G. Hunault

Licence DSCS

Angers, décembre 2018

Développement *Web*

Partie 1 : du PHP conceptuel

Traduire le fragment XHTML strict suivant en PHP conceptuel à l'aide des fonctions de std.php. On supposera que les instructions include("std.php") ; et debutPage(...) ; ont été exécutées.

```
<h2>Qui a trouvé le secret de Poppy ?</h2>

<form action='evalue.php' method='get' onsubmit="return verifie()">
   <input type='text' id='qui' name='qui' value='' size='50'  />
       
   <input type='submit'  value='envoi' />
</form>

<div class='invisible' id='non'>
   Il faut quatre mots dans la réponse :
   un sujet, un adjectif, un nom et un autre adjectif, par exemple :
   par exemple : le jeune garçon russe
</div>
```

On ne cherchera pas à reproduire exactement le formatage (espaces, retours à la ligne, indentation...).

Partie 2 : vérification via Javascript

Que faut-il ajouter dans la partie <head> de la page Web de la question 1 si on veut que la définition de la fonction Javascript nommée *verifie* soit lue dans le fichier poppy-verif.js ?

Donner le texte complet de la fonction Javascript nommée *verifie* pour qu'elle accomplisse les actions suivantes :

- la fonction calcule le nombre de mots du contenu de l'élément associé à l'identifiant qui du formulaire et nomme nbm ce nombre ;

- si nbm vaut 4, la fonction renvoie true ;

- si nbm ne vaut pas 4, la fonction fait apparaitre l'élément qui était précédemment invisible et la fonction renvoie ensuite false.

Pour effectuer le calcul de nbm, on pourra supposer l'existence d'une fonction nbmots(chaine) ou l'existence d'une fonction split(separateur,chaine) similaire à la fonction preg_split de **PHP**.

Partie 3 : une fonction en PHP

On décide maintenant de construire une fonction PHP nommée presence telle que `presence($mot,$phrase)` renvoie 0 si le mot n'est pas trouvé dans la phrase et 1 si le mot est trouvé. Ainsi `presence("petit","le petit chat")` renvoie 1 alors que `presence("petit","la grosse souris")` renvoie 0.

Donner le code PHP complet de cette fonction presence.

On prévoira les cas où soit le mot, soit la phrase est vide. Pour ces deux cas, on doit envoyer 0.

Partie 4 : évaluation en PHP

On s'attend à ce que la réponse au formulaire de la question 1 comporte 4 mots, comme par exemple `le petit enfant blond` ou bien encore comme `la jeune fille anglaise`.

On associe à ces 4 mots un tableau **$points** de 4 valeurs numériques ou "points" censés récompenser des bonnes réponses. Par exemple le tableau avec les valeurs `1 2 4 2` indique que l'on accorde un point si le premier mot (l'article) est trouvé alors que le mot numéro 3 rapporte 4 points.

Question 4.1

Donner le code PHP qui construit ce tableau exemple.

Question 4.2

On suppose maintenant que la variable **$reponse** contient la réponse de l'utilisateur ou de l'utilisatrice au formulaire Web de la question 1, que le tableau **$points** contient autant de valeurs que la réponse contient de mots et que la variable **$solution** est la phrase qu'il fallait répondre.

Donner le code PHP conceptuel qui :

- passe en revue chacun des mots de la réponse et l'évalue à l'aide de la fonction `presence()` pour lui associer soit 0 soit le nombre de points prévu ;

- affiche le détail de l'évaluation de chaque mot dans un élément invisible ;

- affiche le nombre de points obtenu et le nombre de points maximum pour la phrase.

Voici un exemple de code HTML possible, on en reproduira l'esprit mais pas forcément le formatage :

```
<div class="invisible">
Réponse utilisateur : le petit chat noir
La bonne phrase     : le petit chat blanc
Points des mots     : 1  2     4    3

Analyse de la réponse :
  mot 1 : le    évaluation : 1 cumul 1
  mot 2 : petit évaluation : 2 cumul 3
  mot 3 : chat  évaluation : 4 cumul 7
  mot 4 : noir  évaluation : 0 cumul 7
</div>

<p>
Score de votre réponse : 7 points sur 10.
</p>
```

Question 4.4

Serait-ce beaucoup plus compliqué d'évaluer la réponse si le code de la solution comportait des expressions régulières, par exemple si la solution était codée **$solution ="le pett ?it ? [cs]hat ? bl[ae]nc ?" ;** au lieu de la simple chaine **$solution ="le petit chat blanc" ;** ?

Question 4.3

Donner la partie de code PHP correspondant au fichier evalue.php qui vient mettre dans la variable **$reponse** la réponse au formulaire Web de la question 1. On s'assurera que, quelle que soit cette réponse, la variable **$reponse** existe et que sa valeur par défaut est la chaine vide.

Partie 5 : expressions régulières

Paraphraser, c'est-à-dire expliquer avec des phrases et des exemples ce que détectent les deux expressions régulières suivantes :

Expression 1 : **gr[aio]s**

Expression 2 : **[AEae]ngers?**

Donner enfin une expression régulière qui permet de détecter si une chaine de caractères correspond à la désignation des salles à la faculté des sciences d'Angers. Pour cela, il faut exactement une lettre de A à L (pour le numéro de bâtiment), puis un chiffre (pour l'étage) : 0, 1, 2 ou 3, puis deux chiffres pour le numéro de salle sachant qu'il n'y a pas plus de 45 salles par étage. Exemples valides : `G102, L215`. Exemples invalides : `u12, W599, T2018`.

Partie 6 : culture Web

Répondre en au moins 5 lignes et 3 phrases au minimum à l'aide d'au moins 3 mots de 3 syllabes ou plus à la question suivante afin de « transmettre un contenu rédactionnel fort » :

> *"Faut-il bien savoir lire en classe de C.P. pour réaliser des exercices sur ordinateur ?"*

CORRIGÉ RAPIDE

Partie 1 : du PHP conceptuel

```
<?php
include("std.php") ;
debutPage() ;

h2("Qui a trouvé le secret Poppy ?") ;

form("evalue.php","get","onsubmit='return verifie()'") ;
 input_text("qui","")  ; nbsp(4) ;
 input_submit("envoi") ;
finform() ;

div("invisible","non");
  p() ;
  echo " Il faut quatre mots dans la réponse : " ;
  echo " un sujet, un adjectif, un nom et un autre adjectif." ;
  finp() ;
findiv() ;

finpage() ;
?>
```

Partie 2 : vérification via Javascript

Il faut ajouter `<script src="poppy-verif.js"></script>` dans le `<head>`.
Contenu possible :

```
function verifie() {

   quitxt = window.document.getElementById("qui").value
   nbm    = quitxt.split(" ").length
   if (nbm==4) { return true }
   window.document.getElementById("non").setAttribute("class","visible")
   return false

} // fin de fonction verifie
```

Partie 3 : une fonction en PHP

```
function presence($mot,$phrase) {

  # renvoie 1 si le mot est vu dans la phrase
  # renvoie 0 sinon, ou si mot ou phrase est vide

  if ($mot=="")    { return 0 ; } ;
  if ($phrase=="") { return 0 ; } ;
  $tdm = preg_split("/\s+/",trim($phrase))  ;
  $nbm = count($tdm) ;
  $vu  = 0 ;

  for ($idm=0;$idm<$nbm;$idm++) {
    if ($tdm[$idm]==$mot) { $vu = 1 ; } ;
  } # fin pour idm

  return($vu) ;

} # fin de fonction presence
```

Partie 4 : évaluation en PHP

```
<?php
include("std.php") ;
debutPage("poppy réponse","std") ;

h1("Le secret de poppy, gestion de la réponse") ;

# exemple : $reponse  = "le petit chat noir"  ;

$reponse = "" ;

if (isset($_GET["qui"])) { $reponse = $_GET["qui"] ; } ;
if ($reponse=="") {
  echo "Aucune réponse fournie. STOP.\n " ;
  exit(-1) ;
} ; # fin si
```

```
$points   = array(1,2,4,3)        ;
$nbt      = array_sum($points)    ;
$solution = "le petit chat blanc" ;

$tmr = preg_split("/\s+/",trim($reponse))  ;
$nbm = count($tmr) ;
$nbp = 0 ;

div("invisible")  ;
echo "Réponse utilisateur : $reponse  \n";
echo "La bonne phrase     : $solution \n";
echo "Points des mots     :" ;
for ($idm=0;$idm<$nbm;$idm++) {
  echo " $points[$idm] " ;
} # fin pour idm
echo "\n\n" ;

for ($idm=0;$idm<$nbm;$idm++) {
  $motc = $tmr[$idm] ;
  if (presence($motc,$solution)==1) {
      $pointc = $points[$idm] ;
  } else {
      $pointc = 0 ;
  } ; # fin si
  $nbp += $pointc ;
  echo "  mot ".(1+$idm)." ".sprintf("%-7s",$motc) ;
  echo " évaluation : $pointc cumul $nbp \n" ;
} # fin pour idm
findiv() ;


p() ;
echo "Score de votre réponse : $nbp points sur ".$nbt."." ;
finp() ;
finPage() ;
?>
```

Utiliser des expressions régulières pour les mots à tester ne serait pas plus compliqué parce que PHP dispose de fonctions pour gérer les expressions régulières.

Partie 5 : expressions régulières

L'expression 1 correspond aux adjectifs gras, gris, gros.

L'expression 2 correspond aux chaines Angers, Engers, angers, engers avec ou sans le s en fin de mot.

L'expression régulière pour les salles est `^[A-L][0-3]([0-3][1-9]|4[1-5])$` si on admet que les numéros de salle commencent à 01, c'est-à dire qu'il n'existe pas de salle numéro 00.