RÈGLEMENT SUR LE PLAGIAT

Extrait du règlement disciplinaire sur le plagiat ou la fraude de l'université de Montréal
Constitue un plagiat: faire exécuter son travail par un autre; utiliser, sans le mentionner, le travail d'autrui; **échanger des informations lors d'un examen**; falsifier des documents. Le plagiat est passible de sanctions allant jusqu'à l'exclusion du programme.

CONSIGNES

- Une feuille de notes US-Letter recto-verso est permise.
- Pas d'appareil électronique (sauf pour consulter l'heure)
- Le barème est donné à titre indicatif.

---

(12) 1. Questions de cours

(a) Qu'est-ce qu'une attaque XSS ?

(b) En quel language est écrit JQuery ?

(c) Qu'est que `sammy.js` et à quoi sert-il ? À quoi correspond une *route* dans ce framework ?

(d) À quoi sert `Bootstrap` ?

(e) Quel est le protocole utilisé pour envoyer une requête AJAX ?

(f) HTTP1.0 est-il un protocole à état ?

(g) Qu'est-ce qu'une faille include ? Comment s'en prévaloir ?

(h) Qu'est-ce que l'adaptation progressive ?

(i) À quoi sert `XQuery` ?

(j) Vous souhaitez offrir une mini calculatrice sous la forme d'une API REST qui offre l'addition et la multiplication de nombres entiers. Indiquez quelles sont les *routes* de votre application et la nature des réponses formulées par chacune.

(k) Qu'est-ce que `Mustache` ?

(l) Vous souhaitez offrir l'URL `<htdocs>/rep1/rep2/index.html` où `<htdocs>` est l'endroit où sont servies les ressources. Quelles priorités **minimales** doivent avoir les répertoires `rep1` et `rep2` et le fichier `index.html` ? Vous répondrez en indiquant les priorités façon Unix (*user*, *other* et *group*).

(20) 2. Le coin XPath

Considérez le bout de code HTML suivant:

```
<body>
 <div id="d1">
     <span id="a">span a</span>
     <div id="d5">div d5
         <span id="f">span f</span>
     </div>
 </div>
 <div id="d2">
     <span id="b">span b</span>
     <div id="d6" title="d5">div d6</div>
 </div>
 <div id="d3">
     <span id="c">span c</span>
     <div id="d7">div d7</div>
 </div>
 <div id="d4">
     <span id="d">span d</span>
     <div id="d8">div d8</div>
 </div>

 <span id="e">span e</span>
 <span> span1 </span>
 <span> span2 </span>

</body>
```

(a) Selon vous, ce code est-il valide au sens de la norme XHMTL 1.0 Strict ? Justifiez.

(b) Indiquer les **identificateurs** de tous les éléments sélectionnés par les commandes XPath qui suivent. Si un élément qui n'a pas d'attribut id est sélectionné, vous indiquerez la nature de l'élément sélectionné (ex: l'élément `span` contenant le texte *span2*).

(a) `//div`
(b) `//div[@id="1"]`
(c) `//div[@id]`
(d) `//span[@*]`
(e) `//div[last()]`
(f) `//div/div[1]`
(g) `//div`
(h) `//div[@id="d5"]/../span`
(i) `//div[position() != 2]`

(j) `//div[@id="d6"]/parent::node()/following-sibling::node()/span`

(k) Écrire deux règles CSS qui entrent en conflit pour la DOM de l'exemple, expliquez la nature du conflit et indiquez quelle règle est préférée et pourquoi.

(l) Écrire une règle CSS qui met en rouge le texte des éléments `span` de l'exemple qui n'ont pas d'attribut `id`.

(10) 3. Scriptage de la DOM via l'interface W3C

Considérez le code HTML de la question précédente, et soit `d` une variable qui pointe sur le nœud de la DOM correspondant à l'élément `div` d'identificateur `d1`:

```
var d = document.getElementById("d1");
```

(a) Complétez la commande javascript suivante de manière à pointer sur l'élément `span` d'identificateur `f`:
`d.getElementsByTagName(...)...`

(b) Même question, mais cette fois-ci vous ne devez utiliser aucune fonction:
`d. ...`

(c) Que vaut `d.previousSibling.nodeName` ?

(d) Que vaut `d.parentNode.nodeName` ?

(e) Écrire du code utilisant l'interface W3C permettant d'aller pointer sur le nœud de la DOM correspondant à l'avant dernier élément `span` du code HTML de la question précédente (celui de contenu `span 1`).

(20) 4. Le coin XML

Vous allez dans cette question gérer un language XML décrivant des instances comme celle qui suit qui contiennent une séquence d'au moins un `produit`. Un produit possède les attributs `prix` et `titre` qui sont obligatoires, `qte` qui est facultatif et qui vaut `1` par défaut. Le contenu d'un produit est une chaîne de caractères qui doit contenir un espace qui sépare le prénom (premier champ) non vide du nom (second champ) non vide.

```
<panier>
      <produit qte="10" prix="15.50" titre="La forêt des mal aimés">
          Pierre Lapointe
      </produit>
      <produit qte="7" prix="14.75" titre="Philémon chante">
         Philémon Cimon
      </produit>
      <produit qte="12" prix="14.50" titre="Un monde pour n'importe qui">
         Jérôme Minière
      </produit>
      <produit qte="5" prix="14.25" titre="L'été">
         Philémon Cimon
      </produit>
```

```
    <produit qte="10" prix="15.50" titre="La nuit éclaire le jour qui suit">
       Jérôme Minière
    </produit>
</panier>
```

(a) L'élément `panier` est-il un élément simple? Pourquoi ?

(b) L'élément `produit` est-il un élément simple? Pourquoi ?

(c) Écrivez un schéma XML qui permet de décrire un tel language. Votre schéma doit être le plus fidèle possible aux spécifications.

(d) Indiquez comment lier l'instance de l'exemple à ce schéma.

(e) Écrire une feuille de transformation XSLT qui prend en entrée une instance du langage défini précédemment et qui affiche en sortie une page HTML qui contient l'information de l'instance visualisée sous forme d'une table. Voici la page générée pour l'instance donnée en exemple. Le code HTML produit par votre feuille de transformation devra être le plus fidèle possible au rendu visuel de cette figure. Vous n'avez pas à vous soucier des entêtes HTML.

Bienvenue dans mon magasin:

auteur	titre	prix	quantité	action	
Pierre Lapointe	*La forêt des mal aimés*	15.50	10	+	-
Philémon Cimon	*Philémon chante*	14.75	7	+	-
Jérôme Minière	*Un monde pour n'importe qui*	14.50	12	+	-

W3C XHTML 1.0

(10) 5. Le coin JQuery

```
<html>
  <head>

    <script type="application/javascript" src="jquery.js"></script>

    <script type="application/javascript">
    $(document).ready(function(){

    jQuery.fn.f = function (n) {
      $("div:eq(" + n + ")").append( " + " +
        this.map(function () {
             alert(this.tagName);
             return this.attr("id");})
                        .get().join("- "));
      return this;
    };
```

```
    $("p").f(0)
          .find("span")
          .f(1)
          .css("text-decoration","underline")
          .end()
          .f(2)
          .css("text-decoration", "overline");

});  // ready
</script>

<style type="text/css">
p, div { margin:1px; padding:1px; font-weight:bold;
         font-size:16px; }
div::before { content: "->"; }
b { color:red; }
</style>

<title>Quiz JQuery / css</title>
</head>

      <body>

          <p id="p1">Hi there <span id="s1">how</span> are you
          <span id="s2">doing</span>? </p>

          <p id="p2"> This <span id="s3">span</span> is one of
              several <span id="s4">spans</span> in this
              <span id="s5">sentence</span>. </p>

          <div> one </div>
          <div> two </div>
          <div> three</div>

      </body>
</html>
```

(a) Indiquez ce qu'affiche un navigateur (ayant accès à javascript et configuré "normalement") lorsqu'il reçoit cette page.

(20) 6. Le coin client - serveur

Une application côté serveur `game.php` génère des grilles de taille aléatoire dans lesquelles un nombre aléatoire de cases dites mystères sont sélectionnées. Écrire une page HTML `index.html` qui contient un bouton `prêt?` qui lorsqu'on clique dessus:

- fait disparaitre le bouton `prêt?`,
- affiche une grille aléatoire de dimension au moins égale à 5x5 produite par `game.php`,
- les cases mystères de la grille sont visibles (mises en rouge) pendant une période de 10 secondes, après quoi elles sont masquées,
- le joueur a alors 10 coups pour identifier les cases mystères en cliquant sur les cases de la grille. À chaque clic, le décompte du nombre de cases mystère et le nombre de coups restant à jouer est affiché,
- lorsque le joueur a sélectionné 10 cases, le nombre de cases mystère identifiées est envoyé au serveur (application `score.php`) qui retourne une information permettant de savoir s'il s'agit du meilleur score à date. Si tel est le cas, un message est affiché,
- le bouton `prêt?` réapparait afin de permettre au joueur de continuer à joueur.

La totalité de ce jeu doit être accessible depuis l'unique page `index.html`. Cette page ne doit contenir aucun code CSS et aucun code javascript. Vous n'avez pas à vous soucier des entêtes de la page. Le scriptage côté serveur doit être réalisé en PhP, le scriptage client en javascript (avec ou sans JQuery). Puisque toute l'interaction doit se faire au sein de la page `index.html`, les requêtes au serveur doivent être faites via Ajax.

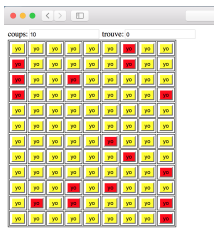
(a) Identifiez toutes les ressources de l'application en expliquant celles qui sont côté client et celles qui sont côté serveur, puis codez l'application.

(b) Commencez par écrire `game.php` en expliquant les entrées de cette application.

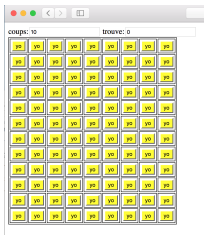
(c) Écrivez ensuite les ressources manquantes.

Voici quelques copies d'écran de ce à quoi peut ressembler une partie.

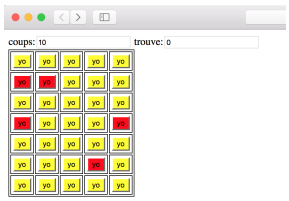
a) Le jeu démarre lorsque le joueur clique sur le bouton `prêt?`.



b) La grille s'affiche alors et les cases mystères sont affichées pendant 10 secondes.



c) Les cases mystères ne sont plus visibles, et le joueur peut commencer à jouer en cliquant sur les cases.



d) Au bout de 10 coups, le bouton `prêt?` s'affiche à nouveau et une nouvelle partie commence lorsqu'il est cliqué.