1 Programmation web

Question 1.1 Le code HTML suivant est-il valide ? Décrivez seulement les erreurs majeurs, c'est à dire celles qui sont communes à toutes les révisions de HTML depuis XHTML 1.0 (i.e. depuis l'introduction du concept de *validation*).

```
<!DOCTYPE html>
<head>
  <meta http-equiv="refresh" content="5" />
</head>
<body>
  <title>Les résultats de la présidentielle</title>
  <table border=1>
    <tr><img src="hollande.txt" alt="François Hollande" />
      <td>François Hollande<td>51,7%</tr>
    <tr><img id="photo-sarko" src="sarkozy.jpeg" />
      <td>Nicolas Sarkozy<td>48,3%</tr>
  </table>
  <p><i>Données <a href="http://www.tns-sofres.com/">
    TNS Sofres</i></a>.</p>
</body>
```

Question 1.2 Réécrivez le fragment de code suivant en déplaçant tous les éléments de présentation en dehors du code HTML, dans une feuille CSS séparée. Si vous ne vous souvenez pas de certains noms ou valeurs de propriétés CSS, inventez-en des similaires. N'hésitez pas à décrire l'affichage voulu dans des commentaires, ou en faisant une esquisse.

```
<body bgcolor="black" align="center">
  <p align="center">
    <table width="80%" bgcolor="white">
      <tr align="center" bgcolor="#eee">
        <td><a href="#1"><div>Link 1</div></a></td>
        <td><a href="#2"><div>Link 2</div></a></td>
        <td><a href="#3"><div>Link 3</div></a></td>
        <td><a href="#4"><div>Link 4</div></a></td>
      </tr>
      <!-- colspan permet à une case d'occuper plusieurs colonnes -->
      <tr><td colspan="4">
        <!-- Note: size 6 correspond approximativement a 30px -->
        <!-- Note:   est un caractère blanc de la meme largeur qu'une
             espace (espace insécable) -->
        <h1 id="1"><font size="6">    Un</font></h1>
        <p><font size="+1"><b>L</b></font>orem ipsum...</p>

        <h1 id="2"><font size="6">    Deux</font></h1>
        <p><font size="+1"><b>D</b></font>olor sit amet...</p>

        <h1 id="3"><font size="6">    Trois</font></h1>
        <p><font size="+1"><b>C</b></font>onsectetor adipiscing elit...</p>

        <h1 id="4"><font size="6">    Quatre</font></h1>
        <p><font size="+1"><b>I</b></font>nteger sagittis...</p>
      </td></tr>
      <tr><td colspan="4">
        <hr><!-- dessine une ligne horizontale -->
        <p align="center">Dernière modification XX-XX-XXXX</p>
      </td></tr>
```

```
    </table>
  </p>
</body>
```

Question 1.3 Un site de paris en ligne utilise le code AJAX suivant pour implanter un *bonneteau* (ou *jeu des trois cartes*, ou *trouvez la dame*).

```
<!-- extrait du code client -->

<p id="msg">Où est la dame de cœurs?</p>
<img id="card-1" src="back.png" onclick="play(1)" />
<img id="card-2" src="back.png" onclick="play(2)" />
<img id="card-3" src="back.png" onclick="play(3)" />

<script>
  function play(choice) {
    $.ajax({
      url     : 'play.php',
      type    : 'get',
      dataType: 'text',
      success : function(data) {
        var card = $('#card-' + choice);
        if (data == 'WIN') {
          card[0].src = 'queen.png';            // change l'image
          $('#msg').html('Vous avez gagné!');
        } else {
          card[0].src = 'jack.png';             // change l'image
          $('#msg').html('Essayez à nouveau.');
        }
      }
    });
  }
</script>
```

```
<!-- extrait de play.php -->

<?php
  if (rand(1, 1000000) == 1) {
    echo 'WIN';
  } else {
    echo 'LOOSE';
  }
?>
```

Si cette technique a l'avantage de rendre le jeu facilement intégrable dans des sites tiers sans gêner la navigation, lorsque l'utilisateur désactive JavaScript rien ne marche plus.

Décrivez comment la technique de la *graceful degradation* (dégradation gracieuse) permet de rendre l'application utilisable même en l'absence de JavaScript, tout en gardant le comportement souhaité lorsque celui-ci est disponible. Servez-vous d'extraits de code pour illustrer vos propos.

2 Analyse de code

Question 2.1 Le code suivant contient trois failles d'injection. Trouvez-les, expliquez comment elle peuvent être exploitées et proposez une correction.

Extrait de `index.php` Cette page vérifie les identifiants d'un utilisateur.

```
<script src="loadstyle.js"></script>

<!-- Ce formulaire permet de choisir la feuille de style par défaut -->
<form method="get" action="choose_style.php">
  <select name="stylesheet">
    <?php
      foreach ($stylesheets as $s) {
        echo "<option value='$s'>$s</option>";
      }
    ?>
  </select>
  <input type="submit" value="ok" />
</form>

<?php
  if($_SESSION['loggedin']) {
    echo '<h1>Bonjour ' . $_SESSION['name'] . '</h1>';
  } else {
    echo '<form method="post" action="login.php">';
    echo '<input type="text" name="login" />';
    echo '<input type="password" name="pwd" />';
    echo '<input type="submit" value="Login" />';
    echo '</form>';
    echo '<a href="new_account.php">Créer un compte</a>';
  }
?>
```

Extrait de `register.php` Cette page permet d'enregistrer un nouvel utilisateur.

```
  $login = mysql_real_escape_string($_POST['login']);
  $pwd   = mysql_real_escape_string($_POST['pwd']);
  $name  = mysql_real_escape_string($_POST['name']);

  $res = mysql_query("SELECT * FROM users WHERE login='$login'") or die();
  if (!$res) {
    mysql_query("INSERT INTO users VALUE ('$login', '$pwd', '$name')") or die();
  }
```

Extrait de `login.php` Cette page vérifie les identifiants d'un utilisateur.

```
  $login = mysql_real_escape_string($_POST['login']);
  $pwd =   mysql_real_escape_string($_POST['pwd']);

  $res = mysql_query("SELECT * FROM users WHERE login='$login' AND pwd='$pwd'")
           or die();
  if ($res) {
    $row = mysql_fetch_assoc($res);
    $_SESSION['login'] = $login;
    $_SESSION['pwd'] = $pwd;
    $_SESSION['name'] = $row['name'];
    $_SESSION['stylesheet'] = $row['stylesheet'];
    $_SESSION['loggedin'] = true;
  }

```

```
15   // redirection vers la page principale
16   header('Location: http://www.example.com/index.php');
```

Extrait de `choose_style.php` Cette page sauvegarde dans un cookie la feuille de style préférée de l'utilisateur

```
1   $s = $_GET['stylesheet'];
2   setcookie('stylesheet', $s);
3   if ($_SESSION['loggedin']) {
4     $login = $_SESSION['login'];
5     $_SESSION['stylesheet'] = $s;
6     mysql_query("UPDATE users SET stylesheet='$s' WHERE login='$login'");
7   }
8
9   // redirection vers la page principale
10  header('Location: http://www.example.com/index.php');
```

Extrait de `loadstyle.js` Ce script charge dynamiquement une feuille de style sauvegardée dans un cookie.

```
1   // on récupère le cookie stylesheet
2   // (le code de getCookie est omis)
3   var stylesheet = getCookie('stylesheet');
4   if (stylesheet) {
5     var link = document.createElement('link');
6     link.rel = 'stylesheet';
7     link.type = 'text/css';
8     link.href = stylesheet;
9     document.head.appendChild(link);
10  }
```

3 Analyse d'application web

Play that game est une maison spécialisée dans la production de jeux multiplayer en JavaScript. Leur dernier succès est une bataille navale pour deux joueurs, comportant aussi une *live chat*.

Les joueurs s'identifient avec leur login, après quoi le serveur leur assigne un adversaire et initialise une partie avec des navires disposés au hasard. Nous passons cette première phase et présentons directement ici la partie du code qui implante la logique du jeu.

Le code repose lourdement sur JavaScript, AJAX et JQuery. Nous rappelons qu'en JQuery la commande `$('#identifiant')` est l'analogue de `document.getElementById('identifiant')`.

Extrait de `battle.html`

```
1   <script src="battle.js"></script>
2
3   <div id="plateau">
4     <!-- Ce bloc contient le plateau de jeu. On omet les détails. -->
5   </div>
6
7   <!-- Ces champs permettent de tirer un coup. -->
8   <h3>Tirer<h3>
9   <p>
10    ligne:   <input type="number" id="row" />
11    colonne: <input type="number" id="column" />
```

```
12    <input id="fire" type="button" value="Tire" onclick="fire()" disabled />
13  </p>
14
15  <!-- Ici la chat. -->
16  <div id="chat-text"></div>
17  <input type="text" id="chat-input" />
18  <input type="button" value="Envoi" onclick="send_message()" />
```

Extrait de `battle.js`

```
1   // Cette variable permet de savoir à qui est le tour de jouer
2   var myturn = false;
3
4   // Cette fonction change de tour et active ou désactive le formulaire de tir
5   function toggle_turn() {
6     if (myturn) {
7       $('#fire').attr('disabled', true);
8     } else {
9       $('#fire').attr('disabled', false);
10    }
11    myturn = !myturn;
12  }
13
14  // Cette variable contient le plateau de jeu Allez voir le fichier plateau.php
15  // pour une description de son contenu.
16  var plateau;
17  // Cette fonction dessine le plateau de jeu
18  function draw_plateau() { /* détails omis */ }
19
20  // Cette fonction est invoquée au démarrage
21  function init() {
22    $.ajax({
23      url      : 'init.php',
24      dataType : 'json',
25      success : function(data) {
26        plateau = data.plateau
27        draw_plateau();                       // on dessine le plateau,
28        myturn = data.myturn;                 // on voit à qui est le tour,
29        if (myturn)                           // si c'est notre tour on
30          $('#fire').attr('disabled', false); // active le formulaire.
31      }
32    });
33  }
34
35  // Cette fonction est exécutée lorsqu'on envoie un coup
36  function fire() {
37    if (myturn) {
38      // on récupère les coordonnées
39      var row = $('#row').val();
40      var column = $('#column').val();
41      // on passe le tour
42      toggle_turn();
43      // on envoie le coup au serveur
44      $.ajax({
45        url      : 'fire.php',
46        data     : 'row=' + row + '&column=' + column,
47        dataType : 'json',
```

```
48        success : function(data) {
49          if (data.ok) {                    // Si le coup est reussi
50            if (data.ok == "hit")
51              alert("Touché!");
52            // on met à jour le plateau
53            plateau = data.plateau;
54            draw_plateau();
55          } else {                          // Si le coup n'est pas valide
56            if (data.error == 'self_shot')
57              alert('Vous ne pouvez pas tirer sur vos navires.');
58            else if (data.error == 'already_hit')
59              alert('Cette coordonnée a déjà été frappée.');
60            else
61              alert('Erreur: mauvaises coordonnées.');
62            toggle_turn();
63          }
64        }
65      });
66    }
67  }
68
69  // Cette fonction interroge périodiquement le serveur pour demander
70  // des mises à jour.
71  function poll() {
72    $.ajax({
73      url      : 'poll.php',
74      dataType : 'json',
75      success : function(data) {
76        if (data.message) {
77          update_chat(data.message);
78        } if (data.fired) {
79          // Si on a reçu un coup, on met à jour le plateau
80          plateau = data.plateau;
81          draw_plateau();
82          // on reprend le tour
83          toggle_turn();
84        }
85      }
86    });
87  }
88
89  window.onload = function() {
90    init();                    // On initialise le plateau
91    setInterval(poll, 500);    // On interroge le serveur 2 fois par seconde
92  }
```

Extrait de `init.php`

```
1  include('plateau.php');
2
3  $pid = $_SESSION['player'];
4  $data = array(
5    'plateau' => get_plateau($pid),
6    'myturn' => myturn($pid));
7  // Cette fonction de librairie transforme un tableau PHP en un tableau JSON
8  echo json_encode($data);
```

Extrait de `fire.php`

```php
include('plateau.php');

$pid = $_SESSION['player'];
// Si c'est bien le tour du joueur
if (myturn($pid)) {
  $plateau = get_plateau($pid);
  $row = intval($_GET['row']);
  $column = intval($_GET['column']);

  // Contrôle que les coordonnées $row et $column définissent un coup valide
  // dans $plateau
  if ($row >= count($plateau)       || $row < 0    ||
      $column >= count($plateau[0]) || $column < 0)
    echo     '{ "error" : true }';
  else {
    $data = array('plateau' => $plateau);
    switch ($plateau[$row][$column]) {
      case 0:
        $data['ok'] = 'miss';
      case 1:
        if (!isset($data['ok']))
          $data['ok'] = 'hit';
        $data['plateau'] = change_plateau($pid, $row, $column);
        break;
      case 2:
        $data['error'] = 'self_shot';
        break;
      case 10: case 11: case 12:
        $data['error'] = 'already_hit';
        break;
    }
    echo json_encode($data);
  }
} else {
  echo '{ "error" : true }';
}
```

Extrait de `plateau.php`

```php
// Cette fonction controle si c'est au joueur en paramètre de jouer,
// ou bien à son adversaire.
function myturn($player_id) { /* détails omis */ }

// Cette fonction se connecte à la base de données et récupère le plateau de
// jeu associé au joueur en paramètre. La valeur de retour est un tableau
// d'entiers à deux dimensions contenant les valeurs suivantes:
//    0 : Case vide
//    1 : Case occupée par un navire de l'adversaire
//    2 : Case occupée par un navire du joueur
//   10 : Case vide et explosée
//   11 : Case occupée par un navire de l'adversaire et explosée
//   12 : Case occupée par un navire du joueur et explosée
function get_plateau($player_id) { /* détails omis */ }

// Cette fonction enrégistre dans la base de données les effets d'un coup
```

```
17 // de feu, passe le tour au joueur suivant, et renvoye le plateau modifié
18 function change_plateau($player_id, $row, $column) { /* détails omis */ }
```

En déhors des cas spécifiés, on suppose que le code qui a été omis ne contient pas de faille de sécurité.

Question 3.1 Les joueurs peuvent-ils tricher ? Comment corriger le code pour les en empêcher ?

Question 3.2 En supposant que le chat contienne un bug qui permet l'injection de code JavaScript dans le div `#chat-text`, que peut faire un utilisateur malicieux pour tricher ?

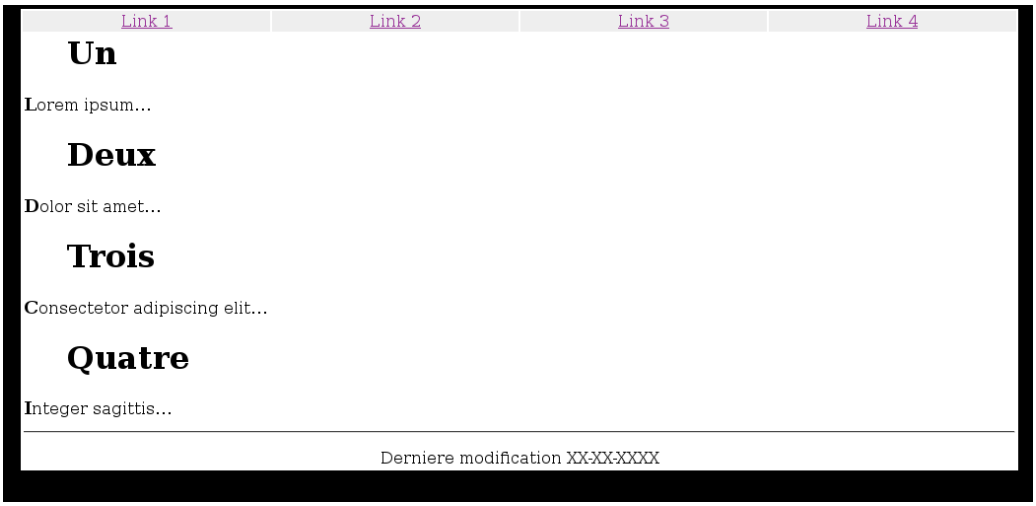
Question 3.3 En supposant que le chat ne contienne pas de faille, mais que le serveur soit configuré avec une entête `Access-Control-Allow-Origin: *`, que peut faire un utilisateur malicieux pour tricher ?

Solution 1.1

1. La balise `<title>` n'est pas autorisée dans le `<body>`. Elle devrait plutôt être placée dans le `<head>`.
2. Les deux balises `<tr>` contiennent autre chose que des balises `<td>` (ou `<th>`).
3. Les balises `<td>` ne sont pas fermées.
4. La deuxième balise `<img>` ne contient pas d'attribut `alt`.
5. Dans le dernier paragraphe, les balises `<a>` et `<i>` sont inversées.

Remarque : Il est vrai qu'il manque une balise `<html>`, cependant cette balise est (de façon surprenante) optionnelle en HTML5 ! Néanmoins, il est toujours recommandé d'utiliser cette balise, ainsi que `<head>` et `<body>` (elles aussi optionnelles).

Solution 1.2 Voici la façon dont la page est visualisée par Firefox 10.



Voici une des manières possibles de convoyer le même contenu en HTML5.

```
<body>
  <div id="content">
    <nav>
    <ul>
      <li><a href="#1">Link 1</a></li>
      <li><a href="#2">Link 2</a></li>
      <li><a href="#3">Link 3</a></li>
      <li><a href="#4">Link 4</a></li>
    </ul>
    </nav>

    <h1 id="1">Un</h1>
    <p>Lorem ipsum...</p>

    <h1 id="2">Deux</h1>
    <p>Dolor sit amet...</p>

    <h1 id="3">Trois</h1>
    <p>Consectetor adipiscing elit...</p>

    <h1 id="4">Quatre</h1>
    <p>Integer sagittis...</p>

    <footer>Dernière modification XX-XX-XXXX</footer>
  </div>
</body>
```

Et voici une feuille de style qui reproduit de près le même affichage.

```css
/* Le cadre principal */
body { background-color: black; }
#content {
  background-color: white;
  width: 80%;
  margin: auto;
}
/* Les trois blocs suivants font en sorte que le <nav>
   se comporte comme un tableau.
   D'autres solutions seraient possibles en utilisant
   display: inline-block ou float: left
*/
nav{
  display: table;
  border-spacing: 2px 0;
  width: 100%;
}
nav ul {
  list-style-type: none;
  display: table-row;
  width: 100%;
}
nav li {
  display: table-cell;
  background-color: #eee;
  text-align: center;
}
/* Ceci permet aux liens d'occuper toute la largeur */
nav li a { display: block; }
/* Le style du corps */
h1 {
  font-size: 30px;
  margin-left: 4ex;
}
p:first-letter {
  font-weight: bold;
  font-size: larger;
}
/* Le bandeau en bas */
footer {
  border-top: solid thin black;
  text-align: center;
}
```

Solution 1.3 La solution présentée ici n'est qu'une des nombreuses possibles. Le premier pas consiste à faire en sorte que les trois images soient cliquables même sans JavaScript. En utilisant un `<input>` de type `submit` et en bidouillant un peu avec CSS on peut obtenir le même résultat de la façon suivante.

```css
<style>
input {
  background : url(back.png) no-repeat;    /* l'image est placée au fond */
  width: 64px;
  height: 80px;                            /* les dimensions de l'image */
}
</style>
```

```
8  <form method="post" action="play.php?nojs">
9    <input id="card-1" name="card-1" type="submit" onclick="return play(1)" />
10   <!-- etc. -->
11 </form>
```

Les attributs `onclick` ne sont pris en compte que si JavaScript est activé. Dans ce cas, il suffit de modifier légèrement le code pour que la présence de JavaScript annule l'action par défaut qui serait la soumission du formulaire : pour cela, il faut renvoyer `false`.

```
1  function play(choice) {
2    $.ajax({
3      // Même code qu'avant
4    });
5    return false;
6  }
```

Il ne reste plus qu'à adapter le code PHP pour réagir de façon différente lorsque JavaScript est désactivé. Le paramètre GET `nojs` passé par le formulaire sert exactement à cela.

```
1  if (isset($_GET['nojs'])) {
2    // Dans ce cas on affiche une page HTML:
3    // On découvre la carte cliquée par le joueur
4    if (isset($_GET['card-1']) { /* première carte */ }
5    // etc.
6  } else {
7    // Dans ce cas, on renvoie du JSON.
8    // Même code qu'avant
9  }
```

Solution 2.1

1. `index.php`, ligne 17, la variable de session `name` provient d'une donnée POST non filtrée à la ligne 3 de `register.php`. Il s'agit d'une injection dans le code HTML affiché, donc d'un potentiel danger de XSS. La fonction de librairie PHP `htmlspecialchars` permet de réparer la faille.
2. `loadstyle.js`, ligne 6, le contenu du cookie provient d'une donnée GET non filtrée à la ligne 1 de `choose_style.php`. Cela permet à un attaquant de faire ouvrir une feuille de style arbitraire à la victime, avec un danger potentiel de *mashup attack* ; ou plus simplement de forcer l'utilisateur à générer une requête GET vers une adresse arbitraire (possibilité d'escalade CSRF). La meilleure méthode pour réparer cette faille consiste à avoir une liste des feuilles de style possibles, et à comparer le chemin fourni par le client avec cette liste.
3. `choose_style.php`, ligne 6, la variable `$s` n'est pas filtrée contre les injections SQL. La fonction de librairie PHP `mysql_real_escape_string` permet de réparer la faille.

Solution 3.1 De façon évidente, la page `init.php` envoye le plateau complet aux joueurs, sans cacher la position des navires adversaires : une simple lecture du code HTML reçu permet de connaître leur position.

La contre-mesure est simple : changer toutes les valeurs 1 en 0 dans le tableau `$plateau` avant de l'envoyer au client.

Solution 3.2 Si l'attaquant peut injecter du code JavaScript arbitraire dans la chat, celui-ci s'executera dans le même contexte que le code de la bataille navale. Il suffira alors de lire le contenu de la variable `plateau` déclarée à la ligne 16 de `battle.js`, de coder son contenu dans une URL sous le contrôle de l'attaquant, et de générer une requête GET vers cette URL (par exemple à travers une balise `<img>`) ; voire même tout simplement d'envoyer le contenu de `plateau` à l'attaquant à travers la chat (mais, cela pourrait mettre en alerte la victime, si celle-ci voit l'historique de la chat) !

Cette attaque est impossible à contrer : le seul moyen de s'en protéger consiste à éliminer la faille XSS.

Solution 3.3 L'attaquant peut mettre en place une CSRF :

1. Il attire la victime (par exemple en lui donnant un lien dans la chat) sur un site qu'il contrôle.
2. Le site de l'attaquant génère une requête AJAX vers `fire.php` avec des coordonnées quelconques (par exemple, des coordonnées qui ont été déjà touchées, pour éviter d'alerter la victime) ;
3. `fire.php` renvoie du code JSON contenant le plateau de jeu **tel qu'il est vu par la victime** ; ce code est lisible par le script de l'attaquant grâce à l'entête `Access-Control-Allow-Origin`.
4. Il ne reste plus qu'à coder ces données dans une URL contrôlée par l'attaquant et à lancer une requête vers cette URL.

Les contre-mesures pour cette attaque sont les mêmes que pour toute attaque de type CSRF : l'entête `Referrer`, les *nonces*, les formulaires de confirmation, etc.