Chapitre :
Interaction web

I. Éléments sur HTML

1. Description courte du langage HTML

L'**Hypertext Markup Language**, abrégé en **HTML**, est un langage utilisé pour écrire des pages Internet. Il utilise des balises (Markups) et permet de faire de l'hypertexte (le fait de pouvoir lier des mots d'un texte à une nouvelle page), d'où son nom.
Le contenu d'une page HTML peut être très varié :

- Du simple texte
- Des images (fixes ou animées)
- De la vidéo
- Des formulaires (permettant d'écrire un mot, de cocher des cases, etc.)
- Et surtout des liens vers d'autres pages

Le type de contenu qui peut être intégré dans une page a évolué avec le langage HTML, qui en est actuellement à la version 5 (la première à permettre entre autres d'inclure directement des vidéos, sans passer par flash). Il existe également d'autres langages, comme le XHTML (le X étant pour *Extensible*) qui lui est très similaire.

2. Structure d'une page HTML

a. Fichier html

Le code d'une page Internet, écrit en HTML, s'écrit dans un simple éditeur de texte (plus simple qu'un logiciel de traitement de texte) tel que notepad++ qui permet la coloration syntaxique. Le fichier créé a en général pour extension .html (ou parfois .htm).
Il est toujours possible d'enregistrer une page d'une site Internet sur son ordinateur à partir du navigateur (avec « Enregistrer sous... »). On peut alors en général voir le contenu (mais pas toujours complet) de la page enregistrée, même sans être connecté.
On peut s'aider de la documentation trouvée sur de nombreux sites pour créer une page Internet et connaître entre autres toutes les balises. Il y a beaucoup de sites en anglais, en particulier les sites officiels des langages (HTML, XHTML), comme par exemple https://html.spec.whatwg.org/multipage/ pour le HTML. Le W3C (World Wide Web Consortium) est une communauté de personnes qui travaillent ensemble au développement de standards du Web, c'est lui qui définit les standards des langages pour les pages Internet.
On trouve évidemment des pages en français, comme http://www.aliasdmc.fr/courshtml/
On peut toujours faire des recherches sur Internet et trouver d'autres sites qui traitent des points précis ou expliquent autrement certaines choses.
Le code source d'une page HTML a en principe au minimum une structure telle que la suivante :

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   Exemple de HTML
  </title>
 </head>
 <body>
  Ceci est une phrase avec un <a href="cible.html">hyperlien</a>.
  <p>
   Ceci est un paragraphe sans hyperlien.
  </p>
 </body>
</html>
```

La toute première ligne indique au navigateur la version du langage utilisée (ici HTML).
Ensuite commence le code proprement dit de la page. Tous les mots entre "<" et ">" sont les balises. Pour chaque balise qui ouvre une structure, il y a une balise qui la referme. Cette dernière porte le même nom précédé d'un slash :
<head> ... </head>, <body> ... </body>, <a> ... , etc.
On remarque que pour certaines balises comme <a>, il est possible d'ajouter des éléments (par exemple dans <a href="cible.html">).
Certaines balises ne nécessitent pas d'être fermées par une autre balise ; on écrit alors un slash à la fin de la balise, comme par exemple <link /> ou <img /> que l'on verra plus bas.
On observe généralement une indentation du texte qui permet une lecture claire de la structure.
En particulier, tout le code est entre <html> et </html>, et on observe deux parties principales à l'intérieur : l'entête (<head>) et le corps (<body>).

b. Entête

L'entête, délimitée par les balises <head> et </head>, contient les informations générales sur la page. C'est ici que l'on donne le titre de la page avec <title>. On peut ajouter d'autres éléments comme :

```
<meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
<link rel="icon" href="Docs/icone.gif"/>
<link rel="stylesheet" type="text/css" href="style.css"/>
<script type="text/javascript" src="java.scripts" ></script>
```

<meta> permet d'indiquer plusieurs choses comme l'auteur, le type de codage des caractères, etc.
<link /> permet de donner des adresses de fichiers qui peuvent être l'icône de la page, contenir le style de la page (que l'on verra plus tard), etc.
<script> permet de donner l'adresse d'un fichier (ou directement le texte), contenant un code (ou script) de nature différente du html qui peut être utilisé dans la page, en particulier du code JavaScript.
Il en existe d'autres.

c. Corps

Le corps, délimité par les balises <body> et </body>, est la partie qui décrit le contenu de la page. C'est donc en principe la partie la plus importante dans le code.
Ici les balises indiquent la structure de la page.

```
<h1>Titre 1</h1>
  <h2>Sous-titre 1</h2>
  <ul>
    <li>Premier item d'une liste</li>
    <li>Second item d'une liste</li>
  </ul>
  <p>Un paragraphe</p>
```

Il existe beaucoup de balises. Principalement :
<h1>, <h2>, ... permettent de mettre des titres hiérarchisés (<h1> étant le plus grand, <h6> le plus petit).
 et <li> permettent de faire des listes. Il existe différentes sortes de listes (numérotées, à puce, etc) et donc de balises.
Et d'autres encore (tableaux, formulaires, vidéos...) !
Il faut tout de même citer deux autres balises parmi les plus communes. Le code permettant de faire un lien vers une autre page est le suivant :

```
<a href="adresse de la page">Texte sur lequel on clique</a>
```

L'adresse de la page (ou du document) peut être **absolue** (http ://...) ou **relative** (Documents/...).
On ajoute une image ainsi :

```
<img src="adresse de l'image" alt="texte si pas d'image"/>
```

Et l'on peut tout à fait rendre une image « cliquable » en utilisant `href` et en insérant une image plutôt qu'en écrivant du texte entre les balises <a> et .

II. CSS

Le CSS (Cascading Style Sheet) est un langage permettant d'ajouter des indications d'apparence (de style) à une page HTML. Il peut être présent directement dans le code HTML, ou bien donné dans un fichier externe dont l'adresse est donnée en entête.
Globalement on peut retenir qu'il est possible de définir des attributs (taille, couleur, etc) à tout élément (titre, texte simple), en partie pour le corps (body) :

```
body{
background-color:#f8f8d6;
color:#1c1c1c;
}
```

mais aussi pour tous les autres éléments :

```
h1{
text-align:center;
color:#101e2e;
font-size:x-large;
text-transform:uppercase;
border-top:2px dotted #101e2e;
padding-top:0.5em;
}

p{
font-size:medium;
margin:0 0 0 1em;
}

a{
color:#0003ac;
text-decoration:none;
background-color:transparent;
}

a:hover{color:#8da0ff;}

a:visited{color:#b74cbb;}
```

Les dernières lignes permettent de donner des couleurs différentes selon que la souris est sur le texte ou que le lien a déjà été visité.
Si l'on veut définir des attributs communs à plusieurs éléments, on liste les différents éléments en séparant par des virgules :

```
h2, h3{
color:#103e2e;
}
```

On peut définir des styles différents selon l'identité (id) ou la classe (class) d'un élément, les identités et classes étant définies dans le code HTML, au sein des balises des éléments en question.
Sur une page il ne peut y avoir qu'un élément ayant une identité donnée, alors que les classes peuvent être partagées par plusieurs éléments.
Ce qui concerne les objets ayant l'identité page est introduit dans le fichier css par un `#page` (le dièse indique une identité) :

```
#page{
margin:1% 1%;
color:#000;
background-color:#ffffbe;
border:3px solid #000;
height:700px;
}
```

Ce qui concerne les objets ayant la classe `c1` est introduit par un `.c1` (la classe est indiquée par un point) :

```
.c1{
color:#655f4b;
background-color:transparent;
}

a.c1{
color:#b00606;
display:block;
font-weight:bold;
margin-left:-3em;
padding-left:4px;
text-decoration:none;
}
```

Un élément ayant cette classe `c1` aura une certaine couleur (`#655f4b`). Un lien (`<a>`) dans un élément de classe `c1` aura une couleur particulière (`#b00606`), mais le fond (`background-color`) restera le même que celui défini pour la classe `c1` (c'est le phénomène de cascade : la définition la plus générale reste la même pour les cas particuliers tant qu'elle n'est pas modifiée).
On peut également définir des attributs différents pour des éléments de même type mais ayant une identité (ou une classe différente) :

```
#menu .c1{
color:#444;
}

#droite .c1{
color:#bbb;
font-size:medium;
}
```

L'apparence d'un élément de classe `c1` n'aura pas la même apparence dans menu que dans droite.

Des éléments d'explications plus complets (en français) se trouvent ici : http://www.zonecss.fr/courscss/.

III. JavaScript

Pour davantage d'interaction entre l'homme et la machine sur le web, on utilise le langage de script JavaScript.
On a pu voir à la section précédente les éléments du code HTML qui introduisent du script JavaScript : c'est avec la balise `<script>`.
Les scripts peuvent être écrits dans le fichier HTML, ou bien dans un fichier externe, dont l'adresse est renseignée en entête du fichier HTML.
Le JavaScript permet de faire beaucoup de choses, en particulier de gérer des événements (comme des clics de souris, des survols d'éléments par la souris, etc.) mais aussi de modifier dynamiquement le code HTML de la page en cours, par le biais du DOM (*Document Object Model*).
C'est aussi un langage de programmation, avec lequel on peut donc manipuler des variables, utiliser des structures conditionnelles, des boucles, etc.
En guise de découverte, suivre les instructions données à la page suivante :
https://pixees.fr/informatiquelycee/n_site/nsi_prem_js.html
Entre autres sources d'informations et de formation sur JavaScript, on peut citer les suivantes :

- https://fr.wikipedia.org/wiki/Syntaxe_JavaScript
- https://developer.mozilla.org/fr/docs/Web/JavaScript
- http://www.gchagnon.fr/cours/dhtml/exercices.html

On donne ici quelques éléments utiles pour réaliser les exercices :
On peut créer des boutons et les associer à une action exécutée par une fonction définie en JavaScript. Pour cela, la partie du code HTML doit ressembler à :

```
<button onclick="fonction_a_executer()">Texte du bouton</button>
```

Côté JavaScript, les variables sont initialisées de manière suivante :

```
x = 2
var s = "Du texte"
const c = 5
```

Les variables initialisées sans mot clé sont globales.
Celles qui sont introduites par le mot clé `var` sont locales à la fonction dans laquelle elles sont définies.
Le mot clé `const` permet de définir une constante, donc non modifiable.
Pour modifier la valeur d'une variable, il ne faut plus utiliser le mot clé :

```
x = x+2
```

L'instruction de saisie peut être effectuée avec la fonction `prompt` :

```
var x = prompt("Donner la valeur de x :")
```

À l'exécution, une fenêtre popup apparaît et permet d'entrer la valeur.
Une manière plus élégante consiste à définir une zone d'entrée dans le code HTML avec une certaine identité :

```
<input type="text" id="n" value="">
```

Puis, dans le code JavaScript, dans l'exécution d'une fonction, aller chercher son contenu (donné entre temps par l'utilisateur) :

```
var n = Number(document.getElementById("n").value)
```

Si c'est une chaîne de caractères que l'on doit récupérer, il suffit de ne pas mettre `Number`.
Pour afficher (une variable, une chaîne de caractère, etc.), on peut utiliser la fonction `alert` qui fait apparaître également une fenêtre popup :

```
alert("la valeur de x est "+x)
```

Pour afficher dans la page HTML elle-même, on peut par exemple créer dans le fichier HTML un contenu ayant une identité donnée :

```
<div id="MonContenu"></div>
```

Puis définir son contenu dans le script JavaScript :

```
document.getElementById("MonContenu").innerHTML="la valeur de x est "+x
```

L'instruction conditionnelle « Si ... Alors » a la syntaxe suivante :

```
if (expression1)
 {
   // instructions réalisées si expression1 est vraie;
 }
 else if (expression2)
 {
   // instructions si expression1 est fausse et expression2 est vraie;
 }
 else
 {
   // instructions réalisées dans les autres cas;
 }
```

La boucle Pour a la syntaxe suivante :

```
for (initialisation;condition;instruction de boucle) {
   /*
      instructions exécutées à chaque passage dans la boucle
      tant que la condition est vérifiée
   */
 }
```

Exemple, une boucle faisant aller `i` de 0 à 9 :

```
for (i=0;i<10;i++) {
    // instructions quelconques
 }
```

La syntaxe de la boucle Tant que :

```
while (condition) {
   // instructions exécutées tant que la condition est vraie ;
 }
```

IV. Formulaires HTML

⊛ **Activité :** fiche sur les formulaires HTML

V. Interaction client-serveur

en HTML « pur », il y a une interaction principale entre l'humain et la machine, celle qui consiste à cliquer sur un lien hypertexte.
Décrivons brièvement et de manière incomplète ce qui se produit lorsque l'on clique sur un lien (rappel de programme de seconde en SNT) :
Cliquer sur un lien revient à taper l'adresse de ce lien dans la barre d'adresse du navigateur. Par suite, le protocole DNS (Domain Name System) associe le nom du site à son adresse IP (identifiant le serveur de la page pour les machines du réseau Internet). Le navigateur se connecte alors au serveur de la page en faisant une **requête HTTP**.
En réponse, le serveur envoie le code contenu dans le fichier HTML de la page voulue. Cela se fait sous forme de paquets dont la bonne réception est gérée par le protocole TCP (Transmission Control Protocol) via IP (Internet Protocol). Les paquets sont alors assemblés par le client (le navigateur) qui reconstitue le fichier HTML et l'affiche pour l'utilisateur.
Lorsque l'on parcourt Internet à l'aide d'un navigateur il s'opère un dialogue entre le client (l'ordinateur duquel on demande les pages Web) et le serveur (l'ordinateur qui possède les ressources sur site visité).
Ce dialogue s'effectue au moyen d'un protocole, le protocole HTTP (HyperText Transfer Protocol). Lorsque le client souhaite une ressource, il fait une requête HTTP (ou HTTPS, dans le cas où la connexion peut être sécurisée par cryptage). Une telle requête contient plusieurs éléments dont :

- la méthode employée pour effectuer la requête. Il peut s'agir de GET, POST, mais aussi HEAD, PUT et bien d'autres.
  - La méthode GET est la plus fréquemment utilisée. Elle permet d'obtenir simplement la ressource souhaitée.
  - La méthode POST permet d'ajouter des données nécessaires au traitement de la requête pour obtenir la ressource
  - La méthode HEAD permet de ne recevoir que l'entête, donc les informations, de la ressource.
  - La méthode PUT permet de modifier la ressource. Cela n'est évidemment possible que si les client a les droits nécessaires pour modifier des données sur le serveur.
- l'URL (Uniform Resource Locator) de la ressource, autrement dit son adresse.
- la version du protocole utilisé par le client, souvent HTTP 1.1 ou HTTP 2.0
- le navigateur employé (Firefox, Chrome) et sa version
- le type du document demandé, généralement text/html

Le serveur, après réception de la requête, renvoie à son tour une réponse. Celle-ci contient :

- La version du protocole utilisée (http 1.1 ou http 2.0)
- Le code de retour de la réponse et son texte associé. Par exemple :
  - 200 OK
  - 403 FORBIDDEN
  - 404 NOT FOUND
  - 500 INTERNAL ERROR

- Si le code de retour est 200, la ressource est disponible et suit l'entête de la réponse.

Il peut être intéressant de savoir que pour afficher une page Web contenant des ressources disponibles à plusieurs adresses, comme par exemple des images, une requête est nécessaire pour obtenir chacune de ces ressources. Autrement dit, pour obtenir une seule page Internet, il peut être nécessaire de faire plusieurs requêtes.
Comme indiqué rapidement plus haut, HTTPS est la version sécurisée du protocole HTTP. Par « sécurisé » en entend que les données sont chiffrées avant d'être transmises sur le réseau.
Voici les différentes étapes d'une communication client - serveur utilisant le protocole HTTPS :

1. le client demande au serveur une connexion sécurisée (en utilisant "https" à la place de "http" dans la barre d'adresse du navigateur web) ;
2. le serveur répond au client qu'il est OK pour l'établissement d'une connexion sécurisée. Afin de prouver au client qu'il est bien celui qu'il prétend être, le serveur fournit au client un certificat prouvant son identité. En effet, il existe des attaques dites "man in the middle", où un serveur pirate essaye de se faire passer, par exemple, pour le serveur d'une banque : le client, pensant être en communication avec le serveur de sa banque, va saisir son identifiant et son mot de passe, identifiant et mot de passe qui seront récupérés par le serveur pirate. Afin d'éviter ce genre d'attaque, des organismes délivrent donc des certificats prouvant l'identité des sites qui proposent des connexions https.
3. À partir de ce moment-là, les échanges entre le client et le serveur seront chiffrés grâce à un système de clé de chiffrement (nous aborderons cette notion de clé de chiffrement l'année prochaine, en terminale). Même si un pirate arrivait à intercepter les données circulant entre le client et le serveur, ces dernières ne lui seraient d'aucune utilité, car totalement incompréhensible à cause du chiffrement (seuls le client et le serveur sont en principe aptes à déchiffrer ces données).

D'un point vu strictement pratique il est nécessaire de bien vérifier que le protocole est bien utilisé (l'adresse commence par https ://) avant de transmettre des données sensibles (coordonnées bancaires...). Les navigateurs permettent de s'en assurer, ajoutant par exemple un symbole de cadenas à l'adresse en https ;// qui montre que la connexion sera bien sécurisée.

On peut voir les requêtes et les réponses à celles-ci en utilisant la console Web du navigateur (sur Firefox par exemple).

Il est à noter que durant l'utilisation d'un navigateur Internet, lors de l'interaction de l'utilisateur avec la page Web, certains processus se font uniquement côté client, comme l'exécution de code JavaScript, alors que d'autres se font côté serveur, comme l'obtention d'une page suite à l'envoi de données via un formulaire, lors du traitement par un fichier php pour obtenir le code html qui en découle.