Le WWWeb : HTML. CSS. Formulaires. Interaction Homme-Machine. Requetes HTTP/HTTPS

Partie A : Les langages

1. Présentation.

Ce cours vous propose dans un premier temps d'écrire quelques pages en HTML, avec un peu de style en CSS, puis d'examiner comment les requêtes web sont traitées. Enfin dans une dernière partie on présente succinctement deux langages qui permettent de créer des pages web dynamiques, soit du côté client, soit du coté serveur.

Nous écrirons notre page dans un fichier de type HTML (HyperText Markup Language), qui est le format utilisé pour représenter les pages web. Ce langage est un langage de description : on se contente de décrire ce qu'il y a dans la page ; comme un titre, une liste, un tableau, un formulaire à remplir, etc…

Pour information, un système hypertexte est un système contenant des « nœuds » (par exemple des pages web, mais aussi des articles d'encyclopédie) reliés entre eux par des hyperliens permettant de passer rapidement d'un nœud à un autre. On peut considérer que l'Encyclopédie de Diderot et D'Alembert est le précurseur de l'hypertexte. Le World Wide Web (conçu en 1989 par Tim Berners-Lee) est le système hypertexte le plus vaste et le plus utilisé. Il existe d'autres formes d'hypertextes, comme des fictions écrites (des « romans »).

HTML n'est pas un langage de programmation : il ne permet aucun calcul, test, ni boucle. Pour cela, il est nécessaire de passer par l'intermédiaire d'un langage de programmation : Python, JavaScript, PHP… ce que l'on fera, très très succinctement.

Ce n'est pas non plus un langage de présentation ; la différence est subtile. On utilisera CSS (cascading style sheet, feuille de style en cascade) pour la présentation. Pour donner un exemple de la différence d'utilisation, dans html on écrira « ce texte est un titre », dans CSS on écrira « les titres doivent être écrit en police Times 24 rouge, sur fond vert, et clignoter » (pour être sûr d'avoir un site vraiment moche).

HTML et CSS sont des langages interprétés par un navigateur : Firefox, Chrome, IE…

Vous trouverez de nombreux tutoriels bien faits sur le web (ceux d'OpenClassroom sont très bien en général).

Pour écrire du code HTML, le mieux est d'avoir un éditeur adapté. Sous Windows, on peut utiliser tout simplement le bloc-notes, ou encore Notepad++ (http://notepad-plus-plus.org/fr/), qui est extrêmement simple (et fait par un Français), et colorie les mots clés lorsque l'on lui spécifie dans le menu : langage>H>HTML (coloration sémantique). Sous OsX, j'utilise SublimeText (en version gratuite). Komodo a une bonne réputation et est multi-plateformes. Avant d'écrire la moindre ligne de code, précisez dans le menu Encodage>Encoder en UTF-8 sans BOM pour avoir une compatibilité maximale.

La version actuelle du HTML est HTML5, pour le CSS c'est CSS4. Le HTML n'a plus de numéro de version, c'est un langage en constante évolution (HTML Living Standard)

2. Introduction au langage HTML

Vous pouvez suivre ce paragraphe, soit en créant une page en exemple, soit en explorant en parallèle le fichier « exemple_site.html », disponible ici http://www.maths-info-lycee.fr/exemple_site/site.html . Ouvrez ce fichier à la fois sous Firefox, et dans NotePad+.

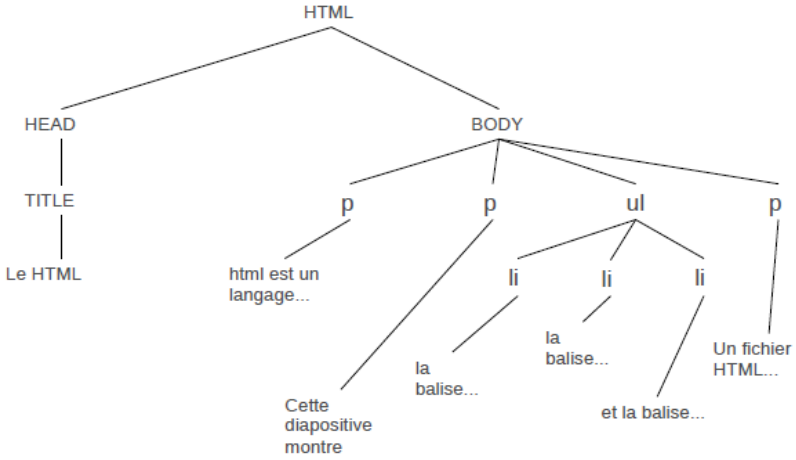
Le langage HTML est un langage à balises. Les balises sont des mots clés insérés entre < et >. Une balise s'ouvre sous la forme `<balise>`, et se ferme sous la forme `</balise>`. On peut enchâsser des balises les unes à l'intérieur des autres, en respectant l'ordre de fermeture inverse de l'ordre d'ouverture ; a contrario `<balise1> <balise2> </balise1> </balise2>` est incorrect.

a. Structure du document ; quelques détails sur l'en-tête.

Un document HTML est structuré en arborescence, il comporte deux branches principales :

- Son en-tête, entre les balises `<head>` et `</head>` ;
- Son corps, entre les balises `<body>` et `</body>`.

Le document HTML commence par `<!DOCTYPE html> <html lang = "fr">` et finit par `</html>`. Ces lignes sont présentes pour que les navigateurs lisent le fichier comme du html. La déclaration `<!DOCTYPE html>` est valable en HTML 5, elle est bien plus compliqué en HTML 4.



L'en-tête est comporte plusieurs balises importantes :

- Les balises `<meta>`, qui précisent les métadonnées du document :
  - ```
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <!-- encodage utilisé-->
    ```
  - D'autres usages de la balise `<meta>` : description de la page, mots clés, auteur, indexation de la page par les robots des moteurs de recherche ou non…
- `<title>Mon titre</title>` *`< !-- le titre de la page -->`*
- `<link rel="stylesheet" href="style.css">` *`< !-- style de la page (définition des couleurs, des polices de caractères) où style . css est un fichier dans lequel est contenu le style de tous les éléments de la page -->`*
- Un autre usage de la balise `<link>` parmi plusieurs : lien pour envoyer un message à l'auteur. Plus généralement, `<link>` est un lien sur une ressource externe.
- `<script src="script.js"></script>` *`< !-- où script.js est le fichier contenant toutes les fonctions en Javascript.-->`*
  - *On peut mettre les fonctions en javascript à l'intérieur de la page en html également, notamment si elles sont courtes et compréhensibles. Il est néanmoins plus propre de mettre toutes les ressources dans des dossiers spécialisés (un pour le css, un pour le javascript, un pour les images etc.)*

b. Quelques exemples de balises dans le corps.

Comme précisé ci-dessus, le langage HTML est un langage à balises. Quand on ouvre une balise, on la ferme ensuite. Du moins en général…HTML est très souple et tolérant avec ce qui serait considéré comme des erreurs de syntaxe dans d'autres langages.

Une balise peut avoir des attributs, qui précisent le fonctionnement de la balise (cf `<img>` ci-dessous)

Des balises courantes :

- `<h1>` de titre principal, `<h2>` titre de niveau 2 etc…
- `<p>` de paragraphe
- `<ul>` pour une liste non ordonnée (unordered list, `<ol>` pour liste ordonnée)
  - puis `<li>` pour chaque élément de la liste
- `<b>` met en gras (bold)
- `<i>` met en italiques.

Comme les caractères « < » et « > » sont utilisés par le langage, on ne peut pas les taper dans un texte. On a les équivalences dans le tableau ci-contre.

Obtenir les caractères réservés dans le corps du hmtl	
<	&lt ; (less than)
>	&gt ;
"	&quot ;
&	&amp ;

Certaines balises servent à renforcer le texte :

- `<em>` (emphase)
- `<strong>` (renforcement), etc...
- Si on ne précise pas dans une feuille de style CSS ce que cela signifie, par défaut elles correspondent respectivement à « italiques » et « gras ».
- La balise `<div>`, utilisée en général avec une feuille de style CSS (cf. ci-dessous), sous la forme `<div class = "mon_style_trop_bo">` permet de donner un style précis à une subdivision de la page.

En fin du document « exemple_site.html », figurent des exemples de balises pour faire un tableau et un formulaire, cf. ci-dessous.

Il existe aussi d'autres balises pour une présentation élégante par blocs (balises du type `<div>`, sémantiques et d'organisation).

Une balise peut comporter des attributs qui précisent ses propriétés : style, langue différente, taille des images (cf. paragraphe multimédia).

c. Les liens : l'essence du HTML
    ```

La balise fondamentale : `<a>.` C'est celle qui permet de définir un lien vers un autre document HTML (ou toute autre ressource web).
Attributs :

- `href = "mon_lien"` . Obligatoire, donne l'adresse du lien, soit localement, soit sur le web. Attention aux adresses avec les caractères réservés <, >, ", &, qui sont à remplacer comme spécifié dans le tableau ci-dessus. Remarquez que `<a href="page2.html">` définit l'hyperlien sur la deuxième page du site, qui dans ce cas doit être présente dans le même répertoire. On peut avoir une page dans un répertoire quelconque, il suffit d'en donner le chemin d'accès complet, ou relatif.
- `href = "mailto :toto@gmail.com"` ouvre l'application de messagerie électronique par défaut, avec l'adresse pré renseignée.
- `href = "mon_fichier.pdf"` permet le téléchargement d'un fichier situé dans le même dossier que la page. Cf. remarques précédentes au cas où le fichier n'est pas dans le même dossier que la page.
- `target = "_blank"` force l'ouverture dans une nouvelle fenêtre. Ne pas abuser.
- `title = "lien vers un autre site"` rajoute une infobulle, qui s'affiche en survolant le lien avec la souris.
- On peut créer une **ancre** dans une page. C'est un point de la page auquel on peut accéder directement à l'aide d'un lien. Par exemple, `<a id="nom_de_l_ancre"></a>` définit une ancre à laquelle on accèdera par `<a href=="#nom_de_l_ancre"> Aller directement à l'ancre </a>.` C'est utile pour les longues pages.

d. Balises multimédia

- Pour tous les noms de fichiers, respectez les règles suivantes pour éviter les problèmes d'encodage : pas d'accents, espaces remplacés par des caractères *underscrore* : _ .
- `<img>` insère une image. L'attribut `src = "adresse du fichier image"`, donnant son chemin d'accès, est obligatoire. Si elle est sur le web, on donnera l'adresse complète `http://www…` Sinon on donnera le chemin d'accès sur l'ordinateur. L'attribut « `alt` » contient une brève description, lue par les interfaces vocales pour les non-voyants, et utilisée par les moteurs de référencement. Cet attribut était obligatoire en HTML4. Même s'il ne l'est plus, il fait partie des bonnes pratiques. Autres attributs facultatifs :
  - `height` ou `width` permet de définir la hauteur, respectivement la largeur, de l'image en pixels. La définition *a priori* permet au navigateur de réserver un espace pour l'image avant affichage de celle-ci et fait gagner du temps lors du chargement de la page. Si vous devenez un pro du web, il est indispensable d'utiliser un de ces attributs. Si vous n'utilisez qu'un seul des attributs, les proportions de l'image seront respectées. Sinon, les résultats sont souvent très laids…
  - `title = « ceci est une image »` rajoute une infobulle, qui apparaît lorsque l'on survole l'image avec la souris

  Une remarque sur les formats d'image à utiliser pour un site. Les formats ont leur spécificités : PNG est un format avec compression et sans perte de données, JPEG avec compression et avec pertes de données. Les formats de type « RAW » sont à proscrire (très lourds). Donc :
  - Pour une photo, utilisez le JPEG en tenant compte du poids.
    - Si par exemple il s'agit du fond de votre site, compressez beaucoup. Changez la définition et le format sous Gimp.
    - Vous pouvez mettez un aperçu, et un lien pour charger la photo (cf. ci-dessus, balise `<a>`). De nombreux logiciels permettent la création de miniatures (thumbnails), comme Easy Thumbnails (Windows), ThumbsUp (OsX), mkthumb (Linux)…
    - Vous pouvez aussi compresser en JPEG, avec un taux de 15 à 20 % de compression, qui est acceptable au niveau qualité. Vérifiez dans ce cas que votre photo n'est pas trop lourde.
  - Pour un graphique avec peu de couleurs (moins de 256) : PNG 8 bits ou GIF
  - Pour un graphique coloré : PNG 24 bits
  - Pour une image animée : GIF. N'abusez pas, c'est assez pénible.
  - Évitez les autres formats.
- Les images ne sont pas toutes libres de droits, voici un lien qui recense certains sites où elles le sont : https://www.reformedulycee.fr/2019/11/documents-libres-de-droits-nos-sites-preferes/
- `<video>` insère une vidéo (comme son nom l'indique). Cette balise est récente, il est utile de prévoir un message d'erreur car des navigateurs un peu anciens ne la prennent pas en charge. Outre l'attribut `src` obligatoire, et les attributs faucltatifs `height/width` comme pour les images, on peut mettre également les attributs facultatifs suivant :
  - `controls` pour afficher les contrôles de lecture, arrêt, et volume (fortement conseillé)
  - `preload` pour télécharger la vidéo dès le chargement de la page. Ceci permet la lecture immédiate, au défaut de la lenteur de chargement. Voir le tutoriel pour l'usage de cet attribut.

- `loop` pour jouer le fichier en boucle. Limiter l'usage de cet attribut.
- `autoplay` pour jouer automatiquement la vidéo. Ce dernier attribut est peu apprécié des utilisateurs en général.
- `poster` pour afficher une image. Par défaut, la première image de la vidéo est affichée.

Aucun navigateur ne reconnaît par défaut tous les formats vidéo… Il est souvent nécessaire pour l'utilisateur final d'installer des plugins, afin de pouvoir lire lesdites vidéos.

- `<audio>` insère… un son. Outre l'attribut obligatoire src, on peut utiliser `controls, preload, autoplay`, et `loop` comme pour les vidéos. La balise est mieux implantée dans les navigateurs que `<video>`, et il y a moins de problèmes de formats également.

3. CSS.

CSS permet de spécifier la forme du document (par opposition au contenu donné par html), en précisant les attributs d'une balise.
La bonne pratique consiste à définir les styles dans un document à part, dont on en précise la localisation dans l'en-tête : `<link rel="stylesheet" type="text/css" href="exemple.css">.`
Les **sélecteurs** CSS comportent une ou plusieurs propriétés auxquelles sont affectées des valeurs. Les sélecteurs peuvent être des sélecteurs d'éléments (1er exemple ci-dessous, tous les paragraphes du document sont concernés), de classe (précisés dans la balise html par `class = "nom_de_classe"`), ou d'identifiant (précisés dans la balise html par `id = "nom_d'identifiant"`). Plusieurs éléments d'une page peuvent être de la même classe, par contre un identifiant est unique. On utilise ces derniers notamment pour associer une action JavaScript à un élément de la page.
Exemples :

- Sélecteur pour tous les paragraphes d'une même page HTML :

```
p { color : red ; border : 1 px black }
    propriété valeur propriété 2 valeurs ici
```

- Sélecteur pour une classe particulière, créée par l'utilisateur. Le nom de la classe est précédé d'un point :

```
.mon_style { color : #00bcd4 ; width : 500px }
             propriété valeur  propriété valeur
```

- Variante où le style ne s'applique que lorsque l'on survole avec la souris l'élément

```
.mon_style:hover { color : #00bcd4 ; width : 500px }
                   propriété valeur  propriété valeur
```

4. Vérification.

Il est indispensable de vérifier les documents html et CSS lorsque l'on fait un site de qualité, on dispose d'outils en ligne pour cela, entre autres :
http://validator.w3.org
http://jigsaw.w3.org/css-validator/

Dans les parties B et C, on va s'intéresser au web dynamique, c'est-à-dire que les pages html vont être modifiées suivant les actions de l'utilisateur. Dans la partie B, avec le PHP, la page sera modifiée sur le serveur (c'est-à-dire sur le site web distant) avant d'être renvoyée au client (le navigateur). Dans la partie C, avec JavaScript, la page sera modifiée sur le client (le navigateur).

PARTIE B : INTERACTION CLIENT - SERVEUR

1. Formulaires.

La saisie d'informations en html se fait au moyen de formulaires, dont un exemple de structure est :

```
<form action = "url_où_les_donnees_sont_envoyées" method = "post">
  <fieldset>
   <legend>Questionnaire intrusif</legend>
   <p>Rentrez votre nom : <input type="text" name = "nom"></p>
   <p>Rentrez votre prénom : <input type="text" name = "prénom"></p>
   <p><label for = "date_n">Rentrez votre date de naissance :</label> <input
   type="date" name = "date_n"></p>
     <p>Une case à cocher <input type="checkbox"></p>
     <p>Choisissez votre profession<br>
     <input type="radio" name="profession" value="nul" checked>magicien<br>
```

```
      <input type="radio" name=" profession " value="raté">champion du monde<br>
      <input type="radio" name=" profession " value="loser"> autre<br>
    </p>
    <p>Donnez votre signe astrologique<br>
    <select name="signe" size="3">
      <option value="boulet">Boulet</option>
      <option value="limace">Limace</option>
      <option value="clou_rouille">Clou rouillé</option>
      <option value="goudron">Goudron</option>
    </select>
    <br>
    <button>Un bouton pour envoyer le formulaire<button type = "submit">
    </fieldset>
</form>
```

La balise encadrant un formulaire est `<form>`, les attributs sont `action`, qui précise l'url où sont envoyées les données, et `method`, qui sera dans notre cas `"get"` ou `"post"` (cf. paragraphe suivant).
La balise `<input>` affiche un champ à renseigner. Comme on peut le voir dans l'exemple ci-dessus, ce champ peut être du texte, une case à cocher, un bouton dit de type « radio », pour sélectionner un choix parmi plusieurs. Le type de donnée à rentrer est précisé par l'attribut « `type` ». Du texte est rentré par défaut. On peut préciser `type="email"` , `type="url"` , `type="password"` pour respectivement une adresse mail, une adresse web, un mot de passe. D'autres types sont possibles (cf. https://www.w3schools.com/tags/att_input_type.asp). La balise `<input>`est vide, on peut lui associer une balise <label> pour donner une étiquette aux éléments à rentrer.
Tous les éléments à envoyer doivent avoir un attribut « `name` », sinon ils ne seront pas envoyés.
La balise `<fieldset>` permet d'avoir une mise en page plus esthétique, en regroupant les items. La balise `<legend>` donne une légende au champ de saisie (au cas où vous auriez un doute).

2. Requêtes http : le passage des paramètres

Le protocole HTTP (HyperText Transfert Protocol) permet à une machine client (en général sur un navigateur) de communiquer avec une machine serveur (en général hébergeant un site). Ce protocole utilise plusieurs méthodes.
Pour se connecter à un site, on utilise GET. Pour passer des paramètres, on peut utiliser GET si, par exemple, il s'agît d'une requête de recherche sur un moteur. Pour passer les paramètres d'un formulaire en vue d'un traitement par le serveur en PHP, on utilise en général POST (cf. ci-dessus `<form … method = "post">`)
Dans les deux cas, GET ou POST, les données sont transmises en clair, non chiffrées. Contrairement à ce que l'on peut lire parfois, la méthode POST n'est donc pas plus sécurisée que la méthode GET. Avant de détailler plus avant ces méthodes, remarquons que le protocole http montre ici ses limites. Un champ « password » sera masqué lors de la saisie, mais pas lors du transfert d'information entre client et serveur. C'est pour cela que le protocole HTTPS (Sécurisé) a été construit, avec chiffrage des données. HTTPS combine HTTP avec un chiffrement du type SSL ou TLS, grâce à un certificat d'authentification délivré par une autorité tierce, de confiance. Comme toujours dans le domaine de la cybersécurité, https peut être piraté, notamment avec des attaques du type Attaque de l'homme du milieu.

Voici un exemple simplifié de requête HTTP GET, du client vers le serveur :

```
GET / ts_exos_reflexion.html/
HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:72.0) Gecko/20100101
Firefox/72.0
Host: www.maths-info-lycee.fr
Accept: text/html
```

La méthode GET demande une ressource au serveur, ici la page « ts_exos_reflexion.html » sur le serveur de « www.maths-info-lycee.fr ». Le protocole utilisé est http, version 1.1, d'un navigateur Firefox version 72, sous système OsX 10.13.

La réponse du serveur est, sous forme simplifiée :

```
HTTP/1.1 200 OK
Date: Fri, 07 Feb 2020 11:13:14 GMT
Content-Type: text/html, charset=UTF-8
Content-Length: 5283
Transfer-Encoding: chunked
Server: Apache /2.0.54 (Debian GNU/Linux) DAV/2 SVN/1.1.4
Connexion : Close

<!DOCTYPE html>
<html>
  ...le corps de la réponse, c'est-à-dire le code de la page demandée (la ressource)
```

```
</html>
```

La réponse commence par la ligne de statut, avec d'abord la version du protocole HTTP utilisée par le serveur, puis un code numérique indiquant le résultat. Les plus fréquemment rencontrés sont : 200 pour le succès, et 404 pour une page non trouvée, 403 : accès refusé, 504 : pas de réponse du serveur etc. Et 418 : « je suis une théière »…en effet il existe une version HTCPCP : Hyper Text Coffee Pot Control Protocol (devinez à quelle date elle a été écrite). Figure ensuite la date à laquelle a été reçue la requête, le type de fichier (html), sa longueur (5283 octets), et le logiciel faisant fonctionner le serveur web (Apache sous Linux, ce dernier en version Debian). Le serveur précise aussi que la connexion est terminée. Cet en-tête permet au client, ici le navigateur de gérer la réponse : faut-il afficher le fichier (type html/text), proposer un enregistrement ou une ouverture avec un autre logiciel (décompresseur pour un zip, lecteur pdf, …).
Après l'en-tête figure une ligne vide, puis le corps de la requête. Ici c'est une page html dans un fichier texte, qui fait 5283 octets.
Le site permet de récupérer à la fois la requête et sa réponse. Plus généralement, un sniffer –renifleur en français– permet d'observer les échanges sur internet en général et le web en particulier. Voir aussi le programme donné en cours renifleur_http.py.

Lorsque l'on envoie un formulaire avec la méthode GET, l'url est du type :
https://www.qwant.com/?client=brz-moz&q=http
La requête sera du type donné en exemple ci-dessus
Avantage : l'url contient les paramètres. On peut la copier et la coller, avec comme objectif d'obtenir exactement la même page. Les paramètres sont précisés après le point d'interrogation, par paire *nom_param = valeur_param*. L'exemple donné ci-dessus est celui d'une recherche des pages comportant le mot « htttp » sur le moteur de recherche Qwant. L'url contient deux paramètres : le client, qui est le navigateur Firefox (*br* comme browser, navigateur, et *moz* comme Mozilla Fireox), et la question/requête q (comme question/query, I don't know which one it is precisely).
Inconvénient : les paramètres sont très lisibles.

Lorsque l'on envoie un formulaire avec la méthode POST, les paramètres ne sont pas passés dans l'adresse, mais dans le corps de la requête.
Supposons que l'on envoie un formulaire, avec deux paramètres, par un bouton « submit », en remplissant un formulaire à l'url (fictive, n'essayez pas elle n'est pas fonctionnelle) : http://www.mats-info-lycee.fr/test/demo_formulaire.php
La requête sera :

```
POST /test/demo_formulaire.php
HTTP/1.1
Host: www.maths-info-lycee.fr
nom_param1=valeur_param1&nom_param2=valeur_param2
```

Inconvénient : on ne peut pas copier/coller la requête, la mettre en mémoire dans les favoris, ni dans l'historique. D'ailleurs si on recharge la page on verra s'afficher un avertissement du type « êtes-vous sûr ? Toutes les données seront perdues. Votre ordinateur va exploser. Les martiens vont envahir la Terre. »
Avantage : les paramètres sont un peu moins visibles (mais ça ne sécurise pas pour autant un mot de passe, qui reste transféré « en clair »). Et surtout, les données passées en paramètre peuvent être aussi longues que l'on souhaite, ce qui n'est pas le cas avec la méthode GET. Les formulaires sont en général passés avec la méthode POST.

Insistons sur le fait qu'un mot de passe est protégé uniquement lors de l'usage du protocole HTTPS, qui rajoute le chiffrement SSL.
SSL (Secure Socket Layer) vient en complément du protocole TCP/IP. Ses avantages sont

- la confidentialité : on ne peut espionner les données. La connexion est chiffrée.
- l'intégrité des données : on ne peut pas les modifier.
- l'authentification : on est sûr de l'identité du client et du serveur grâce à des certificats présents dans le navigateur, fournis par des sociétés tierces à qui on fait implicitement confiance. Des soucis fréquents se posent avec les certificats, certains navigateurs décidant à un moment donnés que des certificats ne sont plus valables. Cette décision est toujours justifiée, mais si un site « honnête » ne met pas à jour ses certificats, il sera considéré comme douteux.

L'évolution récente de SSL est TLS (Transport Layer Security) qui accompagne la version 2 de HTTP.
Les informations transmises peuvent toujours être piratées sur le serveur. Ceci arrive fréquemment, avec des conséquences plus ou moins fâcheuses, certains sites ne chiffrant pas les données. C'est notamment arrivé sur un site dont la spécialité est « l'infidélité conjugale », les noms et adresses des utilisateurs étant stockés en clair...

3. Un très rapide aperçu du langage PHP

PHP signifie PHP Hypertext Processor (remarquez la petite blague d'informaticien, la définition est récursive : PHP Hypertext Processor signifie PHP Hypertext Processor Hypertext Processor etc.) Le PHP est un langage de programmation open-source utilisé sur les serveurs. Il est incorporé à une page web, dont le suffixe n'est plus « .html » mais « .php ». Il sert principalement à l'accès aux bases de données, ce qui dépasse amplement l'objectif de ce cours. Le PHP nécessite un interpréteur, qui est côté serveur. Comme Python, écrire le programme ne suffit pas : il est ensuite nécessaire qu'il soit interprété, afin de produire du code html. Si on écrit une page en PHP localement sur son ordinateur, et qu'on l'ouvre dans un navigateur sans avoir préalablement installé un interpréteur, seul le code html de la page sera affiché, sans exécution du PHP.

Lors de l'envoi d'une page en PHP sur un serveur, le serveur génère le code html correspondant et renvoie celui-ci au client.

Remarques :

- Pour exécuter du PHP dans une page html sur votre ordinateur, utilisez UWAMP ou WAMP (sous windows) ou équivalent (XAMPP et MAMP sous Linux/OsX). Ce n'est pas simple au début. Si nécessaire, le professeur peut vous fournir un tutoriel écrit pour toutes les étapes d'installation, ainsi que pour lancer votre premier projet PHP.
- On peut utiliser PHP comme un langage de script (comme Python) sous Windows. Un double clic sur un fichier en ".php" lancera le script dans un shell.
- Il existe d'autres langages pour créer des pages web dynamiques sur un serveur (par exemple Python grâce à Python Flask ou Bottle), mais ils sont moins utilisés. PHP pesait pour 82% dans les langages de programmation web côté serveur en 2016. Ceci dit Flask est très puissant, facile à utiliser, et moderne.

Quelques bases :

- Le code PHP est dans la balise `<?php code ?>`. On peut écrire plusieurs balises PHP dans une page web. On utilise aussi `<script language="php">` et `</script>`, comme en JavaScript
- Les commentaires sont précédés de // sur une seule ligne, ou entre /* et */
- Les instructions sont séparées par des « ; »
- Les variables sont déclarées avant utilisation, et leur nom commence toujours par $. On peut leur affecter directement une valeur, ou pas :

```
$nombre = 42      //variable $nombre déclarée avec la valeur 42
```

- `echo "texte"` écrit le texte dans la page web, on peut rajouter du code html pour préciser la forme (par exemple un `echo 'bonjour<br>'` pour passer à la ligne). On peut aussi utiliser `print`, avec la même syntaxe que `echo`.

```
echo $nombre." est la réponse universelle"
```

écrit sur la page « 42 est la réponse universelle »

- Les variables booléennes ont comme valeur `true` ou `false` (sans majuscules)
- Instruction conditionnelle :

```
if($x > 0){
      echo '$x contient une valeur strictement positive <br>';
}elseif($x == 1){
      echo '$x contient la valeur 0 <br>';
}else{
      echo '$x contient une valeur strictement négative <br>';
}
```

Remarque : on dispose d'opérateurs de comparaisons plus nombreux qu'en Python, par exemple === est Vrai sssi les valeurs mais aussi les types sont égaux. On dispose également de ! (not), AND, OR et XOR.

- Boucle « pour » :

```
<?php
      for($i = 0; $i <= 5; $i++){
            echo '$i contient la valeur ' .$i. '<br>';
      }
?>
```

Remarque : `($i = 0; $i <= 5; $i++)` donne l'incrémentation sur la variable d'itération `$i` : de 0 à 5 inclus, de 1 en 1 (la partie `$i++`)

- Boucle « tant que » :

```
<?php
      $i = 0;
      while($i <= 10){
            echo '$i contient la valeur ' .$i. '<br>';
            $i = $i + 1;      //incrémentation de i
            // on peut aussi écrire $i++
      }
?>
```

- Si vous voulez approfondir, un cours très bien fait est ici : https://www.pierre-giraud.com/cours/, que ce soit pour le PHP ou pour JavaScript ci-dessous

En conclusion, Remarquez les similitudes et les différences avec Python. *Pour ceux qui ont programmé en C (arduino), remarquez aussi les similitudes, notamment dans la boucle « for ».*

PARTIE C : INTERACTION AVEC L'UTILISATEUR DANS UNE PAGE WEB

Un très rapide aperçu de Javascript

Originellement conçu –en dix jours– comme un langage de scripts, JavaScript s'est imposé comme le langage principal de programmation de pages web dynamiques, côté client. Contrairement au PHP, le Javascript n'est pas interprété côté serveur, mais côté client, dans le navigateur lui-même. Il est présent dans 95% des sites web.

Quelques bases :

- Le code JavaScript, s'il est court et ne concerne qu'une page, est entre les balises `<script>` et `</script>`. En général, on préfère mettre le code dans une page à part, de suffixe ".js", que l'on importera dans l'en-tête de la page html avec une balise `<script type="text/javascript" src= "mon_code.js" async></script>`.
- Les commentaires sont précédés de // sur une seule ligne, ou entre /* et */
- Les instructions sont séparées par des « ; » . Vous pouvez trouver des exemples où cela n'est pas respecté, cependant ne pas les mettre peut conduire à des erreurs difficiles à corriger.
- Les variables sont déclarées avant utilisation, en utilisant le mot-clé `let`. On peut leur affecter directement une valeur, ou pas :
  ```
  let nombre = 42      //variable nombre déclarée avec la valeur 42
  ```
- `alert("texte")` écrit le texte dans une « boite », en dehors de la page web proprement dite.
  `alert(nombre + " est la réponse universelle")` écrit « 42 est la réponse universelle ».
- Les variables booléennes ont comme valeur `true` ou `false` (sans majuscules)
- Instruction conditionnelle :
  ```
  //affichage d'une boite de dialogue pour rentrer un nombre
  //comme en Python, par défaut l'utilisateur rentre une chaine
  let x = prompt('Rentrez un nombre ','0');
  x = Number(x)
  //modification du paragraphe d'identifiant p1 pour affichage du nombre
  document.getElementById("p1").innerHTML +=
  'la valeur de x est ' + x + '<br>'
  //affichage d'une boite d'alerte suivant le signe du nombre
  if(x > 0){
        alert('x contient un nombre strictement positif');
  }else if(x == 0){
        alert('x contient 0');
  }else{
        alert('x contient un nombre strictement négatif');
  }
  ```
  Remarque : on dispose d'opérateurs de comparaisons plus nombreux qu'en Python, par exemple === est Vrai sssi les valeurs mais aussi les types sont égaux. On dispose également de ! (not), AND, OR.
- Boucle « pour », avec modification dans la page du paragraphe d'identifiant « p3 » :
  ```
  let compteur = 1
  for(let i = 0; i < 10; i = i + 3){
      document.getElementById('p3').innerHTML +=
      'i stocke la valeur ' + i + ' lors du passage n°'
      + compteur + ' dans la boucle<br>';
     compteur++;
  }
  ```
  Remarque : `(let i = 0; i < 10; i = i + 3){`donne l'incrémentation sur la variable d'itération `i` : de 0 à 10 exclu, de 3 en 3 (la partie `i = i + 3)`
- Boucle « tant que », avec écriture directement dans la page html, dans le paragraphe d'identifiant p2 :
  ```
  let i = 0;
  while(i <= 10){
     document.getElementById('p2').innerHTML +=
     "i contient la valeur " + i + "<br>";
     i = i + 1;  //incrémentation de i
     // on peut aussi écrire $i++
  }
  ```

- Quelques exemples d'interaction avec les éléments de la page HTML.
  - Action sur un survol d'élément de la page
    On définit une fonction dans la page des scripts (ou dans le corps du html) :
    ```
    function ma_fonction() { alert('Autodestruction enclenchée...')}
    ```
    Et on précise dans le corps du html :
    ```
    <span onmouseover="javascript:ma_fonction()">Ne pas passer le curseur ici</span>
    <!-- span encadre l'action-->
    ```

  - Action sur un bouton et sur le titre de la page avec `document.querySelector('élément')`. *Ici un exemple où un clic sur un bouton change le titre de la page*
    Dans la page HTML, on rajoute un bouton :
    ```
    <button>Changer le titre</button>
    ```
    Dans la page des scripts :
    ```
    //On définit deux variables qui contiennent le titre h1 et le bouton
    let mon_bouton = document.querySelector('button');
    let titre_h1 = document.querySelector('h1');

    //On crée un gestionnaire d'événement qui se déclenche au clic sur le bouton
    //et modifie le titre de la page
    mon_bouton.addEventListener('click', function() {
      titre_h1.textContent = 'Nouveau titre';
    });
    ```

  - S'il y a plusieurs éléments dans la page du même type (boutons, titres d'un certain niveau, paragraphes, etc.) alors on utilise un identifiant HTML pour agir sur un élément précis de la page avec `document.getElementById('identifiant d'élément')`. *Ici un exemple où un clic sur un bouton demande un nouveau nom d'utilisateur et change le titre de la page en conséquence.*
    Dans la page HTML, on rajoute un bouton :
    ```
     <button id = 'b2'>Changer le titre</button>
    ```
    Dans la page des scripts :
    ```
    //On définit deux variables qui contiennent le titre h1 et le bouton
    //Les deux variables sont récupérées par deux méthodes différentes
    let mon_bouton2 = document.getElementById('b2');
    let titre_h1 = document.querySelector('h1');

    //On crée un gestionnaire d'événement qui se déclenche au clic sur le bouton
    //et modifie le titre de la page. L'événement est l'appel de la fonction
    //demander_nom_util()
    mon_bouton2.addEventListener('click', function() {
      demander_nom_util();
    });

    //La fonction qui demande le nom et modifie le titre en conséquence
    function demander_nom_util() {
      let nomUtilisateur = prompt('Veuillez saisir votre nom?');
      localStorage.setItem('nom', nomUtilisateur);
      if (nomUtilisateur == 'Yoda'){
     titre_h1.textContent = 'Mes respects, Maître ' + nomUtilisateur;
      }else{
     titre_h1.textContent = 'Salut, loser ' + nomUtilisateur;
      }
    }
    ```

  - Un exemple de bouton qui dévoile du texte caché sur une page :
    Dans la page HTML, on rajoute un bouton, associé à une fonction cacher/montrer. Le texte de la division suivante n'est pas affiché (ce que précise l'attribut `style="display:none;"`) :
    ```
    <button onclick="cacher_montrer()">Texte caché</button><br>
    <div id="cache" style="display:none;">
          Très confidentiel.<br>
          lecture dangereuse !
    </div>
    ```
    En théorie dans la page des scripts (et en pratique ça fonctionne moins aléatoirement chez l'auteur de ce cours, si le script est dans la page HTML, c'est probablement dû à un appel différent de la fonction JavaScript, ce que vous avez remarqué bien sûr) :

```
function cacher_montrer() {
  var x = document.getElementById("cache");
  if (x.style.display === "none") {
      x.style.display = "block";
  } else {
      x.style.display = "none";
  }
}
```

En conclusion, remarquez les similitudes et les différences, notamment sur les variables et structures de contrôle, avec Python et PHP.

UN TRUC QUI N'EST PAS UN EXERCICE

Le programme `requete_http.py` vous permet de construire une requête http et de recevoir la réponse.
Les programmes « ébauche de client et de serveur en http » montrent –assez maladroitement, il subsiste des bugs- le fonctionnement d'un serveur et d'un client.
La bibliothèque `requests` permet de récupérer le contenu d'une requête.
Exemple (envoi d'une requête GET) :

```
import requests
resultat = requests.get('http://www.maths-info-lycee.fr')
print(resultat)             # donne le code 200/404/etc.
print(resultat.headers)     # donne les en-têtes de la réponse
print(resultat.text)        # donne le texte de la réponse (le code html
                            # pour un site)
```

MINI PROJET HTML CSS JAVASCRIPT

Ce projet consiste en la réalisation d'un mini-site web, de deux pages minimum. Il est à faire seul ou à deux.
Vous pouvez vous baser sur l'exemple http://www.maths-info-lycee.fr/exemple_site/site.html (les sources sont téléchargeables : http://www.maths-info-lycee.fr/templates/exemple_site.zip) ou sur des templates plus sophistiqués, dont on trouve des wagons sur le ouèbe.
Attention : toutes les feuilles HTML, CSS et JavaScript doivent être nettoyées de toute la graisse superflue…ce qui n'est pas simple, ce qui devient très compliqué lorsque l'on utilise un template.
Le mini-site porte sur le sujet de votre choix. Vous pouvez récupérer du texte existant, dans ce cas vous citerez les sources en hyperlien, dans une page ou une division dédiée.
La page d'accueil s'appelle forcément index.html ou index.php
Le site doit comporter :

- Des hyperliens, à la fois entre les pages du site et sur des pages extérieures au site.
- Une partie interactive avec :
  - Un ou plusieurs scripts JavaScript, qui figureront sur une feuille en .js.
  - Éventuellement une page PHP, avec notamment un formulaire à remplir et une action à effectuer sur ce formulaire. Faites quelque chose de simple (sauf si vous souhaitez un effet waouh). Vous ne pourrez tester le PHP que si vous hébergez votre site (cf. ci-dessous) , ou si vous installez un serveur PHP sur votre ordinateur (ci-après le guide pour UWamp ou WampServer, pas forcément simple).

Le style doit être géré à part dans une feuille CSS. Il doit être cohérent sur l'ensemble du site, lisible, et autant que possible esthétique.
Le site est organisé comme suit :
Au « niveau 0 », les pages HTML et PHP, ainsi que 3 dossiers de noms css, js, images (et éventuellement d'autres dossiers si vous mettez par exemples des pdfs, etc. chaque type de document ayant son propre dossier). Dans les dossiers figureront les pages ou documents du type indiqué. Faites attention à créer cette structure dès le début, afin que les balises `<link>` fonctionnent.
Tous vos fichiers seront codés en UTF-8. Les noms des fichiers (y compris leurs extensions) seront en minuscules.
Vous me rendrez également un fichier lisezmoi.txt (format txt simple, surtout pas de docx ou odt) avec vos nom et prénom et annonçant le thème choisi et

- le nom de la nouvelle feuille de style CSS
- le nom de la nouvelle page JavaScript en précisant l'effet des scripts
- la page où est situé votre formulaire

D'une manière générale vous indiquerez dans ce fichier lisezmoi.txt tout point particulier mettant en valeur votre travail et sur lequel vous voulez attirer l'attention du correcteur.
Vous m'enverrez le site sur Pronote, dans un dossier compacté à votre nom.
Vous pouvez télécharger votre site sur 000webhost (un hébergeur gratuit, il vous faudra créer un compte). Dans ce cas donnez-m'en l'adresse dans le fichier lisezmoi.txt

WAMP SERVER, UWAMP, XAMP, MAMP…

Pour faire du PHP, il est indispensable d'avoir un serveur Apache-MySQL-PHP. De nombreux logiciels existent, plus ou moins compliqués à mettre en œuvre. Une autre solution est d'héberger son projet chez un hébergeur gratuit, là il n'y aura aucun problème d'installation. Lorsque l'on débute en PHP, les problèmes de serveur sont souvent bien plus difficiles à résoudre que les problèmes de programmation !

Le plus simple sous Windows semble être UWamp :

- Installer les différentes versions de Visual C++. On peut utiliser le lien ci-dessous (dans Wamp Server), ou les liens dans la page de téléchargement de UWamp. Installez tout pour ne pas vous poser de questions.
- Pour Uwamp proprement dit, la version en .zip sur une clé USB externe permet un usage immédiat et transportable
- Mettre votre projet dans le dossier www, et le lancer par le bouton « navigateur » de UWamp, sinon ça ne fonctionne pas.

Installation de Wamp Server :

- suivre avec attention les instructions lors de l'installation ; pour une fois il faut les lire !
- Notamment :
  - Présence des différentes versions de Visual C++. Lien pour l'outil disponible dans le premier écran d'installation (logiciel check_vcredist.exe). Le lancer. S'il manque des logiciels, les installer **avant** wampserver. Les liens sont disponibles là aussi dans le premier écran d'installation, la totalité étant là : http://wampserver.aviatechno.net/files/vcpackages/all_vc_redist_x86_x64.zip .
  - Le répertoire d'installation doit être à la racine du disque, et le nom ne doit contenir ni espaces ni caractères diacritiques.
- Lancer wampserver. Une icône « wampmanager » rouge puis orange puis verte apparaît dans le menu accessible par la petite flèche. Si elle ne passe pas au vert, un clic gauche dessus permet de « démarrer les services ».
- Créer un projet, en ajoutant un dossier dans le dossier www de wampserver. Y mettre les fichiers html, css, php etc.
- Un clic gauche sur l'icône wampmanager permet d'accéder au menu « virtualhost » puis « gestion virtualhost ». Ceci lance une page php, suivre les indications pour remplir le formulaire.
- Comme indiqué après complétion du formulaire, redémarrer le serveur DNS ; accès par clic droit sur l'icône wampmanager.
- Pour que les pages PHP fonctionnent correctement, il est indispensable de passer par le menu « virtualhost » de wampmanager. Un clic sur l'hôte virtuel désiré ouvre une page html donnant accès aux différentes pages du projet, et permet de les ouvrir ainsi dans le navigateur.
- Ouvrir les pages directement depuis le système de fichiers ne lance pas l'interpréteur PHP ; ça n'est pas fonctionnel.