# TP Noté de Web (2016)

*Correction réalisée par Kévin Cocchi. Énoncé disponible ici.*

## — HTML —

### Question 1

La partie `scolarite/notes` désigne un **répertoire**.

### Question 2

L'avantage d'une page web dynamique par rapport à une page web statique est que **les données peuvent être modifiées** en conservant une structure de page similaire.

### Question 3

L'encodage utilisé pour afficher la page web est celui défini dans **l'attribut** `charset` de la balise `meta`.
Il est donc possible que les caractères soient mal affichés si l'encodage du fichier ne coincide pas avec celui défini dans la balise `meta`.

### Question 4

La balise `<b>` (*obsolète*) n'a pour objectif que de créeer une **mise en forme purement visuelle** (le plus souvent, du gras).
La balise `<strong>` a pour objectif de **mettre en valeur** une portion du texte.

### Question 5

La balise `<span>` est utilisée pour **encadrer** des mots, des phrases ...
*On peut ensuite appliquer un style `CSS` pour mettre en valeur cet élément.*

### Question 6

Le code pour définir un **lien hypertexte** vers la page `plandusite.html` est `<a href="plandusite.html">lien</a>`.

### Question 7

L'attribut `alt` de la balise `<img>` permet d'afficher un **texte alternatif** si l'image ne peut pas être affichée.

## Question 8

Les balises `media` de `HTML5` apportent un **support natif** pour l'affichage de contenu *vidéo* ou *audio*.

## Question 9

Produire un code `HTML` valide garantit que la structure de la page sera la **même sur chaque navigateur**.

*Avec des balises mal fermées, le comportement peut varier d'un navigateur à l'autre.*

## Question 10

Pour afficher un **tableau** en `HTML` :

```
<table>
    <tr><th>Nom</th><th>Note</th></tr>
    <tr><td>Durand</td><td>14</td></tr>
    <tr><td>Dupond</td><td>15</td></tr>
</table>
```

## Question 11

Pour afficher un **champ de choix** en `HTML` :

```
<legend>Choisissez l'UE :</legend>
<input type="radio" name="choixUE" value="IA"><br>
<input type="radio" name="choixUE" value="ILO"><br>
<input type="radio" name="choixUE" value="RIIA">
```

## Question 12

Pour afficher un **bouton de validation** en `HTML` :

```
<input type="submit" value="Saisir les notes">
```

## Question 13

Pour définir un **formulaire** en `HTML` envoyant les entrées vers la page `affichenotes.php` par la méthode `POST` :

```
<form action="affichenotes.php" method="post">
    ...
</form>
```

## Question 14

`h2 { color: red; }` **colore** le titre de second niveau en rouge.

## Question 15

Il suffit de changer le premier élément de liste en **appliquant la classe** : `<li class="valide">valide</li>`.

## Question 16

La **première ligne de chaque paragraphe** est mise en `small-caps`.

# — PHP —

## Question 17

Le code `PHP` est executé sur le **serveur web**.

## Question 18

Le code suivant affiche : *Notes déposées le 05/03/2016.*

## Question 19

Le code suivant affiche : *Notes déposées par Martin.*

## Question 20

Le code suivant affiche : *Notes déposées par $prof.*

## Question 21

Le code suivant affiche : *Jacques Blanc*.

## Question 22

Le code qui permet d'afficher la note de *Jacques Blanc* est : `<?php echo $notes["Jacques Blanc"]; ?>`.

## Question 23

Le code suivant affiche : *14*.

*En `PHP`, les arguments sont passés par valeur aux fonctions.*

## Question 24

L'inclusion permet d'**inclure du code répétitif** et n'avoir à l'écrire qu'une seule fois.

## Question 25

`<?php echo $_POST["choixUE"]; ?>` permet d'afficher la valeur de `choixUE` passé par la **méthode `POST`**.

## Question 26

```php
<?php
$requete = "SELECT id_ue, intitule FROM ue WHERE responsable='Toto'";
$db = pg_connect("host=pgsql2 dbname=ensiie");
$query = pg_query($db, $requete);
$tuple = pg_fetch_assoc($query);
while($tuple) {
    echo $tuple["id_ue"].' - '.$tuple["intitule"];
    $tuple = pg_fetch_assoc($query);
}
?>
```

## Question 27

`pg_fetch_assoc` renvoie un `array`.

## Question 28

On utilise un champ caché lorsque l'on veut passer un paramètre **d'une page à l'autre**.

*En pratique, l'utilisateur peut modifier ce champ, alors qu'il ne devrait pas … Don't trust user input.*

## Question 29

Les **sessions** permettent de stocker les identifiants lors de la navigation (*via les variables* `$_SESSION`).

## Question 30

L'extension *PDO* de `PHP` permet de ne plus se soucier des **fonctions spécifiques à un SGBD**.

## Question 31

Selon le patron *MVC*, le code permettant l'accès aux données de la base se situe dans le **modèle**.

## Question 32

Le code *AJAX* permet de réaliser des **requêtes** `PHP` **sans changer de page**.

## Question 33

Une injection `HTML` correspond à l'ajout de code `HTML` à la page après une saisie de l'utilisateur non sécurisée.

*Il est alors aussi possible d'injecter du code JavaScript malveillant … Penser à* `stripslashes`.

# — XML —

## Question 34

La balise *"ue"* n'est pas **correctement fermée** : `<ue id=PW2>...</ue>`.

## Question 35

De nouveau, la balise *"responsable"* **n'est pas fermée** : `<responsable id="toto" />`.

## Question 36

Les balises ne sont pas correctement **imbriquées** :

```
<enseignant>
    <nom>Toto</nom>
```

```
    <prenom>Pierre</prenom>
</enseignant>
```

### Question 37

La **casse des balises** est différente : `<ue><intitule>Programmation web</intitule></ue>`.

*En `XML`, la casse n'importe pas, du moment qu'elle reste consistante pour les deux balises.*

## — DTD —

### Question 38

```
<!ELEMENT brochure (enseignements, enseignants)>
<!ELEMENT enseignements ANY>
<!ELEMENT enseignants ANY>
```

### Question 39

```
<!ELEMENT enseignants (enseignant)*>
<!ELEMENT enseignant ANY>
```

### Question 40

```
<!ELEMENT ue ANY>
<!ATTLIST ue id ID #REQUIRED
             lang CDATA #IMPLIED>
```

## — Schémas XML —

### Question 41

Les schémas `XML` ont un typage plus fort que `DTD`, et permettent la **description des contraintes sur le contenu** du document.

### Question 42

```
<enseignants>
    <enseignant>
```

```
        <nom>Cliché</nom>
        <prenom>Pierre</prenom>
        <bureau>42</bureau>
        <statut>permanent</statut>
    </enseignant>
    ...
<enseignants>
```

`statut` *ne peut valoir que contractuel ou permanent.*

### Question 43

```
<semestre>
    <ue id="FH" lang="fr">
        <module>LV1</module>
        <module>LV2</module>
        <module>Communication</module>
    </ue>
    <ue id="ECO" lang="fr">
        <module>Micro-économie</module>
        <module>Analyse financière</module>
        <module>Initiation à l'entrepreneuriat</module>
    </ue>
    <ue id="ILO" lang="fr">
        <module>ILO</module>
    </ue>
</semestre>
```

## — Accessibilité —

### Question 44

Le second principe de *WCAG* est celui d'être **utilisable** : par exemple, laisser le temps à l'utilisateur de lire et utiliser le contenu.

### Question 45

Le **texte alternatif** pour l'image n'est pas explicite : il faudrait par exemple indiquer qu'il s'agit du logo de la RATP, ou laisser cet attribut vide.

## Question 46

Il se peut qu'en aggrandissant les caractères, la page soit entièrement **déformée**.