COURS DE PROGRAMMATION WEB DYNAMIQUE

PARTIE 1 :

Bases de la programmation en PHP

PARTIE 1 :

Les fonctions

LES FONCTIONS

- Une fonction est un bloc d'instructions qui porte un nom et qui calcule une valeur et la retourne (généralement) au programme appelant.

- Deux sortes de fonctions en php:
  - les fonctions intégrées à php : plus de 1000
  - les fonctions dites utilisateurs (construites par lui) :

  Comme tout langage de programmation, php permet l'écriture de fonctions.

LES FONCTIONS

- **Définition et appel d'une fonction**

```
function nom_fonction($arg1, $arg2, …, $argn)
{
        instructions ;
        return ; // non obligatoire
}
```

LES FONCTIONS

- Les identificateurs de fonctions sont insensibles à la casse.
- Même sans paramètre, un entête de fonction doit porter des parenthèses **( )**.
- Les différents arguments sont séparés par une virgule **,** et le corps de la fonction est délimité par des accolades **{ }**.
- Les fonctions peuvent prendre des arguments dont il n'est pas besoin de spécifier le type. Elles peuvent de façon optionnelle retourner une valeur.
- L'appel à une fonction peut ne pas respecter son prototypage (nombre de paramètres).

- Une fonction peut être définie après son appel.

LES FONCTIONS

- exemple : fonction sans return

```
<?php
        function increment($x) {
                $x += 1;
                echo " Valeur dans la fonction = $x<br> " ;
        }
        $a=5;
        echo 'Valeur initiale = '. $a . '<br>';
        increment($a);

?>
```

LES FONCTIONS

- ***exemple : fonction sans return***

  fonction affichant en rouge et gras le texte passé en paramètre:

```
<?php
        AfficheRougeGras("cette ligne est en rouge gras <br>");
        print("cette ligne est en police normale<br>\n");
        function AfficheRougeGras($texte) {
                $texteEnGras="<B>";
                $texteEnGras.="<font color=red>";
                $texteEnGras.=$texte;
                $texteEnGras.="</font></b>";
                print($texteEnGras);
        }
?>
```

LES FONCTIONS

- ***exemple : fonction avec return***

```
<?php
    function produit($a,$b) {
            return ($a*$b);
    }
    print (produit(3,7));
?>
```

LES FONCTIONS

- ***exemple : fonction en paramètre d'une autre fonction***

```
<?php
        function produit($a,$b) {
                return ($a*$b);
        }
        print (produit(3,7))."<br>";
        AfficheRougeGras(produit(3,7));
        function AfficheRougeGras($texte) {
                $texteEnGras="<b> <font color=red>";
                $texteEnGras.=$texte;
                $texteEnGras.="</font></b>";
                print($texteEnGras);
        }
?>
```

LES FONCTIONS

- **Portée des variables**

  - La portée d'une variable dépend de son emplacement.

  - Les variables ont une portée globale si elles sont définies en dehors d'une fonction, sinon, elles sont locales.

  - A noter que l'accès à une variable globale à l'intérieur d'un bloc n'est pas automatique.

  - La variable est globale mais il faut explicitement autoriser son accès dans le bloc.

LES FONCTIONS

- **Portée des variables**

```
<?php
        $a=$b=1;  // ici variables à portée globale
        echo somme();
        Function somme(){
                $result=$a+$b;  //ici variable a portee locale
                return $result;  //ce ne sont pas les memes que ci-haut
        }
?>
```

Notice: Undefined variable: b in **C:\wamp\www\...\ PorteeDeVariables1.php** on line **5**
Notice: Undefined variable: a in **C:\wamp\www\...\ PorteeDeVariables1.php** on line **5**
0

LES FONCTIONS

- **Portée des variables**

```
<?php
        $a=$b=1;  // ici variables à portée globale
        echo somme($a,$b);
        Function somme($a,$b){
                $result=$a+$b;  //ici variable a portee locale
                return $result;  //ce ne sont pas les memes que ci-haut
        }
?>
```

```
2
```

LES FONCTIONS

- **Portée des variables**

➢ **On peut modifier la portée des variables locales à une fonction.**

global: permet de travailler sur une variable de portée globale au programme.

exemple :

```
function change() {
        global $var; // définit $var comme globale
        $var++; // cela sera répercuté dans le reste du programme
}
```

LES FONCTIONS

- **Portée des variables**

```
<?php
        $a=$b=1;
        echo somme();
      function somme() {
        global $a,$b;
        $result=$a+$b;
        return $result ;
      }
?>
```

```
2
```

LES FONCTIONS

- **Portée des variables**

Utilisation d'une variable static

Une variable déclarée "static" garde la dernière valeur entre deux appels de fonction ==> elle reprend sa dernière valeur à chaque nouvel accès à la fonction ou au bloc.

Syntaxe:

```
static $variable;  // à l'intérieur de la fonction
```

exemple :

```
function change() {
        static $var=0;  // définit $var comme statique
        $var++;   // sa valeur sera conservée jusqu'au prochain appel
}
```

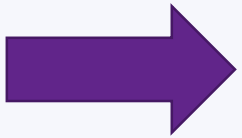
LES FONCTIONS

- **Portée des variables – variable static**

```
<?php
function somme(){
global $a,$b;
$resultat=0;
$resultat=$resultat+$a+$b;
return $resultat;
}
$a=$b=1;
echo '-1er appel: '.somme().'<br>';
echo '-2ème appel: '.somme();
?>
```

```
static $resultat=0;
```

- 1er appel: 2
- 2ème appel: 2



Avec static

- 1er appel: 2
- 2ème appel: 4

LES FONCTIONS

- **Portée des variables – variable static**

exemple:

```
<?php
        function compte () {
                static $compteur = 0 ;
                $compteur++ ;
                echo "$compteur - " ;
                if ($compteur<10) compte(); // appel récursif de compte()
        }
        compte() ;
?>
```

Affiche: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 –

LES FONCTIONS

- **Portée des variables – variable static**

On peut aussi utiliser le tableau associatif $GLOBALS pour accéder à une variable globale au sein d'un bloc.

Il aura comme clé le nom de la variable globale.

```
<?php
        $var1 = 1 ;
        $var2 = 2 ;
        $var3 = 3 ;
        function affiche(){
                global $var1;
                echo '--> $var1='.$var1.'<br>';
                echo '--> $var2='.$GLOBALS[var2].'<br>';
                echo '--> $var3='.$var3;
        }
        affiche();
?>
```

Affiche :

--> $var1=1

--> $var2=2

--> $var3=

Les deux variables $v1 et $v2 sont globales.
$v1 est déclarée globale, on y accède sans problèmes.
On accède à $v2 par l'intermédiaire du hash $GLOBALS.

LES FONCTIONS

Passage de paramètres par référence

Pour passer une variable par référence, on fait précédé son nom dans la définition de la fonction par le symbole **&**

***Le passage de paramètres est alors permanent*.**

LES FONCTIONS

Arguments d'une fonction

On peut donner une valeur par défaut aux arguments lors de la déclaration d'une fonction. Ainsi, si un argument est «oublié» lors de l'appel de la fonction, cette dernière lui donnera automatiquement une valeur par défaut décidée à l'avance par le programmeur.

exemple :

```
function Set_Color($color="black") {
        global $car;
        $car["color"] = $color;
}
```

LES FONCTIONS

Les fonctions de date et heure: date()

la fonction date() permet d'afficher la date du jour ainsi que l'heure courante. Elle affiche en fait, la date du serveur.

```
$date-jour = date("d-m-y") ;
$heure-courante = date ("H:i") ;
// d, m, y, H et i peuvent être placées dans n'importe quel ordre
```

Les valeurs retournées par la fonction date() sont affectées aux variables $date-jour et $heure courante

LES FONCTIONS

Les fonctions de date et heure: date()

exemple :

```
<?php
        $date_jour = date ("d-m-Y");
        $heure = date ("H:i");
        echo 'Nous sommes le : ‘;
        echo $date_jour;
        echo ' et il est : ‘;
        echo $heure;
?>
```

LES FONCTIONS

Codes à utiliser avec la fonction date() :

format	description	exemple
d - j	jour du mois: sur deux chiffres - sans 0 en entête	05 - 5
D - l	jour de la semaine en anglais: 3 lettres - textuel	mon - Mon
F	nom du mois, textuel, en anglais	February
m	mois de l'année en chiffres (0 en entête)	04
M	mois de l'année en 3 lettres	Feb

LES FONCTIONS

Codes à utiliser avec la fonction date() :

format	description	exemple
n	mois de l'année en chiffres; sans le 0 en entête	4
y - Y	année à 2 chiffres - année en 4 chiffres	09 - 2009
h - g	heure (format 12 heures): avec 0 en entête et sans	02 - 2
H - G	heure (format 24 heures): avec 0 en entête et sans	07 - 7
i	minutes	44

LES FONCTIONS

Les fonctions de date et heure: time() et getdate()

la fonction ***time()*** permet d'afficher la date au format système.

La fonction ***getdate()*** permet de ressortir les infos de ***time()***

exemple :

```
<?php
        $time = time(); // date au format système
        print 'la date système du jour est '.$time.'<br>';
        $date = getdate($time); // passage de la variable time dans getdate
        print 'Nous sommes le '.$date['mday'].' - '.$date['mon'].' –
        '.$date['year']. ' il est '.$date['hours'].':'.$date['minutes'];
?>
```

LES FONCTIONS

Clés du tableau associatif retourné par getdate() :

Clés	description	exemple
seconds	secondes	22
minutes	minutes	13
hours	heures de la journée de 0 à 23	12
mday	jour du mois de 1 à 31	23
wday	jour de la semaine de 0 à 6	4

LES FONCTIONS

Clés du tableau associatif retourné par getdate() :

Clés	description	exemple
yday	jour de l'année de 0 à 365	53
weekday	nom du jour de la semaine (anglais)	Thursday
mon	mois de l'année	2
month	mois de l'année (en anglais)	February
year	année en 4 chiffres	2017

LES FONCTIONS

La fonction unclude

- La fonction ***include()*** de PHP permet d'évaluer et d'insérer, à chaque appel, le contenu d'un fichier dans un autre (lors de l'exécution de ce dernier).
- Un élément HTML répétitif qui apparaît dans toutes les pages de votre site (un menu par exemple) pourra être isolé dans un fichier PHP. Un appel de ce fichier grâce à la fonction ***include()*** apparaîtra dans toutes les pages de votre site. Ainsi si le menu doit être par exemple modifié il suffira uniquement de changer le fichier contenant le menu.

  syntaxe :

  <?php include("nom_fichier.php"); ?>

  Insère et exécute un fichier externe dans la page d'appel.

LES FONCTIONS

La fonction unclude

exemple :

1ère page :

```
<html> <body>
        <font size="2" face="Arial">Texte en HTML</font>
                <?php
                        include("page2.php"); // on inclue le fichier
                ?>
</body> </html>
```

2ème page : (page2.php)

```
<?php
        $heure = date("H\hi"); // \pour afficher h
        print("<center><font size=\"2\" face=\"Arial\"> et texte en PHP.
                Il est $heure.</font></center>");
?>
```

Affiche :

Texte en HTML

et texte en PHP. Il est 10h43.

LES FONCTIONS

La fonction unclude

exemple :

page1.php:

```
<?php
        $couleur = 'jaune';
        $fruit = 'banane';
?>
```

pageprincipale.php

```
<?php
        echo "Une $fruit $couleur ";  // Une (avant include)
        include 'page1.php';
        echo "<br>Une $fruit $couleur"; //Une banana jaune
?>
```

Notice: Undefined variable: couleur in **C:\wamp\www\Cours ENS\fonctions\include\pageprincipale.php** on line **3**
Notice: Undefined variable: fruit in **C:\wamp\www\Cours ENS\fonctions\include\pageprincipale.php** on line **3**

Une
Une banane jaune

LES FONCTIONS

La fonction require

require permet d'inclure dans le code le contenu d'un fichier et de l'exécuter.

exemple :

```
<?php
        require 'fichier.php';
        require $fichier;
        require ('fichier.txt');
?>
```

require et **include** se différencient dans leur gestion des erreurs:

include produit un **Alert (warning)** tandis que **require** génère une **erreur fatale**.

Autrement, on utilise **require** si on veut que le script s'interrompe dans le cas où un fichier d'inclusion manque, et **include** si on veut que le script continue son exécution.