université de BORDEAUX

TP8 - Techno Web

1 Réaliser un Site Web

Le but de ce TP est de réaliser un petit site web au CREMI mettant en évidence votre compréhension des différentes technologies Web : HTML, CSS, JavaScript et PHP. En guise de rapport, vous devez rendre sur Moodle le lien vers votre site web au CREMI, ainsi que l'ensemble des fichiers source de votre site.

Important : Ce petit site web devra avoir un contenu personnalisé (autour d'une passion par exemple).

1.1 Préambule

Afin de réaliser ce TP, vous allez devoir consulter des documentations de référence :
— w3schools, avec de nombreux examples ;
— MDN Web Docs ;
— World Wide Web Consortium.

En outre, le site du *W3C* offre un outil de validation en ligne qui vous permet de vérifier la syntaxe de la page Web que vous avez produite : `https://validator.w3.org/`.

Le CREMI possède un serveur Web *Apache2* sur lequel chaque étudiant possède un espace personnel, consultable à l'URL `https://<prenom>-<nom>.emi.u-bordeaux.fr/` et dont les ressources sont accessibles dans le répertoire `/net/www/<login>/`. Vous pouvez vérifier votre URL exact en consultant cette page : `https://services.emi.u-bordeaux.fr/intranet/mesinfos/`.

Afin de travailler sur sa machine personnelle, il est également possible d'installer un serveur Web comme *Apache2* (ou *Nginx*) en local de la façon suivante (sous Debian/Ubuntu) :

```
$ sudo apt update
$ sudo apt install apache2 php libapache2-mod-php
$ sudo chmod 777 /var/www/html/                    # accès en écriture à tous
$ echo "hello world!" > /var/www/html/index.html   # nouvelle page d'accueil
```

Dans ce cas là, il faut ajouter vos ressources web directement dans le répertoire `/var/www/html/` et la page web sera alors directement accessible à l'URL `http://localhost/`.

Pour tester votre site web, nous vous conseillons d'utiliser le navigateur web *Chrome* ou *Firefox*, qui possède des outils de débogages très puissants (`F12`), qui permettent d'inspecter le code source à tous les niveaux, d'afficher les erreurs et même de modifier en direct le code à l'aide d'une console *JavaScript* interactive ! Plus d'info sur https://developer.chrome.com/docs/devtools/overview/.

Si vous utilisez un éditeur de code comme *VS Code*, il existe de nombreuses extensions utiles pour le développement Web, que nous vous recommandons d'installer comme HTML CSS Support.

1.2 HTML

Dans ce TP, nous utiliserons la version 5 du langage HTML. Ce document débute par une balise spéciale `<!DOCTYPE html>` et se compose de balises (ou *éléments*) ouvrantes (`<html>`) et fermantes (`</html>`), qui servent à décrire la structure du document. Une page simpliste est donc :

```
<!DOCTYPE html>

<!-- un commentaire -->

<html lang="fr">

<head>
    <meta charset="utf-8" />
    <title>Demo Web</title>
</head>

<body>
    <h1>Ma Page Web</h1>
    <h2>Un Sous-Titre</h2>
    <p>Ma page est chouette !</p>

</body>

</html>
```

Travail demandé :

- En partant de l'exemple précédant, écrire une page HTML basique sur un sujet de votre choix. Utilisez un large éventail de balises, comme par exemple : `<h1>`, `<h2>`, `<p>`, `<pre>`, `<br>`, ... Notez que pour mettre du texte en évidence, il faut privilégier l'utilisation des balises `<em>` et `<strong>`, qui apporte plus de sémantique plutôt que `<i>` (*italic*) et `<b>` (*bold*).
- Ajoutez encore une liste (balises `<ul>`, `<ol>`, `<li>`), une image (`<img>`), un tableau (`<table>`), ...
- Ajoutez des liens externes à votre page web avec l'attribut `href=http://...` de la balise `<a>`. Vous pouvez également ajouter un lien interne `href=#foo`, qui pointe sur un élément quelconque possédant l'attribut `id=foo`.
- Ajoutez des *meta* informations dans la partie `<head>`, comme par exemple `author`, `description`, `keywords`, ...
- Enfin, structurez votre document avec les balises : `<header>`, `<nav>`, `<main>`, `<section>`, `<footer>`, ... Votre document devra obligatoirement contenir plusieurs sections, ainsi qu'un menu en en-tête avec des liens internes vers les sections.

— Dans cette partie, on s'interdira volontairement d'effectuer de la mise en forme en changeant par exemple les polices de caractères (`<font>`), l'alignement (`<center>`), la couleur, ... Nous allons faire cela de manière beaucoup plus élégante dans la section suivante avec une feuille de style CSS.

Incluez ensuite dans l'en-tête des liens de navigations correspondants : vous pouvez pointer avec l'attribut `href="#foo"` un élément quelconque possédant l'attribut `id="foo"`.

Pour réaliser le travail demandé, vous pouvez vous aider de la documentation complémentaire suivante :

— Un premier tutoriel : https://developer.mozilla.org/fr/docs/Web/HTML
— Un autre tutoriel https://www.w3schools.com/html/default.asp
— Les différentes balises : https://developer.mozilla.org/fr/docs/Web/HTML/Element ou https://www.w3schools.com/tags/default.asp
— Spécification : https://html.spec.whatwg.org/

1.3 CSS

Jusqu'à présent, notre page a le style par défaut du navigateur. Pour appliquer notre propre style et ainsi contrôler l'aspect de la page, on intègre à la page une feuille de style, défini dans le fichier *mystyle.css*, via la balise `<link>` dans l'en-tête du document (`<head>`).

```
<head>
    <meta charset="utf-8" />
    <title>Demo Web</title>
    ...
    <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Une feuille de style se décompose en blocs de règles encadrées par des accolades, s'appliquant à un ensemble d'éléments HTML décrit par un ou plusieurs sélecteurs séparés par des virgules. Dans l'exemple suivant, on change les couleurs du texte et de l'arrière-plan de tous les éléments `<h1>` et de tous les liens (`<a>`) de classe `important` se trouvant dans l'élément d'id `section2`. Les couleurs peuvent s'écrire dans plusieurs formats comme `#RRGGBB` en hexadécimal ou encore en utilisant des noms de couleur standards.

```
h1, #section2 .important {
    color: darkblue;
    background: #eeeeee;
}
```

Dans cet exemple, les éléments `h1` et `a` vont donc changer d'apparence mais pas l'élément `p` :

```
<h1>...</h1>
<section id="section2"><a class="important">...</a></section>
<p class="important">...</p>
```

Travail demandé :

— Ajoutez une feuille de style et appliquez un style CSS à plusieurs éléments de votre page : `<h1>`, `<h2>`, `<p>`, `<strong>`, ...
— Ajoutez des attributs *class* et *id* à plusieurs éléments de votre page afin de leur appliquer un nouveau style défini dans votre CSS. Notez que vous pouvez utiliser la balise `<div>` comme un simple conteneur pour appliquez votre style à tout un bloc de HTML. La balise `<span>` permet de faire la même chose à l'inérieur d'une ligne.
— Changez le style pour les liens à l'aide des pseudo-classes `visited`, `hover`, ...
— Mettez en place un *layout* dans votre document avec le *header* dans un bloc en haut, les *sections* répartis sur deux ou trois colonnes au milieu et le *footer* dans un bloc en bas (Fig. 1.3).
— A l'aide d'un style CSS, mettez en place une barre de navigation dans le bloc *header/nav* de votre page.

FIGURE 1 – Exemple de *layout*.

Pour réaliser le travail demandé, vous pouvez vous aider de la documentation complémentaire suivante :

— Un premier tutoriel : https://developer.mozilla.org/fr/docs/Learn/Getting_started_with_the_web/CSS_basics
— Une page, 5 styles : https://www.w3schools.com/css/demo_default.htm
— Pseudo-Classes : https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes
— Template de layout : https://www.w3schools.com/css/css_templates.asp
— Les couleurs : https://www.w3schools.com/colors/default.asp
— Barre de navigation : https://www.w3schools.com/css/css_navbar.asp

1.4 JavaScript

Une fois la page HTML, les styles CSS et les images chargés, la page (son contenu et son aspect) ne change plus. On dit que la page est statique. Pour ajouter un peu de dynamisme, on intègre à de notre notre page un script écrit en langage *Javascript*.

```
<body>
    ...
    <script type="text/javascript" src="myscript.js"></script>
</body>
```

En pratique, le script *myscript.js* va s'exécuter à la fin du chargement de la page et va principalement contenir la définition de fonctions, qui seront appelées en réaction à des

évènements, comme un clic souris. Dans l'exemple ci-dessous, la première fonction modifie le titre `<h1>`, tandis que la seconde fonction modifie le style d'un élément avec l'id *foo*.

```
function func1() {
  let elm = document.querySelector("h1");
  elm.textContent = "Un Nouveau Titre";
}

function func2() {
  let elm = document.getElementById("foo");
  elm.style.fontWeight="bold";
}
```

Afin de réagir à un évènement particulier, il est nécessaire d'affecter le nom d'une fonction *JavaScript* à certains attributs spéciaux (`onclick`, `onmousemove`, `onmouseover`, `onload`, `onfocus`, ...) des différents éléments. Par exemple :

```
...
<h1 onmouseover="func1()">Un Titre</h1>
...
<p id="foo">Ma page est chouette.</p>
<button onclick="func2()">Clic</button>
```

Prérequis : Commencez par lire la documentation ci-dessous.

- Les bases : https://developer.mozilla.org/fr/docs/Learn/Getting_started_with_the_web/JavaScript_basics
- Un tutoriel :https://www.w3schools.com/js/default.asp

Travail demandé :

- Ajoutez à votre site web du *JavaScript* pour modifier son contenu dynamiquement en réaction à des évènements : `onclick`, `setInterval`, `onload`, `onresize`, etc.
- Ajoutez ensuite un exemple de modification dynamique du style appliqué à un élément comme `<div>` ou `<span>` pour changer par exemple la couleur de fond du texte par exemple lors du survol de la souris.
- En mettant la propriété style `display` d'un élement à la valeur `"none"`, il est possible de le masquer complétement. Et il est ensuite possible de l'afficher à nouveau en mettant sa valeur à `"null"` (ou `"block"`).
- Ajouter quelques fonctions plus sophistiquées en fonction de votre humeur : rotation image, élément qui suit la souris, calcul mathématique, petit jeu, ...

Pour réaliser le travail demandé, vous pouvez vous aider de la documentation complémentaire suivante :

- Des exemples : https://www.w3schools.com/js/js_examples.asp
- Les évènements : https://www.w3schools.com/js/js_events.asp
- Modification de la page : https://developer.mozilla.org/fr/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents
- Timer et intervalle : https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Timeouts_and_intervals

1.5 PHP

Le contenu des sites web est souvent dynamiques (moteur de recherche, site de streaming, réseau social). Une manière de faire varier ce qu'envoie le serveur Web est de générer le contenu des pages plutôt que de fournir des pages statiques. Ici, on se propose d'utiliser PHP 7 qui est présent sur les pages personnelles du CREMI.

Un fichier PHP se présente comme un fichier HTML avec des portions de code inséré dedans entre `<?php ... ?>`.

```
<!DOCTYPE html>
<html>
<body>
<?php echo "Mon premier script!"; ?>
</body>
</html>
```

Prérequis : Commencez par lire la documentation ci-dessous.

- Courte introduction à PHP : https://www.php.net/manual/fr/tutorial.php
- https://www.w3schools.com/php/default.asp
- https://www.php.net/manual/fr/tutorial.forms.php

Travail demandé :

- Créez un fichier PHP séparé dont le contenu est variable : affichage de la date, d'un nombre aléatoire, ...
- Ajoutez dans votre fichier HTML un formulaire avec un bouton *submit*.
- Traitez la requête dans le fichier PHP en affichant un message qui dépend des données du formulaire.

Pour réaliser le travail demandé, vous pouvez vous aider de la documentation complémentaire suivante :

- Doc PHP, Afficher une date : https://www.php.net/manual/fr/function.date.php
- Exemples de code : https://www.w3schools.com/php/php_examples.asp

1.6 Rendu Final

Au final, publiez votre site web au CREMI et rendez votre travail sur Moodle, conformément aux consignes indiquées.