Exercices sur les interactions Homme-Machine

Table des matières

4.1 Énoncés

4.1.1 Structuration HTML

Exercice 4.1 (Listes)
Écrire une page HTML avec le titre « Mes emplacements » avec les éléments suivant :

1. La liste non numérotée (`ul`) des cinq pays « que je souhaite visiter ».
2. La liste numérotée (`ol`) des quatre villes « que je préfère » (dans l'ordre)
3. La liste descriptive (`dl`) des trois « derniers établissements fréquentés »
   (en `dt` : les années, en `dd` : l'établissement et la ville)
4. La liste (`ul`) de « mes deux endroits préférés » avec des puces de type `square`.

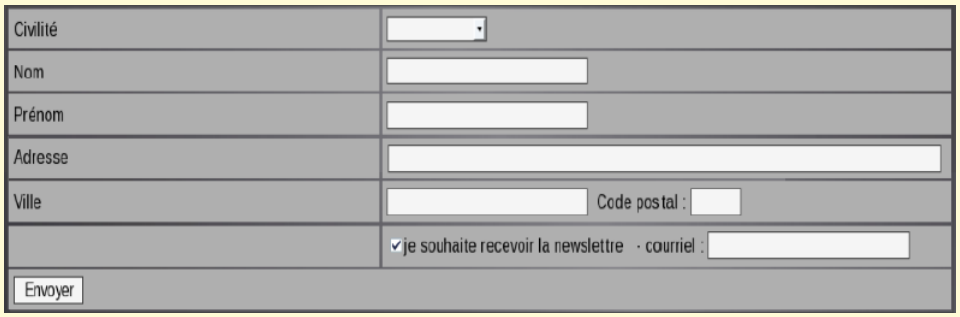
Exercice 4.2 (Emploi du temps)

1. Concevoir le fichier HTML d'un tableau donnant votre emploi du temps des semaines A (les colonnes correspondent aux jours et les lignes aux plages horaires).
2. Recopier le fichier précédent en l'adaptant aux semaines B, puis ajouter un lien pour passer d'une semaine A à une semaine B et réciproquement.

Exercice 4.3 (CV)
Créer un nouveau document HTML nommé « CV.html » avec le titre CV qui comportera les éléments suivants :

- Des titres de niveaux différents pour les informations usuelles (Nom, Prénom, âge)
- Des paragraphes pour décrire le cursus scolaire (collège, classe de seconde, spécialités de 1ière, options) ainsi que le projet de formation et le métier envisagé.
- Au moins un lien interne au document (avec dispositif de retour) et un lien vers un site Internet
- Une image (portrait) stockée localement dans le même répertoire que le document et une image (établissement) issue d'un site internet (donc pas stockée localement)

Exercice 4.4 (Formulaire)
Concevoir le fichier HTML d'un formulaire d'inscription du type ci-dessous :



Vous noterez que les éléments de ce formulaire sont placés dans un tableau ; la zone de civilité est un sélecteur à choix multiple (options : Monsieur, Madame) ; il y a une case à cocher ; le bouton est de type `button` ; les autres zones sont des champs texte de longueurs préétablies.
Donner à l'attribut `action` la valeur `"https://webia.lip6.fr/~muratetm/diu/Form_data.php"` afin de tester les deux méthodes d'envoi (attribut `method`) : `post` ou `get`.

4.1.2 Mise en forme CSS

Exercice 4.5 (Listes (suite))
Reprendre le fichier « Mes emplacements » créé précédemment. Définir la police par défaut pour l'ensemble de la page `font-family:"Trebuchet MS", Verdana, sans-serif;` (les guillemets encadrent un nom de police contenant un espace) et la taille par défaut des caractères `14px`.

1. Donner à la liste des cinq pays « que je souhaite visiter » un fond gris clair et une bordure de `5px` d'épaisseur. Attention : l'attribut`border` attend 3 données : l'épaisseur, le type de trait (`solid`, `dotted` ou `dashed`) et sa couleur.
2. Donner à la liste des quatre villes « que je préfère » des numéros en chiffres romains et une police de type `"Lucida Console", Courier, monospace` (dans cette famille de polices, chaque caractère utilise le même espace).
3. Encadrer d'une bordure bleue d'épaisseur `2px` la liste des trois « derniers établissements fréquentés » et mettre un `padding` de `20px` à l'élément `dl`.
4. Remplacer les puces carrées de la liste de « mes deux endroits préférés » par des images (définir l'attribut `list-style-type` ou `list-style-image` selon qu'il s'agit d'un caractère unicode ou d'une image), mettre un `margin` de `20px` et encadrer d'une ligne pointillée rouge.

Exercice 4.6 (Emploi du temps (suite))
Reprendre les fichiers « emploi du temps » créés précédemment.

1. Lier les deux fichiers d'emploi du temps (semaines A et B) à un seul fichier CSS. Mettre la colonne des heures et la ligne des jours sur fond gris avec la police de caractères en italique (appliquer à la balise `th`).
2. On souhaite que toutes les cases de la même matière aient une couleur de fond spécifique. Installer ce fonctionnement à l'aide d'un attribut de `class` pour chaque matière.
3. Au survol de la souris au-dessus de la case d'un jour, on aimerait que la ligne entière se détache avec un fond bleu pâle (heure comprise).
4. Difficile en CSS : au survol de la case d'un jour, on aimerait que la colonne entière, jour compris, se détache avec un fond gris clair (on peut adapter une solution trouvée sur internet)

Exercice 4.7 (CV (suite))
Reprendre le fichier « CV.html » créé précédemment. Améliorer la présentation de ce CV en réalisant ce cahier des charges.

1. On souhaite placer la photo en haut à gauche et les informations usuelles (Nom, Prénom, âge) en haut à droite. La hauteur de cette photo est fixée à `200px` et on souhaite que le texte occupe la même hauteur que l'image. Encadrer ce 1er ensemble et colorer légèrement son fond.
2. Le cursus scolaire, le projet de formation et le métier envisagé est placé en-dessous du 1er ensemble, dans une police plus petite et en italique.
3. Les liens sont écrits et soulignés en gris, les liens visités sont écrits en noir.
4. L'image de l'établissement est placée en regard du paragraphe le mentionnant, la hauteur de cette image est fixée à `100px`.

Exercice 4.8 (Menu)

1. Créer la structure d'un projet de site : cinq fichiers HTML (un contenant le futur menu et quatre autres qui seront les pages de destinations des liens de notre menu). Se contenter d'écrire la structure minimale d'une page HTML dans chacun d'entre eux afin de les rendre valides. Un titre et une phrase de texte suffiront pour les distinguer.
   Appeler ces fichiers : accueil.html, html.html, css.html, javascript.html et contact.html (enregistrer tous ces fichiers dans le même dossier).
2. Création de la page accueil.html : utiliser un élément `ul` pour le menu et des éléments `li` pour chaque onglet du menu. Insérer l'ensemble dans une balise `<nav>`. Á l'intérieur des éléments `li`, créer des liens ramenant vers les autres pages HTML grâce à l'élément `a`.
   Donner à chacun de ces liens un attribut `class` avec une valeur différente (par exemple `class="menu_HTML"`, etc.).
   Vérifier que cet ensemble est totalement fonctionnel bien que peu attractif : lorsqu'on clique sur un lien, on est bien ramené vers la page correspondante.
3. Créer un fichier CSS appelé menu.css, puis lier accueil.html et menu.css entre eux en utilisant l'élément HTML `link` d'en l'en-tête.
   - Mettre les bordures (`margin`) et le `padding` à `0px` pour tous les éléments `body` et `nav ul` ; Fixer la police dans `body` (`font-family: 'source code pro', Calibri, serif;`).
   - Enlever les puces des éléments `nav li` et placer ces éléments côte-à-côte grâce la propriété CSS `float` (`list-style-type:none; float:left;`).
   - Appliquer un `overflow : hidden` à l'élément contenant les éléments flottants (donc ici à `nav ul`), pour rétablir la hauteur de l'élément `ul`.
   - Ajouter une couleur de fond (`background-color=#142857`) à l'élément `nav` et étendre sa taille (`width`) à `100%` de son élément parent (l'élément `body`).
   - Changer le type des liens (`nav a`) en `inline-block`, ajouter un `padding` (`20px 30px`), supprimer le soulignement (`text-decoration:none;`) et modifier la couleur (`color:#fff`).
4. Attribuer une couleur ayant de la transparence à chaque onglet survolé par la souris : par exemple `.menu-js:hover{background-color:RGBa(240,210,80,0.15)}`[1]. Ajouter à ce sélecteur une bordure supérieure de `5px` de la couleur du voile, mais avec une transparence moins forte (`border-top: 5px solid RGBa(240,210,80,0.85)`). Adapter le `padding` supérieur à la présence de cette bordure en donnant au sélecteur `nav li:hover a` la valeur `padding:15px 30px 20px 30px`.

1. La couleur RGB(240,210,80) est un bleu opaque ; lui ajouter une transparence (0 :transparent-1 :opaque) à 0.15 pour obtenir un léger voile teinté. Tester les couleurs sur la page `https://web-color.aliasdmc.fr/convertisseur-couleur-rgb-hsl-hexa-predefini.php`

Exercice 4.9 (Animation)

1. Créer un document HTML contenant un ensemble d'articles, chaque article possède une image, un titre, une courte description et un bouton "En savoir plus" qui envoie vers les détails de l'article. Constituer le bloc d'un article avec la structure suivante :

```
<article class="view">
  <img src="images/****.jpg" alt="****" />
  <div class="mask">
    <h2>****</h2>
    <p>**** le court article ****</p>
    <a href="**** l'article ****" >En savoir plus</a>
  </div>
</article>
```

   Choisir un thème et au moins deux sujets pour ce thème. Les détails de l'article seront la page de Wikipédia traitant du sujet choisi, tandis que le court article sera les deux premières lignes de cet article. L'image de cet article doit être carrée et enregistrée dans un dossier `images` placé dans le même dossier que le fichier HTML.

2. Définir ainsi les attributs de six sélecteurs dans un fichier CSS :

Attributs de la classe `mask` (`.mask`)	`position:absolute; left:0; top:300px; background:rgba(0,0,0,0.3); color:white; text-align:center; width:300px; height:300px; transition:top 0.5s linear;`
Attributs du lien de la classe `mask` (`.mask a`)	`text-decoration:none; border-radius:5px; background:#000; color:white; font-size:1.5em; padding: 5px 10px 5px 10px; font-family:fantasy,sans-serif;`
Attributs de la classe `view` (`.view`)	`position:relative; width:300px; height:300px; border:10px solid #fff; overflow:hidden; float:left; margin:5px;`
Attributs de l'image (`.view img`)	`width:300px;height:300px;`
Attribut de `mask` au survol de `view`	`.view:hover .mask{top:0;}`
Attributs du corps (`body`)	`background: #FFFF77;`

   Les réglages CSS donnés ne sont pas indépendants les uns des autres : l'ordre compte ! Retrouver l'ordre de ces cinq sélecteurs pour que l'animation soit opérationnelle (Indication : il suffit de déplacer deux parties).

3. Vérifier que l'animation au survol d'un article fonctionne correctement, puis modifier la taille de l'image (passer à `400px`) et de la police de caractère (passer à `2em`).

4.1.3 Programmation Javascript

Exercice 4.10 (Diaporama)

1. Prévoir au moins trois images dans un même dossier `images`, les nommer `img1, img2, etc.`, puis créer une page HTML montrant `img1`.
2. Programmer l'alternance régulière entre cette image et les suivantes, en utilisant une instruction `setTimeout(changeImg, 2000);` où `changeImg` est une fonction Javascript.
3. Ajouter deux boutons : le 1er pour arrêter le diaporama et le 2e pour le reprendre.

1. Pour en apprendre plus sur les transitions CSS : `https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Transitions/Utiliser_transitions_CSS`

Exercice 4.11 (Jeu)

1. Utilisons la fonction `Math.random()` qui génère un nombre aléatoire entre 0 (inclus) et 1 (exclu) et la fonction `Math.floor(n)` qui donne la partie entière d'un nombre `n`, pour déterminer un nombre entier aléatoire compris entre `n1` (inclus) et `n2` (inclus) : expliquer pourquoi l'instruction `var rand=n1+Math.floor((n2-n1+1)*Math.random())` réalise cet objectif.
2. Écrire un script qui génère un nombre aléatoire entre 0 et 1000 et demande à l'utilisateur d'entrer un nombre (boîte de saisie de type `prompt`). Si le nombre entré est trop grand, il affiche le paragraphe « Ce nombre est trop grand, réessayez ! » ; adapté le message pour le cas où le nombre est trop petit. Quand l'utilisateur a trouvé le nombre, le message de félicitation contient le score (nombre d'essais).
3. Améliorer le script précédent en proposant une nouvelle partie. Le message final rappellera le ou les score(s) précédent(s).

Exercice 4.12 (Années bissextiles)

1. Créer un document HTML qui affiche une boîte de dialogue de saisie (de type `prompt`) avec le message suivant : « Entrez une année : ».
2. Capturer la réponse de l'utilisateur dans une variable et tester si l'année saisie par l'utilisateur est une année bissextile (l'année doit être au format `aaaa`).
   Pour cela, écrire une fonction `isAnneeBissextile()` qui reçoit en argument l'année à tester et retourne `true` dans le cas où l'année est bissextile et `false` sinon.
3. Afficher dans la page HTML le résultat du test : « aaaa est une année bissextile » ou bien « aaaa n'est pas une année bissextile ».
4. Tester dans un navigateur le bon fonctionnement de votre page WEB.

Rappel sur les années bissextiles :
Le pape Grégoire XIII a mis au point en 1582 un calendrier, qualifié encore aujourd'hui de grégorien, qui introduit les règles de calcul des années bissextiles : les années divisibles par 4 sont bissextiles et, lorsqu'il s'agit de la première année d'un nouveau siècle (par exemple 1900), l'année doit être divisible par 400 pour être bissextile. Ainsi 1700 et 1786 ne sont pas des années bissextiles alors que 1600 et 1784 le sont.

4.1.4 Dynamisme évènementiel

Exercice 4.13 (Quizz)

1. Écrire un texte à trou en HTML en prenant un proverbe ou une expression connue et en y remplaçant un des mots par des pointillés. Par exemple « Quand les ..... auront des dents ».
   L'espace pointillé à compléter doit être disposé dans une balise DIV sensible à un clic de souris : en cliquant sur la zone, on voit apparaître une zone de saisie qui permet de compléter le texte.
2. Prévoir un bouton pour la correction : en pressant le bouton, si la réponse est correcte, on obtient un message de félicitation et la réponse colorée en vert ; en cas de mauvaise réponse, le message et la couleur sont adaptés.
3. Finaliser le projet en écrivant dix phrases à trou.
   Le message final indique, cette fois, le nombre de bonnes réponses.

Exercice 4.14 (QCM)

1. Écrire une question de QCM en HTML en prévoyant quatre réponses assorties de cases à cocher. Le thème choisi n'a pas d'importance (éviter les thèmes grossiers). L'utilisateur doit pouvoir cocher une case ou aucune ; s'il en coche au moins deux, un message informatif lui rappelle qu'une seule réponse est correcte (prévoir une zone pour ce message).
2. Prévoir un bouton pour la correction : en pressant le bouton, si la réponse est correcte on obtient un message de félicitation ; en cas de mauvaise réponse, le message est adapté et la bonne réponse s'affiche dans la zone du message informatif.
3. Finaliser le projet en écrivant un QCM de cinq questions sur le même thème. Le message final indique la note finale, calculée en donnant 2 points par bonne réponse et en enlevant 0,5 point par mauvaise réponse.

Exercice 4.15 (Formulaire (suite))

Reprendre le formulaire d'inscription de la partie 1.
L'objet de cet exercice est d'enrichir et de valider les contenus saisis par un utilisateur :

1. On veut prévoir que la zone de saisie de l'adresse mail soit cachée (attribut `visibility` à `hidden`) tant que la case à cocher est désactivée ; elle ne devient visible que lorsque la case est activée.
2. Si la case à cocher est activée, donner le focus à la saisie de l'adresse mail. Le contrôle de l'adresse sera effectué à la perte du focus : on y cherchera la présence du caractère @.
3. Le champ du code postal ne doit contenir que 5 chiffres : contrôler chaque caractère entré en réagissant à l'évènement `keypress` pour cet élément, puis ajouter à la fonction déclenchée à la perte du focus un contrôle du nombre de chiffres.
4. Tous les champs doivent être remplis.
   Cette validation s'effectue à la fin, lorsque l'utilisateur clic sur le bouton « Envoyer ».

Vous noterez que les éléments de ce formulaire sont placés dans un tableau ; la zone de civilité est un sélecteur à choix multiple ; il y a une case à cocher ; le bouton est de type `button` ; les autres zones sont des champs texte de longueur préétablies.

Exercice 4.16 (Taquin)

L'objet de cet exercice est de créer un jeu de Taquin : un tableau de 4 fois 4 cases carrées dans lesquelles sont écrits les nombres de 1 à 15, la 16e case étant vide, ce qui permet de bouger une des cases avoisinant ce trou. Au départ, les nombres sont dans le désordre, le but du jeu est de les remettre dans l'ordre.

1. Créer l'apparence du jeu, chaque case étant un bouton HTML. Le désordre initial ne pouvant pas être aléatoire (sinon, il n'y aurait pas de solution une fois sur deux), reproduire la figure ci-contre. Régler le style CSS des boutons pour obtenir des carrés de `96px` de côté régulièrement espacés, par exemple avec une marge de `2px` ; fixer alors la largeur du jeu à `400px`.

10	9	11	8
1	12	5	4
7	2	6	13
15	3	14	

2. Écrire le code JavaScript qui, lorsqu'on clique sur un bouton voisin de la case vide, l'échange avec la case vide. Utiliser `document.getElementById()` pour récupérer un bouton, la méthode `bouton.removeChild(bouton.firstChild)` pour effacer le texte (`bouton.firstChild`) du bouton et la méthode `bouton.appendChild(valeur)` pour écrire un nouveau texte (`valeur`) dans le bouton.

Facultatif : Ajouter un compteur de coups ; enregistrer d'autres positions de départ, etc.