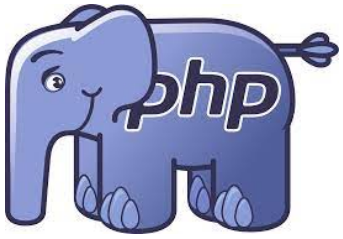


Systèmes & Technologies de l'Informatique

Création d'un site Web dynamique

Le langage PHP

4ème Sciences de l'informatique

Enseignant : Chakib Ben Jlijel

Année scolaire: 2024-2025

I- Introduction

I. Introduction

1. Principe de fonctionnement

Un site Web dynamique est un site dont les pages pouvant être générées dynamiquement en suite à une demande d'un utilisateur.

I. Introduction

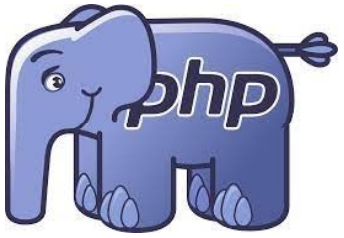
2. Environnement de Travail

Pour pouvoir développer en PHP, on a besoin :

- ❖ Un serveur Web (Apache);
- ❖ Un interpréteur PHP;
- ❖ Un SGBD (MySql), si le site web contient une base de données.

Dans la suite du cours, on utilisera le serveur « **XAMPP** » qui est un ensemble de logiciels permettant d'installer un serveur Web local **Apache**, **PHP**, **MySQL** et **phpMyAdmin** qui est une interface Web pour la gestion des bases de données.

II- Le langage PHP



II. Le langage PHP

1. Introduction

PHP (Hypertext Preprocessor) est un langage de programmation de scripts côté serveur permettant de produire des pages web dynamiques.

Qu'est-ce qu'un fichier PHP ?

- ✓Les fichiers PHP peuvent contenir du texte, du code HTML, CSS, JavaScript et PHP;
- ✓Le code PHP est exécuté sur le serveur et le résultat est renvoyé au navigateur sous forme de HTML brut;
- ✓Les fichiers PHP ont l'extension **".php"**

II. Le langage PHP

Syntaxe PHP

- ✓Un script PHP peut être placé n'importe où dans le document.
- ✓Un script PHP commence par **<?php** et se termine par **?>** :

```
<?php
// Le code PHP se met ici
?>
```

- ✓L'extension par défaut pour les fichiers PHP est "**.php**".
- ✓Les instructions PHP se terminent obligatoirement par **un point-virgule** (;).

II. Le langage PHP

2. Fonction echo

La fonction **echo** écrit(affiche) une ou plusieurs chaînes dans la sortie.

Syntaxe :

echo(*strings***)**

```php
<?php
echo "Bonjour le monde!";
?>
```

```php
<?php
$str = "Bonjour le monde!";
echo $str;
?>
```

```php
<?php
//Concaténer deux variables de type chaîne ensemble :
$str1="Bonjour le monde!";
$str2="Quelle belle journée!";
echo $str1 . " " . $str2;
?>
```

II. Le langage PHP

3. Les commentaires en PHP

Il est possible d'ajouter des commentaires en PHP pour décrire le fonctionnement du code.

PHP prend en charge plusieurs façons de commenter :

```
// Ceci est un commentaire sur une seule ligne

/* C'est un
commentaire sur plusieurs lignes */
```

II. Le langage PHP

4. Les constantes

Une constante est un identifiant (nom) pour une valeur simple. La valeur ne peut pas être modifiée pendant le script.

Un nom de constante valide commence par une lettre ou un trait de soulignement.

Pour créer une constante, utilisez la fonction **define()**.

Syntaxe :

define (*"NOM_CONSTANTE"* **,** *valeur***)**

II. Le langage PHP

5. Les variables en PHP

a. Règles pour les variables PHP :

- ✓ Une variable commence par le signe **$**, suivi du nom de la variable;
- ✓ Un nom de variable **doit commencer** par une lettre ou le caractère de soulignement "_" ;
- ✓ Un nom de variable ne peut pas commencer par un chiffre ;
- ✓ Un nom de variable ne peut contenir que des caractères alphanumériques et des traits de soulignement (A-z, 0-9 et _);
- ✓ Les noms de variables sont sensibles à la casse ($age et $AGE sont deux variables différentes).

II. Le langage PHP

b. Les différents types de variables

PHP prend en charge les types de données suivants :

- ✓Chaîne de caractères (string)
- ✓Entier (int)
- ✓Réel (float)
- ✓Booléen (bool)
- ✓Tableau (array)
- ✓...

II. Le langage PHP

c. Changer le type de données

Si vous attribuez une valeur entière à une variable, le type sera **automatiquement** un entier.
Pour modifier le type de données d'une variable existante, mais sans modifier la valeur, vous pouvez utiliser les opérateurs de transtypage.

Opérateur de transtypage	Rôle
(int) variable	Convertit une variable en entier
(float) variable	Convertit une variable en réel.
(string) variable	Convertit une variable en chaîne
(bool) variable	Convertit une variable en booléen
(array) variable	Convertit une variable en tableau

Exemple :

```php
<?php
$x = 2024;
$x = (string) $x;
var_dump($x);
?>
```

Affichage :

```
string(4) "2024"
```

II. Le langage PHP

6. Les opérateurs PHP

Opérateur d'affectation

Opérateur	Nom
=	affectation

Opérateurs sur les chaînes

Opérateur	Nom
.	concaténation

II. Le langage PHP

Opérateurs arithmétiques

Opérateur	Nom
+	addition
-	soustraction
*	multiplication
/	division
%	modulo

II. Le langage PHP

Opérateurs de comparaison

Opérateur	Nom
==	égal à
<> ou bien !=	différent de
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à

II. Le langage PHP

Opérateurs logiques

Opérateur	Nom
&&	ET
||	OU
!	NON

II. Le langage PHP

7. Les fonctions mathématiques en PHP

Fonction	Rôle
abs()	Retourne la valeur absolue d'un nombre
sqrt()	Retourne la racine carrée d'un nombre
round()	Arrondit un nombre réel à son entier le plus proche
rand()	Génère un nombre aléatoire

```php
<?php
//génère un nombre aléatoire
echo(rand());

//génère un entier aléatoire compris entre 10 et 100 (inclus)
echo(rand(10, 100))
?>
```

II. Le langage PHP

8. Fonctions diverses

Fonction	Rôle
die()	Affiche un message et termine le script courant.
isset()	Vérifie si une variable est définie.
require()	Inclut et exécute le fichier spécifié en argument.

```php
<?php
$a = 0;
// True car $a est défini
if (isset($a)) {
  echo "La variable 'a' est définie.<br>";
}

?>
```

II. Le langage PHP

9. L'instruction if en PHP

En PHP, nous avons les instructions conditionnelles suivantes :

- ❑ **Instruction if** - exécute du code si une condition est vraie
- ❑ **Instruction if ... else** - exécute du code si une condition est vraie et un autre code si cette condition est fausse
- ❑ **Instruction if ... elseif ... else** - exécute des codes différents pour plus de deux conditions

II. Le langage PHP

a. L'instruction if

Syntaxe :

if (*condition***) {**

code à exécuter si la condition est vraie ;

}

```php
<?php
$t = 14;

//Affiche "Bonne journée !" si $t est inférieur à 20 :
if ($t < 20) {
  echo "Bonne journée!";
}
?>
```

II. Le langage PHP

b. L'instruction if ... else

Syntaxe :

if (*condition***) {**
code à exécuter si la condition est vraie ;
}
else {
code à exécuter si la condition est fausse ;
}

```php
<?php
$t = date("H");

/*Affiche "Bonne journée !" si l'heure actuelle est
inférieure à 20 heures, et "Bonne nuit !" sinon
*/
if ($t < "20") {
  echo "Bonne journée!";
} else {
  echo "Bonne nuit!";
}
?>
```

II. Le langage PHP

c. L'instruction if ... else if ... else

Syntaxe :

if (*condition1***) {**
code à exécuter si cette condition est vraie ;
}
else if (*condition2***) {**
code à exécuter si la première condition est fausse et que cette condition est vraie ;
}
else {
code à exécuter si toutes les conditions sont fausses ;
}

```php
<?php
$t = date("H");

/*
Affiche "Bonjour !" si l'heure actuelle est inférieure
à 10 heures, et "Bonne journée !" si l'heure actuelle
est inférieure à 20. Sinon, il affichera "Bonne nuit !"
*/
if ($t < "10") {
  echo "Bonjour!";
} elseif ($t < "20") {
  echo "Bonne journée!";
} else {
  echo "Bonne nuit!";
}
?>
```

II. Le langage PHP

10. Les boucles en PHP

En PHP, nous avons les types de boucles suivants :

- **while** – exécute un bloc de code tant que la condition spécifiée est vraie.
- **do... while** - exécute un bloc de code une fois, puis répète la boucle tant que la condition spécifiée est vraie.
- **for** - exécute un bloc de code un nombre de fois spécifié.
- **foreach** - exécute un bloc de code pour chaque élément d'un tableau.

II. Le langage PHP

a. La boucle while

Syntaxe :

while (*condition***) {**

code à exécuter tant que la condition spécifiée est vraie ;

}

```php
<?php
$i = 1;
//Afficher $i tant que $i est inférieur à 6 :
while ($i < 6) {
  echo $i;
  $i++;
}
?>
```

II. Le langage PHP

b. La boucle do ... while

Syntaxe :

do {

code à exécuter tant que la condition spécifiée est vraie ;

} while (*condition***);**

```php
<?php
$i = 1;
//Afficher $i tant que $i est inférieur à 6 :
do {
  echo $i;
  $i++;
} while ($i < 6);
?>
```

II. Le langage PHP

c. La boucle for

Syntaxe :

```
for (expression1; expression2; expression3) {
  // bloc de code
}
```

```php
<?php
//Afficher les nombres de 0 à 10
for ($x = 0; $x <= 10; $x++) {
  echo "Le nombre est: $x <br>";
}
?>
```

II. Le langage PHP

d. La boucle foreach

Deux syntaxes, avec ou sans clés :

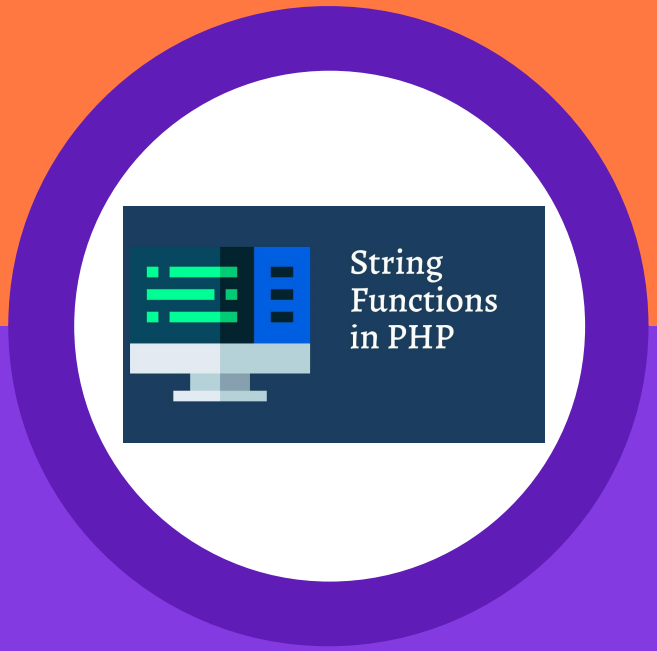
```php
<?php
$couleurs = array("rouge", "vert", "bleu", "jaune");

//Parcourir les éléments d'un tableau indexé
foreach ($couleurs as $c) {
  echo "$c <br>";
}
?>
```

```php
<?php
$membres = array("Sameh"=>"35", "Najla"=>"37", "Omar"=>"43");

//afficher à la fois la clé et la valeur du tableau associatif $membres
foreach ($membres as $cle => $valeur) {
  echo "$cle : $valeur <br>";
}
?>
```

III- Les chaînes



III. Les chaînes

Une chaîne est une séquence de caractères, comme "Bonjour tout le monde !".

Les chaînes en PHP sont entourées soit de guillemets doubles, soit de guillemets simples.

```php
<?php
echo "Bonjour";
print "Bonjour";
?>
```

```php
<?php
$x = "Antoine";
echo "Bonjour $x";
?>
```

1. Longueur de chaine

La fonction PHP **strlen()** renvoie la longueur d'une chaîne.

```php
<?php
echo strlen("Bonjour le monde!");
?>
```

III. Les chaînes

2. Rechercher du texte dans une chaîne

La fonction PHP **strpos()** recherche un texte spécifique dans une chaîne.

Si une correspondance est trouvée, la fonction **renvoie la position** du caractère de la première correspondance. **Si aucune correspondance n'est trouvée**, il **retournera FALSE**.

```php
<?php
echo strpos("J'aime php, j'aime php aussi !","php");
?>
```

III. Les chaînes

3. Modifier les chaînes

PHP dispose d'un ensemble de fonctions intégrées que vous pouvez utiliser pour modifier des chaînes.

a. Majuscule

La fonction **strtoupper()** renvoie la chaîne en majuscule :

```php
<?php
$x = "Bonjour le Monde!";
echo strtoupper($x);
?>
```

BONJOUR LE MONDE!

b. Minuscule

La fonction **strtolower()** renvoie la chaîne en minuscule :

```php
<?php
$x = "Bonjour le Monde!";
echo strtolower($x);
?>
```

bonjour le monde!

III. Les chaînes

c. Remplacer une chaîne

La fonction PHP **str_replace()** remplace certains caractères par d'autres caractères dans une chaîne.

```php
<?php
//Remplacer le texte « le Monde » par « Monsieur »
$x = "Bonjour le Monde!";
echo str_replace("le Monde", "Monsieur", $x);
?>
```

Bonjour Monsieur!

d. Supprimer les espaces

La fonction PHP **trim()** supprime tout espace du début ou de la fin.

```php
<?php
$x = " Bonjour le Monde! ";
echo trim($x);
?>
```

III. Les chaînes

4. Concaténation des chaînes

Pour **concaténer** ou combiner deux chaînes, vous pouvez utiliser **l'opérateur « . »** :

```php
<?php
$x = "Bonjour";
$y = "le Monde";
$z = $x . " ". $y;
echo $z;
?>
```

Bonjour le Monde

III. Les chaînes

5. Découpage de chaînes

Tranchage

Vous pouvez renvoyer une sous chaîne en utilisant la fonction **substr()**.

Spécifiez l'index de départ et le nombre de caractères que vous souhaitez renvoyer.

```php
<?php
//Démarrer la tranche à l'index 11 et terminez la tranche 5 positions plus tard
$x = "Bonjour le Monde!";
echo substr($x, 11, 5);
?>
```

Monde

Trancher jusqu'à la fin

```php
<?php
$x = "Bonjour le Monde!";
//Commencez la tranche à l'index 11 et allez jusqu'à la fin
echo substr($x, 11);
?>
```

Monde!

III. Les chaînes

6. Comparaison de deux chaînes

La fonction PHP **strcmp()** compare deux chaînes.

Elle retourne :

- **0** - si les deux chaînes sont égales
- **<0** - si chaîne1 est inférieure à chaîne2
- **>0** - si chaîne1 est supérieure à chaîne2

Remarque : La fonction strcmp() est **sensible à la casse**.

```php
<?php
echo strcmp("Bonjour le Monde!","Bonjour le Monde!")."<br>"; // les deux chaînes sont égales
echo strcmp("Bonjour le Monde!","Bonjour")."<br>"; // Chaîne 1 est supérieure à la chaîne 2
echo strcmp("Bonjour le Monde!","Bonjour le Monde! Bonjour!")."<br>"; // Chaîne 1 est inférieure à la chaîne 2
?>
```

```
0
10
-9
```

III. Les chaînes

7. Fonction chr()

La fonction PHP **chr()** retourne un caractère de la valeur ASCII spécifiée.

```php
<?php
//Utiliser les valeurs décimales 49, 52 et 53 pour ajouter les caractères ASCII : "1", "4" et "5".
$str = chr(49);
$str2 = chr(52);
$str3 = chr(53);
echo("$str + $str2 = $str3");
?>
```

```
1 + 4 = 5
```

8. Fonction ord()

La fonction PHP **ord()** retourne le code ASCII du premier caractère d'une chaîne.

```php
<?php
//Retourne le code ASCII de "b" :
echo ord("b")."<br>";
echo ord("bonjour")."<br>";
?>
```

```
98
98
```

IV- Les tableaux

IV. Les tableaux

1. Qu'est-ce qu'un tableau ?

Un tableau est une variable spéciale qui peut **contenir plusieurs valeurs** sous un seul nom, et vous pouvez accéder aux valeurs en faisant référence à **un numéro d'index** ou à **un nom**.

2. Les types de tableaux

En PHP, il existe deux types de tableaux :

- ✓ **Tableaux indexés** : Tableaux avec un index numérique
- ✓ **Tableaux associatifs** : Tableaux avec **clés** nommées

IV. Les tableaux

3. Créer des tableaux

✓Vous pouvez créer des tableaux en utilisant la fonction **array()** :

```php
<?php
$voitures = array("Volvo", "BMW", "Toyota");
?>
```

✓Vous pouvez également utiliser une syntaxe plus courte en utilisant les **crochets [ ]** :

```php
<?php
$voitures = ["Volvo", "BMW", "Toyota"];
?>
```

✓Vous pouvez d'abord déclarer **un tableau vide**, puis y ajouter des éléments ultérieurement :

```php
<?php
//Déclarer un tableau vide
$voitures = [];
$voitures[0] = "Volvo";
$voitures[1] = "BMW";
$voitures[2] = "Toyota";
?>
```

✓Il en va **de même** pour les **tableaux associatifs** :

```php
<?php
$maVoiture = [];
$maVoiture["marque"] = "Ford";
$maVoiture["modèle"] = "Mustang";
$maVoiture["année"] = 1964;
?>
```

IV. Les tableaux

4. Éléments du tableau

Les éléments du tableau peuvent être de n'importe quel type de données (chaînes , nombres (int, float), ...).

Vous pouvez avoir **différents types** de données dans le même tableau.

Exemple

Éléments du tableau de **quatre types de données** différents :

```php
<?php
$user =  array("Mickael Andrieu", "mickael.andrieu@exemple.com","S3cr3t", 34);
?>
```

IV. Les tableaux

5. Tableaux indexés PHP

Dans les tableaux indexés, chaque élément a **un numéro d'index**.

Par défaut, **le premier élément a l'index 0**, le deuxième élément a l'élément 1, etc.

```php
<?php
//Créer un tableau indexé
$voitures = array("Volvo", "BMW", "Toyota");
?>
```

a. Accéder aux tableaux indexés

Pour accéder à un élément du tableau, vous pouvez vous référer au **numéro d'index**.

```php
<?php
//Créer un tableau indexé
$voitures = array("Volvo", "BMW", "Toyota");
//Afficher le premier élément du tableau indexé
echo $voitures[0];
?>
```

IV. Les tableaux

b. Modifier la valeur

Pour modifier la valeur d'un élément du tableau, utilisez le **numéro d'index** :

```php
<?php
//Créer un tableau indexé
$voitures = array("Volvo", "BMW", "Toyota");
//Modifier la valeur du deuxième élément :
$voitures[1] = "Ford";
?>
```

c. Boucler sur un tableau indexé

Pour parcourir et afficher toutes les valeurs d'un tableau indexé, vous pouvez utiliser une boucle **foreach**, comme ceci :

```php
<?php
//Créer un tableau indexé
$voitures = array("Volvo", "BMW", "Toyota");
//Afficher tous les éléments du tableau :
foreach ($voitures as $v) {
    echo "$v <br>";
}
?>
```

IV. Les tableaux

6. Tableaux associatifs PHP

Les tableaux associatifs sont des tableaux qui utilisent **des clés nommées** que vous leur attribuez.

```php
<?php
//Créer un tableau  associatif :
$maVoiture = array("marque"=>"Ford", "modèle"=>"Mustang", "année"=>1964);
?>
```

a. Accéder aux tableaux associatifs

Pour accéder à un élément du tableau, vous pouvez vous référer au **nom de la clé**.

```php
<?php
$maVoiture = array("marque"=>"Ford", "modèle"=>"Mustang", "année"=>1964);
//Afficher le modèle de la voiture
echo $maVoiture["modèle"];
?>
```

IV. Les tableaux

b. Modifier la valeur

Pour modifier la valeur d'un élément du tableau, utilisez le **nom de la clé** :

```php
<?php
$maVoiture = array("marque"=>"Ford", "modèle"=>"Mustang", "année"=>1964);
//Changer l'élément de l'année
$maVoiture["année"] = 2024;
?>
```

c. Boucler sur un tableau associatif

Pour parcourir et afficher toutes les valeurs d'un tableau associatif, vous pouvez utiliser une boucle **foreach**, comme ceci :

```php
<?php
$maVoiture = array("marque"=>"Ford", "modèle"=>"Mustang", "année"=>1964);
//Afficher tous les éléments, clés et valeurs du tableau
foreach ($maVoiture as $k => $v) {
    echo "$k: $v <br>";
}
?>
```

IV. Les tableaux

7. Fonctions de tableau

La véritable force des tableaux PHP réside dans les fonctions de tableau intégrées, comme la fonction **count()** pour **compter les éléments du tableau** :

Exemple

Combien d'éléments y a-t-il dans le tableau $voitures :

```php
<?php
$voitures = array("Volvo", "BMW", "Toyota");
//Afficher le nombre d'éléments d'un tableau
echo count($voitures);
?>
```

V- Date et heure

V. Date et heure

1. Fonction date()

La fonction PHP **date()** permet de formater une date et/ou une heure.

```php
<?php
//L'exemple ci-dessous formate la date du jour de trois manières différentes
echo "Aujourd'hui c'est " . date("Y/m/d") . "<br>";
echo "Aujourd'hui c'est " . date("Y.m.d") . "<br>";
echo "Aujourd'hui c'est " . date("Y-m-d") . "<br>";
?>
```

Aujourd'hui c'est 2024/01/08
Aujourd'hui c'est 2024.01.08
Aujourd'hui c'est 2024-01-08

```php
<?php
$d=date("Y-m-d");          //Retourne la date système
$d=date("H:i:s");          //Retourne l'heure système
$d=date("Y-m-d H:i:s");    //Retourne la date et heure système
$d=date("Y-m-d",$t);       //convertit le temps $t en une date
?>
```

V. Date et heure

2. Fonction checkdate()

La fonction **checkdate()** est utilisée pour valider une date grégorienne.

```php
<?php
var_dump(checkdate(12,31,-400));
echo "<br>";
var_dump(checkdate(2,29,2003));
echo "<br>";
var_dump(checkdate(2,29,2004));
?>
```

```
bool(false)
bool(false)
bool(true)
```

V. Date et heure

3. Fonction strtotime()

La fonction PHP **strtotime()** est utilisée pour convertir une chaîne de date lisible par l'homme en un horodatage Unix (le nombre de secondes depuis le 1er janvier 1970 00:00:00 GMT).

```php
<?php
$d=strtotime("10:30pm April 15 2014");
echo "La date de création est " . date("Y-m-d h:i:s", $d);
?>
```

La date de création est 2014-04-15 10:30:00

```php
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:s", $d) . "<br>";

$d=strtotime("next Saturday");
echo date("Y-m-d h:i:s", $d) . "<br>";

$d=strtotime("+3 Months");
echo date("Y-m-d h:i:s", $d) . "<br>";
?>
```

2024-01-09 12:00:00
2024-01-13 12:00:00
2024-04-08 10:51:57

V. Date et heure

4. Fonction time()

La fonction PHP **time()** retourne l'heure actuelle sous forme de timestamp Unix, puis formate-la en date.

```php
<?php
$t=time();
echo($t . "<br>");
echo(date("Y-m-d",$t));
?>
```

```
1704711372
2024-01-08
```

VI- Les fonctions

V. Les fonctions

Outre les fonctions PHP prédéfinies, il est possible de créer vos propres fonctions.

- ✓Une fonction est un bloc d'instructions pouvant être utilisé de manière répétée dans un programme.
- ✓Une fonction ne s'exécutera pas automatiquement lors du chargement d'une page.
- ✓Une fonction sera exécutée par un appel à la fonction.

1. Créer une fonction

Une déclaration de fonction définie par l'utilisateur commence par le mot clé **function**, suivi du nom de la fonction.

```php
<?php
//Déclaration de la fonction monMessage
function monMessage() {
  echo "Bonjour le monde!";
}
?>
```

V. Les fonctions

Remarque : Un nom de fonction doit **commencer par une lettre ou un trait de soulignement**. Les noms de fonctions **ne sont PAS sensibles à la casse**.

2. Appeler une fonction

Pour appeler la fonction, il suffit d'écrire son nom suivi de parenthèses ()

```php
<?php
//Déclaration de la fonction monMessage
function monMessage() {
  echo "Bonjour le monde!";
}

//Appel de la fonction monMessage
monMessage()
?>
```

Remarque : Les arguments sont spécifiés après le nom de la fonction, entre parenthèses. Vous pouvez ajouter autant d'arguments que vous le souhaitez, il suffit de les séparer par une virgule.

VII- Gestion des formulaires PHP

V. Gestion des formulaires PHP

Les variables superglobales PHP **$_GET** et **$_POST** sont utilisées pour collecter les données de formulaire.

GET vs. POST

Tous les deux GET et POST créent un tableau (par exemple **array** (clé1 => valeur1, clé2 => valeur2, clé3 => valeur3, ...)).

Ce tableau contient des paires clé/valeur, où **les clés** sont les **noms des champs de formulaire** et **les valeurs** sont **les données d'entrée de l'utilisateur**.

V. Gestion des formulaires PHP

GET et POST sont traités comme $_GET et $_POST. Ce sont des **variables super globaux**, ce qui signifie qu'ils **sont toujours accessibles, quelle que soit leur portée** - et vous pouvez y accéder à partir de n'importe quelle fonction ou fichier sans rien faire de spécial.

- ✓**$_GET** est un tableau de variables transmis au script actuel **via les paramètres URL**.
- ✓**$_POST** est un tableau de variables transmis au script actuel via **la méthode HTTP POST**.

VIII- PHP avec MySQL

VIII. PHP avec MySQL

1. Ouvrir une connexion à MySQL

Avant de pouvoir accéder aux données de la base de données MySQL, nous devons pouvoir nous connecter au serveur.

La fonction PHP **mysqli_connect()** ouvre une nouvelle connexion au serveur MySQL et sélectionne une base de données.

mysqli_connect(serveur, nomUtilisateur, motDePasse, nomBd)

```php
<?php
// Ouvrir une nouvelle connexion et sélectionner la base de données nommée "test"
$conn = mysqli_connect("localhost", "root", "", "test");
?>
```

VIII. PHP avec MySQL

3. Fermer une connexion à MySQL

La fonction PHP **mysqli_close()** est utilisée pour fermer une connexion à une base de données précédemment ouverte.

mysqli_close(connexion)

```php
<?php
// Ouvrir une nouvelle connexionau à un serveur MySQL.
$conn = mysqli_connect("localhost", "root", "");

// Sélectionner la base de données nommée "test"
mysqli_select_db($conn, "test");

// ....du code PHP...

// Fermer la connexion $conn
mysqli_close($conn);
?>
```

VIII. PHP avec MySQL

4. Exécuter une requête

La fonction PHP **mysqli_query()** exécute une requête sur une base de données.

mysqli_query (connexion, requête)

```php
<?php
// Ouvrir une nouvelle connexionau à un serveur MySQL.
$conn = mysqli_connect("localhost", "root", "", "test");

$sql = "INSERT INTO MesInvites (nom, prenom, email) VALUES ('Castor', 'Mickael', 'm.castor@exemple.com')";
// Exécute une requête
$res = mysqli_query($conn, $sql);

// Fermer la connexion $conn
mysqli_close($conn);
?>
```

Remarque : La requête doit être une chaîne de caractères.

VIII. PHP avec MySQL

4. Obtenir le nombre de résultats retournés par la requête SELECT

La fonction PHP **mysqli_num_rows()** renvoie le nombre de lignes retournées par une requête SELECT.

mysqli_num_rows(resultat)

```php
<?php
// Ouvrir une nouvelle connexionau à un serveur MySQL.
$conn = mysqli_connect("localhost", "root", "", "Monde");

$sql = "SELECT Code, Nom FROM Pays ORDER BY Nom";
// Exécute une requête
$res = mysqli_query($conn, $sql);

// Obtenir le nombre de lignes dans le jeu de résultats
$nbl = mysqli_num_rows($res);

echo "L'ensemble de résultats comporte $nbl lignes.";

// Fermer la connexion $conn
mysqli_close($conn);
?>
```

VIII. PHP avec MySQL

5. Obtenir le nombre de lignes affectées

La fonction **mysqli_affected_rows()** est utilisée pour obtenir le nombre de lignes affectées par une requête SQL exécutée sur la base de données, comme une requête INSERT, UPDATE ou DELETE.

mysqli_affected_rows(connexion)

```php
<?php
// Ouvrir une nouvelle connexionau à un serveur MySQL.
$conn = mysqli_connect("localhost", "root", "", "Monde");

// Mettre à jour des lignes
mysqli_query($conn, "UPDATE Langue SET Statut=1 WHERE Pourcentage > 50");
echo "Lignes affectées (UPDATE) : ".mysqli_affected_rows($conn)

// Supprimer des lignes
mysqli_query($conn, "DELETE FROM Langue WHERE Pourcentage < 50");
echo "Lignes affectées (DELETE) : ".mysqli_affected_rows($conn)

//  sélectionner toutes les lignes
mysqli_query($conn, "SELECT CodePays FROM Langue");
echo "Lignes affectées (SELECT) : ".mysqli_affected_rows($conn)

// Fermer la connexion $conn
mysqli_close($conn);
?>
```

VIII. PHP avec MySQL

Fonctionnement de la fonction mysqli_affected_rows() :

- ✓**INSERT** : Elle renvoie le nombre de lignes insérées.
- ✓**UPDATE** : Elle renvoie le nombre de lignes **modifiées**.
- ✓**DELETE** : Elle renvoie le nombre de lignes supprimées.
- ✓Si la requête **n'affecte aucune ligne** (par exemple, si une mise à jour ne modifie aucune donnée ou si une suppression n'a pas trouvé de ligne correspondante), elle **renverra 0**.
- ✓Si la **requête échoue**, elle **renverra -1**.

VIII. PHP avec MySQL

6. Récupère une ligne d'un ensemble de résultats

La fonction PHP **mysqli_fetch_row()** récupère une ligne d'un ensemble de résultats et la renvoie sous forme de tableau indexé.

mysqli_fetch_row(resultat)

```php
<?php
// Ouvrir une nouvelle connexionau à un serveur MySQL.
$conn = mysqli_connect("localhost", "root", "", "Monde");

$sql = "SELECT Nom, CodePays FROM Ville ORDER BY ID DESC";
$res = mysqli_query($conn, $sql);

// Récupère une et une ligne sous forme de tableau indexé
while ($ligne = mysqli_fetch_row($res)) {
    echo $ligne[0]." ".$ligne[1]."<br>";
}

// Fermer la connexion $conn
mysqli_close($conn);
?>
```

VIII. PHP avec MySQL

La fonction PHP **mysqli_fetch_array()** la ligne suivante d'un ensemble de résultats sous forme d'un tableau associatif, d'un tableau indexé ou les deux.

mysqli_fetch_array(resultat, typeResultat)

Remarque :

typeResultat : Facultatif. Spécifie le type de tableau à produire. Il peut s'agir de l'une des valeurs suivantes :

- MYSQLI_ASSOC
- MYSQLI_NUM
- MYSQLI_BOTH (c'est la valeur par défaut)

VIII. PHP avec MySQL

```php
<?php
// Ouvrir une nouvelle connexionau à un serveur MySQL.
$conn = mysqli_connect("localhost", "root", "", "Monde");

$sql = "SELECT Nom, CodePays FROM Ville ORDER BY ID DESC";
$res = mysqli_query($conn, $sql);

// Tableau indexé
$ligne = mysqli_fetch_array($res, MYSQLI_NUM);
echo $ligne[0]." ".$ligne[1]."<br>";

// Tableau associatif
$ligne = mysqli_fetch_array($res, MYSQLI_ASSOC);
echo $ligne["Nom"]." ".$ligne["CodePays"]."<br>";

// Tableau associatif et indexé
$ligne = mysqli_fetch_array($res, MYSQLI_BOTH);
echo $ligne[0]." ".$ligne["CodePays"]."<br>";

// Fermer la connexion $conn
mysqli_close($conn);
?>
```

VIII. PHP avec MySQL

7. Gestion des erreurs

La fonction PHP **mysqli_error()** retourne une description sous forme de chaîne de la dernière erreur.

mysqli_error(connexion)

```php
<?php
$conn = mysqli_connect("localhost", "root", "", "Monde");

// Exécuter une requête et vérifier l'erreur
if (!mysqli_query($conn,"INSERT INTO Utilisateurs (nom) VALUES ('Alex')")) {
  echo("Description de l'erreur :" . mysqli_error($conn));
}
// Fermer la connexion
mysqli_close($conn);
?>
```

VIII. PHP avec MySQL

7. Gestion des erreurs

La fonction PHP **mysqli_error()** retourne une description sous forme de chaîne de la dernière erreur.

mysqli_error(connexion)

```php
<?php
$conn = mysqli_connect("localhost", "root", "", "Monde");

// Exécuter une requête et vérifier l'erreur
if (!mysqli_query($conn,"INSERT INTO Utilisateurs (nom) VALUES ('Alex')")) {
  echo("Description de l'erreur :" . mysqli_error($conn));
}
// Fermer la connexion
mysqli_close($conn);
?>
```