Interactions entre l'homme et la machine sur le Web
Voici une synthèse du cours *openclassroom*.

0.1 HTML et CSS

HTML est un langage de balisage. Si la balise enveloppe un élément commme le texte d'un paragraphe, on utilise la syntaxe avec une balise ouvrante et une fermante. Par exemple, pour un paragraphe, les balises sont:

 ..... .

Le document html ci-dessous comporte les balises les plus basiques que vous devez maintenant connaitre.

Ci-contre une capture d'écran de la page obtenue dans un navigateur. Il manque la mise en page qui se réalise avec CSS.

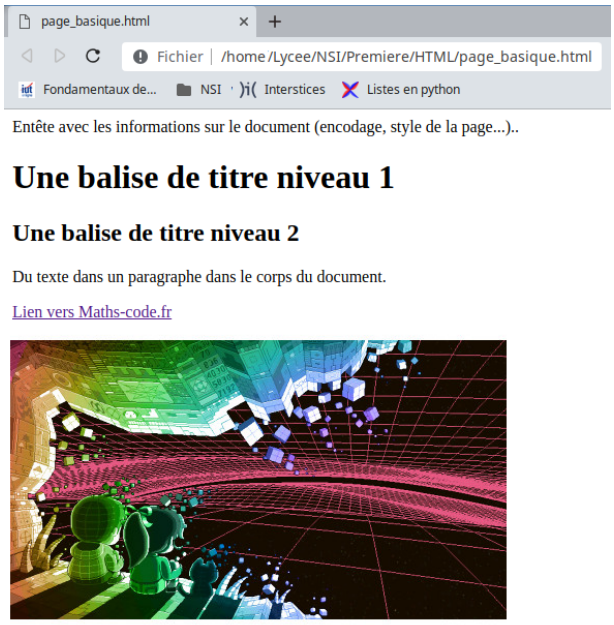

Figure 1:La page correspondant au code HTML ci-dessus.

```
<!DOCTYPE html>
<html>

    <head>
        Entête avec les informations sur le document (encodage, style de la page...)..
    </head>

    <body>
        <h1>Une balise de titre niveau 1</h1>
        <h2>Une balise de titre niveau 2</h2>
        <p>
            Alan Turing est un mathématicien Britannique né en 1910.
            Sa machine de Turing et la théorie du lambda calcul d'Alonzo Church
            sont considérés comme les fondations du modèle théorique des ordinateurs.

        </p>
        <img src="images/Trous_noir_8bits.png" alt="Un trou noir en pixels" />$
        <a href="http://maths-code.fr" >Lien vers Maths-code.fr </a>
    </body>

</html>
</html>
```

0.2 Exemple plus élaboré

0.2.1 Le code HTML

Voici un extrait du code HTML de la page d'accueil de `maths-code.fr`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Forum: Maths et NSI-Lycée Carnot Bruay-Labuissière</title>
    <meta http-equiv="Content-Type" content="charset=utf-8" />
    <meta name="description" content="Lycéee Carnot, Artois" />
    <script language="Javascript" type="text/javascript">
        function ouvrir(){
        fenetre=window.open("","fen_img","width=640, height=480, scrollbars=no,
                                toolbar=no, location=no, directories=no, status=no")}
    </script>
    <link rel="stylesheet" type="text/css" href="style_index.css" />
</head>

<body>
        <div id="page">
            <div style="background-image:url(images/Fond-entete3.png);
                                color:rgb(119, 117, 95);height:170px;width:800px">
                <p style="color:white;">
                    <i>Des maths et du code</i>
                </p>
            </div>

            <?php include "entete2";?>

            <p style="text-align : justify;" >
                Bienvenue sur le
                <a href="http://maths-code.fr/forum/viewforum.php">forum de discussion</a>
                <i>Maths Code</i>, une place virtuelle où discuter et échanger.
            </p>

            <br />D'autre part, ce forum ne collectera pas vos données.....
        </div>
</body>
</html>
```

Analyse de ce code On peut utiliser CSS avec la balise `<style>` toujours placée dans `<head></head>`. Ce n'est pas la solution idéale, il suffit de voir les balises `<div style=""....>` ou `<p style="">` dans le code ci-dessus. L'idéal est d'appeler le fichier CSS à l'aide de la balise `<link rel />`:

```
<link rel="stylesheet" type="text/css" href="style_index.css" /> .
```

Attention aux chemins indiqués pour accéder aux fichiers. La remarque est identique pour les script Javascript et PHP. Ils ont été écrits respectivement dans le code ci-dessus:

- pour javascript: `<script> ..... </script>` .
- pour php: `<?php ..... ?>` .

0.2.2 Le code CSS

Voici un extrait de la feuille CSS liée au code HTML précédent:

```
body{font-size:15px;font-family:Times;background: #000 url(images/Essai5.jpg);
     background-repeat:repeat; margin: 0px;}

#conteneur {background: #000 url(images/nuit-ubuntu850.jpg);
            color:#98e5ff; background-repeat:no-repeat;
            background-position:0% 240px;
            margin-left: auto; margin-right:auto; margin-top: 5px;
            width: 850px; height:1120px;
            border:1px solid gray}


.bottom {float:left;margin-left: 10px;
         border:1px solid gray;width : 480px; height:300px;width:480px;
         background:url(images/Trou_noir.gif) no-repeat;background-size:contain;}

.bottom p{float:right}
h4{ color :#FFFFFF;font-size:0.9em;text-align: left}
a {text-decoration: none;}
a:link {color: #6495ED;}
a:visited {color:#228B22;}
a:hover {color:#CD5C5C; }
a:active {color: lime;}
```

Une partie de ses éléments sont détaillés ci-dessous.

0.2.3 Sélecteurs

Pour chaque déclaration, la structure est toujours la même :

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

Le sélecteur: c'est la balise (X)HTML:

1. `body, h1, h4, p,` ...etc...
2. l'identifiant (id).
3. ou la classe (class).

La propriété: c'est l'attribut qu'on veut appliquer (`font ; background ; margin` ; ...etc...)

Les valeurs: ce sont les caractéristiques de la propriété.

Identifiant

C'est le symbole dièse # comme sélecteur qui permet de cibler un élément par son id. Cet attribut peut être utilisé en javascript, et aussi servir d'ancre.

Classe

Le symbole `.` est un sélecteur de classe. La principale différence entre les ids et les classes est que ces dernières vous permettent de cibler plusieurs éléments. Utilisez des classes quand vous souhaitez que votre mise en forme s'applique à un groupe d'éléments. Alternativement, vous pouvez donc recourir aux ids, un peu comme on chercherait une aiguille dans une botte de foin, et mettre en forme l'élément spécifique ainsi ciblé, et lui seul.

le sélecteur descendant

`.bottom p` dans notre exemple est un sélecteur descendant. Ici , on applique un style au paragraphe de la classe `bottom`.

0.3 Eléments d'interaction dans une page web

Le Web s'appuie sur un dialogue permanent entre clients et serveurs. Les clients sont les machines ou leurs applications qui se connectent au Web, les serveurs sont les machines qui leur répondent, et rendent donc un *service*. La première étape pour accéder à un site Web est la saisie de l'adresse ou URL (*Uniform Resource Locator*) de ce site dans la barre d'adresse du navigateur.

0.3.1 Requête HTTP et interaction client/serveur

HTTP est un protocole de transmission au même titre que FTP. Il est de type **client/serveur.** Il permet à un client de récupérer auprès d'un serveur web des données. Si le client est un navigateur web, ce protocole permet d'obtenir les données nécessaires à l'affichage d'une page internet. Le protocole fonctionne par l'intermédiaire de requêtes émises par le client et de réponses fournies par le serveur.

Exemple Examinons la demande d'une page web telle que `maths-code.fr` par un navigateur internet.

1. Le navigateur commence par demander, à un serveur de noms de domaines (DNS), l'adresse IP de `maths-code.fr`. DNS signifie Domain Name Server, le serveur effectue la traduction adresse IP/ nom du site. Celui-ci lui fournit 92.222.139.190.
2. En utilisant le protocole de transport TCP sur le port 80, le navigateur se connecte à la machine dont l'adresse IP est 92.222.139.190 .
3. Une fois la connexion établie, le navigateur envoie un certain nombre de message appelés **requêtes** en se conformant au protocole HTTP pour demander la ressource *index.html*.
4. Le serveur Web se trouvant à l'adresse 92.222.139.190 envoie en réponse le contenu du fichier.
5. Le navigateur peut alors parcourir le fichier et afficher la page correspondante.

Voici la requête du client:

```
GET / HTTP/1.1
Host: maths-code.fr
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
```

Le serveur répond:

```
HTTP/1.1 200 OK
Date: Sun, 12 Apr 2020 13:33:27 GMT
Content-Type: text/html; charset=UTF-8
```

Le code `200` indique que tout s'est passé normalement. La page retournée est du **html** encodé en `UTF-8`.

0.3.2 Méthodes pour requête HTTP

Pour utiliser ce protocole, nous allons devoir passer par un navigateur web (Chromium, Brave, Firefox, etc.) qu'on va alors appeler un "client http". Lors de la demande d'une page Web, un navigateur prépare une requête HTTP. Cette requête est mise en forme dans un ou plusieurs paquets par le protocole TCP puis sont encapsulés dans des paquets munis des adresses IP pour pouvoir circuler. Ces paquets sont eux-mêmes encapsulés dans des trames Ethernet ou Wifi. Ces trames ajoutent aux paquets les adresses MAC (Medium Access Control), qui sont les adresses physiques des appareils. Le processus inverse est réalisé chez le récepteur jusqu'à récupérer le contenu du message de demande HTTP.

HEAD, GET et POST sont des méthodes d'accès définies dans le protocole HTTP et reprises dans la spécification HTML. Le choix de la méthode dépend de la façon dont les données sont reçues, de la taille et la nature des données. La syntaxe générale d'une requête client est toujours la même:

```
Méthode utilisée\\
En-tête de requête \\
<saut de ligne>\\
Corps de la requête (éventuellement)
```

L'en-tête contient des informations comme le nom du site, le type de connection ou la longueur du corps de la requête:

Exemple:

```
HEAD /fichier.html HTTP/1.1
Host: www.site.fr
Connection: Close
<saut de ligne>
```

La méthode appliquée à fichier.html est HEAD, cette méthode demande une réponse identique à une requête GET pour laquelle on aura omis le corps de la réponse

0.3.3 Les formulaires

Un formulaire est un élément codé en HTML. Les formulaires sont utilisés avec les méthodes GET et POST. Pour créer un formulaire, il faut obligatoirement lui spécifier une méthode d'envoi à choisir entre "get" et "post" ainsi que l'url du script qui va recevoir les données du formulaire. Celui se traduit par le code html :

```
<form method="post" action="traitement.php"></form>
```

Sur cet exemple, les données seront envoyées au fichier "traitement.php" grace à l'attribut `action`.

- Tous les éléments du formulaire seront placés entre les balises `<form...> ..... </form>`.
- La balise `<input>` permet à l'utilisateur de saisir des données. L'attribut `type` indique la manière de fonctionner de `<input>`, par exemple: `text`, `password` ou `image`.
- La balise `<label>` permet de donner un intitulé à un champ de formulaire comme par exemple un `input` de type `text`, un `textarea` ou encore les champs de type `radio` ou `checkbox`, parmi les plus utilisés.

  Son utilisation permet aux navigateurs d'associer cet intitulé au champ de formulaire. Ainsi, un clic sur l'intitulé donnera le focus au champs de formulaire, ce qui est très pratique par exemple pour les boutons radio ou cases à cocher parfois trop petites pour certains utilisateurs.

0.3.4 PHP

Ce langage de programmation s'inclut dans le code HTML via un fichier ou la balise `<?php ?>`.

Attention: PHP est un langage interpété coté serveur, il vous faut donc un serveur PHP pour lire les pages correectement avec votre navigateur. Dans ce qui suit , nous allons surtout utiliser les variables, les blocs de code usuels se repèrent facilement. Les variables sont faciles à repérer, car elles commencent toutes par un signe dollar.

Exemple: Le script suivant se comprend aisément, retrouvez les variables, commentaires, affectation, chaine de caratères, structure conditionnelle.

```
<?php
            $x = 4; //On affecte la valeur 4 à £x
            $y = 2; //On affecte la valeur 2 à £y

            if($x > 1){
                echo '$x contient une valeur supérieure à 1';
            }

            if($x == $y){
                echo '$x et $y contiennent la même valeur';
            }
            else{
                echo '$x et $y contiennent des valeurs différentes <br>';
            }
        ?>
```

Un type de variable réservée va nous servir rapidement: les variables superglobales ou auto-globales. Ces variables sont disponibles dans tous les contextes du script local ou global et son au nombre de 9. En voici quelques-unes:

`$_GET` : Un tableau associatif (ou dictionnaire) des valeurs passées au script courant via les paramètres d'URL (aussi connue sous le nom de "query string"). Elle est utilisée pour manipuler les informations envoyées via un formulaire HTML.

`$_POST`: Commme la précédente, un tableau associatif des valeurs passées au script courant via le protocole HTTP et la méthode POST.

`$_SERVER` est un tableau contenant des informations comme les en-têtes, dossiers et chemins du script. Les entrées de ce tableau sont créées par le serveur web. Il n'y a aucune garantie que tous les serveurs les rempliront tous ;

0.3.5 Méthode GET

GET demande une ressource au client **sans la modifier**. Dans un formulaire, elle est spécifiée ainsi:

```
<form method="get" action="page.html"> </form>
```

Avec cette méthode, les données du formulaire seront encodées dans une URL. Celle-ci est composée du nom de la page ou du script à charger avec les données de formulaire empaquetée dans une chaîne. Les données sont séparées de l'adresse de la page par le symbole `?` et entre elles par le symbole `&`. Le symbole `=` assigne une valeur à une variable.

Exemple: Si on envoie à page.html les valeurs "couleur bleu" et "forme rectangle", l'URL construite par le navigateur sera:
`http://maths-code.fr/page.html?couleur=bleu&forme=rectangle`

Exemple Tapez une recherche dans le moteur de recherche lilo.org, observez le changement d'URL après avoir validé la recherche. GET est la méthode utilisée. La spécification HTML 5 demande que l'on utilise GET quand la requête ne cause pas de changement dans les données, donc opère une simple lecture.

Les données de formulaire doivent être uniquement des codes ASCII. La taille d'une URL est limitée par le serveur, souvent un peu plus de 2000 caractères, en comprenant les codes d'échappement.

0.3.6 Méthode POST

POST modifie la ressource. Sa spécification est identique à `GET` dans un formulaire:

```
<form method="post" action="page.html"> </form>
```

Elle envoit un en-tête et un corps de message au serveur. Le corps est généralement constitué des données entrées dans le champ de formulaire par l'utilisateur.

Les données du formulaire n'apparaissent pas dans l'URL.

Exemple On demande à un utilisateur de saisir son nom et son âge:

```
<form method="post" action="page.html"> </form>

<form action="action.php" method="post">
 <p>Votre nom : <input type="text" name="nom" /></p>
 <p>Votre âge : <input type="text" name="age" /></p>
 <p><input type="submit" value="OK"></p>
</form>
```

On peut récupérer les entrées de la manière suivante en incluant un script php:

```
Bonjour, <?php echo htmlspecialchars($_POST['nom']); ?>.
Tu as <?php echo (int)$_POST['age']; ?> ans.
```

On comprend l'utilisation de la variable `$_POST`: la clé utilisée est telle qu'elle a été définie dans le script d'appel précédent :'nom' et on affichera sa valeur `$_POST['nom']`. Les variables en php sont toujours précédées d'un $

En conséquence, il n'est pas possible de récupérer directement les données en JavaScript , il faut ajouter du code PHP dans la page. toujours d'après l'exemple précédent:

```
<?php
 $couleur = $_POST['couleur'];
 $forme = $_POST['forme'];
?>
... code HTML ...
```

On peut toutefois assigner les données récupérées en PHP à un script JavaScript:

```
<script>
 var couleur = <?php echo $couleur;?>;
 var forme = <?php echo $forme;?>;
</script>
```

Conclusion: La méthode GET est la valeur de méthode par défaut. On l'utilise de préférence sauf si on ne veut pas que les paramètres soient ajoutés à l'URL. Elle permet de récupérer les données passées à la page avec du code *JavaScript*. **Cependant lors de l'utilisation de la méthode GET dans un formulaire, les données sont transmises en clair dans la barre d'adresse du navigateur**. Cela pose des problèmes de sécurité. En effet, au lieu du type "text", on peut utiliser le type "password". Lors de l'entrée du mot de passe, les lettres sont remplacées par des points mais lors de l'envoi des données, le mot de passe apparaît en toutes lettres dans la barre

d'adresse. De plus la limitation de la longueur de l'URL (2000 caractères ASCII environ) limite la longueur de la requête

La méthode POST transmet les données dans le corps de la requête : les données sont donc moins visibles. Pour autant la méthode POST n'est pas plus sécurisée que GET pour une personne mal intentionnée voulant intercepter des données. Il faut alors sécuriser la connection par TLS/SSL; le protocole sera alors HTTPS.

Exercice 1 Que font les script suivants ?

1.

2.
```php
<?php
                function mystere1(){
                    $s = 0;
                    for($x = 0; $x <= 5; $x++){
                        $s = $s+$x;
                    return $s;
                    }
                }
        mystere1();
            ?>
```

3.
```php
<?php
                $x = 0;
                $y = 20;

                do{
                    echo '$x contient la valeur ' .$x. '<br>';
                    $x++;
                }while($x <= 5);

                do{
                    echo '$y contient la valeur ' .$y. '<br>';
                    $y++;
                }while($y <= 5);
            ?>
```

4.
```php
<?php
                $prenom = 'Rick';
                $x = 4;
                $y = 5;

                function mystere1($p){
                    echo 'Bonjour ' .$p. '<br>';
                }

                function mystere2($p1, $p2){
                    echo $p1. ' + ' .$p2. ' = ' .($p1 + $p2). '<br>';
                }

                mystere1($prenom);
                mystere1('Bender');
                mystere2($x, $y);
                mystere2(1, 1);
            ?>
```

5.
```php
<?php
    /*Note : l'opérateur "*" a une priorité plus grande que "."
     *pas besoin de () ici donc pour faire le calcul*/

    function mystere3(float $a, float $b){
        return $a. ' * ' .$b. ' = ' .$a * $b. '<br>';
    }


    mystere3(4, 5);
?>
```

Exercice 2

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP et formulaire</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="feuille_style.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            //Si notre variable _POST['prenom'] est bien définie, echo...
            if(isset($_POST['prenom'])){
                echo 'Bonjour, tu t\'appelles ' .$_POST['prenom'];
            }
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

Rédiger un formulaire correspondant à cette page.