



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**
Membre de
HONORIS UNITED UNIVERSITIES

Module :

Programmation PHP et Fremawork

Chapitre 2: Programmation PHP



Pr. Zakia EL UAHHABI



Plan du cours



1

Syntaxe de base

2

Variables et types

3

Structures de contrôle

4

Fonctions

Syntaxe de base

- Balises de début et de fin d'un script

```
<?php  
// code PHP  
?>
```

- Ex. **<?php** echo "Bonjour tout le monde"; **?>**

- Toute instruction se termine par un point virgule **;**

- Sensible à la casse.

- Les commentaires

```
/* voici un commentaire */  
// un commentaire sur une ligne
```

Syntaxe de base

- Balises de début et de fin d'un script

<?php
// code PHP
?>

PHP (code)

HTML content

< ?php
PHP code
?>

HTML content

< ?php
PHP code
?>

HTML content ...

- Le code (PHP) contenu dans **< ?php et ?>** est exécuté et le résultat transmis au client.
- Le code (HTML) non contenu dans **< ?php et ?>** est transmis tel quel au client

Variables et types

Déclaration et affectation

- En PHP, les noms de variables sont préfixées par le symbole dollar (\$) et commencent par une lettre minuscule, majuscule ou le caractère souligné (_).
- Les espaces ne sont pas autorisés.
- PHP ne nécessite pas de déclaration explicite du type de variable .

- Types de données :
 - Nombres entiers : int, integer
 - Nombres réels : real, double, float
 - Chaînes de caractères : string
 - Tableaux : array
 - Objets

Exemples:



```
$i = 1;  
$pi = 3.14;  
$ch = "oui";
```

- Conversion de type :
 - Ex. \$x = 1.25;
 \$x = (int) \$x; // \$x est égal à 1

Variables et types

Chaînes et interprétation des variables

```
<?php
$age = 16 ;
print "You are " . $age . " years old."
print "You are $age years old." ; # You are 16 years old.
?
```

-  Les variables dans les chaînes délimitées par " " sont interprétées.
-  Les variables dans ces chaînes sont remplacées par leur valeur.

Variables et types

Chaînes et interprétation des variables

- Les variables dans les chaînes délimitées par ' ' ne sont pas interprétées :

```
<?php
$age = 16 ;
print 'You are $age years old.' ; # You are $age years old.
?
```

- Pour lever toute ambiguïté, on peut placer le nom de la variable entre :

```
<?php
$age = 16 ;
print "Today is your $ageth birthday." ; # ageth not found
```

Variables et types

opérateurs
d'incrément/d
décrément

Opérateurs prédéfinis

opérateurs
de
comparaison

Opérateur	Nom	Résultat
++\$a	Pré-incrément	Incrémente \$a d'un, puis renvoie \$a.
\$a++	Post-incrément	Renvoie \$a, puis incrémente \$a d'un.
--\$a	Pré-décrément	Décrémente \$a d'un, puis renvoie \$a.
\$a--	Post-décrément	Renvoie \$a, puis décrémente \$a d'un.

Opérateur	Signification
=	<u>affectation</u> simple
+=	<u>addition</u> puis affectation
-=	<u>soustraction</u> puis affectation
*=	<u>multiplication</u> puis affectation
/=	<u>division</u> puis affectation
%=	<u>modulo</u> puis affectation
=	<u>ou</u> puis affectation
&=	<u>et</u> puis affectation
=>	<u>associe</u> une valeur à une clé dans un tableau
->	<u>réalise</u> un appel de méthode

Exemple	Nom	Résultat
\$a == \$b	Egal	Vrai si les valeurs de \$a et \$b sont égales
\$a === \$b	Identique	Vrai si \$a == \$b et si \$a et \$b sont de même type.
\$a != \$b	Non égal	Vrai si \$a n'est pas égal à \$b.
\$a <> \$b	Non égal	Vrai si \$a n'est pas égal à \$b.
\$a !== \$b	Non identique	Vrai si \$a n'est pas égal à \$b, ou si \$a et \$b sont de types différents (PHP 4).
\$a < \$b	Plus petit que	Vrai si \$a est plus petit que \$b.
\$a > \$b	Plus grand que	Vrai si \$a est plus grand que \$b.
\$a <= \$b	Plus petit ou égal à	Vrai si \$a est plus petit ou égal à \$b
\$a >= \$b	Plus grand ou égal à	Vrai si \$a est plus grand ou égal à \$b.

opérateurs
d'affectation

Variables et types



Portée des variables



Portée dans un script:

Une variable déclarée dans un script est reconnue juste après l'instruction de sa déclaration.

Portée dans une fonction :




- 👉 Une variable déclarée dans le corps d'une fonction n'est accessible que dans le corps de cette même fonction. Elle sera considéré comme une variable locale à la fonction.
- 👉 Utiliser le mot clé global à l'intérieur d'une fonction pour utiliser des variables déclarées dans votre script auparavant.

Variables et types



Variables prédéfinies: superglobaux



-  PHP propose toute une série de variables prédéfinies qui sont déjà présentés dans le langage sans que vous n'ayez à les déclarer: variables du serveur, variables d'environnement et celles d'entrées
-  Elles s'écrivent toujours en majuscules et fournissent divers renseignements.
-  Elles sont automatiquement globaux. Elles sont dites '**Superglobaux**'

Variables et types

Variables prédéfinies: superglobaux

Nom	Description
\$GLOBALS	Contient toutes les variables disponibles dans l'environnement d'exécution global.
<u>\$_SERVER</u>	Contient les variables fournies par le serveur web.
\$_GET	Contient les variables fournies en paramètre au script via la méthode GET du protocole HTTP.
\$_POST	Contient les variables fournies par un formulaire via la méthode POST du protocole HTTP.
\$_COOKIES	Contient les variables fournies par les cookies via le protocole HTTP.
\$_FILES	Contient les variables fournies suite à un chargement de fichier par un formulaire via la méthode POST du protocole HTTP.
\$_ENV	Contient les variables fournies par l'environnement. Ce peut être des variables du Shell sous lequel s'exécute PHP, les variables CGI...
\$_REQUEST	Contient les variables fournies au script par n'importe quel mécanisme. (il contient par défaut le contenu des variables <u>\$_GET</u> , <u>\$_POST</u> et <u>\$_COOKIE</u>)

Variables et types

Variables prédéfinies: variables d'environnement.

Variable	Utilité	Affichage
<code>\$_SERVER['REQUEST_METHOD']</code>	La méthode d'appel	POST
<code>\$_SERVER['SERVER_NAME']</code>	Nom du serveur	localhost
<code>\$_SERVER['SERVER_ADMIN']</code>	L'email de l'administrateur du serveur	email@domaine.tld
<code>\$_SERVER['SERVER_ADDR']</code>	L'Adresse IP du serveur	195.14.0.256
<code>\$_SERVER['QUERY_STRING']</code>	<u>Les paramètres indiqués à votre script</u>	url=toto.html&id=234
<code>\$_SERVER['REMOTE_PORT']</code>	Port HTTP de la requête	80
<code>\$_SERVER['REMOTE_ADDR']</code>	Adresse IP de l'internaute	88.101.2.255
<code>\$_SERVER['REQUEST_URI']</code>	Chemin du script	/exemple.php
<code>\$_SERVER['PATH_TRANSLATED']</code>	Chemin physique (complet) du script	/home/www/domain.fr/exemple.php
<code>\$_SERVER['HTTP_USER_AGENT']</code>	User agent du navigateur du client	Mozilla/5.0 (Windows ; U; Windows NT 5.1; fr; rv:1.8.1) Gecko/20061010 Firefox/2.0
<code>\$_SERVER['HTTP_REFERER']</code>	L'URL de la page d'où provient l'internaute	https://oseox.fr/exemple.php
<code>\$_SERVER['HTTP_HOST']</code>	Le nom de domaine où est exécuté le script	example.fr
<code>\$_SERVER['HTTP_ACCEPT_LANGUAGE']</code>	Langue acceptée par le navigateur de l'internaute	fr
<code>\$_SERVER['DOCUMENT_ROOT']</code>	Adresse de la racine du serveur	/var/www/exemple.fr/

Variables et types

Variables prédéfinies: variables d'environnement.

Exemple:

```
<?php  
echo 'votre nom du serveur est : ' . $_SERVER['SERVER_NAME'] . '<br>';  
echo 'votre chemin du script est : ' . $_SERVER['REQUEST_URI'];  
?>
```



votre nom du serveur est : localhost
votre chemin du script est : /test-php/variable-predefinie.php

Variables et types

Variables externes à PHP

Formulaires HTML (GET et POST)

Lorsqu'un formulaire est envoyé à un script PHP, toutes les variables du formulaire seront automatiquement disponibles dans le script.

Exemple:

```
<form action="test.php" method="post">
  <p>Nom : <input type="text" name="nom" /> </p> <br />
  <p>prenom: <input type="text" name="prenom" /> </p> <br />
  <p><input type="submit" name="submit" value="Envoyer" /> </p>
</form>
```

formuaire.html

```
$nom=$_POST['nom'];
$prenom=$_POST['prenom'];
echo "<h3> Bienvenue ".$prenom." ".$nom."</h3>";
```

test.php

Variables et types

Variables externes à PHP

Formulaires HTML (GET et POST)

Exemple:

formulaire.html

```
<form action="test.php" method="GET">
  <p>Nom : <input type="text" name="nom" /> </p> <br />
  <p>prenom: <input type="text" name="prenom" /> </p> <br />
  <p><input type="submit" name="submit" value="Envoyer" /> </p>
</form>
```

test.php

```
$nom=$_GET['nom'];
$prenom=$_GET['prenom'];
echo "<h3> Bienvenue ".$prenom.' '.$nom."</h3>";
```

les paramètres passés directement dans l'URL sont des paramètres GET :
<http://domain/test.php?nom=value1&prenom=value2>

Variables et types

Variables externes à PHP

Cookies HTTP :

- 👉 Les cookies sont un mécanisme permettant de stocker des données sur la machine cliente à des fins d'identification de l'utilisateur.
- 👉 Pour créer un cookie en PHP, on utilise la fonction `setcookie()`.
- 👉 La fonction `setcookie()` doit être appelée avant d'écrire tout code HTML pour qu'elle fonctionne puisque les cookies doivent être envoyés avant toute autre sortie.
- 👉 La syntaxe de base de `setcookie()` est la suivante:
`setcookie(name, value, expire, path, domain, secure, httponly)`

Variables et types

Variables externes à PHP



Cookies HTTP :

`setcookie(name, value, expire, path, domain, secure, httponly)`

Exemple:

```
setcookie("TestCookie","valeur de test", time() - 3600, "/test/","domaine.com",1,1);
```

value	des informations sensibles.
expires	La date d'expiration du cookie sous forme d'un timestamp UNIX
path	Le chemin sur le serveur sur lequel le cookie sera disponible.
domain	Indique le domaine ou le sous domaine pour lequel le cookie est disponible.
secure	Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client.
httponly	Indique si le cookie ne doit être accessible que par le protocole HTTP. Pour que le cookie ne soit accessible que par le protocole http, on indiquera la valeur true.

Variables et types

Variables externes à PHP

Cookies HTTP :

 Exemple

```
<?php  
  
    setcookie('user_id', '1234');  
  
?>
```

Variables et types

Variables externes à PHP

Cookies HTTP :

Récupérer la valeur d'un cookie

Pour récupérer la valeur d'un cookie, on utilise la variable superglobale **`$_COOKIE`**.

```
<?php
    setcookie('user_id', '1234');
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Cookies</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>recuperer la valeur de cookie</h1>
        <?php
            if(isset($_COOKIE['user_id'])){
                echo 'Votre ID de session est le ' .$_COOKIE['user_id'];
            }
        ?>
    </body>
</html>
```

Variables et types

Variables externes à PHP

Cookies HTTP :

Modifier la valeur d'un cookie

Pour modifier la valeur d'un cookie, on utilise la fonction `setcookie()`.

```
<?php
    setcookie('user_id', '1234');
    //modifier id user
    setcookie('user_id', '12');
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Cookies</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>recuperer la valeur de cookie</h1>
        <?php
            if(isset($_COOKIE['user_id'])){
                echo 'Votre ID de session est le ' .$_COOKIE['user_id'];
            }
        ?>
    </body>
</html>
```

Variables et types

Fonctions de gestion des variables

■ Il existe une multitudes de fonctions qui agissent sur les variables en PHP :

<u>Empty()</u>	:Détermine si une variable est vide;
<u>isset()</u>	:permet de vérifier si la variable passée en paramètre existe ou non;
<u>Unset()</u>	:Détruit une variable;
<u>Gettype()</u>	:Retourne le type de la variable;
<u>is_numeric()</u>	:Détermine si une variable est un nombre ou une chaîne numérique;
<u>is_float()</u>	:Détermine si une variable est de type nombre décimal;
<u>is_int()</u>	:Détermine si une variable est de type nombre entier;
<u>is_array()</u>	:Détermine si une variable est un tableau;
<u>is_bool()</u>	:Détermine si une variable est un booléen;
<u>is_object()</u>	:Détermine si une variable est de type objet;
<u>var_dump()</u>	:Affiche les informations d'une variable;
<u>var_export()</u>	:Retourne le code PHP utilisé pour générer une variable.

Structure de contrôle

Structure conditionnelle : **if ... else if ... else**

```
if (condition)
{
    code qui sera exécuté si la condition1 est vraie ;
}
else if (condition) //optionnel
{
    code qui sera exécuté si la condition1 est vraie ;
}
else //optionnel
{
    code qui sera exécuté si toutes les conditions ci-dessus sont fausses
}
```

On peut utiliser elseif (sans espace) au lieu de else if .

Structure de contrôle

Structure conditionnelle : Switch

```
Switch ($variable)
{
case valeur_1 :
    instruction-ou-bloc1 ;
    break ;
case valeur_2 :
    instruction-ou-bloc1 ;
    break ;
case valeur_n :
    instruction-ou-bloc1 ;
    break ;
default :
    instruction-ou-bloc1 ;
    break ;
}
```

Structure de contrôle

Structure répétitives : Boucle For

```
for (init_expr; condition;incr_expr ){  
  
instructions;  
  
}
```

Avec :

- **init_expr** : c'est une initialisation du compteur de nombre d'itération
- **condition d'arrêt** : c'est la condition d'arrêt du boucle for c'est-à-dire arrêt du compteur d'itération
- **incr_expr** : c'est l'incrémentement du compteur

Exemple:

```
<?php  
for($i=1;$i<6;$i++){  
echo "$i <br>" ;  
}  
?>
```


Structure de contrôle

Structure répétitives : Boucle WHILE

```
while (condition){  
    instructions;  
}
```

```
do {  
    séquence d'instructions;  
} while (condition )
```

Exemple:

```
<?php  
$i=1;  
while ($i<6) {  
    echo "$i <br>" ;  
    $i++;  
}  
?>
```

Exemple:

```
<?php  
$i=1;  
do{  
    echo "$i <br>" ;  
    $i++;  
}while ($i<6)  
  
?>
```

Structure de contrôle



Mots-clés break et continue



- ➡ Le mot clé **break** permet de mettre fin à une boucle prématurément.
- ➡ Le mot-clé **continue** permet d'ignorer l'itération courante et passer immédiatement à l'itération suivante.

Fonctions

Définition

- Fonctions se sont des méthodes définis et implantés par l'utilisateur. Le principe consiste à regrouper dans un bloc un ensemble d'instructions nécessaire pour accomplir une tâche, et à l'appeler à chaque fois qu'on a besoin d'effectuer la tâche.
- Une fonction peut être définie en utilisant la syntaxe suivante :

```
<?php  
function nom-fonction()  
{  
    instructions;  
}  
?>
```

Fonctions

Déclaration de fonction

```
function nom_de_la_fonction ($arg1, $arg2...)
{
    instruction;
    reutrn nom_de_variable ; //optionnel
}
```

```
function nom_de_la_fonction () // sans arguments
{
    instructions ;
}
```

- Une fonction peut avoir des arguments, qui sont facultatives.

Fonctions

Appel de fonction

■ L'appel se fait de deux manières :

- ➡ soit directement en donnant le nom de la fonction et les arguments qu'elle accepte quand la fonction ne retourne pas de valeurs
- ➡ soit en affectant le résultat de la fonction à une variable.

Fonctions

Exemple d'application

Exemple 1:

```
function resultat($x, $y) {  
    $valeur = $x+$y;  
    return 2* $valeur + 1 ;  
}
```

Appel

Exemple 1:

```
$Result=resultat(5, 7);
```

Exemple 2:

```
function afficher() {  
    echo "bonjour tout le monde ";  
}
```

Appel

Exemple 2:

```
afficher()
```

Fonctions

Fonction « include » et « require »

- Les fonctions « **include** » et « **require** » permettent d'insérer le contenu d'un fichier dans un autre.

Syntaxe:

```
<?php
.....code.....
include ("chemin absolu du fichier extension« );
// ou require ("chemin absolu du fichier extension« );
.....code.....
?>
```

Exemple:

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

fichier.php

```
<?php
include('fichier.php');
somme(4,8);
echo ($som);
?>
```

index.php

Fonctions

Fonctions sur le type string

Nom	Fonctions équivalentes en Java
<code>strlen</code>	<code>length</code>
<code>strpos</code>	<code>indexOf</code>
<code>substr</code>	<code>substring</code>
<code>strtolower, strtoupper</code>	<code>toLowerCase, toUpperCase</code>
<code>trim</code>	<code>trim</code>
<code>explode, implode</code>	<code>split, join</code>

Exemple:

```
$name = "Emsi - 3iir"
$length = strlen($name) ; # 11
$cmp = strcmp($name, "Emsi") ; # > 0
$index = strpos($name, "i") ; # 3
$last = substr($name, 8, 4) ; # "3iir"
$name = strtoupper($name) ; # "EMSI - 3IIR"
```


Fonctions

Exercice d'application 1 :

- ➡ Créer un programme en PHP qui génère une table de multiplications d'un nombre donné par un utilisateur. Votre affichage doit être le suivant :

Donnez la valeur à multiplier:

Table de multiplication de : 9

9 x 0 =0

9 x 1 =9

9 x 2 =18

9 x 3 =27

9 x 4 =36

9 x 5 =45

9 x 6 =54

9 x 7 =63

9 x 8 =72

9 x 9 =81

Fonctions

Solution :

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>table multiplication</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <form method="post" action="index.php">
    <label for="multiplication" id="idlabel" >Donnez la valeur à multiplier:</label><br>
    <input type="number" name="valeur" id="idinput"><br>
    <input type="submit" name="valider" value="calculer" id="idbutton">
  </form>
  <hr>
  <div id="container">
    <?php include('multiplication.php') ?>
  </div>
</body>
</html>
```

index.php

```
<?php
function multiplication ($a)
{
  for ($i=0; $i <= 10; $i++)
  {
    echo "<h3>$a x $i =".($a * $i)."</h3>";
  }
}
if(isset($_POST['valider']))
{
  if (isset($_POST['valeur']))
  {
    $valeur=$_POST['valeur'];

    if(!$valeur==" " AND is_numeric($valeur))
    {
      echo "<h2>Table de multiplication de : $valeur</h2>";

      echo "<h3> ".multiplication ($valeur)."</h3>";
    }
  }
}
?>
```

multiplication.php