Royaume du maroc
ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ

Ministère de l'Education Nationale
du préscolaire & des sports
ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵎⴰⵡⴰⵙⵉⵏ ⴰⵏⴰⵎⵓⵔ
ⴰⵏⴰⵎⵓⵔ ⴰⵎⴰⵡⴰⵙ ⴰⵏⴰⵎⵓⵔ ⴷ ⵜⵙⵏⵙⵓⵜ

Lycée Technique de Taza

شهادة التقني العالي

Brevet de Technicien Supérieur

Multimédia et Conception Web

Module : Développement Multimédia

Année professionnelle : 2024 / 2025

Professeur : HICHAM ZAARAOUI

- MCW

Chapitre 3

Les variables en PHP

I. Introduction

- Les variables sont l'un des concepts les plus fondamentaux en programmation, et cela s'applique également à PHP. Elles permettent de stocker, manipuler et accéder à des données au sein d'un script.

- Une variable est un conteneur qui va nous permettre de stocker des données de différents types(entiers, flottants, chaines de caractères, tableaux, booléens, ressource ou NULL).

- En PHP, une variable commence par le signe $, suivi du nom de la variable.

II. Syntaxe de base

- L'extension des fichiers PHP est **.php**
- Un script PHP peut être placé n'importe où dans le fichier.
- Un script PHP commence par `<?php` et se termine par `?>`
- Les instructions doivent se terminer par `;`
- Pour les commentaires en PHP, vous pouvez utiliser `//` ou `#` sur une seule ligne et `/*.....*/` sur plusieurs lignes.

Exemple :

```php
<?php
echo "Hello World!";
//ceci est un commentaire
?>
```

II. Syntaxe de base

Règles pour les variables en PHP

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

III. Les variables en PHP: Affichage et modification d'une variable

- Pour afficher le contenue d'une variable, on utilise l'instruction **echo** ou **ECHO**.

Exemple 1 :

```php
<?php
$a=3;
echo $a;// resultat est 3
?>
```

- Pour modifier le contenue d'une variable, on affecte une nouvelle valeur a la même variable, car le PHP est un langage interprété.

Exemple 2:

```php
<?php
$a=3; echo $a;// resultat est 3
$a=27; echo $a;//resultat est 27
?>
```

III. Les variables en PHP: Type des variables

- Le **typage** des variables est **implicite** et **dynamique** en PHP. Donc, il n'est pas nécessaire de déclarer leur type au préalable ou même de le déclarer.

- **Les types scalaires de base** :
  - ✓ **Entiers**, avec le type **integer**, qui permet de représenter les nombres entiers dans les bases décimale(10), octale(8) et hexadécimale(16).
  - ✓ **Flottants**, avec le type **double** ou **float**, au choix, qui représentent les nombres réels, ou plutôt décimaux au sens mathématique.
  - ✓ **Chaînes de caractères**, avec le type **string**.
  - ✓ **Booléens**, avec le type **boolean**, qui contient les valeurs de vérité **TRUE** ou **FALSE** (soit les valeurs 1 ou 0 si on veut les afficher). Les **Booléens** sont insensibles à la casse.

III. Les variables en PHP: Type des variables

- **Les types composés** :
  - **Tableaux**, avec le type **array**, qui peut contenir plusieurs valeurs.
  - **Objets**, avec le type **object**.
- **Les types spéciaux** :
  - Type **resource** : il correspond à une référence à des informations sur le serveur.
  - Type **null** : represente une variable sans valeur.

Exemple :

```php
<?php
$a=3;//entier
$aa=011;//entier en notation octale(9 en decimal)
$aaaa=0x11;//entier en notation hexadecimal(17)
$b=3.5;//float
$bb=3.5e12;//double en notation exponentielle
$c='abc'; $c="abc"; //chaine de caractères
$d=true;//Booléen
?>
```

III. Les variables en PHP: Affectation par valeur et par référence

❑ Affectation par valeur

L'affectation par valeur signifie que lorsque vous assignez une variable à une autre, une copie de la valeur est créée.

Exemple :

```php
<?php
$a = 5; // Affectation de la valeur 5 à $a
$b = $a; // Affectation par valeur de $a à $b
$b = 10; // Modification de $b
echo $a; // Affichera 5, car $a n'a pas été modifié
echo $b; // Affichera 10, car 5 a été ecrasé
?>
```

III. Les variables en PHP: Affectation par valeur et par référence

❑ Affectation par référence

L'affectation par référence en PHP signifie que vous créez une référence à une variable existante plutôt qu'une copie de sa valeur. Lorsque vous modifiez la valeur de la nouvelle variable, la variable d'origine est également modifiée, car les deux variables pointent vers le même emplacement mémoire.

Exemple :

```php
<?php
$a = 5; // Affectation de la valeur 5 à $a
$b = &$a; // Affectation par référence de $a à $b
$b = 10; // Modification de $b
echo $a; // Affichera 10, car $a a été modifié via $b
echo $b; // Affichera 10
?>
```

III. Les variables en PHP: Conversion de type

- Parfois, vous devez modifier une variable d'un type de données à un autre, et parfois vous souhaitez qu'une variable ait un type de données spécifique. Cela peut être fait à l'aide du casting.

Transtypage explicite(le casting)

Syntaxe :

```
$var=(type)$var;
```

Le casting en PHP se fait avec ces instructions :

- (string)- Convertit en type de données String
- (int)- Convertit en type de données Integer
- (float)- Convertit en type de données Float
- (bool)- Convertit en type de données booléen
- (array)- Convertit en type de données Array
- (object)- Convertit en type de données Objet
- (unset)- Convertit en type de données NULL

III. Les variables en PHP: Conversion de type

❑ Transtypage explicite(le casting)

Exemple :

```php
<?php
$a = 5; // Integer
$b = 5.34; // Float
$c = "hello"; // String
$d = true; // Boolean
$e = NULL; // NULL
$a = (string) $a;
$b = (string) $b;
$c = (string) $c;
$d = (string) $d;
$e = (string) $e;

/*Pour vérifier le type de
n'importe quel objet en PHP,
utiliser la fonction var_dump()*/
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
?>
```

Remarque

La transtypage explicite peut se faire aussi avec la fonction settype():

```
Boolean settype(mixed var,string type).
```

III. Les variables en PHP: Portée des variables PHP

- La portée d'une variable est la partie du script où la variable peut être utilisée.
- PHP a trois portées de variables différentes :
  - Locale
  - Global
  - Statique

Exemple 1: Variable globale

```php
<?php
$x = 5; //porté global
function myTest() {
//utiliser $x à l'intérieur de cette fonction génèrera une erreur
//Solution utiliser le mot clé global $x ou $GLOBALS['x']
echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>La valeur de x à l'exterieur de la focntion est: $x</p>";
?>
```

III. Les variables en PHP: Portée des variables PHP

Exemple 2: Variable local

```php
<?php
function myTest() {
$x = 5; // portée locale
echo "<p>la valeur de x à l'intérieur de la focntion est: $x </p>"; }
myTest();
// utiliser x à l'extérieur de la focntion générera une erreur
echo "<p> La valeur de x à l'exterieur de la function est : $x</p>";
?>
```

Exemple 3: Variable statique

```php
<?php
function myTest() {
static $x = 0;
echo $x;
$x++; }
myTest(); myTest(); myTest();
?>
```

IV. Les variables prédéfinies en PHP

❑ Les **variables prédéfinies** en PHP, également appelées **superglobales**, sont des variables globales disponibles partout dans le script. Ces variables permettent d'accéder facilement à diverses informations du serveur, des requêtes, des sessions, etc. Voici quelques-unes des plus couramment utilisées :

- **$_GET**

Contient les données envoyées via l'URL en méthode GET. Par exemple, si l'URL est `example.com?page=2`, alors `$_GET['page']` vaudra ***2***.

- **$_POST**

Contient les données envoyées via un **formulaire** en méthode **POST**. Par exemple, si un formulaire contient un champ **name**, alors `$_POST['name']` contiendra la valeur saisie.

- **$_REQUEST**

Contient les données envoyées via les méthodes GET, POST et COOKIE. Il regroupe les valeurs de `$_GET`, `$_POST` et `$_COOKIE`.

IV. Les variables prédéfinies en PHP

- **$_SERVER**

  Contient des informations sur le serveur et l'environnement d'exécution, comme l'adresse IP du client, le chemin du fichier script en cours d'exécution, etc.

  - ✓ `$_SERVER['PHP_SELF']` : Le nom du fichier du script en cours d'exécution.
  - ✓ `$_SERVER['HTTP_HOST']` : Le nom de domaine de l'hôte.

- **$_SESSION**

Contient les variables de session, qui persistent entre les requêtes. C'est utilisé pour conserver les données utilisateur entre les pages.

- **$_COOKIE**

Contient les données des cookies, qui sont des petites données stockées sur le navigateur de l'utilisateur.

- **$_FILES**

Contient des informations sur les fichiers téléchargés via un formulaire HTML, tels que le nom du fichier, son type, sa taille, etc.

IV. Les variables prédéfinies en PHP

- **$_ENV**
  Contient des informations d'environnement, comme les variables d'environnement du système d'exploitation.

- **$_GLOBALS**
  Une variable superglobale qui contient toutes les autres variables globales du script, accessibles n'importe où.

Ces superglobales sont très utiles pour manipuler des données venant de l'utilisateur, gérer les sessions, les téléchargements de fichiers, et obtenir des informations sur l'environnement d'exécution du script.

V. Les constantes en PHP

❑ Les constantes personnalisées en PHP

Pour définir une fonction personnalisée en PHP, on utilise la fonction define() qui a la syntaxe suivante:

```
Boolean define(String name, mixed value,[bool case_insensitive=false])
```

Exemple

```
<?php
// Définir une constante
define("SITE_NAME", "MonSiteWeb");
// Utiliser la constante
echo "Bienvenue sur " . SITE_NAME;
?>
```

Remarque:

✓ Pour vérifier l'existence d'une constante, on utilise la fonction defined() qui a la syntaxe:

```
Boolean defined(String name)
```

✓ Redéfinir une constante existante est interdit en PHP.

V. Les constantes en PHP

❑ Les constantes prédéfinies en PHP

PHP fournit plusieurs constantes prédéfinies qui sont très utiles dans diverses situations. Voici quelques-unes des constantes prédéfinies les plus courantes.

- **PHP_VERSION:** Contient la version actuelle de PHP en cours d'utilisation.
- **PHP_OS:** Contient le nom du système d'exploitation sous lequel PHP est en cours d'exécution.
- **__FILE__**: Contient le chemin complet et le nom du fichier PHP en cours d'exécution.
- **__DIR__**: Contient le répertoire du fichier PHP en cours d'exécution
- **__LINE__**: Contient le numéro de ligne actuel dans le fichier
- **__FUNCTION__**: Contient le nom de la fonction dans laquelle elle est utilisée.
- **__CLASS__**: Contient le nom de la classe dans laquelle elle est utilisée.

VI. Les opérateurs en PHP

- Les opérateurs permettent de manipuler les variables et les valeurs.

- PHP divise les opérateurs dans les groupes suivants :
  - Opérateurs arithmétiques
  - Opérateurs d'affectation
  - Opérateurs de comparaison
  - Opérateurs d'incrémentation/décrémentation
  - Opérateurs logiques
  - Opérateurs de chaîne
  - Opérateurs de tableau

VI. Les opérateurs en PHP

❑ Opérateurs arithmétiques en PHP

Opérateur	Opération	Exemple	Description
+	Addition	$x + $y	Somme de $x et $y
-	Soustraction	$x - $y	Difference entre $x et $y
*	Multiplication	$x * $y	Produit de $x et $y
/	Division	$x / $y	Quotient de $x et $y
%	Modulos	$x % $y	Le reste de la division entre $x et $y
**	Exponentiation	$x ** $y	$x exponentielle $y

VI. Les opérateurs en PHP

❑ Opérateurs d'affectation PHP

L'opérateur d'affectation de base en PHP est « = ». Cela signifie que l'opérande de gauche est défini sur la valeur de l'expression d'affectation de droite.

Affectation	Similaire à...	Description
$x = $y	$x = $y	L'opérande de gauche est défini sur la valeur de l'expression de droite
$x += $y	$x = $x +$y	Addition
$x -= $y	$x = $x - $y	Soustraction
$x *= $y	$x = $x * $y	Multiplication
$x /= $y	$x = $x / $y	Division
$x %= $y	$x = $x % $y	Modulos

VI. Les opérateurs en PHP

❑ Opérateurs de comparaison PHP

Opérateur	nom	Exemple	Résultat
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

VI. Les opérateurs en PHP

❑ Opérateurs d'incrémentation/décrémentation PHP

Les opérateurs d'incrémentation/ décrémentation PHP sont utilisés pour incrémenter/ décrémenter la valeur d'une variable.

Operator	Same as...	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

VI. Les opérateurs en PHP

❑ Opérateurs logiques PHP

Les opérateurs logiques PHP sont utilisés pour combiner des instructions conditionnelles.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

VI. Les opérateurs en PHP

❑ Opérateurs de chaîne

PHP dispose de deux opérateurs spécialement conçus pour les chaînes.

Opérateur	Nom	Example	Resultat
.	Concatenation	$txt1 . $txt2	Concatenation de $txt1 et $txt2
.=	Concatenation et assignment	$txt1 .= $txt2	Ajoute $txt2 à $txt1

VI. Les opérateurs en PHP

❑ Opérateurs de tableau

Les opérateurs de tableau PHP sont utilisés pour comparer des tableaux.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y