INFORMATIQUE
Sciences
Université Paris Cité

Programmation Web
Poly du TP n° 1 : Révisions de HTML et CSS

Un site Web fonctionne sur deux niveaux : sur le serveur et sur la machine de l'utilisateur (le client). La connexion entre les deux se fait via le protocole HTTP. Le langage `html` permet de décrire le contenu des pages Web. Le langage `css` peut être associé à `html` pour mettre en forme ce contenu. Ces deux langages sont interprétés par les navigateurs Web sur la machine client.

1 HTML

1.1 Document HTML

Un document (ou page) `html` prend l'extension `.html`. Voici un document `html` minimal, avec son entête (`<head></head>`) et son corps (`<body></body>`).

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <meta name="description" content="courte description">
    <title>Titre de la page</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h1>Ma page web</h1>
    <p>Un paragraphe</p>
    <p>Un <em>autre</em> paragraphe</p>
  </body>
</html>
```

Les balises (*tags*) peuvent aller par paire (`<head></head>`) et entourer un contenu, ou être seules (`<link>`). Dans les deux cas, elles peuvent avoir des *attributs* indiqués avec la syntaxe `attribut="valeur"`. Pour une liste des balises et des attributs existants, on pourra se référer à http://www.w3schools.com/tags.

1.2 Head

Les balises dans l'entête servent à donner des méta-informations sur la votre page. Voici trois balises importantes pouvant être mises dans `<head></head>` :

- `<title></title>`, le titre de la page, qui apparaîtra dans l'onglet de votre navigateur ou dans le titre d'un résultat d'un moteur de recherche.
- `<meta charset="UTF-8">` précisant l'encodage de la page, une information non obligatoire mais essentielle.
- `<meta name="description" content="courte description du contenu">`, une courte description du contenu de la page qui peut être utilisée par les moteurs de recherche ou les réseaux sociaux.

D'autres balises d'entête existent, comme par exemple `author` et `copyright`, mais elles n'ont qu'une faible utilité, sauf éventuellement pour les développeurs regardant le code source.

1.3 Body

Texte. L'élément le plus basique du corps d'une page `html` est le texte. Des balises peuvent être utilisées pour le mettre en forme, par exemple `<em></em>` pour mettre un ou plusieurs mots en évidence — souvent en italique par défaut – mais il vaut mieux n'utiliser ce type de balises que ponctuellement. Si un motif de mise en forme se répète dans le texte, il est préférable d'utiliser du `css`, plus modulable et aidant la page à garder une unité.

Balises de mise en page. La mise en page du texte peut être contrainte par des balises telles que `<h1></h1>`, ... `<h6></h6>` pour former des titres, `<p></p>` pour découper le texte en paragraphes, `<br/>` pour sauter une ligne, etc. On peut créer des listes non numérotées, (`<ul><li>e1</li><li>e2</li>...</ul>`) des listes numérotées (`<ol><li>e1</li><li>e2</li>...</ol>`) ... Les balises génériques `<div>` et `<span>` peuvent aussi servir à créer des mises en formes personnalisées par du code `css`.

Liens et images. Des liens et images peuvent être insérés dans une page sur le modèle suivant :

```
<a href="http://www.exemple.com">Exemple de lien</a>
<img src="logoUPC.jpg" alt="Le logo de Université Paris Cité">
```

où la valeur de `alt` est le texte affiché si l'image ne peut pas être affichée.

1.4 Balises sémantiques

En principe, une page `html` pourrait être entièrement écrite avec un nombre très réduit de balises génériques, mais la norme du langage a récemment introduit un grand nombre de balises *sémantiques*, qu'il est à présent recommandé d'utiliser.

Les balises sémantiques spécifient, via leur nom, la nature de ce que devrait être leur contenu et leur fonction – article, section, barre de navigation etc. Elles sont un moyen pour le rédacteur de la page (ou le relecteur de son code source) de clarifier les rôles de ses différents éléments. Les balises sémantiques les plus importantes sont :

- `<header></header>` contenant l'entête affichée de la page (à ne pas confondre avec `<head>`,
- `<footer></footer>` contenant le pied de page, éventuellement le pied d'une section,
- `<nav></nav>` introduisant une section dédiée aux liens de navigation,
- `<section></section>` introduisant une section générique, habituellement avec un titre,
- `<article></article>` introduisant un contenu qui pourrait exister indépendamment du reste de la page – un article de blog, un message, une description de produit etc.

Toutes les balises sémantiques s'affichent de la même façon au premier abord. Les différences entre elles seront apportées par le code `css` associé à la page.

1.5 Formulaires

Un formulaire est introduit par les balises `<form></form>`. Le rôle de ces deux balises est moins de faire de la mise en page que de délimiter un ou plusieurs éléments permettant à l'utilisateur de saisir des données, celles qui seront envoyées au serveur. Voici tout d'abord des exemples de balises de saisie. Dans chaque cas, la valeur de l'attribut `name` permettra d'identifier l'élément du côté du serveur :

- `<input type="text" name="nom" value="remplissage">`, un champ texte mono-ligne, pré-rempli par un texte dans la zone de saisie qui disparaîtra dès qu'on sélectionnera cette zone,
- `<input type="password" name="nom">`. un champ de texte affichant des étoiles au lieu des caractères saisis,

- `<textarea name="nom" cols="40" rows="4"></textarea>`, un champ texte multiligne – la taille peut aussi être spécifiée en `css`
- `<input type="checkbox" name="nom">` un carré que l'on peut cocher et décocher,
- plusieurs boutons-radio de même nom mais pas de même valeur, l'utilisateur ne pouvant en cocher qu'un :

```
<input type="radio" name="nom" value="1">
<input type="radio" name="nom" value="2">
<input type="radio" name="nom" value="3">
```

- une liste déroulante – les valeurs affichés sont celles entre les balises `<option><option/>` (et non les valeurs de `value`) :

```
<select name="Nom">
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
</select>
```

- `<input type="submit" value="Envoyer">` pour envoyer le contenu du formulaire au serveur. méthode `method`.

Tous ces éléments de saisie ne sont accompagnés d'aucun texte. Pour y associer un texte, on peut utiliser la balise `label` :

```
<input type="checkbox" id="id" name="nom"><label for="id">case a cocher</label>
```

Les différents éléments d'un formulaire peuvent être organisés soit utilisant des blocs (`<div>`, `<span>`, etc.), soit avec les balises `<fieldset></fieldset>`. Voici un exemple de formulaire. Son attribut `method` spécifie le type de requête `html` (`GET`/`POST`) à envoyer ici à un serveur local :

```
<form method="post" action ="http://localhost:8080/" >
  <fieldset>
    <legend>Vos coordonnées</legend> <!-- Titre du fieldset -->
    <label for="nom">Quel est votre nom ?</label>
    <input type="text" name="nom" id="nom" />
     <label for="prenom">Quel est votre prénom ?</label>
     <input type="text" name="prenom" id="prenom" />
  </fieldset>
  <fieldset>
    <legend>Votre souhait</legend> <!-- Titre du fieldset -->
     <p>Faites un souhait que vous voudriez voir exaucé :
       <input type="radio" name="souhait" value="riche" id="riche" />
       <label for="riche">Etre riche</label>
       <input type="radio" name="souhait" value="célebre" id="célebre" />
       <label for="celebre">Etre celebre</label>
       <input type="radio" name="souhait" value="autre" id="autre" />
       <label for="autre">Autre...</label>
     </p>
     <p>
       <label for="precisions">Si "Autre", veuillez préciser :</label>
       <textarea name="précisions" id="précisions" cols="40" rows="4"></textarea>
     </p>
  </fieldset>
  <input type="submit" value="Envoyer" >
</form>
```

1.6 Validation et débogage

Le W3C fournit un validateur de pages `html` que l'on trouve en http://validator.w3.org/. On peut y entrer une URL. Une page est valide si le validateur affiche "This document was successfully checked as HTML5 !". Il est fréquent qu'il y ait des warnings.

1.7 DOM

Pour l'instant nous avons considéré les documents `html` comme des fichiers texte (le code source de la page) rendus comme des pages web dans un navigateur. Une autre manière de représenter un document `html` est de considérer sa structure arborescente, le DOM (Document Object Model). On peut visualiser le DOM et naviguer entre ses différents éléments via les outils de débogage du navigateur, mais aussi via des langages de script tournant du côté client – ce que nous ferons sous peu avec `Javascript` – qui peuvent modifier ou enrichir ce DOM de manière directe, voire le construire totalement au chargement de la page.

2 CSS

La manière dont les éléments d'une liste s'affichent dans un page – par exemple, la couleur du texte, ou la forme des points qui précèdent chaque item – peut être spécifiée de manière directe en `html` via l'attribut `style` :

```
<ul style="color:red; list-style-type:square;">
  <li>Un élément</li>
  <li>Un autre</li>
</ul>
```

Il est clair que cette forme d'écriture est très peu flexible : si une page contient plusieurs listes devant être affichées avec le même aspect, il faudra repréciser ce style pour chacune d'entre elles. Faire un autre choix d'affichage imposera en outre de corriger l'attribut `style` de chacune des listes.

Une solution bien plus flexible est de centraliser dans une *feuille de style*, un fichier `css`, les différents styles d'affichage des éléments d'une page : une unique correction de ce fichier affectera tous les éléments d'un même style. Plusieurs fichiers `css` candidats peuvent par ailleurs coexister

pour une même page web, le choix de l'un ou l'autre se faisant à la volée par exemple suivant le navigateur ou la taille de l'écran, ou encore par un choix de l'utilisateur.

La liaison entre une page et une feuille de style se fait dans la partie `head` de la page à l'aide de la balise `link` :

```
<link rel="stylesheet" href="maFeuillle.css">
```

On pourra par exemple écrire dans le fichier `css` les spécifications suivantes, qui affecteront respectivement les listes non numérotées (`<ul>`), les paragraphes (`<p>`) et tous les éléments (`*`). Dans chaque groupe, les composantes gauche de chaque entrée (`color`, `list-style-type`, ...) sont appelées *propriétés* `css`, les parties droites (`red`, `square`, ...) étant les *valeurs* données à ces propriétés. La combinaison d'une propriété est de sa valeur est appelée une *directive* `css` :

```
ul {
  color : red;
  list-style-type : square;
}
p {
  color : blue;
  font-size : small;
}
* {
  font-family : Impact, "Arial Black", Arial, Verdana, sans-serif;
  text-align : justify;
}
```

2.1 Les sélecteurs

Il y a plusieurs moyens de spécifier les éléments d'une page web affectés par l'un des styles d'un fichier `css`. On peut sélectionner toutes les balises d'un type donné, par exemple toutes les balises `<p></p>` :

```
p {
  // ...
}
```

On peut sélectionner une balise unique grâce à son `id` – dans le fichier `css`, l'id doit être précédé d'un dièse. Pour décorer par exemple l'élement

```
<img src="images/logo.png" alt="Logo du site" id="logo" >}
```

on écrira dans le fichier `css` :

```
#logo {
  // ...
}
```

Enfin, on peut sélectionner un ensemble de balises – pas nécessairement du même type – en leur donnant une valeur commune d'attribut `class`. Dans le `css`, cette valeur d'attribut sera précédée d'un point. Ainsi pour décorer les deux balises suivantes :

```
<div class="gallerie"><img src="images/foo.png" alt="foo"></div>
<div class="gallerie"><img src="images/bar.png" alt="bar"></div>
```

on écrira dans le `css` :

```
.gallerie {
  // ...
}
```

Ces différents sélecteurs peuvent être combinés : par exemple, `p.liens` concerne les balises `p` de classe `liens`. Un élément peut aussi être spécifié comme étant de plusieurs classes, et être donc affecté par une combinaison des choix de styles pour chacune de celles-ci.

Une liste exhaustive des sélecteurs est disponible sur le site du W3C https://www.w3.org/TR/css3-selectors/#selectors. Nous reverrons ces sélecteurs lors de l'étude de `jQuery`.

2.2 Positionnement

Du point de vue de la mise en page, il existe deux grands types de balise : les *blocs* et les *inlines* :

- Les balises blocs, par exemple `<p></p>`, positionnent leurs éléments internes en introduisant un retour à la ligne avant et après de groupe d'éléments. La balise *bloc* générique est `<div></div>`.
- Les balises *inline* en revanche n'introduisent aucun retour à la ligne. En font partie les balises liens `<a></a>`. La balise *inline* générique est `<span></span>`.

Les blocs sont affichés par défaut les uns à la suite des autres, dans l'ordre de leur déclaration dans le fichier `html` – ils obéissent à la directive par défaut `position:static`. Cette suite de blocs positionnés « naturellement » s'appelle *flux* des éléments. Il est possible de modifier ce flux.

- **Positionnement relatif.** La directive `position:relative` permet de positionner un bloc en le décalant (*e.g* de 20px vers la gauche et 30px vers le bas) de sa position naturelle. Ce déplacement ne modifie pas le flux : les autres blocs se comportent comme si le bloc positionné relativement se situait à sa place « normale ». Les propriétés `top`/`bottom` et `left`/`right` permettent de spécifier ce décalage pour chaque axe.
- **Positionnement absolu.** La directive `position:absolute` permet de positionner un bloc de manière absolue dans le système de coordonnées de son premier ancêtre n'ayant pas un positionnement `static`, ou dans le système de coordonnées de la page si un tel ancêtre n'existe pas. Le bloc est dans ce cas retiré du flux et n'aura plus d'impact sur les autres blocs.
- **Positionnement fixe**. La directive `position:fixed` permet de positionner un bloc de manière absolue dans le système de coordonnées de la fenêtre. Le bloc est alors retiré du flux, est affiché au même emplacement indépendamment de tout défilement de la page. Les propriétés `top`, `bottom`, `left` et `right` permettent de spécifier les distances entre chaque bord du bloc et le bord de la fenêtre qui lui fait face.
- **Positionnement flottant** La propriété `float` permet de modifier le flux d'une manière plus complexe que la propriété `position`. En fonction de sa valeur (`left ou right`), celle-ci a pour effet de positionner le bloc dans le flux à sa position naturelle puis de le retirer et de le réinsérer le plus à droite (ou à gauche) possible, jusqu'au bord de son bloc parent ou du premier élément flottant rencontré.
  La propriété `clear` permet de spécifier qu'aucun élément flottant n'est autorisé d'un côté (gauche ou droite) ou des deux côtés de l'élément. Ses valeurs sont `left, right et both`.
- **Z-index.** L'atération du flux entraîne parfois le masquage non désiré d'un bloc par un autre. La propriété `z-index`, à valeur entière, permet de forcer un bloc à être affiché au dessus d'un autre : plus sa valeur pour un bloc donné est élevée, plus le bloc sera mis en avant.