> z-index:10

Exercice

L'effet « stretching » permet d'étendre le contenu d'un élément (et seulement son contenu §1.2 page 2). Cherchez une solution en CSS qui permettrait à une image d'occuper tout l'espace du div article au passage de la souris sur la vignette. Complétez le CSS fourni. On considère que la vignette doit faire 50 % de sa taille et qu'elle est alignée à gauche du texte.

HTML

```
<div id="article">
        <p class="titre">...</p>
        <p class="corps"><img src="photo.png">....</p>
</div>
```

CSS

```
div {
        position :              relative ;
        background-color :      #e0aa14 ;
        width :                 800px ;
        height :                        600px ;
}
p{
        background-color :      #e0e0e0 ;
        font-size :             20px ;
        color :                 blue ;
}
img{
        position :              relative ;
        height :                        50 %;
        width :                 50 %;
        float :                 left ;
}
img:hover{
        position :              absolute ;
        left :                  0 ;
        right :                 0 ;
        top :                   0 ;
        bottom :                0 ;
}
```

3.2-Positionnement fixé

Les similitudes avec le positionnement absolu sont nombreuses :

- la déclaration se fait de la façon suivante « `position:fixed ;` »
- parce qu'on lui applique la propriété position il est considéré comme « positionné » et sert de référence aux éléments en position absolu
- il sort du flux courant et se place dans un autre plan
- la boite d'un élément fixé est dimensionnable

Il se distingue d'absolu par son rendu :
Un élément fixé demeure ancré même lorsque l'utilisateur fait défiler la page avec les ascenseurs.

On peut donc considéré qu'un élément fixé est positionné par rapport à la partie visible de la fenêtre quel que soit son ancêtre d'origine.

Exercice
Créez un bandeau bas qui se superpose à la page (et son contenu) pour afficher une alerte. Le fond de ce bloc sera en noir, il sera placé à 15 % de la gauche, d'une largeur de 500px et de 50px de haut. La bordure sera en rouge, trait plein de 3 pixels d'épaisseur.

Capture du bandeau bas "alerte"

Code CSS :

```
div#alerte{
  margin:        0;
  bottom:        0;
  left:          15%;
  position:      fixed;
  z-index:       10;
  background:    black;
  color:         white;
  height:        50px;
  text-align:    center;
  width:         500px;
  border:        3px     solid red;
}
```

3.3-Positionnement relatif

Un élément relatif se place par rapport à sa position initiale dans le flux et être ensuite décalé avec les propriétés `top` – `right` – `left` et `bottom`.

Le texte « 100px » est placé dans un span qui fait partie d'un paragraphe.
La position relative du span est définie dans le flux soit par rapport à p (qui est le parent).

Code HTML

```
<p>Paragraphe 2 - hauteur <span>100px</span> -
margin 10px</p>
```

Code CSS

```
span{
        top: 10px;
        position:       relative;
        font-weight:    bold;
        background:     #880000;
}
```

Remarques :
L'élément positionné en relatif est toujours dans le flux et interagit avec ses voisins.
Enfin pour finir, l'élément est dit positionné car définit en relatif, il devient donc référent dans le flux et utile pour un positionnement absolu d'autres éléments.

3.4-Positionnement flottant

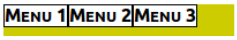
Un élément flottant est considéré hors du flux et positionné à l'extrême gauche ou droite de son conteneur d'origine tout en demeurant sur sa ligne initiale du flux.

Remarque :
Lorsqu'un élément flottant est poussé dans la même direction qu'un autre élément flottant, il demeure sur le même plan et se positionne à ses cotés. Cela permet de créer des menus horizontaux par exemple.

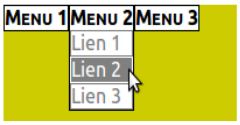
CSS de la capture :

```
div {
                        background-color: #CCCC00;
}
p {
                        border:           thin  solid;
                        background-color: #440000;
                        color:            white;
                        margin:           10px;
                        height:           150px;
                        float:            left;
                        z-index:          10;
}
```

Exercice

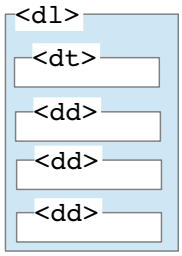


La capture ci-contre correspond à un menu déroulant, qui au passage de la souris dévoile les liens associés.



Le menu déroulant quand la souris passe sur la tête de colonne.

La meilleure technique pour concevoir les menus consiste à utiliser des balises imbriquées et complémentaires. Dans le cas présent nous allons employé `<dl>` (definition list), `<dt>` (définition text) et `<dd>` (definition data). L'association des balises est définie de la façon suivante :



dl contient les balises dt et dd, elle est assimilable à un div.

dt contient le texte affiché en haut de block.

dd contient les textes qui sont affichés dans la liste

HTML

```
<dl>
        <dt>titre</dt>
        <dd>texte</dd>
        <dd>texte</dd>
        ...
</dl>
```

1-Créer la page HTML correspondante à la capture en supposant qu'il y a 3 liens pour chaque menu.

```
<dl>
        <dt>Menu 1</dt>
        <dd>Lien 1</dd>
        <dd>Lien 2</dd>
        <dd>Lien 3</dd>

</dl>
<dl>
        <dt>Menu 2</dt>
        <dd>Lien 1</dd>
        <dd>Lien 2</dd>
        <dd>Lien 3</dd>

</dl>
<dl>
        <dt>Menu 3</dt>
        <dd>Lien 1</dd>
        <dd>Lien 2</dd>
        <dd>Lien 3</dd>
        <dd>Lien 4</dd>
</dl>
```

2-Coder le CSS pour que le menu se déroule au passage de la souris. On s'appuiera sur <dl>.

```
dl {
                        border:         thin    solid;
                        background-color:       white;
                        color:                  black;
                        margin:         0px;
                        float:                  left;
                        top:                    50;
}
dt{
                        font-weight:            bold;
                        font-variant:           small-caps;
                }
```

```
dd{
                        display:            none;
                        z-index:            10;
                        color:              grey;
                        border:       solid 1px;
                        margin-right:      auto;
                        margin-left:       auto;
                    }
dl:hover dd {
                        display:            block;
}
```

3-Le passage de la souris sur les liens déroulants est contrasté par un fond gris, un texte en blanc et le pointeur de souris. Modifier en conséquence le CSS.

```
dd:hover{
                        background-color: grey;
                        color:                  white;
                        cursor:           pointer;
}
```

3.5-Cumuler les règles de positionnement

Priorités

Certaines règles sont contradictoires, par exemple placer un élément en `float:left` avec `position:absolute` et `right:0`.

Pour palier à ce problème, CSS propose une convention de priorité entre les propriétés « `display` », « `float` » et « `position` » :

1. Si « `display:none` » est appliqué alors l'élément ne créé par de boite en conséquence « `float` » et « `position` » sont inactives
2. Sinon si une propriété « `position` » est appliquée avec une valeur « `absolute` » ou « `fixed` », « float » est forcée à « `none` » et l'élément est placé en fonction de « `top` », « `left` », « `bottom` » et « `right` ».
3. Sinon si « float » est fixée à « right » ou « left » ce sont ces valeurs qui sont prises en compte
4. Sinon l'élément est positionné suivant son mode de rendu par défaut via la propriété « display » (valeurs : inline, block etc.)

Block et inline

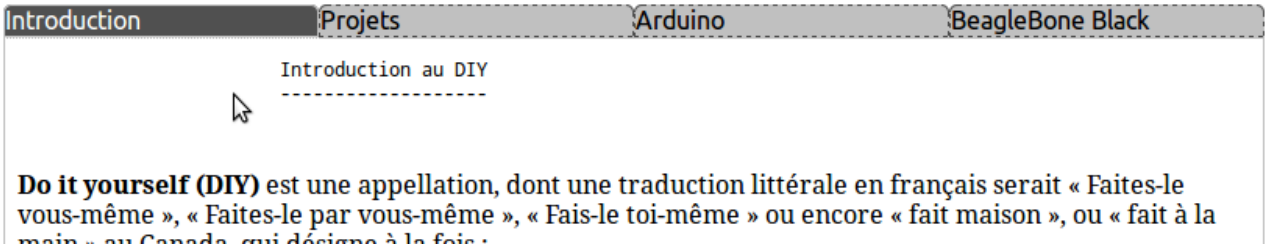
Nous avons vu que les modes block et inline ne peuvent être employés en même temps, pour rappel :

- block oblige les éléments à se positionner les uns au dessus des autres et autorise l'utilisation des propriétés «width » et « height »
- inline permet le positionnement côte à côte mais interdit l'utilisation des propriétés citées précédemment.Le navigateur calcule l'espace utilisé à partir du contenu de l'élément (longueur du texte par exemple) afin de placer les éléments suivants

L'hybridation entre « inline » et « block » est possible grâce à « inline-block ». Cette valeur de « display:inline-block » autorise le rendu côte à côte et la définition de la taille de l'élément avec « width » et « height ».

Exercice

Imaginons un menu à onglet dont les parties actives sont des liens. Par défaut, un élément « a » est de type « inline » et si on souhaite des onglets de tailles identiques le « display:inline-block » est obligatoire :



Capture des onglets

Partie HTML

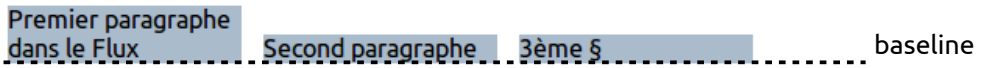
```
<a href="intro.html" target="cadre">Introduction</a>
<a href="projets.html" target="cadre">Projets</a>
<a href="arduino.html" target="cadre">Arduino</a>
<a href="bbb.html" target="cadre">BeagleBone Black</a>
<div>
      <iframe src="intro.html" name="cadre" frameborder="0">
</div>
```

Partie CSS ***à compléter***

```
div{
      width:                  806px;
      border:                 1px solid #c0c0c0;
      padding:          0;

}
iframe {
      width:                  100%;
      height:                 600px;
      margin:                 0;
      padding:          0;
}
a,a:visited{
      display:                inline-block;
      width:                  200px;
      color:                  black;
      background:             #c0c0c0;
      border:                 1px dashed #505050;
      text-decoration:        none;
      border-top-left-radius: 5px;
      border-top-right-radius: 5px;
}
a:active,a:focus{
      color:                  white;
      background:       #505050;
      border:                 1px solid #c0c0c0;
}
```

3.6-Alignement vertical

Le rendu « inline-block » force les éléments à se positionner sur la ligne de texte (« `baseline` ») :



Code HTML

```
<p>Premier paragraphe dans le Flux </p>
<p>Second paragraphe </p>
<p>3ème §</p>
```

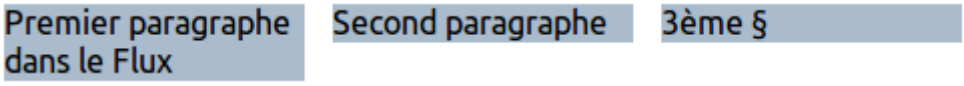
Code CSS

```
p {
      display:          inline-block;
      width:            150px;
      margin:           0 10px 0 0;
      background:       #abc;
}
```

La propriété « vertical-align », réservée aux éléments de contenu ou aux cellules de tableau, modifie l'alignement vertical. Appliquée avec un rendu « inline-block », elle aligne l'élément au sein de son parent :

Valeur	Description
`baseline`	Aligne la base de l'élément avec celle de son parent (valeur par défaut).
`bottom`	Aligne le bas de l'élément avec le bas du parent.
`text-bottom`	Aligne le bas de l'élément avec le bas de la police de l'élément parent.
`top`	Aligne le haut de l'élément avec le haut du parent.
`text-top`	Aligne le haut de l'élément avec le haut de la police de l'élément parent.
`sub`	Aligne la base de l'élément avec la ligne de base indice de son parent, tel l'élément HTML `<sub>`.
`super`	Aligne la base de l'élément avec la ligne de base exposant de son parent, tel l'élément HTML `<sup>`.
`middle`	Aligne le milieu de l'élément avec le milieu de son parent.
`<longueur>`	Aligne la base de l'élément à la longueur donnée au-dessus de la ligne de base de son parent.
`<pourcentage>`	Idem à la valeur `<longueur>`, où le pourcentage est calculé par rapport à la propriété `line-height`.

Dans l'exemple précédent, avec la valeur « top » on obtient :



Conclusion :

Nous n'avons pas abordé de façon exhaustive le positionnement avec CSS, la notion de template, grilles, les rendus tableaux etc. l'objectif de ce cours est d'aborder la notion de positionnement et les règles générales y attenant. Vous pouvez toutefois approfondir la question sur les sites comme w3school (http://www.w3schools.com/cssref/default.asp).

4.Les Frameworks

Les frameworks sont des « collections » d'outils permettant de mettre en œuvre rapidement des fonctions, programmes etc.
Dans le cadre du CSS, les frameworks fonctionnent pratiquement tous de la même façon en fournissant des classes CSS prédéfinies. On peut ainsi générer des boutons, cadres, menus et disposer d'un rendu personnalisable.
Dans tous les cas, cela ne vous affranchit pas d'apprendre le fonctionnement du framework, de décortiquer la documentation et de respecter scrupuleusement les recommandations. Si cela peut apporter un gain de temps certains pendant la conception, il est parfois nécessaire de ne plus les utiliser en phase de production afin d'optimiser les performances du site web.

Quels frameworks ?
Parmis les frameworks CSS disponibles sur internet, certains se distinguent nettement des autres par leurs possibilités et performances. On peut citer:

- « Bootstrap - http://getbootstrap.com/» de twitter, complètement « responsive design » et probablement le plus utilisé aujourd'hui
- « foundation - http://foundation.zurb.com/ » lui aussi complètement « responsive design »
- « blueprint - http://www.blueprintcss.org/ » le plus ancien
- « gumby -http://gumbyframework.com/ » « responsive design »

Le choix du framework est plus une question d'affinité et du contexte dans lequel sera conçu le projet.

JavaScript()

Chapitre 6 – JavaScript
Chapitre 7 – DOM
Annexes

BTS Systèmes Numériques
Informatique & Réseaux

FÉLIX
LE DANTEC
du secondaire au supérieur

Chapitre 6
JavaScript

Le langage HTML, même associé au langage CSS, est statique. Il ne peut effectuer de modification sur la page, interagir avec l'internaute, vérifier la validité des données échangées etc.
Pour palier à cela, il a été introduit, dans les navigateurs, des interpréteurs de langages plus évolués qui ne nécessitent pas de compilation. Le plus couramment employé est « **JavaScript** ».

1.Présentation de JavaScript

JavaScript est un langage inventé en 1995 par Netscape et ajouté au navigateur éponyme puis à tous ceux qui en sont issus.
C'est un langage ***interprété*** par le navigateur et par conséquent **exécuté coté client uniquement**. En général, il est employé pour dynamiser les pages HTML et effectuer des tâches de prétraitement (validité d'une adresse mail par exemple) avant envoi vers le serveur.
Il ne faut pas le confondre avec le Java qui est beaucoup plus puissant et évolué, en effet JavaScript est encapsulé dans l'environnement du navigateur (aucune action avec les fichiers par exemple).

Qu'est-ce que signifie « interprété » ?

Il existe deux grandes familles de langages, les « ***compilés*** » et les « ***interprétés*** ». Les premiers nécessitent un programme nommé compilateur pour les convertir en langage binaire (langage machine) et les rendre exécutables. Les plus connus sont le C, C++, Pascal et l'assembleur.
En revanche les langages « interprétés » ne sont pas convertis en langage machine, ils sont directement exécutés par l'intermédiaire d'un interpréteur comme PHP, PERL ou JavaScript. Il existe des langages hybrides (entre interprété et compilé) qui sont convertis en « byte code » puis interprétés à l'aide d'une machine virtuelle comme Java ou Python.

2.Caractéristiques de JavaScript

C'est un langage objet à prototype, cela signifie que les objets ne sont pas instanciés à partir de classe mais dispose d'un constructeur avec lequel sont crées dynamiquement méthodes et propriétés.
Note : Le paradigme « objet » sera abordé en génie logiciel.

Il va permettre de manipuler/créer les objets de la page HTML (correspondant aux balises HTML) par l'intermédiaire du DOM (Document Object Model – cours C7). Il est possible de modifier le comportement de la page ou des éléments qui la constitue y compris les styles.

Les domaines d'applications de JavaScript sont vastes et multiples :
- communications réseaux, « Ajax » - « websockets* » et « sse* »
- interaction et graphisme dynamiques avec les « canvas* », « drag & drop* » etc.
- le multitâche avec « Web workers* »
- la gestion des fichiers « File API* »
- les données « Web Storage* » , « Web Messaging* » et « Indexed DataBase & Web SQL Database* »

Modes d'exécution
Du fait qu'il soit interprété à partir du navigateur, on peut distinguer deux modes d'exécution :
- au chargement de la page HTML, le code est donc interprété qu'une fois.
- en fonction des événements détectés par le gestionnaire (mouvement ou clic à la souris, appui sur le clavier etc.) de « call-backs ». Le code est exécuté autant de fois que nécessaire.

* Recommandations HTML5

Qu'est ce qu'un call-back ?

En programmation on parle de « call-back » à propos d'une fonction ou méthode qui est exécutée à la suite d'un événement ou action particulière sans que cela soit anticipé. En règle général , la fonction « call-back » est lancée à partir d'une autre fonction.

3.Utilisation de JavaScript dans une application Web

3.1-Insertion du code JavaScript

Avec JavaScript la boucle est bouclée, en effet nous ajoutons la 3ème et dernière brique qui constitue HTML5.

HTML structure la page, définit les éléments mis en place au chargement de la page, en quelque sorte le squelette.

CSS vise à définir le style et l'aspect graphique du contenu. Permet aussi de créer quelques effets dynamiques relativement limités.

JS apporte le dynamisme à la page web et permet surtout la création d'application exécutée dans la navigateur.

Pour insérer du code JavaScript il faut donc s'appuyer sur HTML. Comme CSS il y a trois méthodes :

- ***Associé à une balise*** – principe utilisé avec la détection d'événement – aussi valable au chargement de la page

  Exemple : `<body onload="//code JavaScript">`
  `onload` est un événement associé à `<body>`, dans le cas présent, le code JS sera exécuté pendant le chargement de body (de son contenu).

- Dans la page HTML – exécuté pendant le chargement de la page HTML et/ou prise en compte des fonctions déclarées – utilisation des balises `<script></script>`

  Exemple : `<script> //code javascript </script>`
  Remarque : Par habitude on place <script> dans l'en-tête mais il est possible de l'utiliser partout dans la page HTML

- Dans un fichier externe, la plus courante. Comme CSS, le code JavaScript peut être placer dans un fichier dédié à l'aide de balise <script>. ***Ce fichier ne doit pas contenir d'instruction HTML ou CSS.***

  Exemple : `<script src="chemin/fichier.js" type="text/javascript"></script>`
  L'extension js n'est pas obligatoire mais très fortement conseillée.

3.2-Gestionnaire d'événements et interactions

Le gestionnaire d'événements est un outils permettant d'apporter l'interaction entre l'utilisateur et l'application mais pas seulement, il offre, avec HTML5, un support complémentaire en phase d'exécution.

La liste des événements pris en charge avec l'API HTML5 est donnée en ***annexe*** – source ***w3school[1].***

Association événement – objet et fonction de traitement

L'interaction consiste, à partir du gestionnaire d'événements, d'exécuter un traitement, une fonction etc.

On distingue deux modes d'écriture :

- dans l'objet HTML
  `<img name="monImage" src="monImage.png" onmouseover="//code javascript lié à l'événement">`
- dans le code javascript (fichier ou entre les balises <script></script>)
  `document.monImage.onmouseover=function(e){//code de traitement}`

La seconde solution sera abordée dans le cours C7 – JavaScript & DOM

Exercices - ***Appuyez vous sur les documents annexe***
1- Soit le champs de saisie multiligne `<textarea></textarea>`, associer l'événement sélection de l'objet.

```
<textarea onfocus="//code js"></textarea>
```

2-Soit le bouton HTML donné ci-dessous, on vous demande d'ajouter l'événement de détection de click sur celui-ci et de lancer la fonction JavaScript « `alert('Bouton cliqué') ;` » (fonction qui affiche un message dans la fenêtre – le message est écrit entre parenthèses).
Code HTML du bouton à modifier : `<input type="button" value="test">`

```
<input type="button" value="test" onclick="alert('Bouton cliqué') ;">
```

3-Il est possible d'associer les événements pour un même objet. On vous demande de modifier le code ci-dessous d'un lecteur vidéo HTML5 en détectant les événements de lecture (marche et arrêt) qui exécutent respectivement les fonctions `videoMarche()` et `videoArret()`.
Code HTML à modifier

```
<video id="maVideo"  width="720" height="480">
            <source src="videos/algore.ogv">
</video>
```

```
<video id="maVideo" width="720" height="480" onplay="videoMarche();"
onstop="videoArret();">
        <source src="videos/algore.ogv">
</video>
```

1 **www.w3schools.com/jsref/dom_obj_event.asp**

En résumé :
Un gestionnaire d'événements permet d'exécuter des actions programmées en fonction des événements interceptés (action de l'utilisateur, temporisation ...).

4.Le langage

Avant même de se jeter sur un clavier, il faut d'abord procéder à la résolution méthodique du problème posé par le programme, d'utiliser les outils de modélisation qui y concourent comme l'algorithmie, l'**UML**[2] etc.. et le développement web n'échappe pas à la règle.

Une fois l'étude terminée, on peut alors passer au code. Les langages permettant de programmer des applications reposent sur le même ***paradigme*** à savoir « ***la programmation orientée objet*** ».

Qu'est ce que signifie paradigme informatique ?

C'est la manière dont on programme à partir d'un langage donné. Il existe de nombreux paradigmes, les plus connus sont « la programmation impérative » dont le C fait partie et « la programmation orientée objet » avec le C++, Java, Python, PHP et JavaScript entre autres.

4.1-Principes de base

Premier programme

Pour aborder le fonctionnement et la syntaxe du langage d'un point de vue global, nous allons écrire un programme qui permet d'afficher une fenêtre au chargement de la page HTML. Le programme est directement inséré :

```
<!doctype html>
<html>
	<head>
		<title>Alerte!</title>
		<meta charset="utf-8">
		<style>
			body{
				font-family:  ubuntu;
				font-size:      30px;
				color:          #ff0000;
				}
			h1{
				position:       fixed;
			}
			h1#tete{
				top:            0;
			}
			h1#pied{
				bottom: 0;
			}
		</style>
	</head>
	<body>
		<h1 id="tete">En tête de la page HTML</h1>

		<script>
			alert("Fenêtre JavaScript");
		</script>
		<h1 id="pied">Pied de la page</h1>
	</body>
</html>
```

1-Encadrez la partie qui intègre du code JavaScript.

2 http://fr.wikipedia.org/wiki/UML_%28informatique%29

2-Que constate t-on au chargement de la page HTML ? **(1.html)**

Est affiché :

- le titre h1 d'en tête
- la fenêtre JavaScript

Le navigateur a bloqué le chargement il attend la validation « ok ».
En conséquence le titre h1 du pied de page ne sera affiché qu'après avoir validé.

3-Comment est interprété le code JS ?

A la volée comme HTML et CSS, c'est à dire dès que le navigateur lit l'instruction.

4-Si on écrit le code JS dans un fichier externe, a t-on le même fonctionnement ? **(2.html)**

Oui même constat.

5-Que peut-on remarquer à propos de la syntaxe JS ?

La ligne se termine toujours d'un « ; ». `alert()` est une **fonction prédéfinie** qui affiche le texte entre parenthèses dans une fenêtre. On remarque aussi que le texte est délimité par des guillemets.
Une fonction en JS est nommée et toujours suffixée par des ().
Remarque :*Cette fonction est souvent utilisée pour afficher des valeurs en phase de développement.*

Second programme
Nous allons afficher un contenu HTML sans balise HTML à partir de JavaScript:

```
<!doctype html>
<html>
        <head>
                <title>Alerte!</title>
                <meta charset="utf-8">
                <style>
                      body{
                                font-family:  ubuntu;
                                font-size:      30px;
                                color:          #ff0000;
                                }
                        h1{
                                position:       fixed;
                                top:            50%;
                        }
                        h1#tete{
                                top:            0;
                        }
                        h1#pied{
                                top:            auto;
                                bottom: 0;
                        }
                </style>
        </head>
        <body>
                <h1 id="tete">En tête de la page HTML</h1>
               <script>
                        document.write("<h1>Titre h1 généré à partir de JavaScript</h1>") ;
                </script>
                <h1 id="pied">Pied de la page</h1>
        </body>
</html>
```

Interprétation avec Firebug :

1-A partir de la capture ci-contre que peut-on constater ? **(3.html)**

Le navigateur a exécuté le code JS qui a permis de créer à la volée la ligne pointée par le curseur de la souris. Puis cette ligne a été interprétée comme étant du HTML car elle est affichée en tant que titre <h1> (cf le style spécifique).

2-La ligne JS permet d'écrire une chaîne de caractères dans la page courante. Que représentent respectivement « `document` » et « `write` » ?

« `document` » caractérise la page en cours d'utilisation sur laquelle on applique la méthode qui succède. JavaScript considère « document » comme un « objet » avec lequel on dispose de « propriétés » et de « méthodes ».
« `write` » est la méthode qui va agir sur « document » et son rôle consiste à écrire une chaîne dans la page.

3-D'un point de vue syntaxique, qu'est-ce qui assure lien entre « l'objet » et « méthode » ?

On utilise le caractère « `.` » : `objet.méthode() ;`

On peut en faire de même avec les propriétés, par exemple le titre de la page HTML en modifiant le script :

```
<script language="javascript">
1 :       document.write("<h1>titre h1 généré à partir de JavaScript</h1>") ;
2 :       alert("Avant modification : Le titre de la page est 'Alerte!' Affiché dans l'onglet");
3 :       document.title="Nouveau Titre";
4 :       alert("Après modification : l'onglet affiche 'Nouveau Titre'");
 </script>
```

4-La ligne 3 permet de modifier la propriété « title » qui contient la chaîne correspondant au titre de la page HTML (équivalent de la balise <title>). Quelles différences y a t-il entre une propriété et une méthode ? **(4.html)**

La propriété contient une valeur qu'il peut être possible de modifier, dans le cas présent avec « = » qui est considéré comme un opérateur d'affectation.
Alors qu'une méthode effectue une action, `write()` permet d'écrire. De plus au niveau syntaxe, la propriété n'est pas suivie de parenthèses.

Notions « objet », « méthode » et propriétés

Les exemples précédents font appel au paradigme « programmation orientée objet », en effet on remarque que les méthodes et les propriétés sont toujours associées à un élément que l'on désigne par le terme d'objet.
En développement web, cela correspondant souvent à un élément HTML mais pas seulement (nous verrons des exemples par la suite).

Syntaxe objet

On applique une méthode à un objet à la condition qu'ils soient compatibles :

```
objet.methode() ;
```

On modifie une propriété à un objet :

```
objet.propriété= valeur;
```

On peut considérer que les attributs HTML relatif à une balise sont aussi des propriétés JavaScript.

Exercice :
1-Écrire en JavaScript l'équivalent de la balise **<img>**[3] décrite ci-dessous en vous appuyant sur la syntaxe JS.
En HTML :

```
<img name="objetImage" src="monImage.png" width="800" height="600" alt="texte flottant">
```

On souhaite appliquer la méthode « `zoom()` » à cette même image.
En JavaScript:

```
document.objetImage.src="monImage.png" ;
document.objetImage.width=800;
document.objetImage.height=600;
document.objetImage.alt="texte flottant" ;
document .objetImage.zoom() ;
```

Sens d'exécution des instructions

2-De part la structure des lignes JS, que remarquez-vous ? Pourquoi est-ce écrit de la sorte ?

(1) Les lignes sont toutes préfixées par document. En effet l'image est incluse dans <body> et son équivalent est « document » en JS, cela signifie que JavaScript respecte la hiérarchie entre les éléments au même titre que HTML. On parle alors de hiérarchie entre les objets.
(2) L'image est nommée en HTML et c'est ce nom que l'on utilise en JS pour la désigner.

3-Quelle conséquence cela peut avoir pour les objets globalement ?

L'accès aux propriétés ou l'application des méthodes d'un objet va dépendre de sa position dans la hiérarchie d'une part, deuxièmement le nommage des objets doit être unique sinon il y a un risque certains de confusion.

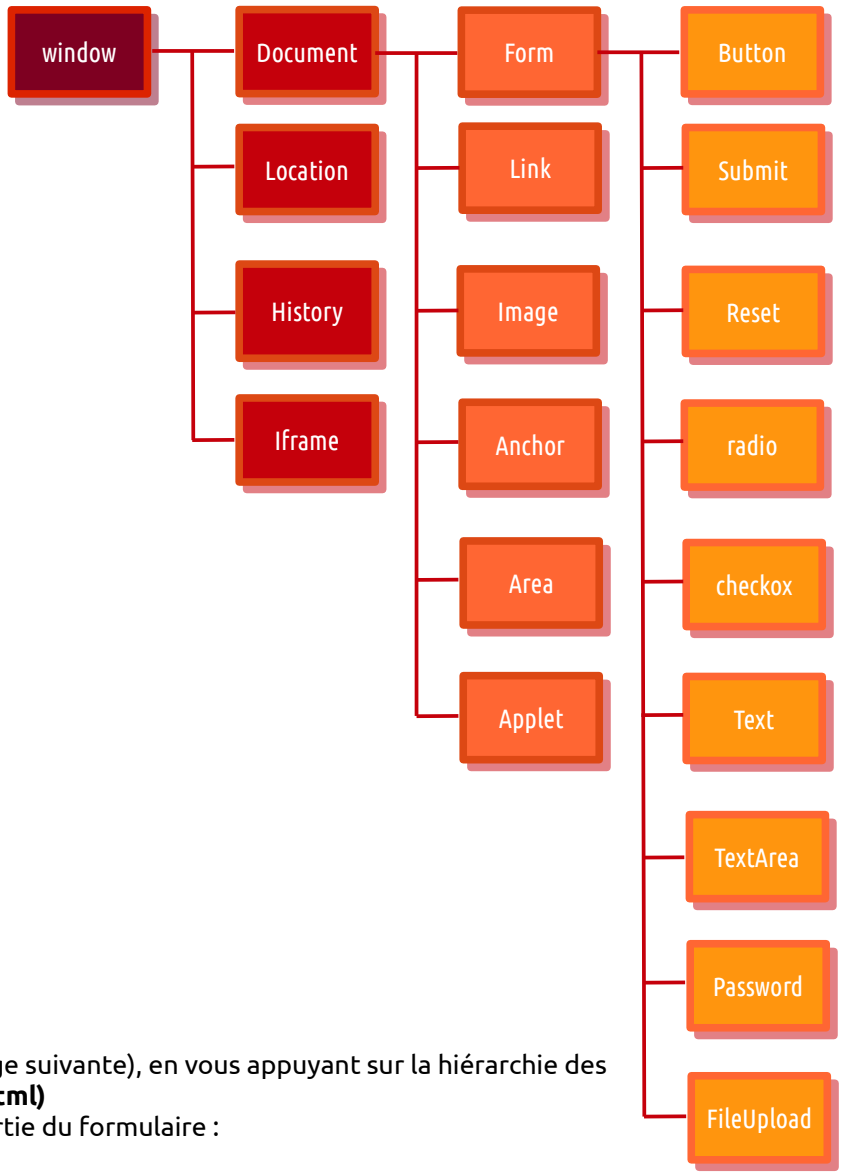
4-Que remarquez-vous au niveau des valeurs de propriétés ?

Elles sont typées dans le sens ou JS distingue un nombre d'une chaîne de caractères grâce aux guillemets. Il existe d'autres types que nous verrons par la suite.

3 http://www.w3schools.com/jsref/dom_obj_image.asp – liste des propriétés de l'objet img (à consulter pour tous les objets)

4.2 Hiérarchie des objets en JavaScript

Comme nous l'avons vu précédemment, les objets sont hiérarchisés en fonction de leur « position » dans la page HTML. Attention, on rappelle que certains objets n'ont pas d'équivalent HTML comme « Date » ou «Math». Liste ci-dessous n'est pas exhaustive, il manque un certains nombre d'objets issus d'HTML5.



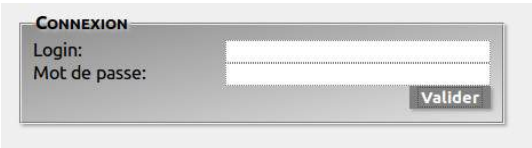
L'objet `window` caractérise la fenêtre du navigateur. Pour accéder à l'historique de navigation ou à l'URL de la page visitée, il est nécessaire d'y faire référence.

L'objet document supporte tous les objets de la page HTML et ceux qui n'ont pas de rendu graphique mais susceptible d'être utilisés dans les scripts.

Exercice

Soit le formulaire (code HTML page suivante), en vous appuyant sur la hiérarchie des objets JS, on vous demande : **(5.html)**
1- de donner les objets faisant partie du formulaire :

input → login , pwd et btnCx



Capture du formulaire de saisie

code HTML du formulaire:

```
1.  <!doctype html>
2.  <html>
3.          <head>
4.               <title>Exercice JS</title>
5.               <meta charset="utf-8">
6.               <style>
7.                   body{
8.                          font-family:  ubuntu;
9.                          font-size:      16px;
10.                         background:     #f0f0f0;
11.                  }
12.                  fieldset{
13.                         width:          400px;
14.                         position:       fixed;
15.                         top:            20%;
16.                         left:           200px;
17.                         box-shadow:     2px 2px 2px #a0a0a0;
18.                         background:     linear-gradient(to bottom right, #909090, #e0e0e0);
19.                  }
20.                  legend{
21.                         font-variant:   small-caps;
22.                         font-weight:    bold;
23.                         text-shadow:    2px 2px 2px #c0c0c0;
24.                  }
25.                  label{
26.                         float:left;
27.                  }
28.                  input{
29.                         display:block;
30.                         margin-left:    auto;
31.                         background:     white;
32.                         transition-property:    background;
33.                         transition-duration:    0.5s;
34.                         border: thin dotted grey ;
35.                  }
36.                  input:focus{
37.                         background:     #D7E8D9;
38.                  }
39.                  input[type="button"]{
40.                         box-shadow:     2px 2px 2px #a0a0a0;
41.                         background:     grey;
42.                         color:          white;
43.                  }
44.                  input[type="button"]:active{
45.                         box-shadow:     0px 0px 0px #a0a0a0;
46.                         background:     white;
47.                         color:          grey;
48.                  }
49.              </style>
50.         </head>
51.         <body>
52.              <form name="formulaire">
53.                  <fieldset><legend>Connexion</legend>
54.                      <label>Login:</label><input type="text" name="login">
55.                      <label>Mot de passe:</label><input type="password" name="pwd">
56.                      <input type="button" value="Valider" name="btnCx">
57.                  </fieldset>
58.              </form>
59.          </body>
60. </html>
```

2-Que remarquez-vous au niveau de la syntaxe CSS de ces objets ?

input[type="button"] permet de spécifier la nature de l'input (ici le bouton). Cette syntaxe CSS permet de sélectionner suivant un attribut HTML.

3-On souhaite insérer du code JavaScript dans la page HTML, sachant que l'on va manipuler des objets du formulaire, à partir de quelle ligne peut-on insérer ce code ? Écrire en HTML alors la déclaration de cette zone de script.

Compte tenu que nous devons accéder au formulaire, il faut que celui-ci soit connu du navigateur quand on exécutera la code JS, donc deux possibilités :

- soit on place le code après la ligne 58
- soit dans l'en-tête mais uniquement avec des fonctions JS (partie traitée plus loin)

```
58          </form>
59          <script language="javascript">
                  … code JS …
??          </script>
```

4-La chaîne de caractères saisie en tant que login est accessible via la propriété « `value` », écrire en JavaScript et en une ligne , la récupération du login dans une variable « `log` » :

```
var log = document.formulaire.login.value ;
```

5-Modifiez la ligne HTML du bouton « `bntCx` » pour que celui lance la fonction javascript « `affiche()` » dès qu'on l'actionne.

```
<input type="button" value="Valider" name="btnCx" onclick="affiche();">
```

6-A partir des 2 questions précédentes, on souhaite que le login soit affiché dans une fenêtre d'alerte quand on clique sur le bouton, complétez en conséquence la fonction JavaScript « affiche() » :

```
function affiche(){
      var log=document.formulaire.login.value ;
      alert(log) ;
}
```

Remarque autre écriture possible : (5.html)

```
function affiche(){
      alert(document.formulaire.login.value) ;
}
```

7-Pourquoi utiliser un gestionnaire d'événement et une fonction ?

La saisie se fait après le chargement de la page donc pour récupérer la saisie, il faut passer par le gestionnaire d'événement (action de l'utilisateur dans ce cas) et une fonction qui sera exécutée qu'à ce moment précis.

Remarque sur les commentaires en JavaScript

Deux façons de faire des commentaires :

- sur une ligne on utilise « `//` » suivi du commentaire
- sur plusieurs lignes

```
/*
      commentaires
*/
```

4.3 Les variables en JavaScript

Les variables sont des objets permettant de mémoriser des informations afin de faire des traitements. Dans la plupart des langages, les variables sont typées, c'est à dire qu'il est nécessaire de leur donner un type à la déclaration (comme le C/C++/Java etc.).

En langage C:

```
int c ;      //déclaration de c en tant qu'entier
…
c=1200 ;     //affectation de valeur
```

En JavaScript, le type n'existe qu'à l'affectation des valeurs :

```
var c ;            // « c » est déclarée mais non typée
c=1200 ;           // « c » est un nombre à l'affectation
```

Règles de déclaration des variables

Les noms donnés aux variables doivent respecter quelques contraintes comme :

-Toujours commencer par une lettre ou le caractère « _ »
-Pas d'espaces, lettres et chiffres autorisés (si ce n'est pas le premier caractère)
-Un certains nombre de mots sont réservés il est donc impossible de les utiliser pour nommer une variable
-Respecter la casse, JavaScript tient compte des minuscules et Majuscules.
-Toujours précédée de l'opérateur « `var` » au moment de la déclaration

Types en JavaScript (hors objet)

Les types natifs :

Type	Description
`Boolean`	Type dont les valeurs possibles sont true et false.
`Null`	Type dont l'unique valeur possible est null. Une variable possède cette valeur afin de spécifier qu'elle a bien été initialisée mais qu'elle ne pointe sur aucun objet.
`Number`	Type qui représente un nombre.
`String`	Type qui représente une chaîne de caractères.
`Undefined`	Type dont l'unique valeur possible est undefined. Une variable définie possède cette valeur avant qu'elle soit initialisée.

Les types natifs (parfois appelés primitifs) sont des « pseudo-objets » qui disposent de propriétés et de méthodes.

Attention : Les booléens (Boolean) sont déclarés à l'aide des valeurs « `true` » et « `false` », ce sont des mots clés utilisés par JavaScript.

Objet référencé:

C'est un type défini par le développeur, il est associé à un espace mémoire et sont stockés dans le tas (heap).
Seules les références à ces valeurs sont stockées dans la pile d'exécution (stack). Ce mécanisme permet d'utiliser deux variables pointant sur une même adresse et donc la même instance.

En JavaScript, tous les objets suivent ce principe, comme l'illustre le code suivant :

```
var monObj1 = new Object() ;
var monObj2 = monObj1 ;
```

L'opérateur « **new** [4]» crée un objet à partir d'une fonction « constructeur » qui va ainsi initialiser l'objet (la variable).

Détermination de type

Il est parfois nécessaire de connaître le type de la variable pour effectuer un traitement adapté, la méthode `typeof` remplit ce rôle.

Exercice
A partir du tableau page 11, donnez les valeurs attendues pour chaque cas ci-dessous :

```
var variable1;
alert(" type de variable1 : "+(typeof variable1));
// variable1 est de type undefined
var variable2 = null;
alert(" type de variable2 : "+(typeof variable2));
// variable2 est de type object
var variable3 = 12.3;
alert(" type de variable3 : "+(typeof variable3));
// variable3 est de type number
var variable4 = "Monsieur Dupont";
alert(" type de variable4 : "+(typeof variable4));
// variable4 est de type string
var variable5 = true;
alert(" type de variable5 : "+(typeof variable5));
// variable5 est de type boolean
```

Il est aussi possible de vérifier le type :

Méthode	Description
`isArray`	Détermine si le paramètre est un tableau.
`isBoolean`	Détermine si le paramètre est un booléen.
`isEmpty`	Détermine si un tableau est vide.
`isFinite`	Détermine si le paramètre correspond à un nombre fini.
`isFunction`	Détermine si le paramètre est une fonction.
`isNaN`	Détermine si la valeur du paramètre correspond à NaN (Not a Number).
`isNull`	Détermine si le paramètre est null.
`isNumber`	Détermine si le paramètre est un nombre.
`isObject`	Détermine si le paramètre est un objet.
`isString`	Détermine si le paramètre est une chaîne de caractères.
`isUndefined`	Détermine si le paramètre est indéfini, c'est-à-dire une référence non initialisée.

Ces méthodes seront utilisées avec les structures de contrôles (`if, else if, swicth case`)

4 https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Op%C3%A9rateurs/L_op%C3%A9rateur_new

Conversion de type
La manipulation des variables n'est pas toujours possible sans avoir fait au préalable une conversion de type. Pour cela on utilise les méthodes suivantes :

```
variable.toString() ;  // convertit la variable en chaîne de caractères
parseInt(variable) ;   // convertit en entier
parseFloat(variable) ; // convertit en flottant
```

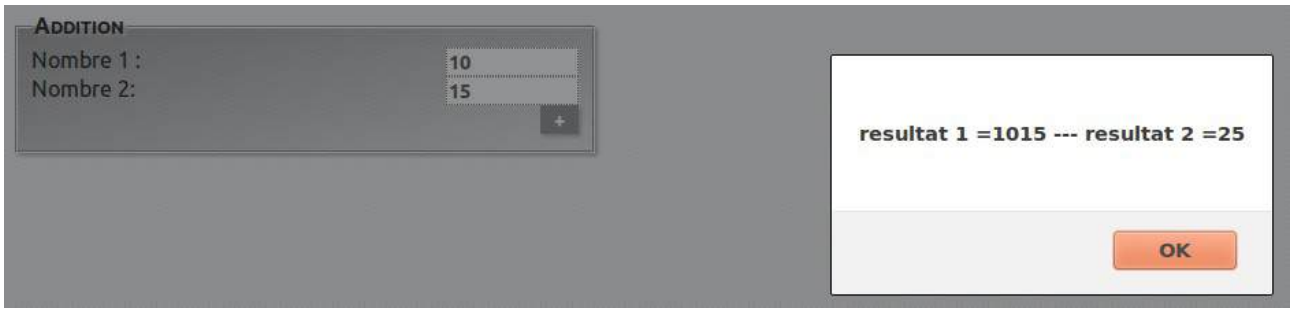
Attention à la syntaxe.

Exercice **(6.html)**

On souhaite additionner deux nombres saisis dans un formulaire :
HTML

```
<form name="formulaire">
      <input type="text" name="nbr1">
      <input type="text" name="nbr2">
      <input type="button" value="+" onclick="add() ;">
</form>
```

Ecrire le programme en JS afin d'afficher le résultat de l'addition dans une fenêtre d'alerte :



JS

```
function add(){
      var nombre1=document.formulaire.nbr1.value ;
      var nombre2=document.formulaire.nbr2.value ;
      var resultat1 = nombre1 + nombre2 ;
      var resultat2 =.parseFloat(nombre1) +.parseFloat(nombre2) ;
      alert("resultat 1 =" + resultat1 + " --- resultat 2 =" + resultat2) ;
}
```

Que remarquez-vous pour « resultat 1 » ? Pourquoi ?

Un formulaire renvoi une chaîne de caractères, nombre1 et nombre2 sont donc considérées comme des variables de type « String ». « resultat1 » contient deux chaînes concaténées (avec +).

4.4 Les opérateurs sur variable

Les opérateurs permettent de lier des variables ou/et des expressions entre elles.

Opérateurs d'affectation

Opérateur le plus courant, il peut être employé avec un autre opérateur.

Symbole	Définition
=	Affectation de base
+=	Addition puis affectation
-=	Soustraction puis affectation
*=	Multiplication puis affectation
/=	Division puis affectation

Opérateurs Arithmétiques

Ils assurent les opérations mathématiques standards.

Symbole	Définition
+	Addition
-	Soustraction
/	Division
*	Multiplication
%	Modulo , retourne le reste d'une division
++	Incrémente
--	Décrémente

Remarque : l'opérateur « + » associé à des chaînes de caractères permet de réaliser la concaténation entre elles.

Opérateurs de comparaison

Les opérateurs de comparaison sont utilisés dans les expressions de condition des structures de programme.
Ils permettent de comparer deux expressions entre elles.
Le résultat de cette comparaison est de type binaire (`true` / `false`).

Symbole	Définition
==	Egalité
<	Strictement inférieur
>	Strictement supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
!=	Différent

Opérateurs logiques

Les opérateurs permettent de réaliser des tests logiques à partir de variables booléennes ou d'autres expressions de condition.

Symbole	Définition
&&	Et logique
||	Ou logique
!	Complément

Exercice
Écrire sous une autre forme :

```
maVar = maVar + 150 ; →        maVar += 150 ;

maVar = maVar -1 ;  →          maVar -- ;
```

Comment calculer de la parité d'un nombre ?

> Il suffit de diviser par 2 et d'en récupérer le reste, s'il est nul c'est un nombre pair sinon nombre impair.
>
> ```
> var parite = nombre%2 ;
> ```

4.5 Structures de contrôle

Le langage JavaScript définit plusieurs structures de contrôle afin de réaliser des conditions et des boucles et de mettre en œuvre des renvois.
Ces structures classiques sont présentes dans la plupart des langages de programmation.
Comme nous allons le voir, elles sont aisément utilisables.

a-Conditions « `if` » , « `else` » et « `else if` »

Les tests conditionnels sont les plus couramment utilisés, ils permettent d'effectuer des actions en à l'aide aux opérateurs logiques et de comparaison (voir page précédente).

Syntaxes

if (*condition*){
  // action exécutée si condition vérifiée
}
	Cas de test d'une condition

if(*condition*){
  // action exécutée si condition vérifiée
} **else** {
  // action exécutée si condition non vérifiée
}
	Deux possibilités attendues

if(*condition*){
  // action exécutée si condition vérifiée
} **else if**(*condition*){
  // action exécutée si autre condition vérifiée
} **else** {
  // action exécutée si aucune condition n'est vérifiée
}
	Possibilités multiples

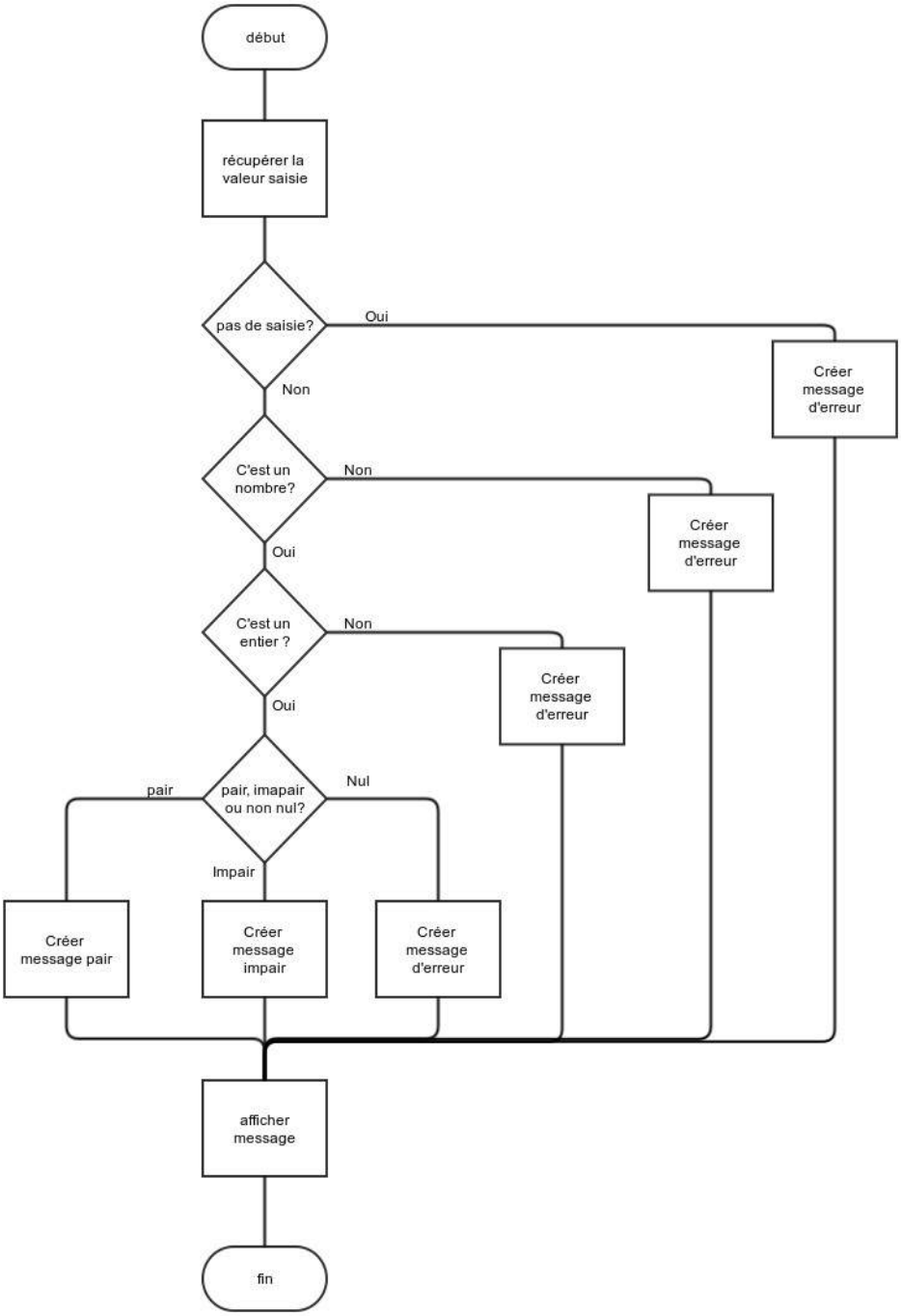
Chaque bloc d'action est encadré par « **{}** » sauf si une seule action est définie. Les conditions sont à mettre systèmatiquement entre « **()** ».

Exercice **(7.html)**

En reprenant l'exercice précédent, écrire le script permettant d'afficher dans une fenêtre d'alerte si le nombre saisi dans un formulaire est pair ou impair. Les conditions à tester sont :

- On cherche la parité si le nombre est entier et non nul (peut être négatif)
- On affiche une erreur texte adaptée si rien a été saisi, si c'est un nombre à virgule, si nombre 0 et si c'est une chaîne de caractère qui a été saisie

Dans un premier temps faire l'algorithme sous forme d'**organigramme** :



HTML +JS
Variables à

déclarer :

« nombre » : contient la valeur saisie par formulaire

« message » : contient le texte à afficher dans la fenêtre d'alerte.

```
<body>
        <form name="formulaire">
           <fieldset><legend>Pair ou Impair</legend>
              <label>Nombre entier à tester :</label>
               <input type="text" name="nbr" size="5">
               <input type="button" value="+" name="btnCx" onclick="tester();">
           </fieldset>
        </form>
        <script language="javascript">
         function tester(){
            var nombre = document.formulaire.nbr.value;
            if(nombre==""){
              reponse = " Veuillez saisir un nombre";
            }else{
             if(isNaN(parseFloat(nombre))){
                var reponse = "Saisie incorrecte, il faut un nombre";
             }else{
                if(parseFloat(nombre) == parseInt(nombre)){
                    nombre=parseInt(nombre);
                    if(nombre % 2 == 0 && nombre != 0){
                        var reponse = nombre +" est pair";
                    }else if(nombre % 2 > 0){
                        var reponse = nombre +" est impair";
                    }else{
                        var reponse = "0 est ni pair ni impair!";
                    }
                }else{
                    var reponse = nombre +" doit etre une valeur entière";
                }
             }
            }
          alert(reponse);
         }
        </script>
</body>
```

b-structure `switch case`

Cette structure de contrôle permet de procéder aux tests de valeurs possibles pour une variable.

Syntaxe

switch(*variable*){
case *valeur1* **:**
//actions
break ;
case *valeur2* **:**
//actions
break ;
…
case default :
//actions
}
	Cette structure impose de délimiter les actions produites pour une valeur par l'instruction break sauf pour les actions définies par défaut (default) qui correspond à tout autre valeur que celles testées.

Cette structure ne permet de tester qu'une seule variable mais il est possible de combiner des `switch case` pour du « multi-test ».

Exercice
Écrire un script qui renvoie un message en fonction de la variable « `systeme` » :

Valeur de « `systeme` »	**Valeur de « `message` »**
`Linux`	`Choix judicieux`
`Windows`	`Comme quoi vitrier mène à tout !`
`OSX`	`Je n'ai jamais vraiment apprécier les pommes`
`Default`	`RAS`

Code :

```
switch(systeme){
      case "Linux" :
            message="Choix judicieux" ;
            break ;
      case "Windows" :
            message="Comme quoi vitrier mène à tout !" ;
            break ;
      case "OSX" :
            message="Je n'ai jamais vraiment apprécier les pommes" ;
            break ;
      case default :
            message="RAS" ;
}
alert(message) ;
```

4.6 les boucles

JavaScript définit quatre types de boucles, `for, for...in, while` et `do...while`.

a-Boucle `for`
La première s'appuie sur le mot-clé `for` afin de spécifier à la fois les traitements d'initialisation réalisés à l'entrée de la boucle, la condition de sortie et les traitements à réaliser après chaque itération. Tant que la condition de sortie est vraie, l'itération continue.

Syntaxe :
```
for(valeur de départ;valeur de fin de boucle; valeur à chaque itération){
      //actions
}
```

Exercices
1-Écrire un script qui permet de répéter 1000 fois la même chose.

```
for(var i=0;i<1000;i++){
      //action
}
```

2-Quelle est la variable testée ? Comment est-elle déclarée et pourquoi ?

C'est la variable i qui est testée. Elle est déclarée dans la boucle en tant que variable ***locale*** afin de ne pas impacter le programme.

3-Modifiez la boucle pour que celle-ci n'affiche que les nombre pairs (en partant de 0 pour aller jusqu'à 1000).

```
for (var i=2;i<1002;i+=2) {
      //action
}
```

4-On souhaite créer un tableau HTML avec les balises `<tr>` et `<td>`. Il compte 10 lignes de 5 cellules chacune. Le tableau sera généré sous forme de chaine de caractères et stocké dans la variable « `tableau` » qui sera ensuite affiché à l'aide de « `document.write()` ». On affichera dans chaque cellule, le numéro de ligne et le numéro de colonne (séparés par une « , »). **(8.html)**

```
<script>
     var tableau="<table border='1'>";
     for(var i=0;i<10;i++){
        tableau+="<tr>";
        for(var j=0;j<5;j++){
            tableau+="<td>"+(i+1)+","+(j+1)+"</td>";
        }
        tableau+="</tr>";
     }
     tableau+="<table>";
     document.write(tableau);
</script>
```

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5
6,1	6,2	6,3	6,4	6,5
7,1	7,2	7,3	7,4	7,5
8,1	8,2	8,3	8,4	8,5
9,1	9,2	9,3	9,4	9,5
10,1	10,2	10,3	10,4	10,5

Remarque : La condition de fin ne doit jamais être testée avec le signe « = » seul, soit « < », soit « > », soit « >= » et « <= ».

b-boucle `for in`

La boucle `for...in` est une variante de la précédente qui se sert du mot-clé `for` conjointement avec le mot-clé `in`, ce dernier permettant de spécifier la variable à utiliser.
Cette boucle permet de parcourir tous les éléments des tableaux indexés ou des objets, comme nous les verrons par la suite.

Sa syntaxe est la suivante :

```
for( variable in structure ) {
   //action
}
```

c-boucles `while , do...while`

La boucle `while` permet de spécifier la condition de fin. Aussi longtemps que cette condition est vraie, les traitements sont répétés.

La syntaxe de cette boucle est la suivante :

```
while( condition de fin ) {
  //actions
}
```

Exercices **(9.html)**
Refaire les exercices de la boucle for avec while cette fois.

1-Écrire un script qui permet de répéter 1000 fois la même chose.

```
var i= 0 ;
while(i < 1000){
      //actions
      i++ ;
}
```

2-Quelle est la variable testée ? Comment est-elle déclarée et pourquoi ? Que se passe t'il sans incrémentation , comment se comporte le navigateur ?

La variable testée est i, elle doit être initialisée avant `while` car on ne spécifie que la condition de fin. Sans incrémentation, i n'atteindra jamais 1000, on reste bloqué dans la boucle et le navigateur est bloqué lui aussi.
C'est le problème majeur du `while`, la boucle sans fin est un système utilisé pour programmer des serveurs mais avec un contrôle sur les variables d'état.

3-Modifiez la boucle pour que celle-ci n'affiche que les nombre pairs (en partant de 0 pour aller jusqu'à 1000). Y a t'il une particularité à propos de la condition de fin ?

```
var i=0 ;
while(i<1000){
      //action
      i+=2 ;
}
```

Oui on ne met pas de « = » en test, cas typique ici avec une condition de test jamais validé i = 1001. On reste alors dans la boucle, c'est pourquoi on ne met jamais de signe « = » seul (comme for).

4-Même question, un tableau 10x5 avec `while`.

```
<script>
        var tableau="<table border='1'>";
        var i=0;
        while(i<10){
           var j=0;
           tableau+="<tr>";
           while(j<5){
               tableau+="<td>"+(i+1)+","+(j+1)+"</td>";
               j++;
           }
           i++;
           tableau+="</tr>";
        }
        tableau+="</table>";
        document.write(tableau);
</script>
```

La dernière boucle `do...while` est une variante de la précédente, qui permet d'effectuer le bloc avant la première condition.
La syntaxe de cette boucle est la suivante :

```
do {
      //action
} while( condition de fin );
```

On est certains que l'action sera faite au moins une fois.

4.7 Les tableaux

Les tableaux en JavaScript sont instanciés, en effet c'est à partir de la classe `Array` (avec un A majuscule). Les tableaux sont des séries de valeurs regroupées sous un même nom.

Il y a deux types de tableaux :
- Les tableaux indicés, chaque élément du tableau est identifié par un indice numérique
- Les tableaux associatifs, chaque élément est identifié par une clé textuelle

a-Tableaux indicés
La déclaration d'un tableau de type `Array` se fait avec l'opérateur **`new`**[5]. L'indice numérique varie de 0 à n (0 étant le premier indice).

```
var fruits = new Array() ;
fruits[0]= pomme ;
fruits[1]= citron ;
fruits[2]= raisin ;
fruits[3]= fraise ;
```

Il est possible de créer et d'initialiser le tableau en même temps :

```
var fruits = new Array("pomme","citron","raisin","fraise") ;
```

5 Voir page 12

Pour accéder à une valeur du tableau il suffit d'utiliser le nom de celui suivi de « [] » comprenant l'indice de la valeur :

```
var fruits = new Array("de la pomme","du citron","du raisin","des fraises");
var x=Math.round(3*Math.random());
resultat.value="Le hasard a voulu qu'on mange "+fruits[x];
```

Dans cet exemple, la valeur recherchée dans le tableau fruits[x], x étant une variable aléatoire générée par `Math.random()`.

Il est possible d'accéder à la totalité d'un tableau à l'aide de la fonction for in :

```
for(i in fruits) {
      document.write(fruits[i]) ;
}
```

L'intérêt de for...in

A supposer que le tableau suivant soit déclaré comme suit :
(10.html)

```
var tab = new Array() ;
tab[0]="pomme" ;
tab[1]="fraise" ;
tab[3]="citron" ;
tab[4]="raisin" ;
```

Écrire avec la boucle for puis avec for in, un script permettant de lire le tableau :

```
for(var i=0;i<tab.length;i++){
      document.write(tab[i]) ;
}
for(var i in tab){
      document.write(tab[i]) ;
}
```

```
Lecture de tab avec une boucle for normale :
tab[0]=Pomme
tab[1]=Fraise
tab[2]=undefined
tab[3]=Citron
tab[4]=Raisin

Même chose avec for in:
tab[0]=Pomme
tab[1]=Fraise
tab[3]=Citron
tab[4]=Raisin
```

Que remarque t-on sur la capture ?

Le cellule d'indice 2 n'existe pas, avec une boucle « `for` » il faut vérifier l'existence de la cellule alors qu'avec « `for...in` » ce n'est pas nécessaire.

b-Tableaux associatifs

Les indices sont remplacés par des mots appelés « clés », la méthode de déclaration est identique que celle des tableaux indicés avec `new` et `Array()`:

```
var fruits = new Array("automne","exotique","été","printemps");
fruits["automne"]= pomme ;
fruits["exotique"]= citron ;
fruits["été"]= raisin ;
fruits["printemps"]= fraise ;
```

Affichage des valeurs avec un tableau associatif :

```
document.write(fruits["automne"]) ;       //affiche la valeur pomme ;
document.write(fruits[2]) ;               // affiche la clé été ;
```

Résumé : Le tableau associatif permet de définir des clés au lieu des indices.

```
tableau[clé] = valeur ;          tableau[indice] = clé ;
```

c-Tableaux multidimensionnels

Dans les paragraphes vu précédemment, les données contenues dans un tableau n'avaient qu'une seule valeur. Il possible d'affecter plusieurs valeurs à une donnée grâce aux tableaux multidimensionnels.
Supposons que l'on souhaite créer un tableau 2 dimensions contenant 8 valeurs qui pourrait être représenté comme suit :

Indices de tableaux	0	1
0	Fraise	Printemps
1	Pêche	Eté
2	Raisin	Eté
3	Pomme	Automne

Il existe plusieurs méthodes pour les créer, en voici deux:

```
//Déclaration avec deux tableaux existants
var fruits = new Array( Fraise , Pêche , Raisin , Pomme ) ;
var saison = new Array( Printemps , Eté , Eté , Automne ) ;
var fruitsEtSaison = new Array(fruits,saison) ;

//Déclaration standard
var fruitsEtSaison = new Array([ Fraise , Pêche , Raisin , Pomme] ,[ Printemps
, Eté , Eté , Aut omne]) ;
```

Pour lire une valeur dans un tableau à plusieurs dimensions, il faut faire référence à sa position dans celui-ci en colonne et ligne :

```
var valeur = tableau [colonne][ligne] ;
```

Exercice **(11.html)**

Créer le tableau suivant, puis à l'aide de boucle généré un tableau HTML (voir la capture ci-après) :

	0	1	2	3
0	Saint Brieuc	Quimper	Rennes	Vannes
1	Lannion	Brest	Saint Malo	Lorient
2	Dinan	Morlaix	Redon	Pontivy

Attention la structure HTML est inversée.

	0	1	2
0	Cellule (0,0) = Saint Brieuc	Cellule (0,1) = Lannion	Cellule (0,2) = Dinan
1	Cellule (1,0) = Quimper	Cellule (1,1) = Brest	Cellule (1,2) = Morlaix
2	Cellule (2,0) = Rennes	Cellule (2,1) = Saint Malo	Cellule (2,2) = Redon
3	Cellule (3,0) = Vannes	Cellule (3,1) = Lorient	Cellule (3,2) = Pontivy

Code JS

```
var dpt22 = new Array("Saint Brieuc","Lannion","Dinan");
var dpt29 = new Array("Quimper","Brest","Morlaix");
var dpt35 = new Array("Rennes","Saint Malo","Redon");
var dpt56 = new Array("Vannes","Lorient","Pontivy");
var dptBZH = new Array(dpt22,dpt29,dpt35,dpt56);
document.write("<table border='1'>");
document.write("<tr><td> </td><td>0</td><td>1</td><td>2</td></tr>");
for(var i in dptBZH){
    document.write("<tr><td>"+i+"</td>");
    for(var j in dptBZH[i]){
        document.write("<td>Cellule ("+i+","+j+") = "+ dptBZH[i][j]+"</td>");
    }
    document.write("</tr>");
}
document.write("</table>");
```

d.Tableaux « objet »

Générer des tableaux objet permet de définir une classe générique applicable ensuite à tout moment, cela donne de la souplesse de programmation.
On souhaite créer un tableau qui enregistre les Rendez-vous avec la date, le nom de la personne et le motif du RV.
Il faut dans un premier temps définir le constructeur :

```
function rendezVous(_dateRV,_nomRV,_motifRV) {
      this.dateRV=_dateRV ;
      this.nomRV=_nomRV ;
      this.motifRV=_motifRV;
}
```

La fonction `rendezVous()` est un constructeur et permet de d'associer des paramètres.
Il suffit ensuite d'instancier :

```
var rvLundi = new rendezVous("31/05","M.Dupont","Achats");
var rvMardi = new rendezVous( 01/06 , Mme.Martin , Repas ) ;
```

Pour accéder aux valeurs, il suffit d'appeler la propriété correspondante :

```
document.write(rvLundi.motifRV) ; //Affiche le motif du Lundi soit Achats
```

Remarque :
L'opérateur `this` prend en considération l'objet actif à l'instant voulu dans l'exemple ci-dessus `rvLundi`. Il est donc nécessaire d'utiliser cet opérateur quand l'objet est non spécifié.

4.8 Les fonctions

La fonction est une partie de programme qui est répétée plusieurs fois dans un même programme.
Elles permettent de rendre le « code » plus lisible en découpant le programme principal.

a-Déclaration d'une fonction

La déclaration d'une fonction est assurée par le mot clé **`function`** qui est placé en tête suivi du nom et entre parenthèses, la liste des arguments attendus séparés par une virgule (sans arguments les parenthèses seront vides).
Le corps de la fonction (partie qui comprend les instructions) est encadré par des accolades « **`{ ....}`** ».
Les fonctions ont en général une valeur de retour désignée par le mot clé **`return`** suivi du résultat retourné au programme.

Exemple **(12.html)**

```
1.    for(var i=0;i<10;i++){
2.        document.write("Rayon :"+i+"m , surface ="+calculSurface(i)+"m²<br>");
3.    }
4.    function calculSurface(rayon){
5.        var surface = Math.round(2*Math.PI*Math.pow(rayon,2));
6.        return surface;
7.    }
```

Les lignes 1 à 3 , boucle appelant 10 fois la fonction de calcul de surface, i étant le rayon passé en paramètre
La ligne 4 permet de déclarer la fonction avec son nom « `calculSurface()` » et elle accepte un paramètre « `rayon` ». C'est une variable qui ne nécessite pas de déclaration avec var (elle est propre à la fonction).
La ligne 5 on effectue le calcul et le résultat est transférée dans la variable « `surface` ».
La ligne 6 renvoie le résultat à l'instruction qui a appelée la fonction.

```
Rayon :0m , surface =0m²
Rayon :1m , surface =6m²
Rayon :2m , surface =25m²
Rayon :3m , surface =57m²
Rayon :4m , surface =101m²
Rayon :5m , surface =157m²
Rayon :6m , surface =226m²
Rayon :7m , surface =308m²
Rayon :8m , surface =402m²
Rayon :9m , surface =509m²
```

La notion de portée des variables

Les variables sont de 2 types, soit globales, soit locale :

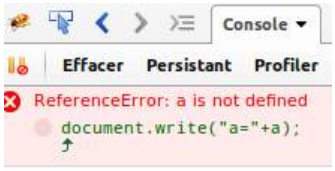
- Elles sont ***locales*** quand elles sont déclarées ***dans une fonction ou dans une boucle***. A la fin de celles-ci elles sont automatiquement détruites, dans l'exemple précédent c'est le cas de `surface`.
- Elles sont ***globales*** quand elles sont ***accessibles de partout dans le script*** (Attention à éviter au maximum).

Dans l'exemple ci-dessous, a est déclarée hors fonction et hors boucle, elle est donc globale.
`testPortee()` peut y accéder :

```
var a = 10;
testPortee();
function testPortee(){
        document.write("a="+a);
}
```

En revanche dans cet exemple: **(13.html)**

```
testPortee();
function testPortee(){
     document.write("a="+a);
}
function varLocale(){
     var a=10;
}
```



Capture firebug

Ici `a` est déclarée dans la fonction `varLocale()`, `testPortee()` ne peut y accéder d'où l'erreur dans firebug.

b-appel de fonction

Le navigateur n'exécute jamais une fonction si celle n'est pas appelée (cela ne signifie pas qu'il n'en connaît pas l'existence). Elles peuvent être placées dans l'en-tête de la page HTML sans que cela génère des erreurs du type objet inexistant (instructions y faisant référence avant que la page HTML ne soit chargée).
L'appel d'une fonction se fait en utilisant le nom employé dans la déclaration et en plaçant ou non des paramètres entre parenthèses.

Soit la fonction :

```
function faireQuelqueChose(parametre1,parametre2){
// code de la fonction
      return valeur ;
}
```

- appel direct dans le code, on attend pas de valeur de retour

```
faireQuelqueChose(valeur1,valeur2) ;
```

- la fonction s'exécute avec deux paramètres puis renvoie une valeur grâce à return, valRetour récupère cette valeur

```
var valRetour = faireQuelqueChose(valeur1,valeur2) ;
```

La valeur de retour est directement testée, elle peut être de type booléen (ici true) ou numérique/chaîne de caractères (avec des " si la chaîne est directement testée) :

```
if(faireQuelqueChose(valeur1,valeur2)) { …}
if(faireQuelqueChose(valeur1,valeur2)==valeurTest) {…}
```

On procède de la même façon avec `while/do...while`.

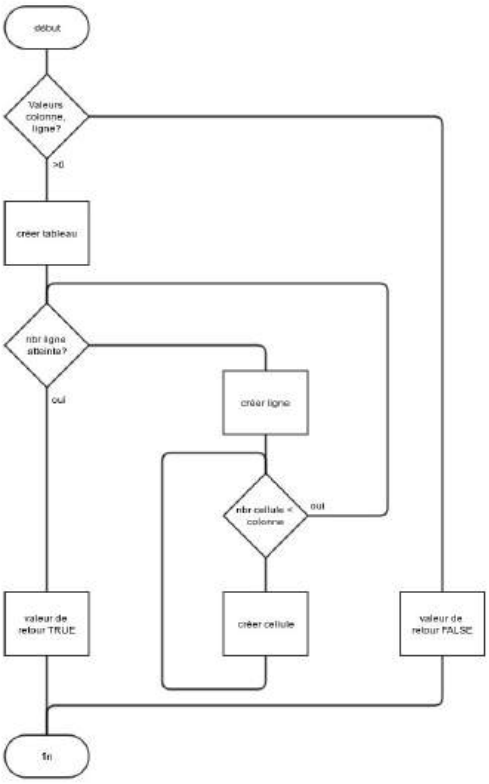
Appel en tant que paramètre d'une autre fonction :
```
document.write(faireQuelqueChose(valeur1,valeur2)) ;
```
Appel à partir d'une fonction :
```
function faireAutreChose(paramètre3){
      … // code de la fonction
      var valRetour=faireQuelqueChose(valeur1,valeur2) ;
      … // poursuite de la fonction
}
```
Remarque : N'oubliez pas la portée des variables, les appels de fonction sont souvent le siège d'erreurs de ce type. L'instruction `return` est alors d'un grand secours !

Exercice :**(14.html)**
On souhaite créer une fonction qui permet de tracer un tableau à partir de 2 paramètres colonne et ligne. La saisie du nombre de colonne et de ligne se fait à partir d'un formulaire, le tableau est généré dans un bloc `div` spécifique après avoir cliqué sur le bouton. Avant de dessiner, on testera la validité de la saisie (valeur positive entière uniquement). La valeur de retour de la fonction attestera de la validité (`true` = ok, `false` = erreur affichée dans une fenêtre d'alerte).

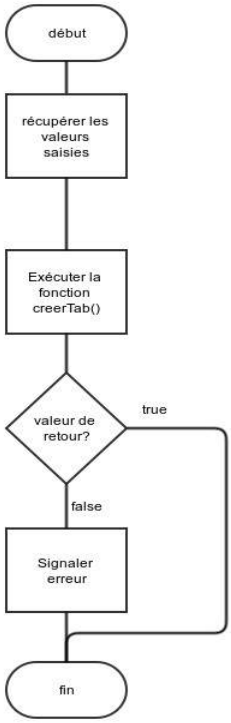
Prototypes des Fonctions à développer :
- `dessine()` , fonction exécutée à partir du bouton, elle lancera la fonction `creerTab()` et testera sa valeur de retour.
-`creerTab(colonne,ligne)`, fonction qui génère le tableau dans le `div` dédié. Elle testera les valeurs passées en paramètre et renverra true ou false à `dessine()`.

1- Créer l'Algorithme de creerTab() et dessine()

creerTab()



dessine()



2-A partir du code HTML ci-dessous, créer le script JS qui comprendra les deux fonctions :

```
<body>
   <form name="formulaire">
       <fieldset><legend>Dessine moi un tableau</legend>
          <label>Nb Colonne :</label><input type="text" name="col" size="5">
          <label>Nb Ligne   :</label><input type="text" name="lgn" size="5">
          <input type="button" value="Dessine un Tableau" onclick="dessine();">
        </fieldset>
   </form>
   <div id="zone"></div>
   <script>
      function creerTab(colonne,ligne){
        if(isNaN(parseFloat(colonne)) || isNaN(parseFloat(ligne))){
            return false;
        }else if(parseInt(colonne) > 0 && parseInt(ligne) > 0){
            colonne=parseInt(colonne);
            ligne=parseInt(ligne);
            document.getElementById("zone").innerHTML="";
            var tab = "<table border='1'>";
            for(var i=0;i<ligne;i++){
              tab+="<tr>";
              for(var j=0;j<colonne;j++){
                tab+="<td>"+i+","+j+"</td>";
              }
              tab+="</tr>";
            }
            tab+="</table>";
            document.getElementById("zone").innerHTML=tab;
        }else{
            return false;
        }
        return true;
      }
      function dessine(){
         var c=document.formulaire.col.value;
         var l=document.formulaire.lgn.value;
         if(!creerTab(c,l)){
              alert("Erreur, impossible de tracer un tableau");
         }
      }
   </script>
</body>
```

Chapitre 7
JavaScript – DOM

Le DOM (**D**ocument **O**bject **M**odel) est une interface de programmation pour tous les documents HTML et **XML**[1]. Elle est définie par le W3C (http://www.w3.org/DOM/). Il s'agit d'une structure représentant les documents tels qu'ils sont interprétés par le navigateur en mémoire. Il est alors possible de modifier sa structure (HTML et CSS) via les méthodes de l'**API**[2] JavaScript.

1.Principe du DOM

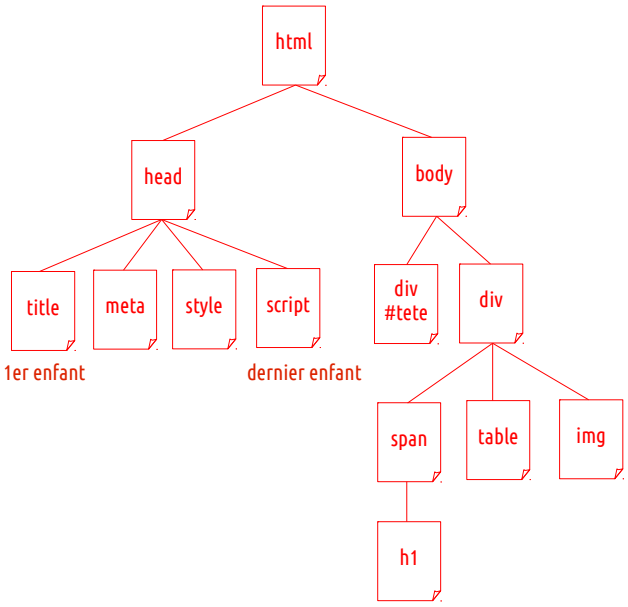
Au chargement du document HTML, la page est définie comme un arbre respectant la hiérarchie et l'encapsulation des balises. Chaque élément (balises entre autres) est vu comme un objet et est associé à un nœud. Par analogie avec l'arbre biologique, un nœud peut être assimilé à une jonction entre des branches.
Les nœuds se distinguent alors par leur lien de parenté (relation parent/enfant/frère et orphelin), il est alors possible d'accéder aux propriétés et méthodes de chacun, d'ajouter ou supprimer des nœuds etc. .

Représentation graphique du DOM

Code HTML

```
<!doctype html>
<html>
  <head>
    <title>Structures DOM</title>
    <meta charset="utf-8">
    <style rel="stylesheet" href="style.css">
    <script src="script.js">
        document.onload=init() ;
    </script>
  </head>
  <body>
    <div id="tete"></div>
    <div>
        <span><h1>Blog TECH</h1></span>
        <table>
          …
        </table>
        <img src="partenaires.png">
    </div>
  </body>
</html>
```

Arborescence DOM



Remarque :
La représentation sous forme d'arbre DOM est possible avec Firefox grâce au module « DOM Inspector ».

1 http://fr.wikipedia.org/wiki/Extensible_Markup_Language
2 Application Programming Interface (en) – Interface de programmation d'application (fr)

2.Terminologie

- Un arbre est la structure hiérarchique qui relie les nœuds entre eux. Sa « racine » correspond au document et son « tronc » à la page HTML. Les nœuds constituent les « branches » du réseau d'éléments.
- Un nœud (`node`) correspond au composant de base de l'arbre. Il existe plusieurs types de nœuds, les principaux sont : élément (type 1), attribut (type 2) ou texte (type 3).
- Le parent (`parentNode`) d'un nœud est toujours unique. C'est celui dont est issu le nœud courant.
- Les enfants (`childNodes`) sont les nœuds issus du nœud courant. Parmi les nœuds enfants on distingue le « premier enfant » (`firstChild Node`) et le dernier enfant (`lastChild Node`) cas d'une liste déroulante par exemple avec les différentes <option>.
- Les nœuds Frères (`sibling nodes`) sont les nœuds qui ont le même parent que le nœud courant. On distingue ceux placés juste avant (`previousSibling`) et juste après (`nextSibling`) du nœud courant .

Exercices
A partir de l'exemple précédent :
1-On se positionne sur le node « `head` » donnez :

parentNode => `html`

childNodes => `title , meta, style et script`

2-Complétez le tableau ci-dessous :

Nodes	Type 1	Type 2	Type 3	Sibling Nodes
`Title`			Structures DOM	Style, meta(NS*),script
`Style`		Rel (stylesheet) href (style.css)		Meta(PS),title,script(NS)
`Script`		src (script.js)	Document.onload=init()	Title,style(PS),meta
`Div`	Span, table,img			div#bandeau(PS), img (NS)
`Span`	h1			table(NS)

*PS : previousSibling NS : nextSibling

3.Méthodes JavaScript d'accès au DOM

La manipulation du DOM repose principalement sur l'accès au nœud (node). L'API permet de balayer l'arbre et de retrouver l'élément à partir de critères précis. Il suffit ensuite de l'associer à un objet JavaScript pour le manipuler.

3.1-Critères d'accès et de sélections, méthodes associées

a-l'identifiant « id »

Nous avons utilisé précédemment, l'attribut « id » en HTML pour associer des propriétés CSS. Il s'avère que l'id sert aussi à pointer le nœud à partir de la méthode DOM « **`getElementById("id")`** ». C'est une méthode de l'objet « **`document`** ».

HTML

```
<img id="monImage" src="image.png">
```

JSDOM

```
var image = document.getElementById("monImage") ;
```

image est une variable JavaScript que l'on associé à l'objet HTML dont l'id est « monImage », il est alors possible de manipuler ce nœud via sa variable :

```
alert(image.src) ;  // renvoie la valeur de src c'est à dire « image.png ».

image.src="autreImage.png" // on change l'image directement via le DOM
```

Exercice **(1dom.html)**
Soit la page HTML ci-dessous. On souhaite afficher par alternance deux images :
- Par défaut, c'est image1.png
- au passage de la souris c'est image2.jpg

Modifiez le code ci-dessous :

```
<!doctype html>
<html>
    <head>
       <title>Exemples DOM - 1</title>
       <meta charset="utf-8">
    </head>
    <body>
       <img  id="monImage" src="images/image1.png" onmouseover="change(true)"
onmouseout="change(false)">
       <script>
           function change(action){
               var image = document.getElementById("monImage");
               if(action){
                     image.src="images/image2.jpg";
               }else{
                     image.src="images/image1.png";
               }
            }
        </script>

    </body>
</html>
```

b.A partir du type de balise

L'arbre DOM est construit à partir de nœud correspondant aux balises de la page HTML. Il est possible de pointer sur une balise en particulier en faisant référence à son type grâce à la méthode « **`getElementsByTagName()`** ». Elle aussi est associée à l'objet « **`document`** ».

```
var image = document.getElementsByTagName("img") ;
```

Particularité de cette méthode
On n'accède pas nommément à une balise contrairement à la précédente méthode mais a un ensemble de balise de même type. De fait, elle renvoie une « nodeList » ce qui est équivalent à un tableau. Dans l'exemple précédent, on manipule l'image de la façon suivante :
- `image.item(x)` ou `image[x]`

Exercice
Même chose que précédemment, mais avec en plus deux images que l'on inverse l'une par rapport à l'autre.

```
<!doctype html>
<html>
   <head>
      <title>Exemples DOM - 1</title>
      <meta charset="utf-8">
   </head>
   <body>
      <img src="images/image1.png" onmouseover="change(true)"
onmouseout="change(false)">
      <img src="images/image2.jpg" onmouseover="change(true)"
onmouseout="change(false)">
     <script>
         function change(action){
            var image = document.getElementsByTagName("img");
            if(action){
                image[0].src="images/image2.jpg";
                image[1].src="images/image1.png";
            }else{
                image[0].src="images/image1.png";
                image[1].src="images/image2.jpg";
            }
         }
      </script>
  </body>
</html>
```

c.A partir de la classe

La classe CSS est aussi un critère de sélection d'un nœud du DOM et pour cela on utilise « `getElementsByClassName()` » . Elle aussi est associée à l'objet « **`document`** » et renvoie une « nodeList » dans un tableau :

```
var image = document.getElementsByClassName("classe") ;
```

On y accède :
- `image[x]` ou `image.item(x)` ;

d.En fonction des sélecteurs CSS

HTML5 nous fournit deux nouvelles méthodes qui permettent d'accéder aux sélecteurs CSS :

- querySelector("selecteur"), retourne le premier sélecteur qu'elle trouve
- querySelectorAll("selecteur"), retourne tous les sélecteurs

Exemple

```
var objHTML = document.querySelector(" div a ") ;
```
Le nœud pointé est le premier lien dans le premier div

```
var objHTML = document.querySelector("div a") ;
```
Tableau de nœuds correspondant à tous les liens des tous les div

Elles sont associées à l'objet « **`document`** ».

3.2-Propriétés du DOM

Attention à ne pas confondre avec les propriétés des nœuds (comme src pour img par exemple), ici les propriétés sont relatives à la manipulation du DOM, accès à un nœud frère par exemple, déterminer le nœud parent etc.

Propriétés	**Description**
`innerHTML`	Contenu d'un élément HTML
`outerHTML`	Contenu HTML et l'élément lui même
`textContent ou innerText`	Contenu texte simple
`style`	Ensemble des propriétés CSS d'un élément
`style.*`	* correspond à une propriété CSS, attention elles ne s'écrivent pas de la même façon en JavaScript. Voir **correspondance CSS-JavaScript**[3].
`tabindex`	Index de tabulation
`id`	Valeur de l'attribut id
`lang`	Valeur de l'attribut lang
`title`	Valeur de l'attribut title
`tagName`	Nom de la balise
`className`	Nom de la classe CSS
`classList`	Enumération des classes
`childNodes`	Liste des nodes enfants d'un élément
`firstChild`	Premier enfant d'un élément
`lastChild`	Dernier enfant d'un élément
`parentNode`	Noeud parent
`nextSibling`	Noeud frère suivant
`prevSibling`	Noeud frère précédant
`attributes`	Liste des attributs d'un noeud

3 http://www.w3schools.com/jsref/dom_obj_style.asp

3.3-Méthodes de manipulation du DOM

Ces méthodes permettent de modifier le DOM, soit en modifiant l'élément visé en lui ajoutant/retirant des attributs ou en ajoutant/supprimant un nœud.

Modification

Liste des méthodes permettant de créer/supprimer ou modifier des attributs d'un nœud du DOM:

Propriétés	Description
`getAttribute("attr")`	Renvoie la valeur de l'attribut attr
`setAttribute("attr", val)`	Modifie la valeur de l'attribut attr avec la valeur val
`removeAttribute("attr")`	Supprime l'attribut attr
`focus()`	Donne le focus à un élément
`Blur()`	Retire le focus

Création et suppression

Liste des méthodes permettant de créer et supprimer des nœuds dans le DOM :

- `createElement()`, crée dynamiquement un nœud DOM :

```
var element = document.createElement("div") ;
```

- `appendChild()`, insère un nœud enfant dans un nœud parent :

```
var enfant = document.createElement("img") ;
document.parent.appendChild(enfant) ;
```

- `removeChild()`, supprime un enfant dans un parent :

```
var enfant = document.querySelector("div#box>a") ;
document.getElementById("box").removeChild(enfant) ;
```

- `insertBefore()`, place un nœud avant un autre :

```
var newLi = document.createElement("li") ;
var listeParente = document.getElementsByTagName("ul") ;
listeParente[0].insertBefore(newLi,listeParente[0].lastChild) ;
```

HTML :

Avant

```
<ul>
        <li>premier li</li>
        <li>deuxième li</li>
        …
        <li>Dernier li</li>
</ul>
```

Après

```
<ul>
        <li>premier li</li>
        <li>deuxième li</li>
        …
        <li>Nouvel li créé</li>
        <li>Dernier li</li>
</ul>
```

- `createTextNode()`, crée un nœud texte (type 3) :

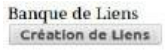
```
var texte = document.createTextNode("Node texte de type 3") ;
document.querySelector("body").appendChild(texte) ;
```

Exercice

On souhaite réaliser une banque de lien, l'utilisateur saisie l'URL et le texte associé pour créer le lien. Pour cela on utilise la méthode prompt(). Elle permet d'afficher une fenêtre de saisie et de la récupérer dans une variable :

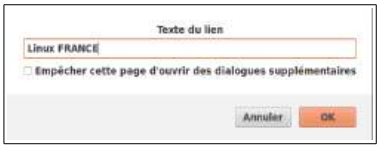
```
var reponse = window.prompt("Phrase ou question","valeur par défaut") ;
```

Chaque lien sera affiché les uns en dessous des autres, les erreurs de saisie doivent signifiées (champs vides ou url = http://) :



1

2



3

4

(plusieurs liens saisis)

```
<!doctype html>
<html>
    <head>
       <title>Exemples DOM - 3</title>
       <meta charset="utf-8">
    </head>
    <body>
       Banque de Liens
       <div id="banque">
       </div>
       <input type="button" value="Création de Liens" id="btn">
      <script>
         var bouton = document.getElementById("btn");
         bouton.onclick=function(e){
           var box = document.getElementById("banque");
           var url=window.prompt("Saisir URL:","http://");
           if(url=="" || url=="http://"){
              alert("Saisie incorrecte!");
           }else{
              var lien = window.prompt("Texte du lien","Lien");
if(lien==""){
                 alert("Erreur de saisie");
              }else{
                 var newA = document.createElement("a");
                 newA.setAttribute("href",url);
                 newA.setAttribute("target","_blank");
                 var texte=document.createTextNode(lien);
                 newA.appendChild(texte);
                 box.appendChild(newA);
                 box.innerHTML+="<br>";
               }
            }
         }

       </script>
   </body>
</html>
```

4.Gestionnaire d'événements

Nous avons vu au chapitre 6 qu'un gestionnaire d'événements permet d'ajouter de l'interactivité dans les applications web (C6 - §3.2 page 3)
Le DOM ne déroge pas à la règle, il offre la possibilité de gérer les événements (liste des événements pris en charge en annexe).

Rappel

Dans une balise :

```
<img id="monImage" src="image.png" onclick="fonction JavaScript">
```

Événement « onclick » lance la fonction JS au « click » sur l'image.

Avec DOM :

```
var image = document.getElementById("monImage") ;
monImage.onclick=function(e){
      … // code de la fonction exécutée dès la détection de l'événement
}
```

On remarque :

- que la fonction n'est pas nommée
- que la valeur passée en paramètre (e) est associée à l'événement, certains événements disposent d'attributs comme la position du curseur de la souris pour « onmousemove ».
- qu'il n'est plus nécessaire d'écrire l'événement dans la balise

Exercice
Écrire la version DOM du gestionnaire d'événements donné ci-dessous :

HTML

```
<img id="monImage" src="image.png" onclick="change(1)" onmouseover="change(2)">
```

DOM JS

```
var image = document.getElementById("monImage") ;
image.onclick=function(e){
      change(1) ;
}
image.onmouseover=function(e){
      change(2) ;
}
```

SGBD

Chapitre 9 – SGBD
Chapitre 10 - SQL

BTS Systèmes Numériques
Informatique & Réseaux

FÉLIX
LE DANTEC
du secondaire au supérieur

Chapitre 9
Introduction aux bases de données

L'accès rapide à l'information via les réseaux intra et internet a nécessité la création de systèmes d'organisation des données. Si l'on souhaite rechercher des personnes portant le même nom dans un département avec un annuaire, on peut aisément imaginer que cela prendrait beaucoup de temps.
Face à de tels problèmes l'informatisation du stockage des données est devenue obligatoire.

1.Base de données

Une base de données est un ensemble d'informations stockées sur un support et doté d'une certaine organisation.

Elle doit répondre à des contraintes précises :

- occupation mémoire la plus restreinte possible
- redondances inexistantes
- les mises à jour et suppression de données doivent laisser la base intègre et ne pas créer d'incohérences
- la recherche d'informations doit être la plus rapide et sûre

2.Modèle entité/association (modèle Conceptuel de Données - MCD)

Le modèle entité/association permet la modélisation abstraite d'une base de données. Il utilise un ensemble de conventions et de représentation graphique pour modéliser les différents concepts contenus dans une information et des liens qui existent entre eux.

2.1-Les entités

On appelle entité une représentation d'un ensemble d'objets réels ou abstraits qui ont des caractéristiques communes.
On distingue deux sortes d'entités :

- les entités fortes, qui ne dépendent pas de l'existence d'une autre entité
- les entités faibles, qui dépendent d'une autre entité.

Avec UML on les modélise par des rectangles :

2.2-Les attributs

Cela représente des caractéristiques particulières comme le `nom`, le `prénom`, l'adresse `email` de l'entité `utilisateur` par exemple.

Chaque entité dispose d'au moins un attribut appelé **identifiant** qui permet de la distinguer des autres, il doit être unique. Des utilisateurs peuvent avoir le même le nom et cela pourrait plus compliquer à les différencier. En règle générale, la clé primaire correspond à un identifiant numérique et unique pour chaque entité.

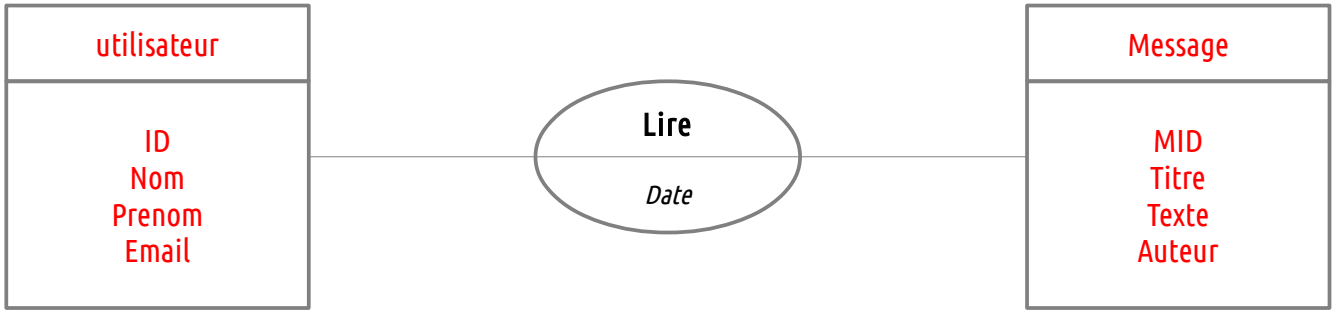
2.3-Les associations

Le concept d'association permet de représenter le lien existant entre deux ou plusieurs entités. Une association entre deux entités est dite « binaire ». Au-delà on dit qu'une association est « n-aire ».

Il est fortement conseillé de créer des associations binaires et à décomposer celles qui seraient ternaires.

Une association est généralement désignée par un verbe d'action, si on prend par exemple «`Utilisateur lit un message` », « `lit` » est l'association entre deux entités « `utilisateur` » et «`message` ».

Elle est modélisée graphiquement par une ellipse dans laquelle on spécifie son nom et ses attributs :

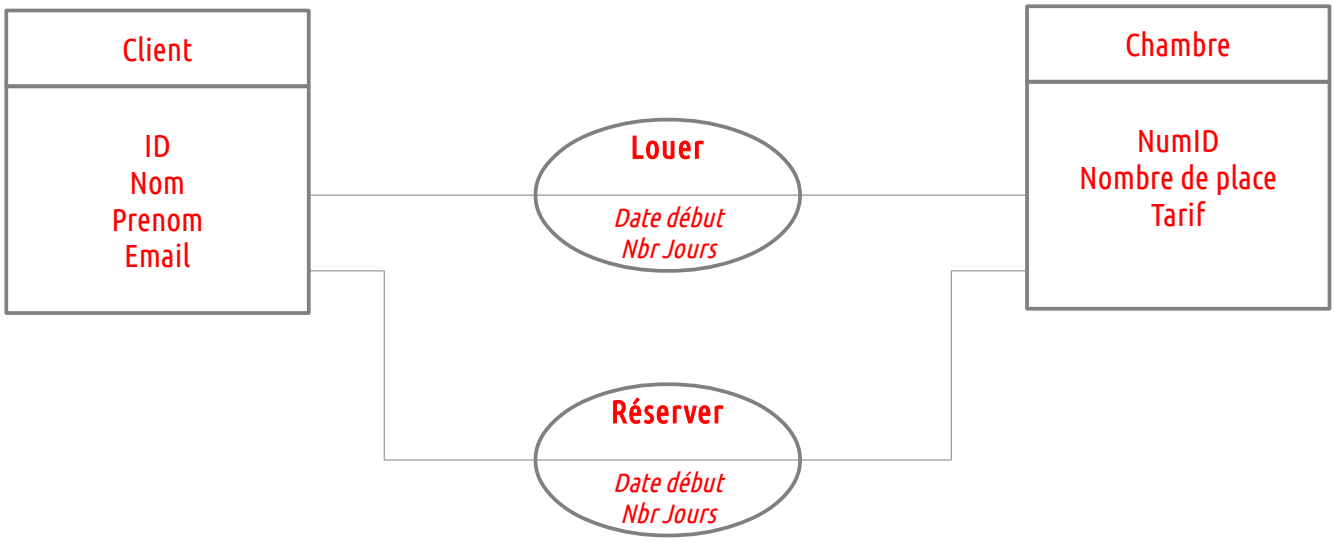


Dans l'exemple ci-dessus, « `Date` » est un attribut de l'association « `lire` ».
Une association est dite réflexive quand elle relie une entité à elle-même.

Exercice

On cherche à modéliser les locations de chambre dans un hôtel.
1-On vous demande de modéliser l'association des entités client ↔ chambre. Vous rechercherez les attributs des deux entités et l'association (éventuellement ses propres attributs).

2- Est-il possible de rajouter un autre association ?



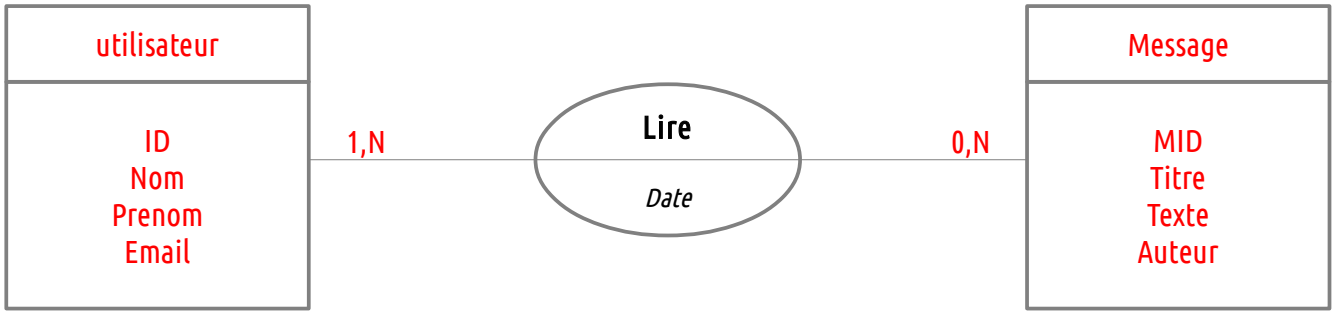
2.4-Cardinalités

La cardinalité mesure le nombre de « liens » qu'il est possible d'avoir entre deux entités.
Par exemple, un Utilisateur peut lire un seul et unique Message en même temps, alors qu'un Message peut être lu en même temps par plusieurs Utilisateurs.
On peut donc considérer que la cardinalité n'est pas forcément la même suivant le sens de l'association.

On distingue des cardinalités minimales et maximales :

- La cardinalité minimale mesure le nombre minimal de participations d'une entité dans l'association. Une personne peut lire un seul message au minimum donc la cardinalité minimale est 1, point de vue Message il peut être lu par personne donc la cardinalité est 0
- La cardinalité maximale mesure le nombre maximal de participations d'une entité dans l'association. Une personne peut lire « N » messages autant qu'en compte la base et de même que l'on peut avoir « N » utilisateurs pour lire le message.

De fait on note la cardinalité suivant ces deux valeurs :

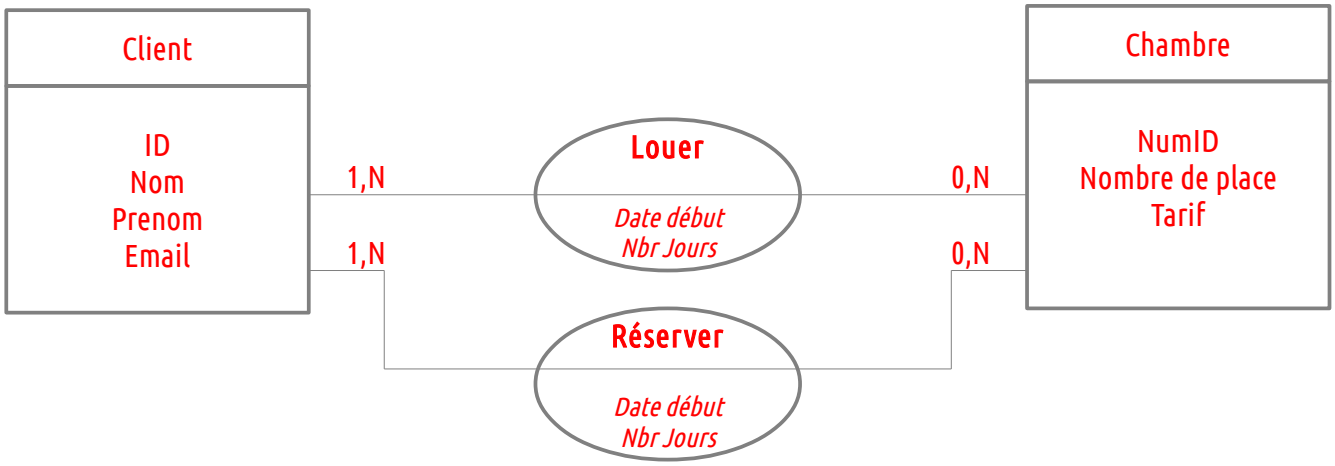


Cardinalités caractéristiques :

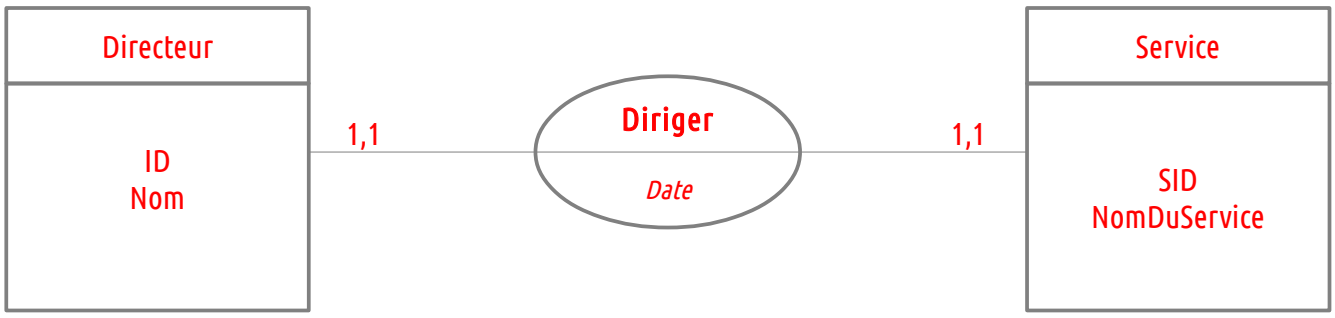
- Association entre la brosse à dent et son propriétaire, elle est de type « 1,1 », on parle alors de cardinalité « un à un »
- Association entre livre et auteur, elle est de type « 1,n », « un à plusieurs »
- Une personne est célibataire ou mariée à une autre personne, de type « 0,1 », on parle alors « optionnel »
- Entre un appartement et locataire, de type « 0,n », « 0 à plusieurs »

Exercices :

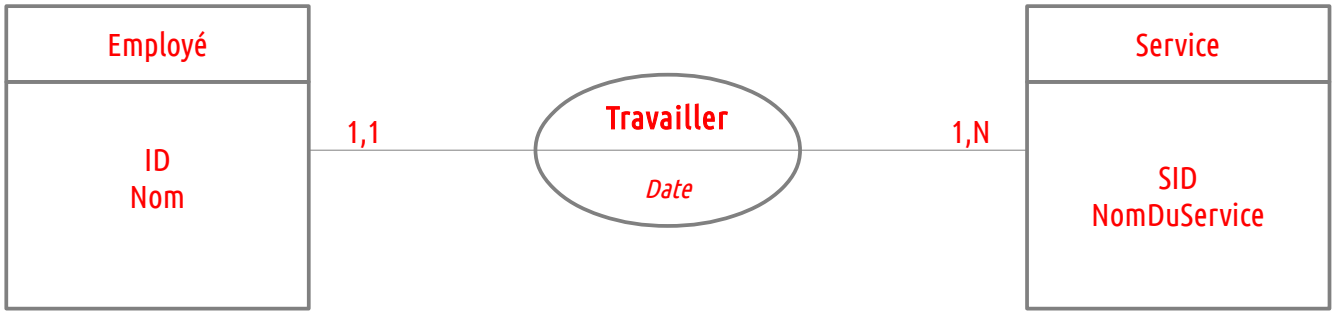
Ex1-Reprendre l'exercice page précédente en rajoutant les cardinalités.



Ex2- Un ministère comprend plusieurs services. Chaque service est dirigé par une seule personne. Pour tenir compte des changements de directeur, l'association « `diriger` » doit prendre en compte la `date` de nomination du directeur. De plus on a besoin de connaître le `nom` et l'`ID` du directeur, de même que l'on doit enregistrer le `nom du service` et son `id`. Représenter les entités, associations et déterminer le nombre de cardinalités.

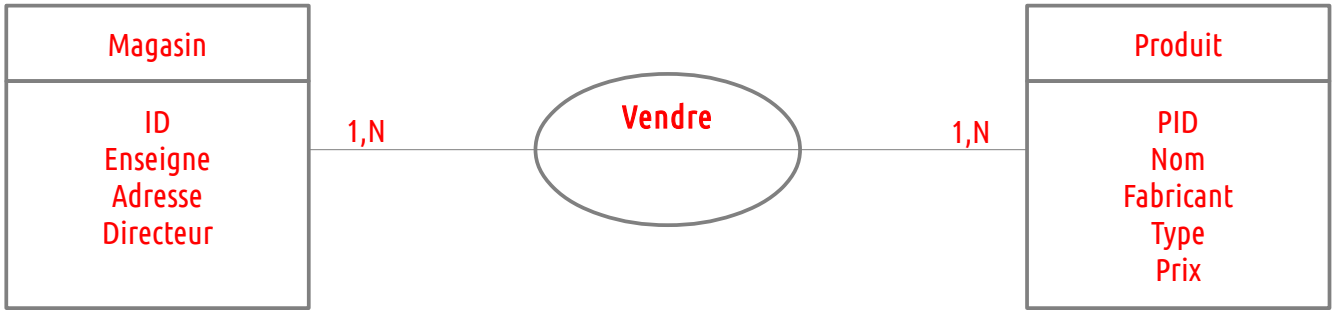


Ex3-Soit le même ministère qu'à l'exercice 2 mais en reprenant l'association entre tous les employés du ministère et les services dans lesquels ils travaillent.

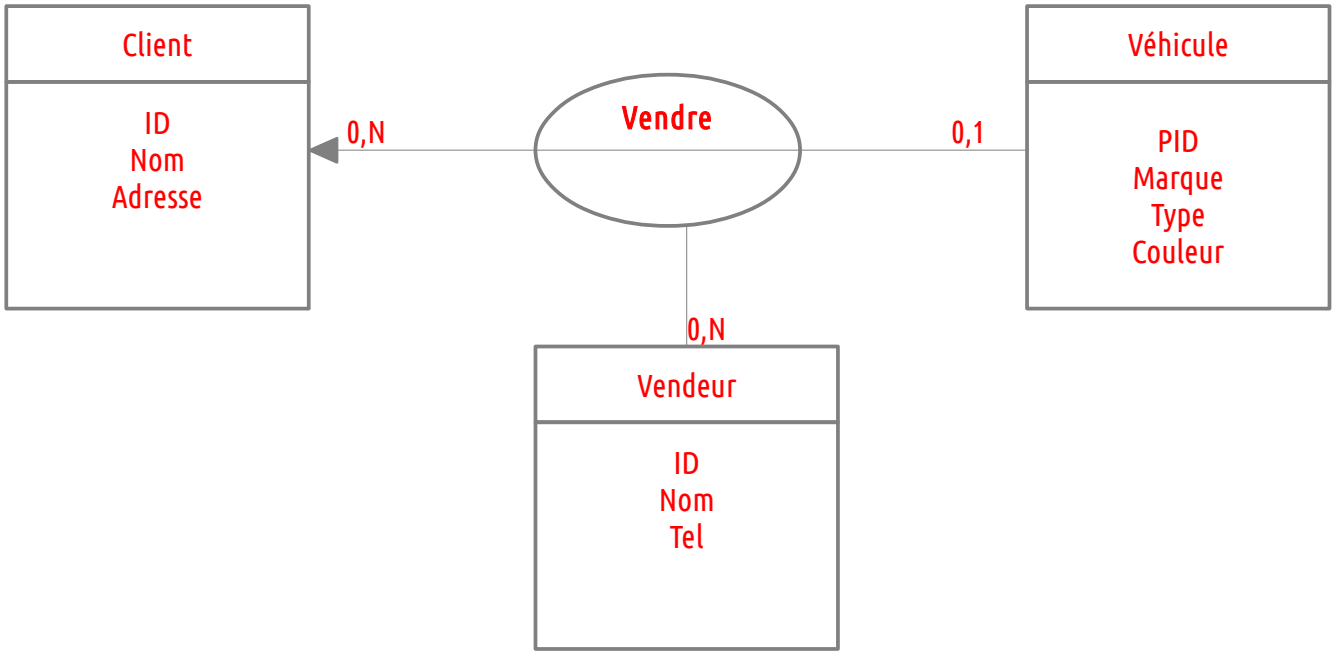


Un employé travaille dans un 1 service (il ne peut pas ne pas faire partie d'un service), un service compte au moins 1 employé.

Ex4-Pour suivre l'évolution des prix, on recense un nombre de produits vendus dans un ensemble de magasins. Recherchez cardinalités, attributs ainsi que l'association entre magasin et produit.



Ex5-On souhaite modéliser un garage de voiture et mettre en évidence l'association « vendre » entre les clients, les vendeurs et les véhicules.
Tracez la représentation graphique du modèle entités/associations, donnez toutes les entités, les attributs ainsi que les cardinalités.



Un vendeur peut vendre aucune ou plusieurs véhicule à aucun ou plusieurs clients. Les véhicules sont ou ne sont pas vendus.

Quelle est la particularité de cette association ? Est-on limité par ce modèle ?

C'est une association ternaire, mettant en évidence l'association de plus de 2 entités. Elle ne peut se lire que dans un sens (lié à l'association).

3.Modèle Relationnel

Ce modèle s'appuie sur le MCD pour être employé dans les SGBDR (Système de Gestion de Base de Données Relationnels). On retrouve une structure similaire au niveau des entités, en revanche les associations sont elles aussi caractérisées en tant qu'entité. Elles sont alors considérées comme des relations.
Les dénominations changent :

- une entité devient une table
- les attributs d'une entité deviennent des colonnes de la table
- l'identifiant unique d'une entité devient une clé primaire (clé de table)
- Pour une association binaire ayant des cardinalités max 1.1, une des tables reçoit comme attribut supplémentaire une copie de la clé primaire (Voir fig 1 page suivante)
- Pour une association binaire ayant des cardinalités maximales 1.N (relation 1.1 – 1.N par exemple), la table représentant l'entité de cardinalité 1.1 reçoit la clé de l'autre entité (voir fig2)
- Pour une association binaire N.M (1N-1N ou 0N-1N par exemple), l'association est toujours traduite par une table. La clé primaire de cette nouvelle table est la concaténation des clés primaires des entités reliées par cette association (voir fig3)

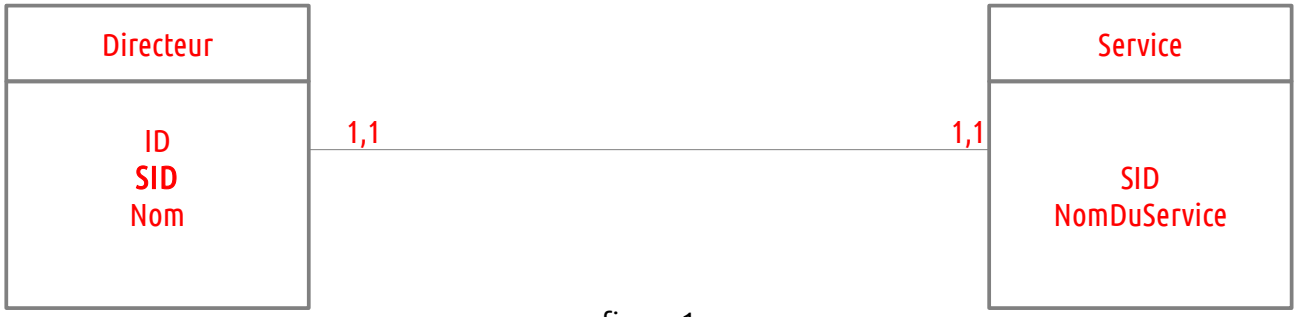


figure 1

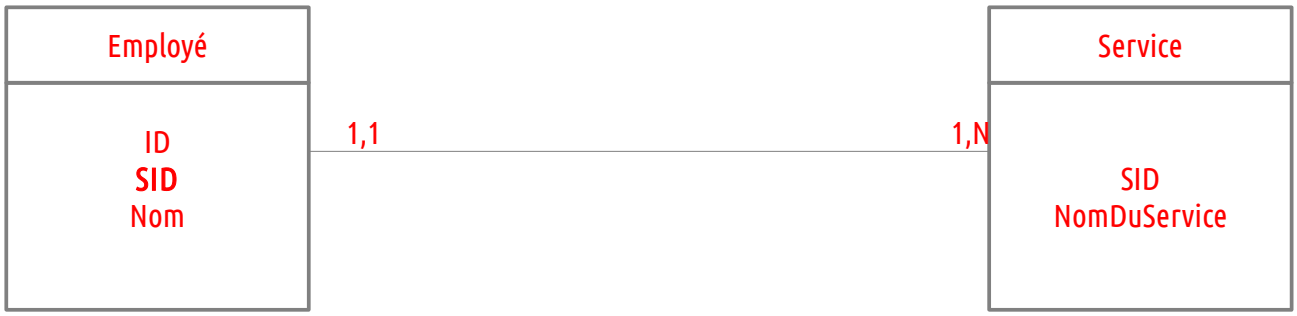


figure 2

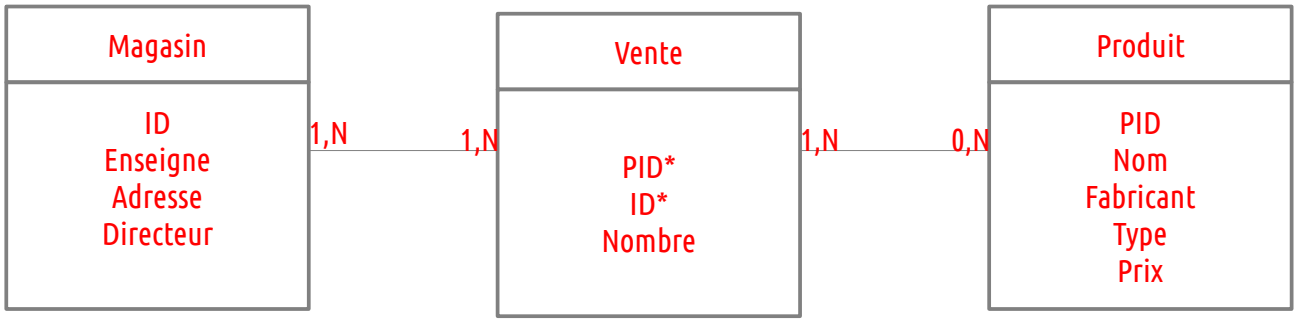
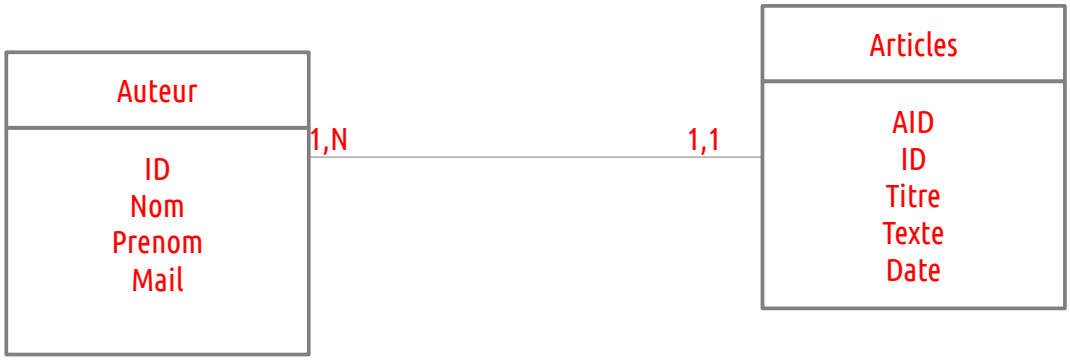


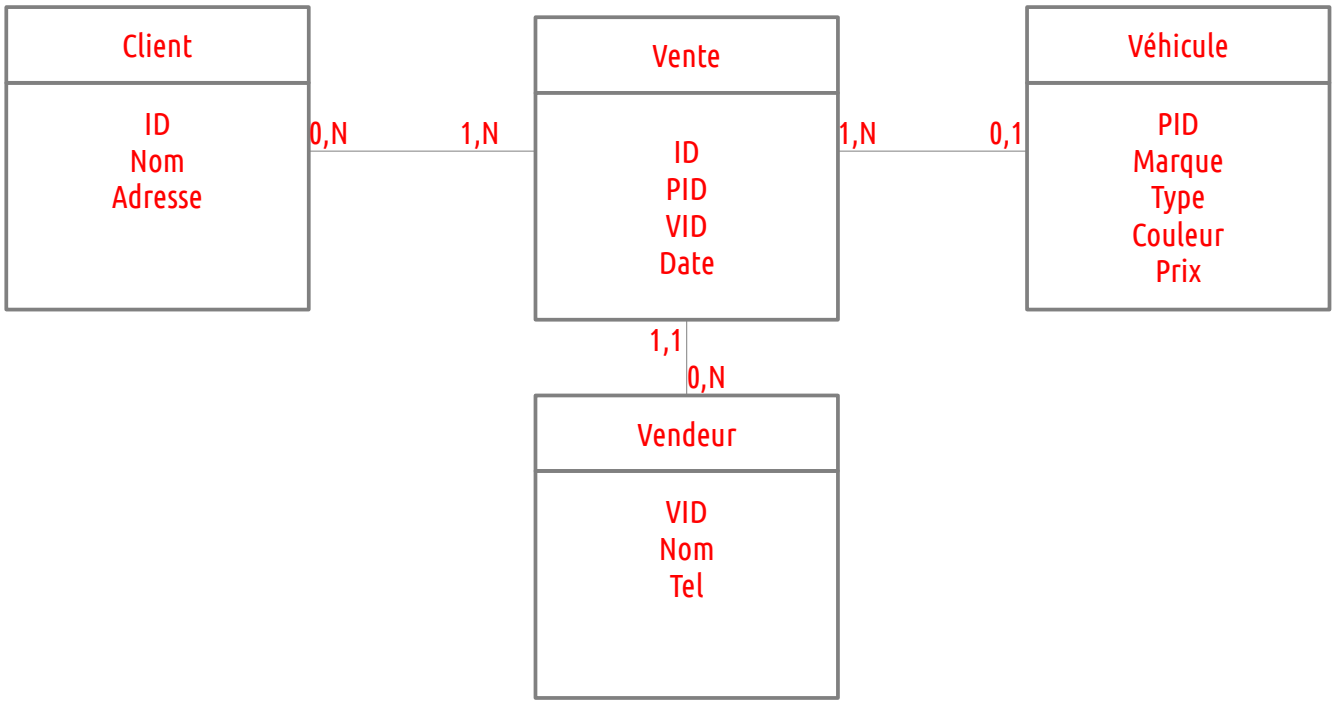
figure 3

Exercices

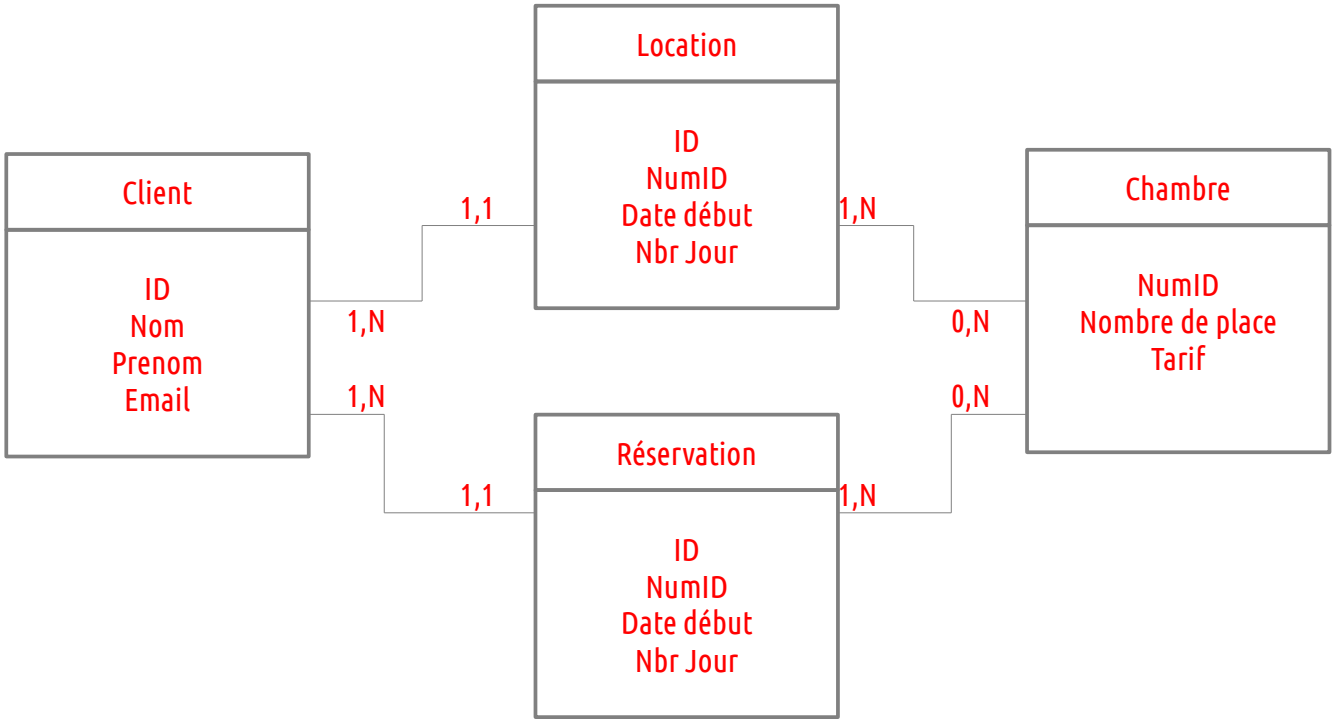
Ex1-Faire le modèle relationnel entre les articles publiés sur un site et leurs auteurs respectifs. Vous indiquerez les tables, colonnes ainsi que les cardinalités.



Ex2-refaire l'exercice 5 page 5 avec le modèle relationnel.



Ex3-refaire l'exercice 1 page 3 en utilisant le modèle relationnel



Chapitre 10
SQL

Créé par IBM en 1970, le SQL (Structured Query Language) est le langage de manipulation de données devenu standard dans la plupart des Systèmes de Gestion de Bases de Données Relationnelles (SGBDR).

Ce langage a fait l'objet de normalisations successives de la part de l'ANSI (American National Standards Institute). Toutefois chaque SGDB adapte SQL en utilisant certaines de ses fonctionnalités et en ajoutant d'autres non standards.

1.Principe

SQL n'est pas un langage procédural, il n'intègre pas de structure de contrôle (if, for, while etc.) en revanche il manipule les ensembles.
Les instructions SQL sont dissociables en 4 familles :

- Manipulation des données et des tables avec les commandes SELECT, INSERT, DELETE et UPDATE
- Création et définition des tables, organisation des données, index etc.
- Définition des droits et permissions des utilisateurs
- Définition de début et fin de transaction et compatibilité avec d'autres langages comme le C/C++,JAVA ou PHP par exemple

2.Base, Tables et Champs

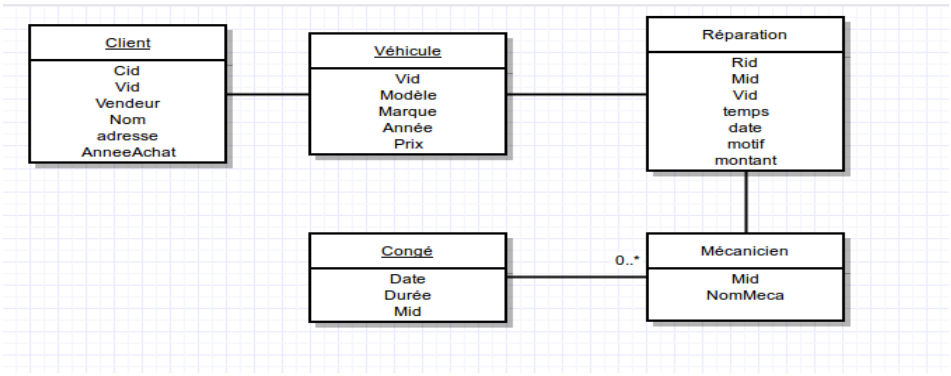
La structure SQL repose sur la façon dont sont vues les données :

- La base correspond à l'ensemble qui contient tout, si on manipule des données, quelque soit leur origine, elles appartiennent toutes à la base
- Les tables décomposent la base en sous-ensembles, les données sont intégrées aux tables
- Les champs, éléments support des données

Exemple :

Base -> `Garage`
Tables -> `Clients, Vehicules, Reparations, Mecaniciens`
Champs de la table Clients : cId, Nom, Adresse, Telephone, anneeAchat, vendeur ,vId
Champs de la table Vehicules : vId, Modele, Marque, Moteur, Annee , prix
Champs de la table Reparations : rId,mId,vId,temps,date,motif,montant
Champs de la table Mecaniciens : mId,nomMeca
Champs de la table conges : date,duree,mId

Les champs doit avoir un typage prédéfini, en revanche la valeur par défaut n'est pas obligatoire et de fait ils sont positionnés à la valeur NULL.
Tracez le modèle relationnel de cette base :



3.Les requêtes type Recherche

3.1.Recherche simple de valeur

Permet de rechercher toutes les données dans la table Clients. * est considéré comme un joker.

```
SELECT * FROM Clients ;
```

Récupère les données des champs nom et Adresse de la table Clients.

```
SELECT Nom,Adresse FROM Clients ;
```

Distinct évite d'avoir de la redondance dans la sélection.

```
SELECT distinct Nom,Adresse FROM Clients ;
```

3.2-Recherche avec clause `WHERE`

`WHERE` permet d'inclure des condition et restriction dans la requête.

```
SELECT modele,marque FROM vehicule WHERE moteur='diesel';
```

La clause WHERE accepte des expressions logiques comme AND,OR mais aussi des opérateurs de comparaison numériques comme < et > par exemple.

```
SELECT vld FROM Clients WHERE vendeur='dupont' AND anneeAchat='2009';
SELECT Modele FROM vehicule WHERE annee >2009 ;
```

L'équivalent pour comparer des chaines nécessite l'emploi des instructions Between ... AND

```
SELECT vld FROM Clients WHERE anneeAchat Between '2009' AND '2010';
```

`Between ... and` dispose d'un contraire `Not between ... and` .

Recherche dans une table à partir d'un groupe de valeur avec l'opérateur IN:

```
SELECT Nom FROM Clients WHERE anneeAchat IN(2005,2007,2008) ;
```

`Not In` est le contraire .

Recherche d'un motif dans une chaine de caractères avec LIKE %motif% :

```
SELECT * FROM Clients WHERE nom LIKE D% ;
```

Ici recherche tous les clients dont le nom commence par D.

```
SELECT * FROM Clients WHERE nom LIKE D_pont ;
```

le _ masque un caractère dans le motif.

```
SELECT * FROM Clients WHERE nom LIKE [D-G]% ;
```

Sélectionne tous les noms dont la première lettre est comprise entre D et G.

Recherche sur des valeurs manquantes :

```
SELECT * FROM Clients WHERE Telephone IS NULL
```

le contraire est aussi applicable avec `IS NOT NULL`

3.3-Recherche avec la clause `ORDER BY`

Permet de classer les réponses de requêtes par ordre croissant ou décroissant, par défaut c'est par ordre croissant.
Pour spécifier cela, on emploie `ASC` et `DESC`.
Recherche par ordre croissant des modèles en fonction des prix :

```
SELECT Modele FROM Vehicule ORDER BY Prix ASC;
```

Il est aussi possible de faire une recherche avec plusieurs critères , les modèles avec prix croissant et marque décroissant, dans ce cas si le prix est le même entre deux modèles le tri portera sur la marque :

```
SELECT Modele FROM Vehicule ORDER BY Prix ASC, Marque DESC;
```

3.4-Les jointures

Les requêtes avec multi -relations ou jointures permet de relier des champs de différentes tables entre eux. Il est impératif que les champs liés soient de même type.
Si on souhaite connaître les noms des clients qui ont acheté un modèle particulier, on constate que «`modele` » et « `nom` » ne sont pas dans les mêmes tables. Il faut donc utiliser des champs (ici vId) :

```
SELECT Nom FROM Clients,Vehicule WHERE Clients.vId=Vehicule.vId ORDER BY Nom ASC;
```

On peut aussi utiliser des alias pour les tables dans le cas précédent :

```
SELECT Nom FROM Clients C,Vehicule V WHERE C.vId=V.vId ORDER BY Nom DESC;
```

4-Fonctions Agrégats

Les fonctions Agrégats permettent de compter, moyenner, déterminer un maximum ou un minimum.
Fonctions :

- `AVG` , calcule la moyenne des valeurs retournées par une requête
- `COUNT`, compte les valeurs retournées par une requête
- `SUM`, additionne les valeurs d'une requête
- `MIN` et `MAX`, déterminent les minimums et maximum des valeurs retournées par une requête

On souhaite compter les réparations par mécanicien dont on veut connaître le nom ainsi que le temps moyen des réparations :

```
SELECT AVG(R.Temps),COUNT(M.rId),nomMeca FROM Mecanicien M, Reparations R WHERE M.mId=R.mId;
```

5-La clause GROUPE BY

Clause qui permet de regrouper les résultats d'une requête.
On veut connaître le nombre de clients par Vendeur à partir de la table clients :

```
SELECT Vendeur, COUNT(cId) FROM Clients GROUP BY Vendeur;
```

Ce qui donnerait :

Vendeur	Nombre de Client
Dupont	5
Martin	1
Petit	6

Il est possible de faire des groupes dans les groupes.
On recherche par vendeur le nombre de client et modèle correspondant, on groupe donc d'abord par vendeur puis par modèle :

```
SELECT C.Vendeur,COUNT(C.cId),V.Modele FROM Clients C, Vehicule V WHERE C.vId=V.vId GROUP BY C.Vendeur,V.Modele;
```

Ce qui pourrait donner un tableau dont les colonnes seraient :

Vendeur	Modèle	Nombre de clients
Dupont	Alpha	3
Dupont	Bravo	2
Martin	Alpha	1
Petit	Bravo	4
Petit	Charly	2

6-Requêtes Imbriquées

Cela consiste à effectuer une requête dans une autre requête en règle générale dans une clause WHERE. Ce qui revient à utiliser le résultat de la requête incluse dans une autre requête.

On recherche le client et le mécanicien qui est intervenu pour des réparations :

```
SELECT C.Nom,M.nomMeca FROM Clients C,Mecaniciens M WHERE vId=(SELECT vId FROM Reparation R,Mecaniciens M WHERE R.mId=M.mId;);
```

7-Requêtes quantifiées

Une requête imbriquée renvoie des valeurs qui peuvent être filtrées. On parle alors de requêtes quantifiées en filtrant ces valeurs à l'aide d'opérateurs.

Opérateurs de requêtes quantifiées :

- ANY, la condition > (ou <) ANY suivie d'une sous-requête teste si l'expression est supérieure (ou inférieur) à au moins un élément de l'ensemble de valeurs fournies par la sous-requête qui ne doit comporter qu'un seul champ extrait
- ALL, la condition > (ou <) ALL suivie d'une sous-requête teste si l'expression est supérieure à tous les éléments de l'ensemble de valeurs fournies par la sous-requête qui ne doit comporter qu'un seul champ extrait
- EXISTS, la condition EXISTS suivie d'une sous-requête a la valeur vrai si la sous-requête extrait un ensemble non vide de valeurs

On souhaite connaître les temps des réparations dont le montant est supérieur à toutes interventions de type « Révision » :

```
SELECT temps FROM Reparation WHERE montant > All(SELECT montant FROM Reparation WHERE motif="Révision";);
```

8-Intersection, Union et Exception d'ensembles

On peut effectuer des opérations ensemblistes comme :

- `INTERSECT` , L'intersection, qui permet de connaître les valeurs communes entre deux ensembles
- `UNION`, L'union, qui consiste à réunir les valeurs de deux ensembles
- `EXCEPT`, L'exception, qui consiste à ne prendre que les valeurs qui ne sont pas communes à deux ensembles qui ont une intersection

Dans le cadre SQL cela correspond à des opérations effectuées entre deux requêtes.
On cherche les clients qui ont comme vendeur « Dupont » et l'année d'achat était 2009 :

```
SELECT Nom FROM Clients WHERE vendeur="Dupont"
      INTERSECT
  SELECT Nom FROM Clients WHERE anneeAchat="2009";
```

9.Insertion de lignes

La commande INSERT into associée à VALUES permet d'ajouter une ligne dans une table. Par exemple Ajout d'un nouveau mécanicien dans la table :

```
INSERT into Mecanicien VALUES('\N','Dubois') ;
```

\N correspond à une clé primaire avec auto-incrément nécessaire pour définir un id différent pour chaque entrée.
Il est important de donner une valeur à chaque champ et si la valeur est inconnue on utilise 'NULL'.

Il est possible de rentrer des valeurs à partir d'une autre table, pour cela on utilise la clause SELECT.

```
INSERT into conges VALUES('12/10/10','14',mId) SELECT mId FROM Mecanicien
WHERE nomMeca='Dubois' ;
```

10.Mise à jour de ligne

La mise à jour s'effectue avec la commande UPDATE.
On souhaite mettre à jour la fiche d'un client nommé 'Martin' en changeant son téléphone:

```
UPDATE Clients SET { Telephone='0296056171'} WHERE nom='Martin' ;
```

11.Création / Suppression d'une base

Avant même d'enregistrer des données dans la base, il faut bien évidemment que celle-ci existe :

```
CREATE DATABASE 'nomBase' ;
```

Et la suppression de celle-ci :

```
DROP DATABASE 'nomBase' ;
```

12.Administration de la base

Il est impératif d'avoir un compte pour accéder au serveur de base de données. Par défaut ce compte est dépendant du serveur, il est possible de demander une authentification par un autre dispositif (exemple annuaire LDAP).
Certains serveurs de base de données, ne demande pas de mot de passe pour le compte administrateur pendant l'installation (cas de MySQL). Il est donc important de lui en fournir un et de créer un autre compte qui permet de faire des opérations sur les bases mais avec des droits plus limités voire même pour une base particulière.
Pour créer un utilisateur :

```
CREATE USER 'test'@ 'localhost ou IP' IDENTIFIED BY 'motdePasse'
   GRANT (Définitions des droits sur les tables)//pour toutes
   INSERT , UPDATE, DELETE, ALTER (etc.)        //les bases
   GRANT ALL PRIVILEGES ON 'base'               //cas particulier pour une base
```

Exemple :

```
CREATE USER 'test'@'localhost' IDENTIFIED BY '***';GRANT SELECT, INSERT, UPDATE, DELETE, CREATE,
DROP ON *.* TO 'test'@'localhost' IDENTIFIED BY '***' WITH MAX_QUERIES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;GRANT ALL PRIVILEGES ON
`asterisk`.* TO 'test'@'localhost';
```

Pour effacer un utilisateur :

```
DROP USER 'test' @ 'localhost ou IP'
```

13.Création de tables

Pour créer une table on utilise la commande CREATE TABLE.
Il y a des règles à respecter :

- Chaque table de la base possède un nom unique
- Une table est composée d'au moins un champ
- Chaque champ au sein d'une table possède un nom unique
- Il est possible de mettre des contraintes sur les champs lors de la création de la table

En reprenant la table Vehicules dans la l'exemple de la 1 ère page :

```
CREATE TABLE `Garage`.`Vehicules` (
      `vId` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
      `Modele` VARCHAR( 30 ) NOT NULL ,
      `Marque` VARCHAR( 30 ) NOT NULL ,
      `Moteur` VARCHAR( 30 ) NOT NULL ,
      `Annee` DATE NOT NULL ,
      `prix` INT NOT NULL )
      ENGINE = innoDB ;
```

On définit avec la table, tous les champs qu'elle contient ainsi que leur typage :

- `Modele` est de type chaine (VARCHAR) et contient au plus 30 caractères.
- `prix` est de type entier (INT)
- `Annee` est une date (DATE)
- `vId` est une clé primaire en auto-incrément

Remarque : `ENGINE` est une instruction qui spécifie le type du moteur de sauvegarde employé (ici => `innoDB`).

14.Champs de table

Le champ permet d'enregistrer la donnée, l'ensemble des champs d'une table constitue une ligne et à chaque fois qu'on enregistre de nouvelles informations cela revient à insérer une nouvelle ligne.
On constate qu'il faut définir un nom par champ et un type à minima.

14.1-Typage des champs

- NUMERIC (numérique), cela comprend les types INT, FLOAT,DOUBLE,BOOL ...
- STRING (Caractère), comprend deux types CHAR(n) de longueur fixe n et VARCHAR(n) variable jusqu'à n caractères
- DATE, contient date et heure suivant plusieurs formats

Les données disposent d'une valeur par défaut qui peut être fixée (optionnel) par la clause DEFAULT. Cela permet, dans certains cas, d'être certains que les champs ne seront pas vides pendant leur création.

14.2-Valeurs attendues avec la clause `DEFAULT`

- Constante Numérique
- Constante Alphanumérique
- mot-clé `USER`
- mot-clé `NULL`
- mot-clé `CURRENT-DATE` (date de la saisie)
- mot-clé `CURRENT-TIME` (heure de la saisie)
- mot-clé `CURRENT-TIMESTAMP`(heure et date de la saisie)

14.3-Les contraintes de champ

On peut imposer des contraintes sur les champs comme la définition d'une clé primaire unique à chaque entrée ou encore imposé qu'un champ ne peut contenir de valeur NULL.

Liste des contraintes :

- `NOT NULL`, valeur `NULL` impossible dans un champ
- `UNIQUE` et `PRIMARY` (`UNIQUE+NOT NULL`), champ unique dans une table comme la clé primaire et qui ne peut être nulle (cas des id par exemple)
- `REFERENCES`, contrainte référentielle, qui pose la contrainte sur le fait qu'un champ peut exister dans une autre table. Cas des clés étrangères.
- `CHECK`, cette contrainte impose que toutes les valeurs d'une colonne satisfassent une condition (par exemple dans la table `conges` , le champ `duree` doit être supérieur à 3 jours)

Sur l'exemple page précédente, la création d'une clé primaire avec l'option `AUTO_INCREMENT` qui permet à chaque entrée dans la colonne une incrémentation automatique et on interdit toutes valeurs `NULL` :

```
`vId` INT NOT NULL AUTO_INCREMENT PRIMARY KEY
```

14.4-Contraintes de tables

Lors de la création des tables il est important de définir des contraintes. Comme nous l'avons vu précédemment, la nécessité des clés primaires pour assurer une bonne cardinalité dans un schéma « Entités/Associations » vu en cours (chap9).
Mais il y a aussi la possibilité de faire les références et en particulier avec les clés étrangères qui, rappelons le, permet d'inclure des clés primaires dans d'autres tables. De fait ces clés sont mises à jour automatiquement si la clé primaire change. On évite ainsi les corruptions de bases.

On insère une clé étrangère dans la table Conges (mId) qui est déjà une clé primaire dans la table Mecaniciens (appelée également mId) :

```
CREATE TABLE `Garage`.`Mecaniciens` (
      `mId` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
      `nomMeca` VARCHAR( 40 ) NOT NULL
      ) ENGINE = InnoDB;

CREATE TABLE `Garage`.`Conges` (
      `date` DATE NOT NULL ,
      `duree` TIMESTAMP NOT NULL ,
      FOREIGN KEY `mId` INT NOT NULL REFERENCES
      `Garage`.`Mecaniciens`(mId)
      ) ENGINE = InnoDB;
```

Il faut aussi pouvoir définir les triggers pour les clés étrangères. En effet si on supprime une clé primaire ou si on modifie il faut pouvoir garder l'intégrité de la table.
Pour cela on emploie les mots clés suivants lors de la création de la table :

- `ON DELETE NON ACTION / ON UPDATE NON ACTION`, ne produit rien en cas de modifications ou de suppression
- `ON DELETE CASCADE / ON UPDATE CASCADE`, suppression ou modification des éléments qui référence la clé. Sur l'exemple précédent cela supprimerai tous les lignes de Conges d'un mécanicien que l'on aurait enlevé de la base par la table Mecaniciens
- `ON DELETE SET NULL / ON UPDATE SET NULL`, à la suppression ou la modification la clé étrangère passe à la valeur NULL (attention il n'y a plus d'intégrité)
- `ON DELETE SET DEFAULT/ ON UPDATE SET DEFAULT`, la clé étrangère prend la valeur par défaut.

```
CREATE TABLE `Garage`.`Conges` (
            `date` DATE NOT NULL ,
            `duree` TIMESTAMP NOT NULL ,
            ON DELETE CASCADE,
            ON UPDATE CASCADE,
            FOREIGN KEY `mId` INT NOT NULL REFERENCES
            `Garage`.`Mecaniciens`(mId)
      ) ENGINE = InnoDB;
```

15.Assertions

Les assertions sont des conditions ou expressions qui doivent être satisfaites lors d'une modification de données pour que celles-ci puissent être réalisées.
On définit une assertion de la façon suivante :

```
CREATE ASSERTION nom_assertion CHECK ( expressions / conditions)
```

D'un point de vue SQL , une assertion est l'équivalent d'une table dont le contenu changera grâce aux expressions.

16.Les Vues

Les vues sont des « tables virtuelles » dans lesquelles sont placées des données venant de tables physiques. L'intérêt d'une vue vient du fait que les données sont regroupées dans une même entité et de sélectionner les données à afficher.
Pour créer une vue on emploie la clause :

```
CREATE VIEW nom (colonnes) AS requete.
```

Pour effacer une vue on utilise la clause :

```
DROP VIEW nom
```

```
CREATE VIEW voiture (col1,col2,col3) AS
      SELECT modele, marque, prix FROM Vehicules WHERE annee='2010';
```

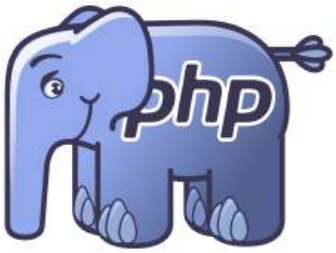
<?PHP?>

Chapitre 11 – Généralités
Chapitre 12 – Chaînes et Tableaux
Chapitre 13 - PDO

BTS Systèmes Numériques
Informatique & Réseaux

FÉLIX
LE DANTEC
du secondaire au supérieur

Chapitre 11
PHP – Généralités



Le sigle **PHP** signifiait à l'origine Personal Home Page. L'auteur, Rasmus Lerdorf, ce langage devait apportait le dynamisme aux pages personnelles qui en manquait cruellement.
Aujourd'hui, l'acronyme PHP signifie Php Hypertext Processor car il renvoie au navigateur un document HTML construit par le moteur de script Zend Engine 2.

1.Introduction

PHP permet de générer des pages interactives, qui permet aux visiteurs de saisir des informations par le biais de formulaire, de les traiter coté serveur (Stockées dans une base de données, un fichier, envoyer un flux etc). L'avènement de PHP était inévitable et il est aujourd'hui le langage « coté serveur » le plus déployé sur internet.

Autres principaux langages utilisés pour la programmation coté serveur :

-Java
-Python
-JavaScript (node.js)
-C/C++ et PERL pour les CGI-BIN

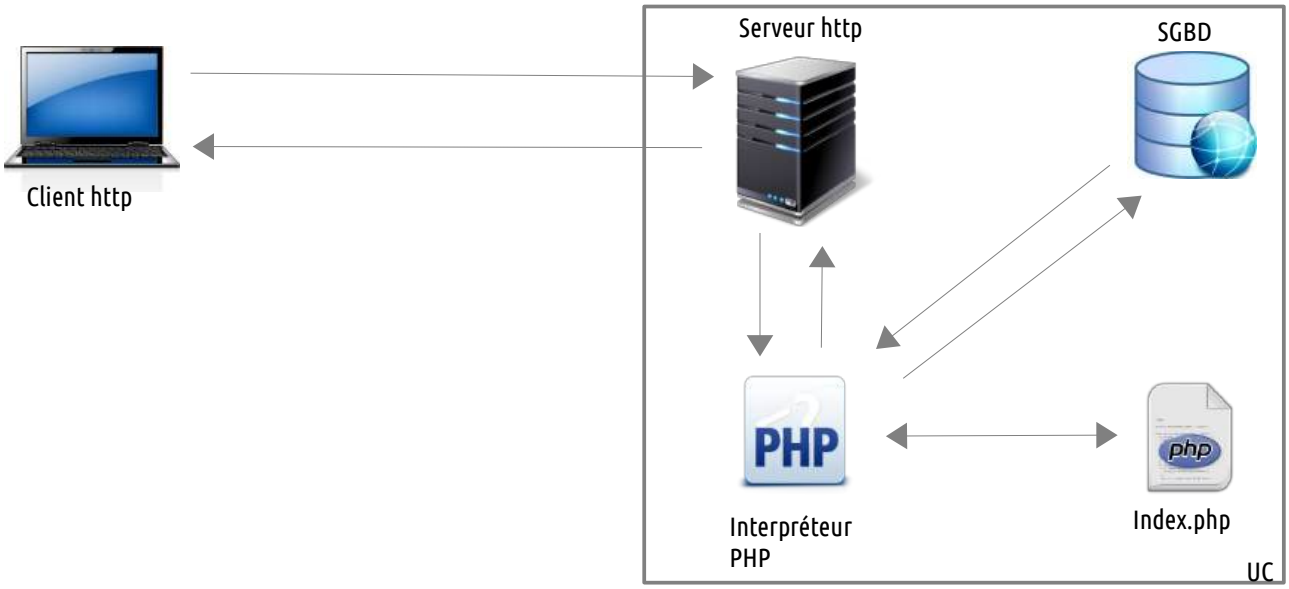
Depuis 1994 (date de sa première version) PHP a suivi une évolution riche tout en prenant en compte des besoins toujours plus grands des développeurs Web. A ce jour, les serveurs sont pour la plupart dotés de la version 5+, on notera que le passage 4.0->5.0 a permis à PHP de devenir un véritable langage orienté objet.
La version 6 a été définitivement abandonnée au profit de la version 7 en cours de test et de développement.

2.Présentation de PHP

PHP repose sur des modules (ou extensions) dont le principal, le module standard qui permet d'accéder aux instructions élémentaires, aux différents types de données et à un grand nombre de fonctions.
A cela viennent s'ajouter un grand nombre de modules spécialisés comme l'accès aux différentes bases de données.
La liste des modules référencés est disponible à l'adresse http://www.php.net qui est aussi le site officiel du langage. (`http://www.php.net/manual/fr/funcref.php`)

2.1-Organisation serveur http, base de données et PHP

Synoptique séquentiel



1 – Le client demande à accéder au serveur http : url = http://serveurhttp/index.php

2- Le serveur récupère la requête et transfère à l'interpréteur la demande d'exécution d'index.php

3-L'interpréteur charge le script index.php puis l'exécute

4-Le programme émet une requête vers le SGBD

5-La réponse du SGBD est traitée par le script

6-Transmission d'un flux HTML5 vers le serveur http grâce à index.php

7-Le serveur renvoi le flux HTML au client en guise de réponse http (HTTP 200)

2.2-Structure d'une page avec PHP

Le code PHP n'est pas renvoyé au client HTTP (synoptique précédent), il doit produire du code HTML qui sera émis alors dans le flux. Pour cela on délimite les portions de code PHP de la façon suivante :

```
//code HTML
<?php

      // code PHP

?>
//code HTML
```

2.3-Méthodes d'inclusion de code PHP

*Dans un fichier HTML (voir http://localhost/index1.php)

```
<!doctype html>
<html>
	<head>
		…
	</head>
	<body>
		<h1>...</h1>
		<?php
			echo "<p>Flux HTML Généré en PHP</p>" ;
		?>
		<div>...</div>
	</body>
</html>
```

*Dans un fichier PHP (voir http://localhost/index2.php)

```
<?php
	include 'genere_entete.php' ;
	include 'genere_corps.php' ;
?>
```

Dans ce cas les fichiers sont externes et leurs codes respectifs sont inclus à l'endroit où l'instruction `include` est exécutée. Les scripts sont ensuite exécutés.

Autre instruction d'inclusion :

```
<?php
	require 'mesFonctions.php' ;
?>
```

La différence avec `include` vient de la réaction de l'interpréteur en cas d'erreur. En effet avec `require` l'erreur est fatale, le script s'arrête immédiatement, alors qu'`include` produit des warnings et le script poursuit quand son exécution.

Remarque :

Il y a deux autres fonctions qui permettent aussi d'inclure des fichiers mais elles ne sont pas exécutées qu'une seule fois :

```
-include_once "fichier" ;
-require_once "fichier" ;
```

3.Variables et constantes

Comme tout langage, PHP manipule des données. Elles sont typées sous forme de texte, de nombres entier ou décimaux, valeurs booléennes, de types composés comme les tableaux, objets etc.

3.1-Variables standards

Les règles de création de variable sont les suivantes :

- Chaque variable possède un identifiant particulier toujours précédé du caractère `$`
- Le nom commence toujours par un caractère alphabétique ou par _ (pas de car. Accentué ni de ponctuation)
- La longueur du nom n'est pas limitée
- La déclaration n'est pas obligatoire tout comme l'initialisation, la variable prend le type dès sa première utilisation
- Les noms de variables sont sensibles à la casse `$mavar=/=$MaVar`

L'affection par valeur et par référence est possible avec PHP :

```
//Affectation par valeurs
1 :$maVar1=12 ;
2 :$maVar2=$maVar1 ;
3 :$maVar1=100 ;
Que vaut ?
Ligne 3 :$maVar2=12
Ligne 3 :$maVar1=100

//Affectation par référence
1 :$maVar1= Brest ;
2 :$maVar2=&$maVar1 ;
3 :$maVar1= Rennes ;
Que vaut ?
Ligne 2 : $maVar2= Brest
Ligne 3 : $maVar2= Rennes
```

L'affectation par référence consiste à prendre en compte l'adresse mémoire d'une variable (équivalent des pointeurs en C).

3.2-Variables prédéfinies

PHP dispose d'un grand nombre de variables prédéfinies, qui contiennent des informations sur le serveur, sur toutes les données qui transitent entre le client et le serveur.
Depuis la version 4.1, les variables sont disponibles sous forme de tableaux « superglobaux ».

Les variables serveur

Variables	Description
$GLOBALS	Contient le nom et la valeur de toutes les variables globales d'un script. Les noms des variables sont les clés du tableau. `$GLOBALS['maVar']` récupère la valeur de `$maVar`.
$_COOKIE	Contient le nom et la valeur des cookies enregistrés sur le poste client. Le nom des cookies correspond à la clé.
$_ENV	Contient les noms et les valeurs des variables d'environnement du serveur.
$_GET	Contient le nom et la valeur des données envoyées par formulaire avec la méthode GET. Le nom des champs correspondent à la clé.
$_POST	Contient le nom et la valeur des données envoyées par formulaire avec la méthode POST. Le nom des champs correspondent à la clé.
$_FILES	Contient le nom des fichiers téléchargés à partir du poste client.
$_REQUEST	Contient l'ensemble des variables superglobales `$_GET`, `$_POST`, `$_COOKIE`, `$_FILES`.
$_SERVER	Contient les informations relatives au serveur Web, tel le contenu des en-têtes http ou le nom du script en cours d'exécution. `$_SERVER["HTTP_ACCEPT_LANGUAGE"]`, code langue du navigateur `$_SERVER["HTTP_COOKIE"]`, nom et valeur des cookies sur le poste client `$_SERVER["SERVER_ADDR"]`, adresse IP du serveur etc.
$_SESSION	Contient l'ensemble des noms de variables de session et leurs valeurs

Utilisation des variables prédéfinies

Elles permettent, entre autre, de récupérer les données envoyées par un formulaire.

Code du formulaire (http://localhost/coursPHP/index3.php) :

```
1.  <!doctype html>
2.  <html>
3.          <head>
4.                  <meta charset="utf-8">
5.                  <title>Formulaire d'envoi de données</title>
6.          </head>
7.          <body>
8.                  <form action="reponse1.php" method="get">
9.                          Nom:<input type="text" name="nom"><br>
10.                         Prénom:<input type="text" name="prenom"><br>
11.                         <input type="submit" name="Envoyer">
12.                 </form>
13.         </body>
14. </html>
```

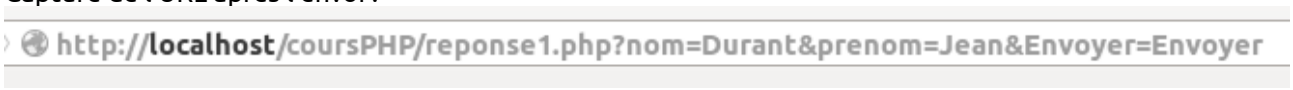
Quel est le rôle de la ligne 5 ? Expliquez chaque paramètre.

Définit le formulaire, la méthode d'envoi – ici GET – et le script appelé pour recevoir les données – ici `reponse1.php`.

Quel est le rôle de la ligne 11 ?

C'est un bouton spécial qui génère la requête http.

Capture de l'URL après l'envoi :



Expliquez la structure de l'URL :

`http://localhost/coursPHP/reponse1.php?nom=Durant&prenom=Jean&Envoyer=Envoyer`

?: Délimiteur entre le script et les données transmises
nom de la data (nom de la balise contenant la donnée) = valeur (saisie)
& : association des différentes données envoyées

Script reponse1.php :

```
1.  <?php
2.          include 'genere_entete.php';
3.  ?>
4.          <body>
5.                  <h1>Données reçues par le serveur :</h1>
6.                  <?php
7.                          echo "Flux HTML généré par PHP:<br>";
8.                          echo "Nom envoyé :<b>".$_GET["nom"]."</b>";
9.                          echo "<br>";
10.                         echo "Prénom envoyé : <b>".$_GET["prenom"]."</b>";
11.                 ?>
12.                 <br>
13.                 <a href="index3.php">retour au formulaire</a>
14.         </body>
15. </html>
```

Donnez les numéros de lignes incluants le code PHP :

Lignes 2, 7 à 10

Que fait la ligne 2 ?

Elle inclut le script genere_entete.php et l'exécute ensuite.

Complétez les lignes 8 et 10 en vous aidant du tableau page 4.

```
echo "Nom envoyé :<b>".$_GET["nom"]."</b>";

echo "Prénom envoyé : <b>".$_GET["prenom"]."</b>";
```

Quel relation y a-t-il entre le formulaire et les variables prédéfinies ?

Les données sont stockées dans un tableaux dont la clé de la cellule est le nom du champs HTML.

Donnez alors les lignes 8 et 10 si la méthode d'envoi est de type POST :

```
echo "Nom envoyé :<b>".$_POST["nom"]."</b>";

echo "Prénom envoyé : <b>".$_POST["prenom"]."</b>";
```

D'après vous qu'est-ce que cela changerait dans l'URL ?

On ne verrait pas les champs et données envoyés.

Capture du flux HTML reçu par le client :

```
1.  <!doctype html>
2.  <html>
3.          <head>
4.                  <title>
5.                          Cours PHP
6.                  </title>
7.                  <meta charset="utf-8">
8.          </head>
9.          <body>
10.                 <h1>Données reçues par le serveur :</h1>
11.                 Flux HTML généré par PHP:<br>Nom envoyé :<b>Durant</b><br>Prénom envoyé :
                    <b>Jean</b><br>
12.                <a href="index3.php">retour au formulaire</a>
13.         </body>
14. </html>
```

Donnez les lignes qui ont été produites par genere_entete.php :

lignes 1 à 8

Quelle partie du code HTML de <body> a été générée par du code PHP ?

La ligne 11

3.3-Les constantes

Il arrive parfois où nous sommes amenés à utiliser de manière répétitive des informations devant rester constantes dans toutes les pages d'un site. Pour ne pas risquer l'écrasement accidentel, l'emploi des constantes est indispensable.

3.3.a Constantes personnalisées

On emploi la fonction define() pour déclarer une constante, il faut lui associer le nom, la valeur et si le nom est sensible ou non à la casse :

```
define( PI ,3.141592,true) ;
define( Planck ,6.6E-34,false);
```

« `true` » insensible à la casse et « `false` » sensible à la casse, dans l'exemple ci-dessus `pi = PI` alors que `Planck=/=PLANCK =/=planck` etc.

`define()` est de type booléen, elle retourne « true » en cas de succès ou « false » en cas d'échec.

Exercice
On souhaite écrire un fichier « ctes.inc.php » qui contient les paramètres de connexion à une base de données. Ce fichier est appelé par « index.php » systématiquement.
On suppose que les deux fichiers se trouvent dans le sous-répertoire.

1-Écrire la ligne à ajouter à `index.php` permettant cela :

```
<?php
        include 'ctes.inc.php' ;
```

2-Pour se connecter à une base de données, il faut indiquer l'adresse du serveur SGBD, le nom de la base, le login et le mot de passe. Ces données ne doivent pas être modifiées par du code malsain c'est pour cette raison que nous déclarerons en tant que constantes. Elles doivent être stockées dans le fichier ctes.inc.php.
On vous demande d'écrire ce script qui ne fait que déclarer les constantes en vous appuyant sur le tableau ci-dessous. La casse caractère doit être respectée :

Nom de la constante	Valeur de la constante
SERVEUR	localhost
BASE	maBase
USER	felix
MDP	felix22

```
<?php
      define('_SERVEUR_','localhost',' false') ;
      define('_BASE_','maBase',' false') ;
      define('_USER_','felix',' false') ;
      define('_MDP_','felix22',' false') ;
?>
```

3.3.b Constantes prédéfinies
Il y a un grand nombre de constantes prédéfinies que l'on peut employer dans les fonctions.

Quelques constantes prédéfinies

Constantes	Description
PHP_VERSION	Version installée sur le serveur
PHP_OS	Nom du système d'exploitation du serveur
DEFAULT_INCLUDE_PATH	Chemin d'accès aux fichiers par défaut
FILE	Fichier en cours d'exécution
LINE	Ligne en cours d'exécution

3.4-Déterminer et changer le type d'une variable

Avant de manipuler les variables, il est parfois utile d'en connaître le type, cela permet de s'assurer que le résultat attendu est conforme, incrémenter une variable de type chaîne alors qu'on pensait travailler avec un nombre par exemple.

3.4.a Détermination et vérification de type

```
$typedemaVar=gettype($maVar) ; //renvoi le type de $maVar
```

Quelques fonctions vérifiant le type :

```
if(is_string($maVar))// Teste si $maVar est une chaîne
if(is_array($mavar)) // Teste si $maVar est un tableau
if(is_ressource($maVar))//Teste si $maVar est de type ressource
```

Les fonctions de vérification de type renvoient toujours un résultat de type booléen d'où la nécessité de tester. Voir le site officiel => http://php.net/manual/fr/book.var.php

3.4.b Changement de type

Il est parfois nécessaire de changer le type de variables en particulier celles provenant d'un formulaire qui sont toujours de type « String » .

Pour cela on effectue un opérateur de « cast » :

```
$maVar=(type désiré) $laVariable ;
```

Exemple affiche « 12 ans » avant puis « 12 » après :

```
$maVar="12 ans";
echo "\$maVar avant changement type string :".$maVar."<br>";
$maVar=(integer) $maVar;
echo "\$maVar après changement type integer :".$maVar."<br>";
//Autre fonction
settype($maVar, type désiré ) ;
```

`settype()` permet de faire en une ligne le changement de type d'une même variable.

3.5-Contrôler l'état d'une variable (http://localhost/coursPHP/index4.php)

C'est un élément important dans le traitement de données envoyées par formulaire, il faut prendre en compte le contrôle des données c'est une question de sécurité avant tout.

Il y a deux fonctions pour vérifier l'état d'une variable :

- `isset($maVar)`, fonction booléenne qui retourne TRUE si la variable existe, FALSE ou NULL dans le cas contraire
- `empty($mavar)`, retourne TRUE si la variable n'est pas initialisée ou si elle vaut 0 ou NULL ou la chaîne "0" et FALSE si elle contient une valeur quelconque

Comme ce sont des fonctions booléennes, il faut les employer avec des instructions de contrôle de type if, while etc.

3.6-Les types divers

PHP offre également deux types particuliers qui sont employés dans des circonstances particulières.

3.6.a Le type ressource

Ce type représente une référence à des informations présentes sur le serveur. Il est le type retourné par certaines fonctions, en particulier celles qui permettent d'accéder aux bases de données lors de la connexion.
Cette information est très utile quand il y a plusieurs connexions simultanées à partir d'un seul ou plusieurs scripts.

3.6.b Le type NULL

Le type NULL est celui qui est attribué à une variable qui n'a pas de contenu ou qui a été explicitement initialisée avec la valeur NULL. Dès que celle-ci se trouve affecter par une valeur, elle change de type.

```
$maVar ;            //$maVar est de type NULL
$maVar= "" ;        //même si elle ne contient pas de chaine $maVar est de type string
```

4.Les instructions de contrôle

La plupart des instructions de contrôle sont identiques à celles qui ont été vues avec JavaScript.
On retrouve :

- les instructions conditionnelles avec `if, if...else,switch...case`
- les instructions de boucle avec `for, while, do...while, foreach, break`

4.1-Opérateur ?

L'opérateur ? permet de remplacer avantageusement une instruction `if...else` en évaluant une expression et en attribuant à une variable une première valeur si la condition est vraie ou une autre valeur si elle est fausse.

```
$maVar= (expression) ? valeur1 : valeur2 ;
//équivaut à
if(expression) {$maVar=valeur1 ;}else{$maVar=valeur2 ;}
```

Exercice

Écrire avec l'opérateur ?, le test de la variable $age qui retourne dans la variable $message soit « Majeur » soit « Mineur » .

```
$message=($age<18) ? 'Mineur' : 'Majeur' ;
```

4.2-Gestion des erreurs

Dans le cadre de la programmation, il est fondamental de prendre en compte les erreurs et de les gérer. Ces erreurs sont bien souvent dues à de mauvaises actions des utilisateurs qui peuvent parfois provoquer un arrêt du script par exemple.

Exemple :

```
function division(){
$a=100/0 ; // une division par 0 provoque le message d'erreur suivant
}
```

Sur la page web :

```
Warning: Division by zero in index.php on line 3
```

Pour éviter ce désagrément, il existe plusieurs solutions :

- Faire précéder l'appel d'une fonction du caractère @, `@division()` par exemple
- Utiliser la fonction `error_reporting()` qui permet de n'afficher que certains messages selon le type d'erreur, `error_reporting(E_WARNING)` bloque l'affichage des alertes comme la division par 0

Liste des niveaux d'erreurs courantes

Constante	Valeur	Description
E_ERROR	1	Erreur fatale qui provoque l'arrêt du script
E_WARNING	2	Alerte ne provoquant par l'arrêt du script
E_PARSE	4	Erreur de syntaxe détectée par l'interpréteur et provoquant l'arrêt du script
E_NOTICE	8	Problème rencontré par le script qui n'est pas considéré comme une erreur
E_ALL	4095	Toutes les erreurs

4.3-Gestion des exceptions

Une exception est un système permettant de détecter une erreur dans un script et de la traiter. Gérer une exception consiste à délimiter une partie du script susceptible d'être à l'origine d'une erreur et d'exécuter une autre action à la place.

```
try {
      if(erreur prévue){
            throw new Exception() ;
      }else{
            //Action normale
      }
} catch(Exception $except){
      //Gestion de l'erreur
}
```

Try...catch est un dispositif qui permet d'isoler la portion de code sensible, ainsi on peut poursuivre l'exécution du script même en cas d'erreur fatale. L'instruction sensible est évaluée avant d'être exécutée.

L'instruction `throw` créée l'objet `Exception`.
On associe ensuite la gestion de l'erreur à l'exception par une instanciation (`$except`).
De fait, 6 méthodes sont associées à l'objet :

Méthode	Définition
`__construct()`	Constructeur de l'objet. Il est appelé automatiquement à la création d'un objet avec le mot clé `new`. Méthode qui ne doit pas être appelée explicitement.
`getMessage()`	Retourne la valeur de la propriété message dans une chaine
`getCode()`	Retourne la valeur de la propriété code sous forme d'un entier
`getFile()`	Retourne la valeur de la propriété file qui contient le nom et le chemin d'accès du fichier dans lequel s'est produite l'erreur.
`getLine()`	Retourne le numéro de ligne à laquelle s'est produite l'erreur
`__toString()`	Retourne une chaine contenant les informations sur l'exception

Exercice

On vous demande d'écrire la gestion d'erreur pour la division. On suppose que `$division = $a/$b`.
Si `$b` est nulle on signale l'erreur fatale à l'aide de la méthode `getMessage()` :

```
try {
      if($b==0){
            throw new Exception() ;
      }else{
            $division=$a/$b ;
      }
}catch(Exception $except){
      echo Erreur dans la division: .$except->getMessage() ;
}
```

Chapitre 12
PHP – Chaînes et Tableaux

Les chaînes et les tableaux sont des éléments essentiels dans le traitement de données par PHP. Que ce soit dans un formulaire ou dans la programmation client/serveur, les données émises au niveau applicatif sont pour la grande majorité au format texte.
A cela s'ajoute l'API que fournit PHP et reposant sur de nombreux tableaux super-globaux permettant de récupérer les requêtes http, de gérer les sessions etc.

1.Affichage et Mise en forme des chaînes et tableaux

PHP dispose d'un certains nombre de fonction pour assurer l'affichage standard ou formaté de chaînes.

1.1-echo

Redirige le flux vers la sortie standard (stdout).

Exemples :

```
1.      echo "Hello World" ;
2.      echo "Bonjour Monsieur $nom" ;
3.      echo "Bonjeur Monsieur ".$_GET["nom"] ;
```

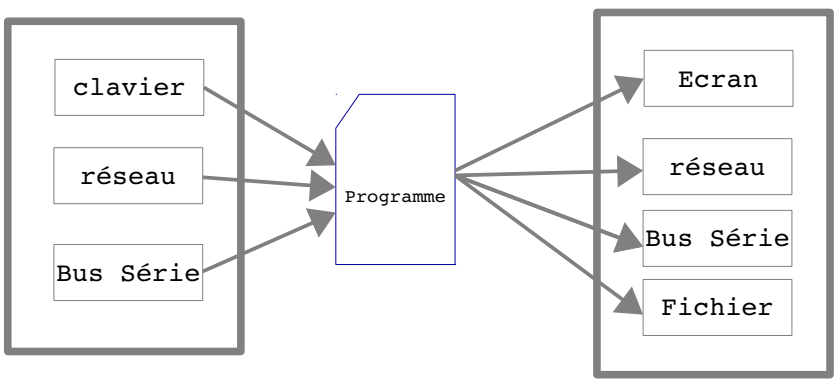
Remarques sur les exemples :

- Dans tous les cas la fonction echo nécessite l'utilisation des " ou ' suivant le contenu
- Une variable peut être insérer dans les " sans avoir besoin de faire de la concaténation
- Contrairement à une variable un tableau ne peut pas être insérer dans une chaîne sans concaténation (« . »)

1.2-Entrée et Sortie Standards

Plus généralement on parle de flux standards en informatique, cela correspond à des canaux gérés par le système d'exploitation et par lesquels les données transitent entre les programmes.

L'entrée standard « `stdin` » correspond au flux de données entrant dans un programme, la sortie standard « `stdout` » correspond au flux de sortie des données provenant du programme.



1.3-print_r()

La fonction `print_r()` affiche la structure et les informations contenues dans un tableau. On utilise cette fonction uniquement en phase de programmation. Elle redirige le flux standard vers « `stdout` ».

Exemples:

```
>print_r($_POST) ;
Array ( [NOM] => Dupont [PRENOM] => Michel [EMAIL] => mdurant@gmail.com [PSEUDO] => jdupont [MDP1]
=> [DDNJ] => 01 [DDNM] => Janvier [DDNA] => 2015 [genre] => Homme [CI] => Array ( [0] => ELN [1] =>
DIY ) )

>print_r($_FILES) ;
Array ( [avatar] => Array ( [name] => linux.jpg [type] => image/jpeg [tmp_name] => /tmp/phpiDsZfZ
[error] => 0 [size] => 22755 ) )
```

Les autres fonctions usuelles sur le traitement des chaînes de caractères sont décrites sur le site officiel : http://php.net/manual/fr/ref.strings.php

2.Protection des formulaires

2.1-Injection de code

Supposons que l'on ait le formulaire suivant(Exemple → http://localhost/coursPHP/chap12/index1.html) :

```
<!doctype html>
<html>
        <head>
                <title>Chap12 - Test injectionS HTML</title>
                <meta charset="utf-8">
        </head>
        <body>
                <form method="get" action="index1.php">
                        Saisie : <input type="text" name="saisie"><br>
                        <input type="submit" value="Envoyer">
                </form>
        </body>
</html>
```

Écrire le programme PHP permettant d'afficher la saisie. On utilisera les scripts `header.php` et `footer.php` pour l'en-tête et le bas de page HTML. `index1.php` affichera le corps.

```
<?php
        require "header.php";

        echo "Valeur reçue non sécurisée:".$_GET["saisie"];

        require "footer.php";
?>
```

Cas typique d'envoi d'une chaîne avec du HTML – URL avec la méthode POST ::

`http://localhost/coursPHP/chap12/index1.php?saisie=%3Ca+href%3D%22http%3A%2F%2Fwww.google.fr%22%3EJean+Dupont%3C%2Fa%3E`

Retrouvez la signification des codes en gras et retrouvez la chaîne qui a été saisie :

`%3C → <` `%3D → =` `%22 → "` `%3A → :` `%2F → /` `%3E → >`

`<a href="http://www.google.fr">Jean Dupont</a>`

Page reçue (point de vue navigateur) :

```
<!doctype html>
<html>
        <head>
                <title>Chap 12 - réponse</title>
                <meta charset="utf-8">
        </head>
        <body>
              Valeur reçue non sécurisée:<a href="http://www.google.fr">Jean Dupont</a>
       </body>
</html>
```

Que peut-on en conclure :
La chaîne de caractère saisie étant renvoyée tel que au client, si celle-ci contient du HTML, elles seront interprétés par le navigateur en tant que balises. Il y a danger car il est possible d'ajouter du code JS malicieux.

2.2-Mesures de protection

Pour éviter du détournement de code ou de l'injection HTML, il faut que PHP génère une page HTML avec une conversion de la chaîne saisie en texte.

Fonction `htmlentities()`
Elle convertit les caractères éligibles à HTML en caractère texte (comme <, >, " etc. liste sur wikipedia : `http://fr.wikipedia.org/wiki/Aide:Liste_de_caractères_spéciaux`)

Exemple :
`$chaineTexte = htmlentities(`*`$chaine_à_convertir`*`) ;`

Modifiez le code d'index1.php en utilisant cette fonction :

```
<?php
        require "header.php";

        echo "Valeur reçue et traitée avec htmlentities():".htmlentities($_GET["saisie"]);

        require "footer.php";
?>
```

Code source de la page reçue :

```
<!doctype html>
<html>
        <head>
                <title>Chap 12 - réponse</title>
                <meta charset="utf-8">
        </head>
        <body>

Valeur reçue et traitée avec htmlentities():&lt;a href=&quot;http://www.google.fr&quot;&gt;Jean
Dupont&lt;/a&gt;
       </body>
</html>
```

La fonction `htmlspecialchars` est très proche mais elle ne convertit que les caractères équivalent HTML.

Il existe aussi la fonction strip_tags() qui élimine les balises HTML dans une chaine de caractère, elle est dépendante de la version HTML, en général on autorise certaines balises avec cette fonction.

```
$chaine=strip_tags($chaine,balises_autorisées) ;
```

Exemples :

```
$chaine = "<span>M. Jean Dupont, adresse <a mailto ="jdupont@jdupont@bzh"> Email</a></span>" ;

$chaine1=strip_tags($chaine) ;                      // élimine toutes les balises
$chaine2=strip_tags($chaine,"<span><a>" );  //n'élimine pas <span> et <a>
```

2.3-Attaque type XSS

XSS – Cross Site Scripting - est un procédé qui permet d'injecter du code malicieux dans une requête HTML. Ce type d'attaque a permis par le passé de propager des virus comme Samy, de faire du fishing ou pire encore d'exploiter des failles coté serveur (spams sur les forums sans authentification par exemple).

3.Les expressions régulières

PHP fournit une collection de fonctions permettant d'exploiter les expressions régulières. Cette famille appartient aux PRCE (`http://php.net/manual/fr/ref.pcre.php`).

Les plus usités sont preg_match, preg_split, preg_replace et preg_grep. Dans tous les cas il faut se familiariser avec les expressions régulières.

3.1-Rappel

L'expression régulière correspond à la définition d'un motif à partir duquel il est possible de faire des comparaison, des tests de validités ou des recherches sur des chaînes de caractères.

Le principe a été abordé en cours, la documentation officielle PHP reprend les bases:

http://php.net/manual/fr/reference.pcre.pattern.syntax.php

Tableau de quelques méta-caractères :

Métacaractère	Nom	Signification
.	Point	N'importe quel caractère présent
[...]	Classe de caractères	Tout caractère parmi ceux énumérés
[^...]	Classe de caractères complémentée	Tout caractère excepté ceux énumérés
^	Circonflexe	Début de chaine
$	Dollar	Fin de chaine
|	Barre verticale	Reconnaît l'un ou l'autre des termes qu'il sépare
(...)	Parenthèses	limite la portée d'un test tel que « | »
?	Optionnel	Présence ou non de la chaine qui le précède
+	Répétition	Une ou plusieurs copies de l'élément qui le précède (1 exigé au minimum)
*	Répétition optionnelle	Comme + avec toutefois la possibilité de ne pas avoir l'élément
{n} {n,} {n,m}	Accolades	Recherche exactement n fois le caractère ou le groupe qui précède Recherche au moins n fois le caractère ou le groupe qui précède Recherche entre n et m fois le caractère ou le groupe qui précède

Exercices :
1-Donnez le motif pour vérifier si le nom commence par une majuscule :

^[A-Z][a-z]+$#

2-Donnez le motif pour rechercher une date de naissance écrite sous la forme JJ/MM/AAAA si J, M et A sont des nombres, JJ ∈ [1-31] pas de 0 pour [1-9], MM ∈ [1-12] pas de 0 [1-9] et AAAA ∈ [19XX-20XX]

^([1-2][0-9]|[1-9]|3[0-1])\/(1[0-2]|[1-9])\/19|20[0-9]{2}$

3.2-Utilisations en PHP

Recherche de motif avec `preg_match()`

Cette fonction retourne « `true` » si la chaîne testée respecte le motif.

Exemple

```
if(preg_match($motif,$chaine)){
      //Si la chaine respecte le motif
}
```

Récupération de valeurs dans une chaine avec preg_split()

Cette fonction extrait d'une chaîne des sous-chaînes à partir d'un caractère ou un motif de séparation. Elle renvoie les sous-chaînes dans un tableau.

```
$tab = preg_split($motif,$chaine) ;
```

Exercice

Un fichier CSV contient les nom, prénom et les adresses email des utilisateurs d'un forum. On lit le fichier et chaque ligne est récupérée dans une variable `$chaine`.

1- On vous demande de traiter la chaîne en récupérant chaque champs dans une cellule d'un tableau.

2- Même chose avec la fonction `list()` et les variables `$nom`,`$prenom` et `$email`.

`list()` :

La fonction `list()` transfère les valeurs d'un tableau dans des variables suivant l'ordre dans lequel elles sont passées en paramètres.

```
$chaine="Jean;Dupont;jdupont@jdupont.fr" ;

1- preg_split()
$tableau=preg_split(" ;",$chaine) ;

2 – avec preg_split() et list() :
list($nom,$prenom,$email)=preg_split(" ;",$chaine) ;
```

4.Les tableaux

4.1-Tableaux indicés et associatifs

Ces notions ont été déjà vues avec JavaScript, on rappelle le principe :

```
$tabIndice = Array("Brest","Rennes","Lorient","Lannion") ;
echo $tabIndice[2] ; // affiche Lorient

$tabAsso = Array("Dpt29" => "Brest","Dpt35" => "Rennes","Dtp56" => "Lorient","Dpt22" =>
"Lannion") ;
echo $tabAsso["Dtp29"] ; // affiche Brest
```

Une chaîne de caractères est vue aussi comme un tableau indicé. Proposez un programme qui permet d'afficher une chaîne de caractères dont chacun d'entre eux serait séparé par un « - » :

```
$chaine="Le PHP est un langage de script coté serveur";
```

Résultat dans le navigateur :

L-e- -P-H-P- -e-s-t- -u-n- -l-a-n-g-a-g-e- -d-e- -s-c-r-i-p-t- -c-o-t-é- -s-e-r-v-e-u-r-

```
<?php
        require 'header.php';
        $chaine="Le PHP est un langage de script côté serveur";
        for($i=0;$i<strlen($chaine);$i++){
                echo utf8_encode($chaine[$i]."-");
        }
        require 'footer.php';
?>
```

Remarque : `strlen()` permet de compter le nombre de caractères d'une chaîne et `count()` le nombre de cellules d'un tableau (déclaré avec `array()`).

4.2-Tableaux multidimensionnels

Comme JavaScript, on peut créer des tableaux à plusieurs dimensions.

Exercice :
Soit le tableau ci-dessous :

```
$villeBzh = Array (
            Array("Saint Brieuc","Dinan","Lannion") ;
            Array("Quimper","Brest","Morlaix") ;
            Array("Rennes","Saint-Malo","Redon") ;
            Array("Lorient","Vannes","Pontivy") ;
) ;
```

Qu'est-ce qu'affiche ?

```
echo $villeBzh[1][2] ;
```

→ Morlaix

```
echo $villeBzh[3][1] ;
```

→ Lorient

```
echo $villeBzh[4][2] ;
```

→ Erreur l'indice 4 correspondrait aux 5ème tableau (il n'y en a 4).

Si on rajoute `Guingamp` aux villes Costarmoricaines, écrire la ligne permettant de l'afficher avec `echo` :

→ `echo $villeBzh[0][3] ;`

4.3-Lecture des tableaux

D'une manière générale on utilise les structures de boucles comme `for`, `while` etc. mais certaines sont à utiliser avec précaution.

Soit le tableau suivant :

```
$tabVille[0]="Lannion" ;
$tabVille[1]="Guingamp" ;
$tabVille[3]="Morlaix" ;
$tabVille[4]="Brest" ;
```

Soit le programme suivant :

```
for($i=0;$i<count($tabVille);$i++){
        echo $tabVille[$i] ;
}
```

D'après vous quels sont les problèmes rencontrés ?

- $tabVille[2] n'existe pas, echo n'affichera rien éventuellement une erreur ou un warning
- count indique 4 cellules donc $i aura pour valeur max 3, on ne verra pas Brest.

Recherchez une solution toujours avec for,while etc. pour afficher le tableau complet même si les cellules ne sont pas contiguës :

```
<?php
        require 'header.php';
        $tabVille[0] = "Lannion";
        $tabVille[1] = "Guingamp";
        $tabVille[3] = "Morlaix";
        $tabVille[4] = "Brest";
        $tabVille[10]= "Rennes";
        $j=0;
        for($i=0;$i<count($tabVille);$i++){
                while(!isset($tabVille[$j])){
                        $j++;
                }
                echo $tabVille[$j]."<br>";
                $j++;
        }
        require 'footer.php';
?>
```

Boucle `foreach` :

Cette boucle permet de régler le problème de l'exercice précédent (même si la solution proposée permet aussi de palier au problème).
La syntaxe de cette boucle est différente si on utilise un tableau indicé ou associatif :

```
//Cas de tableau indicé
foreach($tableau as $valeur){
      // instructions
}
```

`$valeur` contient la valeur de la cellule, attention cette valeur est écrasée à la prochaine itération.

```
//Cas d'un tableau associatif
foreach($tableau as $clef => $valeur){
      //Instructions
}
```

`$clef` contient le « nom » de la cellule `$tableau[$cle] = $valeur`
Même remarque que précédemment avec `$valeur`.

Exercices

1- Écrire le programme de l'exercice précédent avec `foreach()`.

```
foreach($tabVille as $valeur){
      echo $valeur ;
}
```

2-Soit le formulaire ci-dessous :

```
<form method="post" action="index.php">
      Nom :<input type="text" name="nom"><br>
      Prénom :<input type="text" name="prenom"><br>
      signature:<textarea name="signature"></textarea><br>
      Langue:<input type="radio" name="lg" value="fr">fr
      <input type="radio" name="lg" value="en">en<br>
      <input type="submit" value="Envoyer">
</form>
```

Écrire la boucle d'`index.php` qui affichera la saisie complète en indiquant les champs et leur valeurs :

```
foreach($_POST as $cle => $valeur){
      echo "$cle : $valeur <br>" ;
}
```

4.4-Manipulation des tableaux

Il existe de nombreuses fonctions pour la manipulation des tableaux :
-fusion de tableaux
-ajout/suppression de cellule
-tri
-suppression de tableau
-etc.
Voir la liste exhaustive → `http://php.net/manual/fr/ref.array.php`

5-Tableaux super-globaux

PHP fournit un certains nombre de tableaux super-globaux (vu au chapitre 11) tel que $_GET ou $_POST. Nous allons nous intéresser à 3 d'entre eux pour ce chapitre.

5.1-$_SERVER

Tableau contenant les informations relatives aux en-têtes HTTP. Il est ainsi possible de connaître l'adresse IP du client, le script PHP exécuté, la nature du client HTTP (firefox, chrome etc.), le système etc. Il est très utile de le consulter en particulier si on souhaite enregistrer les visites sur le site.
Le descriptif du tableau est donné → http://php.net/manual/fr/reserved.variables.server.php

Exemples :

```
echo $_SERVER['REMOTE_ADDR'] ; // Affiche l'adresse IP du client
$_SERVER['PHP_SELF'] ;          //Contient le nom du script ciblé par le paramètre action
                               //d'un formulaire
```

5.2-$_FILES

Tableau qui contient les informations relatives aux fichiers envoyés par un formulaire. On peut ainsi récupérer le nom d'origine du fichier, son nom dans /tmp du serveur, sa taille, son type (si reconnu) etc.

Attention ce tableau n'existe que si le formulaire d'envoi dispose du paramètre :
`enctype="multipart/form-data"`
C'est un tableau à 2 dimensions même si un seul fichier est envoyé.

Configuration PHP :

L'envoi de fichier est limité par la taille du fichier à envoyer ainsi que par la taille des données envoyées par formulaire. La configuration se fait au niveau du fichier `php.ini` situé dans le répertoire `/etc/php5/apache2/` :

```
;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;
; Whether to allow HTTP file uploads.
; http://php.net/file-uploads
file_uploads = On
...
; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 10M
```

Puis la limite de la taille des données émises par formulaire :

```
; Maximum size of POST data that PHP will accept.
; Its value may be 0 to disable the limit. It is ignored if POST data reading
; is disabled through enable_post_data_reading.
; http://php.net/post-max-size
post_max_size = 12M
```

`post_max_size` doit être plus grande que `upload_max_filesize`.

Redémarrez le serveur apache .

Précautions à prendre
Le fichier est placé dans le répertoire `/tmp` du serveur sous un nom qui ne correspond pas à celui d'origine. De plus il est supprimé à la fin d'exécution du script qui est censé le traiter coté serveur. Le déplacement de celui-ci est assuré par la fonction `move_uploaded_file()`.

Descriptif de $_FILES sur le site officiel : http://php.net/manual/fr/reserved.variables.files.php

5.3-`$_SESSION`

Avec HTTP, les données ne sont pas persistantes entre deux requêtes (pour un même site) y compris dans le cas où on utilise le même script. Les données sont supprimées à la fin du script.
Il devient donc compliquer les connexions, le suivi des utilisateurs etc. .

Le tableau `$_SESSION` permet de compenser ce problème en mémorisant les données définies par le développeur du site.

On peut y stocker tout ce qui servira au dispatcher ainsi que dans la gestion de la connexion.

Le système de session repose sur les cookies et sur les variables d'environnement de PHP, à chaque exécution du script, qui gère les sessions, il y a un contrôle du cookie et une association à $_SESSION.

Gestion des sessions
Une session est créée si la fonction session_start() est utilisée. Elle doit être placée dès le début du script :

```
<?php
        session_start() ;
        …
?>
```

Le cookie correspondant est créé à ce moment précis. Il renferme un ID que l'on peut voir dans firefox : → Préférences>Vie Privée>Règles de conservation → Utiliser les paramètres personnalisés pour l'historique puis cliquez sur « afficher les cookies », recherchez le site hôte (localhost dans le cas présent ou adresse ip). Firefox utilise SQLITE pour gérer les cookies.

Il est possible de récupérer cet ID avec la fonction `session_id()` :

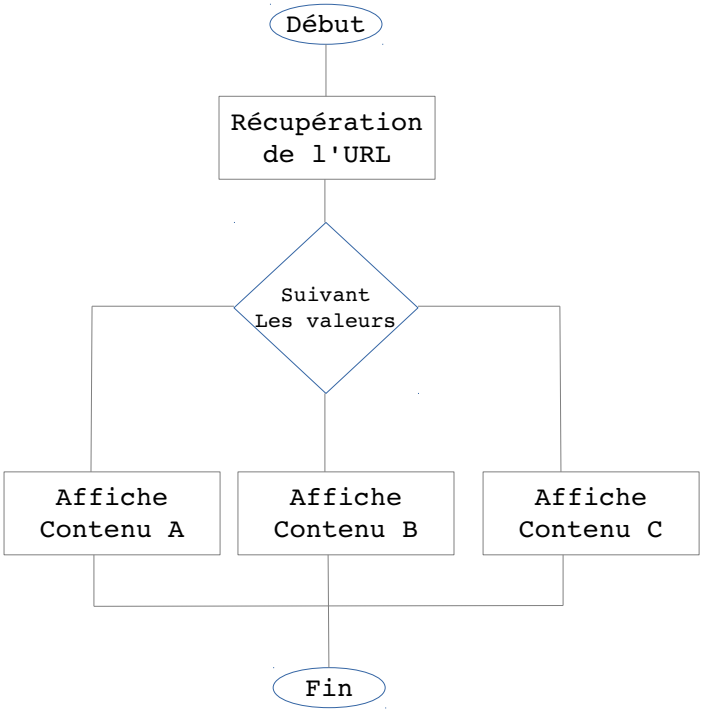
```
        $maSession=session_id() ;
```

Puis de la stocker dans le tableaux de session :

```
        $_SESSION['userID']=$maSession ;
```

Notion de Dispatcher
Le Dispatcher et les sessions seront abordés au TP12.

Un dispatcher est un algorithme permettant d'afficher des contenus dynamiques et différents tout en appelant toujours le même script.

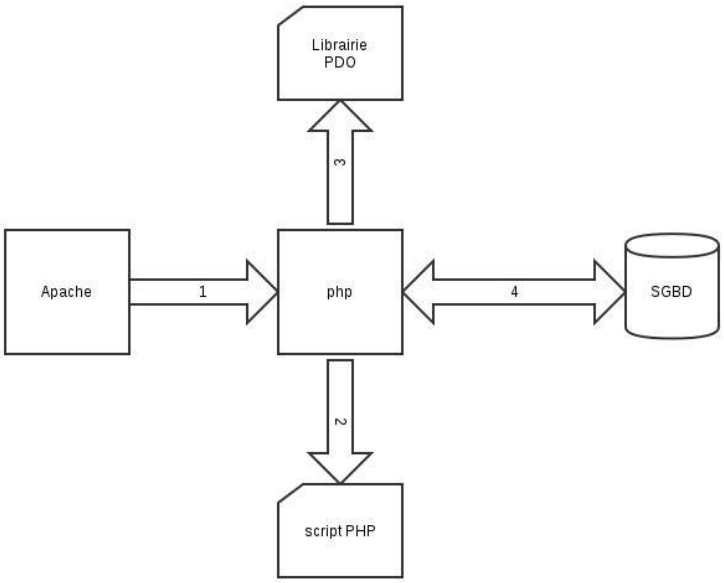


Chapitre 13
PHP & PDO

Les applications utilisent massivement les Systèmes de Gestion de Bases de Données (SGBD) et les sites internet en particulier.
PDO[1] est une couche d'abstraction logicielle permettant de faire des requêtes vers un SGBD à partir d'un script PHP et donc très utile pour les sites internet.

1.Liaison application / SGBD

Un SGBD est un moteur d'analyse et de traitement des requêtes transmises par un client compatible. Il décode la demande, puis l'exécute en recherchant, ajoutant, mettant à jour ou supprimant des données.
Le client s'appuie très souvent sur un langage spécifique pour effectuer une requête (SQL par exemple – voir chapitre 10).



1 – Apache demande l'exécution d'un script et transmet des valeurs envoyées via formulaire

2- Php lit le script et l'exécute, celui utilise la librairie pdo

3- Php charge la librairie

4- puis poursuit l'exécution du script en se connectant à la base et en effectuant des requêtes

La connexion avec le SGBD se fait avec des « sockets » (programmation client/serveur) et le flux de données transitant entre le script et le serveur est en général en binaire. Il faut donc utiliser l'API de développement fourni par le concepteur du SGBD.

La librairie PDO permet ainsi d'envoyer des requêtes au format SQL et de recevoir des données au format texte.

2.Librairie / Classes PDO

PHP dispose de deux méthodes pour communiquer avec un SGBD :

- Soit avec des librairies spécifiques aux moteurs de base de données, mysqli[2] , pgSQL[3], sqlite[4] etc.
- Soit avec PDO pour l'ensemble des moteurs de base de données

La première solution est moins souple en cas de migration vers un autre SGBD, PDO est une couche d'abstraction et dans ce cas il est très facile de migrer.

1 http://php.net/manual/fr/book.pdo.php
2 http://php.net/manual/fr/book.mysqli.php
3 http://php.net/manual/fr/book.pgsql.php
4 http://php.net/manual/fr/book.sqlite3.php

Description de PDO au niveau logiciel
L'extension PDO comprend les trois classes suivantes :

- La classe **`PDO`**, qui permet de créer des objets représentant la connexion de base et qui dispose des méthodes permettant de réaliser les fonctions essentielles, à savoir l'envoi de requête, la création de requêtes préparées et la gestion de transactions.
- La classe **`PDOStatement`**, qui représente, entre autre, une requête préparée ou un résultat de requête « **`SELECT`** ». Ses méthodes permettent de gérer les requêtes préparées et de lire le résultat obtenu.
- La classe **`PDOException`**, qui permet de gérer et d'afficher des informations sur les erreurs à l'aide d'objets.

3.Accès séquentiel

PDO nous oblige, au même titre que les autres méthodes, de respecter un ordre séquentiel pour utiliser un SGBD.

Différentes phases à respecter

1 – On se connecte au moteur de base de données en fournissant le type de SGDB, l'adresse du serveur, le nom de la base, le compte et le mot de passe de l'utilisateur

2 - On prépare la requête, souvent associée à une variable de type chaîne

3 – On émet la requête (méthodes différentes entre une requête de type recherche ou de type modification)

4* – Si recherche, on récupère le résultat au format binaire

5* – On transforme les données reçues en format texte et bien souvent dans un tableau

6* – On traite les données*

7 – On clôt la connexion

4.Description des différentes phases

4.1-Connexion au SGBD (MySQL)

Pour la connexion, PDO s'appuie sur la classe **`PDO`** (via un appel du constructeur `__construct()`) :
Elle attend :

- le driver appelé sous-entendu le type de SGDB , `mysql – sqlite3 – pgsql` etc.
- le serveur SGDB (adresse IP ou nom)
- le nom de la base de données
- le compte utilisateur et son mot de passe

Toutes ces valeurs sont des chaînes de caractères qui peuvent être contenues dans des variables ou constantes (***on préférera des constantes***).

* En cas de recherche de données

Syntaxe :
objet = **PDO("***driver***:host=***adresse_serveur***;dbname=***base_de_données***","***compte***","***mot_de_passe***")**

`PDO()` instancie ***un objet*** sur lequel il faudra s'appuyer pour la suite.

Il faut anticiper les problèmes éventuels liés aux erreurs de connexion, pour cela on utilise le gestionnaire d'exception. Il est utilisé avec toutes les extensions objet de php comme PDO.

Cela consiste à « tenter » une connexion et en cas d'erreur, on intercepte l'exception pour la gérer par la suite. Cela permet de contrôler toutes les phases d'exécution du script, il n'y a pas de place à un mode de fonctionnement aléatoire.

Pour cela on utilise `try … catch` :

```
try{
      //tentative de connexion au SGBD
      $connexion = … ;
catch(PDOException $except){

      //Gestion de l'exception en récupérant  le message d'erreur par exemple :
      echo "Message d'erreur : ".$except->getMessage() ;
}
```

La classe `PDOException` fournit les méthodes nécessaires pour gérer l'exception. Dans l'exemple ci-dessus, l'objet `$except` (instance de la classe) dispose de toutes les méthodes et propriétés (voir la liste → `http://php.net/manual/fr/class.pdoexception.php`).

Remarque
En P.O.O avec PHP, la syntaxe objet/methode et objet/propriété est :

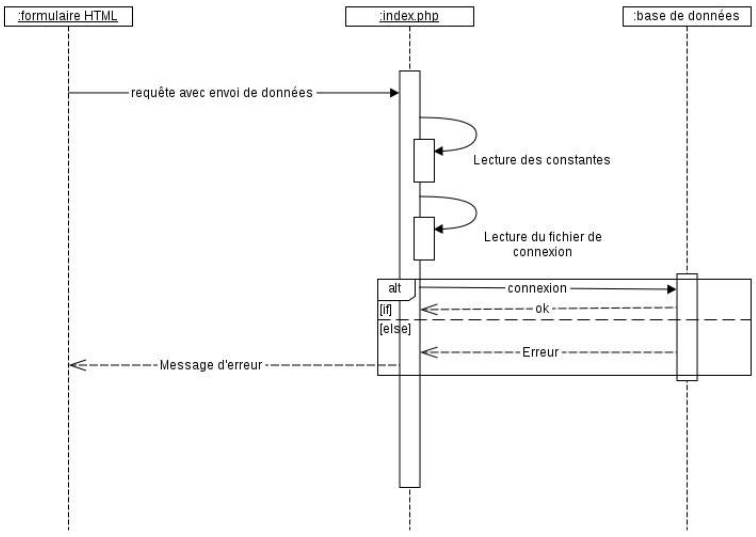
```
      objet->methode() ;
      objet->propriete ;
```

Le symbole **->** assure la liaison entre les éléments. Comme dans l'exemple ci-dessus

```
$except->getMessage() ;
```

Exercice :

Soit le diagramme de séquence ci-dessous :



Fichiers :

- `index.php`, appelé par défaut (dispatcher)
- `ctes.inc.php`, contient les constantes nécessaires pour la connexion à la base de données
- `connexion.php`, script qui gère la connexion au SGBD
- `afficheMessage.php` contient un message d'erreur formaté
- `body.php` contient le corps HTML a généré en cas de connexion

« index.php » prends en compte les fichiers externes (ctes.inc.php et connexion.php) avant de procéder à la connexion.

Dans le fichier ctes.inc.php sont définies les constantes suivantes :

- _HOTE_ dont la valeur est « localhost »
- _BASE_ valeur est « maBase »
- _USER_ compte utilisateur mysql valeur «felix »
- _MDP_ mot de passe mysql valeur « felix22 »

1-Écrivez le code d'index.php en supposant que la fonction connexionBDD() , du fichier connexion.php, assure la connexion avec le SGDB et est de type booléen :

valeur de retour true → la connexion s'est bien passée
valeur de retour false → connexion impossible

et que le fichier contenant les constantes de connexion se trouve dans le même répertoire qu'index.php. Dans le cas d'une erreur de connexion, on affichera le contenu HTML généré par « afficheMessage.php » et le script s'arrêtera. Dans le cas contraire on affichera le corps de la page avec le fichier « body.php ».

```php
<?php
	require "ctes.inc.php" ;
	require "connexion.php" ;
	$temp="";
	if(!($connexion=connexionBDD())){
		include "afficheMessage.php" ;
		die() ;
	}
	include "body.php" ;
?>
```

2-On vous demande d'écrire le fichier ctes.inc.php :

```php
<?php
	define("_HOTE_","localhost") ;
	define("_BASE_","maBase") ;
	define("_USER_","felix") ;
	define("_MDP_","felix22") ;
?>
```

3-Écrivez le code de connexion.php

```php
<?php
	function connexionBDD(){
		$serveur="mysql:host="._HOTE_.";dbname="._BASE_ ;
		try{
			$cnx=new PDO($serveur,_USER_,_MDP_);
			return $cnx ;
		}catch(PDOException $except){
			$temp=$except->getMessage() ;
			return false ;
		}
	}
?>
```

4.2-Émission de requêtes

On distingue deux types de requêtes :

- Celles qui n'exigent pas de résultat comme `INSERT`, `UPDATE` ou `DELETE`
- La recherche avec `SELECT` qui retourne un résultat

PDO distingue ces deux types en utilisant deux méthodes différentes à savoir `exec()` et `query()`.

Exemple 1 – Requête de mise à jour avec `exec()`

On souhaite mettre à jour l'email d'un utilisateur identifié avec son uid, en supposant que l'on travaille sur une seule table.

```
1.  <?php
2.          require "ctes.inc.php" ;
3.          require "connexion.php" ;
4.          $temp="";
5.          if(!($connexion=connexionBDD())){
6.                  include "afficheMessage.php" ;
7.                  die() ;
8.          }
9.          $requete = "update utilisateur set adresseEmail='$mail' where id='$uid'" ;
10.         if(($nb=$connexion->exec($requete))===false){
11.                 include "afficheMessage.php" ;
12.                 die() ;
13.         }
14.         echo "Nombre de champs modifié : $nb" ;
15. ?>
```

La ligne 9, la requête est une chaîne de caractère dans laquelle on peut insérer des valeurs de variables. On notera la présence des apostrophes encadrant les valeurs utilisées en SQL.

Ligne 10, la requête est transmise au SGBD avec `exec()` , le test de la valeur de retour permet de s'assurer que cela s'est bien passé ou non. On notera les 3 = qui s'assurent de la valeur et du type (booléen & false). $nb contient le nombre de champs affectés par la requête (ici ce sera 1).

Exemple 2 – Recherche de valeur

On veut tous les champs des utilisateurs triés par ordre alphabétique.

```
1.      <?php
2.              require "ctes.inc.php" ;
3.              require "connexion.php" ;
4.              $temp="";
5.              if(!($connexion=connexionBDD())){
6.                      include "afficheMessage.php" ;
7.                      die() ;
8.              }
9.      $requete = "select * from utilisateur order by nom asc;" ;
10.     if(!($envoi=$connexion->query($requete))){
11.             include "afficheMessage.php" ;
12.             die() ;
13.     }
14.     //Récupération des résultats…
15.     ?>
```

Ligne 9, la requête est une chaîne de caractère comme précédemment.
Ligne 10, la méthode query() retourne false en cas d'erreur sinon un objet (ici $envoi) qui permet de récupérer par la suite les résultats provenant du SGBD.

4.3-Récupération des résultats

La communication avec un SGBD se fait via des sockets, système de dialogue entre client/serveur. Cela permet d'établir logiciellement des connexions à partir des protocoles réseaux usuels (TCP/UDP) ou même à partir d'un protocole spécifique hors datagramme réseau comme MySQL (raw sockets – sockets brutes).
Les requêtes/données émises sont au format binaire et encapsulées, PDO fournit les méthodes qui assurent l'extraction et la conversion texte ↔ binaire.

Plus d'infos : `https://dev.mysql.com/doc/internals/en/client-server-protocol.html`

En reprenant l'exemple 2 :

La table « `utilisateur` » compte 4 champs :

Id (clé primaire)	Nom	Prenom	Email

Donc pour chaque enregistrement on doit récupérer 4 valeurs :

1	Dupont	Jean	jdupont@dupont.fr
2	Martin	Marie	martin@martin.org
3	Durant	Paul	Paul-durant@durant.fr
4	Thomas	Morgane	morgane@thomas.bzh

D'un point de vue programmation, il faut récupérer les 4 champs dans un tableau. Et si on balaye la totalité de la table, chacun de ces tableaux (pour chaque enregistrement) pourrait être stocké dans un tableau global. Cela signifie que la requête doit générer un tableau à 2 dimensions, chaque ligne de celui-ci serait un tableau.

PDO dispose de plusieurs méthodes permettant cela :

-fetch(TYPAGE_DU_RESULTAT), retourne le résultat d'un enregistrement sous forme de tableau
-fetchAll(), retourne tous les résultats dans un tableau

Principaux arguments de fetch() :

- `PDO::FETCH_NUM` , le tableau des résultats est de type indicé
- `PDO::FETCH_ASSOC`, tableau de type associatif, les clés portent les noms des colonnes de la table
- `PDO::FETCH_BOTH`, combine les deux types précédents (tableau à la fois indicé et associatif)

La lecture d'un enregistrement au format indicé en partant de l'exemple 2 :

```
1.      $requete = "select * from utilisateur order by nom asc;" ;
2.      if(!($envoi=$connexion->query($requete)){
3.              include "afficheMessage.php" ;
4.              die() ;
5.      }
6.      $resultat = $envoi->fetch(PDO::FETCH_NUM) ;
```

Ligne 6, $resultat est un tableau indicé qui ne contient que le premier enregistrement :

```
$resultat[0] → contient l'id  = 1
$resultat[1] → le nom         = Dupont
$resultat[2] → le prenom      = Jean
$resultat[3] → l'email        = jdupont@dupont.fr
```

Exercice

En reprenant le code ci-dessous (cf exemple 2), modifiez le de façon à ce que l'on affiche la totalité des résultats en s'appyuant sur un tableau associatif pour chaque lecture :

id : 1 ;nom : Dupont ;prenom : jean ;email : jdupont@dupont.fr ;
id : 3 ;nom : Durant ;prenom : Paul ;email : Paul-durant@durant.fr ;
id : 2 ;nom : Martin ;prenom : Marie ;email : martin@martin.org ;
id : 4 ;nom : Thomas ;prenom : Morgane ;email : morgane@thomas.bzh ;

← Capture du résultat attendu

```php
<?php
	require "ctes.inc.php" ;
	require "connexion.php" ;
	$temp="";
	if(!($connexion=connexionBDD())){
		include "afficheMessage.php" ;
		die() ;
	}
	$requete = "select * from utilisateur order by nom asc;" ;
	if(!($envoi=$connexion->query($requete))){
		include "afficheMessage.php" ;
		die() ;
	}
	while($ligne=$envoi->fetch(PDO::FETCH_ASSOC)){
		foreach($ligne as $cle => $valeur){
			echo "$cle : $valeur ;" ;
		}
		echo "<hr>" ;
	}
	...
```

Remarque
L'écriture **`PDO::`** permet de définir l'espace de nom auquel appartient la méthode ou la propriété. Cela évite d'avoir des effets de bords sur des nommages.

4.4-Fin du traitement de la requête

Il faut libérer la ressource pour l'exécution d'une nouvelle requête, pour cela on utilise `closeCursor()`.

La fin du code précédent doit se terminer ainsi :

```php
	$envoi->closeCursor() ;
```

Une seconde requête pourra être exécutée. Attention cette méthode est exigible suivant les drivers utilisés.

4.5-Gestion des erreurs de requêtes

Les erreurs peuvent être interceptées pendant toute la phase de traitement. L'objet identifié par la connexion dispose de la méthode errorInfo(). Elle crée un tableau stockant les erreurs renvoyées par le SGBD.

Par rapport à l'exemple précédent :

```
$erreur = $connexion->errorInfo() ;
```

$erreur est un tableau indicé à 3 cellules :
`$erreur[0]` => code erreur SQL State
`$erreur[1]` => code erreur spécifique au driver
`$erreur[2]` => Message textuel d'erreur spécifique au driver

4.6-Préformater les données

Le formalisme des requêtes nous impose une syntaxe particulière pour l'ajout de variable dans la requête. D'un point de vue SQL, une valeur doit toujours être comprise entre des apostrophes (exemple 1 page 5). Il faut s'assurer que le format est respecté avec une protection de la chaîne (requête) :

```
$requete = $connexion->quote($requete);
```

4.7-Compter les occurrences

PDO fournit la méthode « `rowCount()` » permettant de compter le nombre de résultats trouvés à partir d'une requête de type « `SELECT` » :

```
$nbReponses = $envoi->rowCount();
```

5.Procédures Stockées et Requêtes Préparées

5.1-Procédures stockées

Les procédures stockées sont l'équivalent de fonctions crées en SQL. On procède à sa déclaration en lui donnant un nom puis on lui associe un « programme » en SQL. L'intérêt est de définir sous un nom, un ensemble d'instructions plus ou moins complexe qui peuvent être appelées autant de fois que possible.

PDO permet d'utiliser les procédures stockées à partir de PHP :

Exemple

coté SQL :

```
DELIMITER |
CREATE PROCEDURE listeUtilisateur()
BEGIN
      select * from utilisateur order by nom asc ;
END |
```

Note, l'appel d'une procédure en SQL se fait de la façon suivante : `CALL listeUtilisateur() |`

Coté PHP avec PDO, une fois la procédure déclarée en SQL :

```
$envoi=$connexion->prepare("CALL listeUtilisateur()") ;
$envoi->bindParam(1,$utilisateurs,PDO::PARAM_STR) ;
$envoi->execute() ;
```

`$utilisateurs` contient les résultats renvoyés par la procédure stockée.

`prepare()`, permet de « compiler » l'instruction ou l'appel de la procédure avant son exécution.

`bindParam()`, associe des variables PHP stockant soit des critères dynamiques soit des valeurs de retours de requête. Les paramètres seront expliqués au paragraphe suivant.

`execute()`, comme son nom l'indique, exécute la requête ou l'appel de la procédure stockée.

Remarque
Il faut respecter l'ordre des appels des différentes méthodes, 1 → `prepare()` , 2 → `bindParam()` puis 3-> `execute()`.

5.2-Requêtes préparées

Les requêtes préparées permettent de créer des requêtes SQL qui ne sont pas directement utilisables mais qui contiennent des paramètres auxquels il est possible de donner des valeurs différentes en fonction des besoins comme des appels répétitifs par exemple.
Elles sont compilées et utilisées à la façon d'une procédure stockées, on retrouve les méthodes présentées précédemment.

Déclaration et utilisation de requête préparée :

1-Déclaration à l'aide de `prepare()`, le SGBD traite la demande (vérifie la validité) puis compile afin d'optimiser le traitement lors d'un appel.
2-On associe les variables (leurs valeurs) à la requête avec `bindParam()`
3-On demande l'exécution de la requête avec `execute()`

Les phases 2 & 3 sont alors répétées autant de fois que possible.

Exemple

```
1.  $requete="update utilisateur set utilisateur.droit =:nvxdroit where utilisateur.nom=:nom" ;
2.  $envoi->prepare($requete) ;
3.  $envoi->bindParam(":nvxdroit","admin",PDO::PARAM_STR) ;
4.  $envoi->bindParam(":nom","jean",PDO::PARAM_STR) ;
5.  $envoi->execute() ;
6.  $envoi->bindParam(":nvxdroit","user",PDO::PARAM_STR) ;
7.  $envoi->bindParam(":nom","marie",PDO::PARAM_STR) ;
8.  $envoi->execute() ;
9.  $envoi->bindParam(":nvxdroit","webmaster",PDO::PARAM_STR) ;
10. $envoi->bindParam(":nom","morgane",PDO::PARAM_STR) ;
11. $envoi->execute() ;
```

ligne 2, la requête est envoyée au SGBD, elle va être vérifiée puis compilée
lignes 3-4, 6-7,9-10, association variables → valeurs, le dernier paramètre permet de typer la valeur transmise soit `PDO::PARAM_STR` = chaîne de caractère (Les autres types ->`http://php.net/manual/fr/pdo.constants.php`)
lignes 5,8 et 11, la requête est exécutée avec les valeurs transmises par bindParam().

Note : Les champs « variables » dans la requête doivent commencer par « **:** ».

Exercice
Supposons que l'on souhaite transférer des utilisateurs inscrits dans un fichier dans la table utilisateur.
Le fichier est de type CSV, sa structure est « `nom;prenom;adresse_email` » pour chaque enregistrement.
La table est définie page 6, il ne sera pas nécessaire de prendre en charge l'id, cela se fait automatiquement en Auto-Incrément par le SGBD. La requête SQL est de type « `insert into` ».
On vous demande d'écrire le programme permettant de faire ce transfert automatiquement en vous appuyant sur les requêtes préparées.

```
$tabUtilisateur=preg_split(";",file("utilisateurs.txt")) ;
$requete="insert into utilisateur(nom,prenom,email) value(:nomU,:prenomU,:emailU)" ;
$envoi=$connexion->prepare($requete) ; //phase 1
foreach($tabUtilisateur as $cle=>$valeur){
      $envoi->bindParam(":nomU",   $tabUtilisateur[$cle][0],PDO::PARAM_STR); //phase 2
      $envoi->bindParam(":prenomU",$tabUtilisateur[$cle][1],PDO::PARAM_STR);
      $envoi->bindParam(":emailU", $tabUtilisateur[$cle][2],PDO::PARAM_STR);
      $envoi->execute(); //phase 3
}
```

6.Les transactions

A supposer que sur un site marchand, un client effectue un achat et il faut, d'un point de vue développement, réaliser simultanément deux insertions dans deux tables différentes. Si pour une raison quelconque, un problème survient et que la seconde insertion ne se fait pas alors que la première l'est déjà. La base contient alors une incohérence.
Les transactions permettent de gérer ce genre de problème en effectuant des commandes tout ou rien : soit aucune des deux soit toutes les deux.
Le langage SQL possède les commandes pour le faire mais PDO nous fournit des méthodes qui permettent de gérer les transactions sans y faire appel.

Une transaction doit commencer par la méthode `beginTransaction()` de l'objet PDO qui désactive le mode « autocommit » de MySQL (chaque modification effectuée est immédiatement enregistrée par défaut).
L'envoi des requêtes est toujours effectué par les méthodes `exec()` et `query()`, après avoir vérifié que les requêtes ont bien été exécutées, il suffit de valider l'ensemble avec la méthode `commit()`.
Dans le cas contraire, les requêtes sont annulées avec la méthode `rollBack()`.

Exemple

```
1.  //début de la transaction
2.  $connexion->beginTransaction();
3.  //définition des deux requêtes
4.  $requeteA="INSERT INTO utilisateur(uid,nom,prenom,email)
          VALUES('/N','".$nom."','".$prenom."','".$email."')";
5.  $requeteB="INSERT INTO listeDiffusion(mail) VALUES('".$email."')";
6.  //Insertion des données
7.  $envoi=$connexion->exec($requeteA);
8.  $envoi+=$connexion->exec($requeteB);
9.  if($envoi==2){
10.       $connexion->commit(); //validation des requêtes
11.       echo "Requêtes réussies : $envoi";
12. }else{
13.       $connexion->rollBack();
14.       $erreur=$connexion->errorInfo();
15.       echo "Opérations annulées, erreur ".$erreur[2];
16. }
```