TP6 – Langages Web

Cette feuille est consacrée à la découverte et à la pratique des langages du World Wide Web. Les exercices suivants vous demanderont successivement de créér localement et d'enrichir un petit site web. Un récapitulatif des éléments de syntaxe utilisés est fourni en annexe à la fin de ce sujet de TP ; vous pouvez le cas échéant consulter une documentation en ligne pour obtenir des informations complémentaire. Le site du World Wide Web Consortium (*W3C*) offre en outre 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/` , par exemple dans l'onglet "File Upload" ou "Direct Input".

1 Une page statique

La première étape consiste à produire un document HTML incorporant le contenu statique de la page ; ce format permet également de fournir des informations structurelles et sémantiques sur son contenu : découpage en sections et paragraphes, en-tête et pied-de-page, et mise en relief des informations exploitables automatiquement (emphase, dates, acronymes, *etc.*). Pour cet exercice, nous utiliserons la version 5 du langage HTML (voir annexe).

Si vous êtes en panne d'inspiration pour le contenu, vous pouvez reproduire celui de cette feuille de TD au format HTML.

1.1 En-tête et métadonnées

Pour des raisons historiques, tout document HTML5 débute par une balise spéciale `<!DOCTYPE html>` ; en outre, il doit comporter au minimum un titre (non-vide). Une page simpliste est donc :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Le titre de ma page</title>
  </head>
  <body>
    <p>Ma page est chouette.</p>
  </body>
</html>
```

Validez sa syntaxe. Ouvrez ensuite le fichier à l'aide du navigateur de votre choix. Continuez à tester de cette manière les changements que vous apportez au fur et à mesure des exercices.

Consultez les informations sur la balise `meta` que nous avons mises à la fin de ce sujet. Ajoutez dans la section head quelques métadonnées au document, sous la forme de balises `meta` dotées d'un attribut `name` (ayant par exemple la valeur `author`, `keywords` ou `description`) et d'un attribut `content` ayant pour valeur les données correspondantes (nom, liste de mots-clés séparés par des virgules ou brève phrase explicative). On peut alors retrouver ces informations quand on regarde "View Page Info" du menu clic-droit.

1.2 Structure du document

Consultez les informations sur les balises de structure que nous avons mises à la fin de ce sujet. Ajoutez dans le corps du document un en-tête et un pied-de-page entourant le contenu principal. Cette délimitation est utile pour les lecteurs d'écran pour personnes aveugles, qui peuvent ainsi accéder directement au contenu utile de la page.

Ajoutez ensuite en guise de contenu un petit nombre de sections et sous-sections. De même, ce découpage est utile pour les lecteurs d'écran pour naviguer facilement entre les grandes sections du document. 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"`. Vous pouvez faire pointer vos liens, au choix, vers une balise `section` ou directement vers le titre (`h1-h6`) correspondant. Il vous faudra bien sûr ajouter du contenu pour que le navigateur doive effectivement effectuer un défilement pour atteindre la cible du lien.

1.3 Contenu et mise en forme

Ajoutez maintenant un peu de contenu dans chacune de vos différentes sections. Consultez les informations sur le marquage HTML pour distinguer dans le code source certaines informations sémantiques, que nous avons mises à la fin du sujet, et utilisez-les sur votre document : séparation en paragraphes, emphase, listes d'éléments, *etc.* Ajoutez également des éléments hypertextuels : liens externes, images, voire contenus multimédias (si vous êtes en avance).

N'oubliez de vérifier la conformité de votre document, et pas seulement en examinant son rendu dans un navigateur !

2 Un coup de peinture

Un document HTML pur permet de transmettre un document, mais sa présentation dépend des réglages par défaut du logiciel utilisé pour le visualiser, qui sont souvent plutôt spartiates. Afin de modifier la présentation du document sans devoir altérer son contenu, on utilise des feuilles de style (CSS), qui déterminent le rendu de chacun des éléments d'un

document HTML. Lisez l'annexe CSS en fin de ce sujet et vous pourrez ausi vous référer au validateur CSS du W3C : `http://jigsaw.w3.org/css-validator/`.

2.1 Inclusion de CSS

Plusieurs méthodes alternatives permettent d'attacher des styles CSS à des éléments HTML. Tout d'abord, il est possible d'attacher des déclarations directement à une balise HTML, sans passer par un sélecteur, de la manière suivante :

```
<p style="background-color:grey; color:red; font-weight:bold;">
Texte en gras, rouge sur fond gris.</p>
```

Il est également possible d'ajouter des règles dans l'en-tête d'un document HTML :

```
<style type="text/css">
p.serious {
    background-color:grey;
    font-weight:bold;
    color:red;
}
</style>

...

<p class="serious">Texte en gras, rouge sur fond gris.</p>
```

Enfin, il est possible de lister un ensemble de règles dans un fichier séparé portant l'extension `.css` et de l'inclure en ajoutant la balise suivante dans l'en-tête du document HTML :

```
<link rel="stylesheet" href="nom-du-fichier.css">
```

Cette dernière méthode est souvent recommandée pour faciliter la mise en cache de fichiers CSS par les navigateurs, un même fichier CSS pouvant décrire le style de nombreuses pages sur un site web. Le fonctionnement de CSS veut qu'en cas de définitions conflictuelles, la règle la plus proche de l'élément s'applique (ou, à source égale, la dernière règle rencontrée).

2.2 Mise en pratique

1. Pour commencer, donnez une couleur propre aux titres de sections et sous-sections (`h1`, `h2`, `h3`) en fonction de leur importance.
2. Changez ensuite le style des liens pour qu'ils apparaissent en rouge par défaut, et en gris lorsqu'ils ont été visités (élément `a:visited`)
3. À l'aide de la propriété `font-style`, pouvant adopter (entre autres) les valeurs `oblique` ou `normal`, modifier l'apparence de tous les paragraphes de texte de votre document HTML. Qu'advient-il lorsqu'un paragraphe affecté contient du texte mis en emphase avec `<em>` ? Corrigez ce problème à l'aide d'une nouvelle règle CSS.

4. Délimitez quelques portions de texte avec des balises `span` ayant un attribut `class` contenant la valeur `"warning"`. À l'aide d'une règle CSS, mettez ces portions en rouge. Vous pouvez également délimiter ainsi de grandes portions du document avec des balises `div`.
5. Ajoutez ensuite un attribut `class` à un élément de votre document HTML, et donnez-lui la valeur `"waldo"`. Ajoutez ensuite une règle CSS qui change la couleur des éléments de cette classe lorsqu'il sont survolés (élément `.waldo:hover`).

2.3 Bonus

En guise d'exercice supplémentaire, ajoutez une bordure autour du texte d'un paragraphe spécifique (en plaçant directement un identifiant unique sur ce paragraphe).

3 Contenu dynamique, côté client

La page web produite jusqu'à présent est, par nature, statique (en dépit du fait que certains éléments peuvent changer d'apparence au survol de la souris). Afin d'en modifier le contenu dynamiquement, il convient de programmer de nouveaux comportements ; le langage employé à cet effet par l'ensemble des navigateurs web actuels est Javascript (voir annexe). Dans ce contexte, son API la plus importante est celle qui permet d'interagir avec l'arbre syntaxique d'un document HTML : le DOM (Document Object Model).

Pour inclure du code JavaScript dans une page web, vous pouvez l'ajouter directement au moyen d'une paire de balises `<script> /* Votre code. */ </script>`, ou lier un fichier portant par convention l'extension `.js` avec `<script src="nom-du-fichier.js"></script>`, ou encore fournir un fragment de code lié à un élément HTML à l'aide de certains attributs spéciaux (`onclick`, `onmousemove`, `onfocus`, *etc.*).

À titre d'exemple, le fragment de code HTML/JS suivant permet d'inclure un fragment de texte dont le style change en cliquant sur un bouton voisin :

```
<script>
function boldText() {
  txt = document.getElementById("le_texte");
  txt.style.fontWeight="bold";
}
</script>

<p id="le_texte">Un fragment de texte.
<button onclick="boldText()">En gras</button>
```

3.1 Masquage de contenu

Ajoutez une liste d'éléments introduite par un fragment de texte dans votre document HTML. Masquez ensuite le contenu de la liste en lui associant la propriété CSS appropriée

(`visiblity: hidden`), puis ajoutez un callback permettant de faire réapparaître le contenu de la liste (`visiblity: visible`) en cliquant sur le texte au dessus.

Comment se comporte votre page web dans un navigateur incapable d'exécuter du code JavaScript (par exemple, un navigateur en mode texte fonctionnant dans un terminal) ?

Si vous avez le temps, étendez votre fonction pour que cliquer de nouveau sur le texte cache de nouveau la liste. On peut utiliser pour cela une variable globale contenant par exemple `true` au départ. La syntaxe du if est la même qu'en C.

3.2 Bonus : Un coup de peinture, phase II

Au moyen d'un champ textuel (`<input id="colorPicker" type="text">`) et de la propriété `value` de l'élément correspondant, créez une classe d'éléments dont le fond adopte la couleur indiquée par dans le champ `colorPicker` lorsqu'on leur clique dessus. Indice : vous pouvez utiliser le style `background-color: #RRVVBB`, où R, V et B sont des chiffres hexadécimaux.

Si vous avez encore de l'avance, ajoutez une vérification qui colore le fond du champ en rouge si son contenu n'est pas une couleur valide (une série de six caractères appartenant à l'intervalle `0-9a-f`).

4 Contenu dynamique, côté serveur

Pour finir, il est également possible de générer dynamiquement le contenu d'une page web côté serveur. Cela permet notamment aux visiteurs de communiquer par le biais du Web, de préserver la confidentialité de leurs données, et de construire des pages web répondant à des requêtes spécifiques. Dans le cadre de cet exercice, PHP sera le langage support, mais il existe de nombreuses alternatives : JavaScript/Node.js, Ruby/Rails, Python/Django, Java/JSP, C#/ASP.NET, Perl, C*, Python,*etc.*/CGI, et bien d'autres. La documentation du langage PHP est accessible à l'adresse suivante : `http://php.net/manual/`.

Pour exécuter un script PHP et en générer une page web, il faut disposer d'un serveur web fonctionnel. Le CREMI en met un à votre disposition, selon les modalités décrites ici : `https://services.emi.u-bordeaux.fr/intranet/spip.php?article11`.

Vous pouvez placer vos scripts PHP dans le répertoire `/net/www/<votre_login>/`, et accéder au résultat à l'URL `http://<prenom>.<nom>.emi.u-bordeaux.fr/`.

Le langage PHP permet l'utilisation d'un fichier HTML comme gabarit (on parle de *templating*) : il est possible d'inclure à tout moment dans du code HTML une balise PHP ouvrante (`<?php`), suivie d'un fragment de code PHP puis d'une balise fermante (`?>`). Le code PHP est alors évalué de manière transparente côté serveur, et remplacé par le résultat de sa sortie standard, produit notamment par la fonction `echo`, comme dans l'exemple suivant, à sauver dans un fichier test.php (il faut l'extension .php pour que l'interprétation php soit effective) :

```
<!doctype html>
```

```
<title>Simulateur de dé</title>

<p> Le résultat est :

<?php

   echo(mt_rand(1,6));

?>.
```

Mettez en place ce code sur votre espace personnel, chargez la page et actualisez-la : observez le résultat.

4.1 Fortune PHP

Constituez un fichier texte contenant un ensemble de phrases, en utilisant un caractère spécial comme séparateur [1]. Placez ce fichier dans votre répertoire Web, puis écrivez un script PHP servant dans un document HTML l'une des phrases du fichier, choisie au hasard. Utilisez au besoin la fonction `explode(delim,str)`, qui convertit une chaîne formée de fragments séparés par un délimiteur en un tableau de chaînes, de la façon suivante :

```
$str = "abc;d;ef;gh;i";
$t = explode(";",$str);
echo ($t[0]); // "abc"
```

Pour lire le fichier, vous pouvez utiliser la fonction `file_get_contents($filename)`, qui retourne le contenu d'un fichier sous la forme d'une chaîne de caractères.

1. Vous pouvez récupérer au besoin l'un des fichiers textes contenus dans le répertoire `/usr/share/games/fortunes/`, à condition de ne pas y perdre trop de temps.

Annexe : Memento langages Web

HTML 5

Un document HTML se compose d'un ensemble de *balises* (*tags* en anglais), formées en encadrant le mot-clé correspondant par des chevrons de la manière suivante : `<tag>`.

Une balise peut englober une partie d'un document HTML en l'encadrant par une balise initiale (dite *ouvrante*) et une balise finale (*fermante*) incluant un antislash : `</tag>`.

Les balises non-fermantes peuvent inclure un ou plusieurs *attributs* séparés par des espaces. Ces attributs suivent le mot-clé et sont éventuellement munis d'une *valeur*, spécifiée entre guillemets après l'attribut et séparée de celui-ci par le symbole `=`, de la manière suivante : `<tag attribut1="valeur1" attribut2="valeur2" attribut3 attribut4>`.

Voici quelques-unes des balises dont vous pouvez avoir l'usage dans le cadre de ce TD ; certaines balises fermantes, listées entre parenthèses, sont optionnelles :

- `<html> </html>`, `<head> </head>` et `<body> </body>` (optionnels) : permettent de délimiter respectivement l'ensemble du document, son en-tête et son corps (son contenu propre)
- `<meta>` : permet d'inclure des méta-données dans l'en-tête du document (date, auteur, langue, encodage, mots-clés, *etc.*)
- `<header> </header>`, `<main> </main>` et `<footer> </footer>` : délimitent les différentes parties du corps du document (en-tête, principal, pied-de-page)
- `<section> </section>` et `<nav> </nav>` : délimitent explicitement une section (ou sous-section) du document et une zone de navigation (par ex. une collection de liens)
- `<div> </div` et `<span> </span>` : délimitent une portion du document, pour y appliquer par exemple des styles, encadrements, liens... Contrairement à `section` et `nav`, il n'y a pas de sémantique attachée à ces balises. `div` effectue un retour à la ligne, `span` non.
- `<h1> </h1>` ... `<h6> </h6>` : marquent un titre de section, par ordre hiérarchique décroissant (`h1` = premier niveau, `h2` = second niveau, *etc.*), et délimite implictement une section
- `<a> </a>` : transforme un contenu en un lien hypertexte, pointant vers l'URL spécifiée dans l'attribut `href`
- `<p>` (`</p>`) : marque un nouveau paragraphe de texte
- `<br/` Retour à la ligne, sans marquer un nouveau paragraphe
- `<em> </em>` : ajoute une emphase sur un fragment de texte
- `<img>` : permet d'inclure une image, dont l'emplacement est désigné par l'attribut `src`. Il faut également ajouter un attribut `alt` dont le contenu sera affiché par le navigateur à la place de l'image quand l'image ne fonctionne pas, et pour les internautes aveugles qui ne peuvent donc pas voir l'image.
- `<ul> </ul>` et `<ol> </ol>` : délimitent une liste d'éléments, respectivement non-ordonnée et ordonnée
- `<li>` (`</li>`) : marque un nouvel élément dans une liste

— `<audio> </audio>`, `<video> </video>` : permettent d'inclure du contenu multimédia dans une page

Document de référence : `https://www.w3.org/TR/html5/`

CSS

Une spécification CSS se compose d'un ensemble de règles, chacune formée d'un *sélecteur* et d'un *bloc de déclarations*. Le sélecteur détermine à quels éléments d'un document HTML appliquer la règle, et chacune des déclarations du bloc spécifie un aspect du rendu de ces éléments.

Voici quelques exemples de sélecteurs :

— `balise` sélectionne les éléments `<balise>` du document
— `.classe` sélectionne les éléments ayant l'attribut `class="classe"`
— `#ident` sélectionne l'élément d'attribut `id="ident"`
— `:hover` sélectionne les éléments au survol de la souris (les mots-clés `link`, `visited`, `active` et `focus` sont également reconnus)
— `bal1 bal2` sélectionne les éléments `<bal2>` inclus dans un élément `<bal1>` dans le document

Par exemple, la règle CSS suivante met en italique les titres de niveau 1 :

```
h1 {
    font-style: italic;
}
```

Ces syntaxes peuvent se combiner ; ainsi le sélecteur `ul.biblio a:visited` désigne tous les liens visités dans les listes ayant la classe `biblio`.

Un bloc de déclaration se compose d'une liste de *déclarations* entre accolades, séparées par des point-virgules. Chaque déclaration associe une *propriété* à une *valeur*, en les séparant par un caractère deux-points.

À titre d'exemple, la règle CSS suivante grise et met en italiques les liens visités dans une bibliographie (une liste non-ordonnée ayant la classe `biblio`) :

```
ul.biblio a:visited {
    font-style: italic;
    color: grey;
}
```

Document de référence : `https://www.w3.org/TR/CSS/`

JavaScript

Le JavaScript est un langage de programmation faiblement typé, dont la syntaxe s'apparente superficiellement à celle du Java. Une spécification utilisable du langage dépasse largement le cadre de ce mémento, mais plusieurs ressources en ligne sont disponibles :

— La documentation du Mozilla Developper Network : `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/About`
— La référence du W3C : `https://www.w3.org/standards/webdesign/script`

Voici cependant quelques éléments utiles dans le cadre de ce TP :

— Le système de typage de JS étant extrêmement laxiste, aucune annotation de type n'est nécessaire lors de la déclaration d'une variable ou d'une fonction (à la différence de Java).
— Toutes les variables sont globales par défaut. Le mot-clé `var` permet de déclarer une variable comme locale à la fonction.
— Les fonctions, déclarées à l'aide du mot-clé `function` sont des objets de premier ordre ; c'est-à-dire que, comme en Python, les noms de fonctions peuvent être employés comme des variables normales.
— Une variable globale `document` est automatiquement créé au chargement d'un script dans une page web ; elle permet d'accéder au contenu du document HTML (une fois celui-ci chargé).
— Dans le cas où un script JS doit s'exécuter à l'ouverture de la page, il convient d'attendre que le document soit entièrement chargé. Ceci peut être accompli en écrivant le contenu du script dans une fonction `main` et en incluant l'instruction suivante dans le script : `window.addEventListener('load', main);`.

Les méthodes suivantes sont disponibles pour la classe `Document`, et retournent un élément ou un tableau d'éléments :

— `document.getElementById("id_html");`
— `document.getElementsByClassName("classe_html");`
— `document.getElementsByTagName("balise_html");`

Les propriétés et méthodes suivantes sont disponibles pour la classe `Element`, et permettent d'accéder (et de modifier) son contenu, sa classe, son identifiant, ses attributs et d'y rattacher des callbacks (fonctions invoquées lorsque l'élément est survolé, cliqué, sélectionné, activé, *etc.*) :

— `elem.innerHTML`
— `elem.class`
— `elem.id`
— `elem.getAttribute("attribut")`
— `elem.setAttribute("attribut", "valeur")`
— `elem.removeAttribute("attribut")`
— `elem.addEventListener("action", fonction);`