
PHP
Bases
1

Php & HTML

```
<!DOCTYPE html>
<html lang="fr">
	<head>
		<title>Première page</title>
		<meta charset="utf-8"/>
	</head>
	<body>
		Une page vide (ou presque)
	</body>
</html>
```

Php & HTML

```
<!DOCTYPE html>
<html lang="fr">
	<head>
		<title>Première page</title>
		<meta charset="utf-8"/>
	</head>
	<body>
<?php   echo 'Une page vide (ou presque)'; ?>
	</body>
</html>
```

Php & HTML

```
<?php echo '<!DOCTYPE html>
<html lang="fr">
	<head>
		<title>Première page</title>
		<meta charset="utf-8"/>
	</head>
	<body>
		Une page vide (ou presque)
	</body>
</html>'; ?>
```

Php & HTML

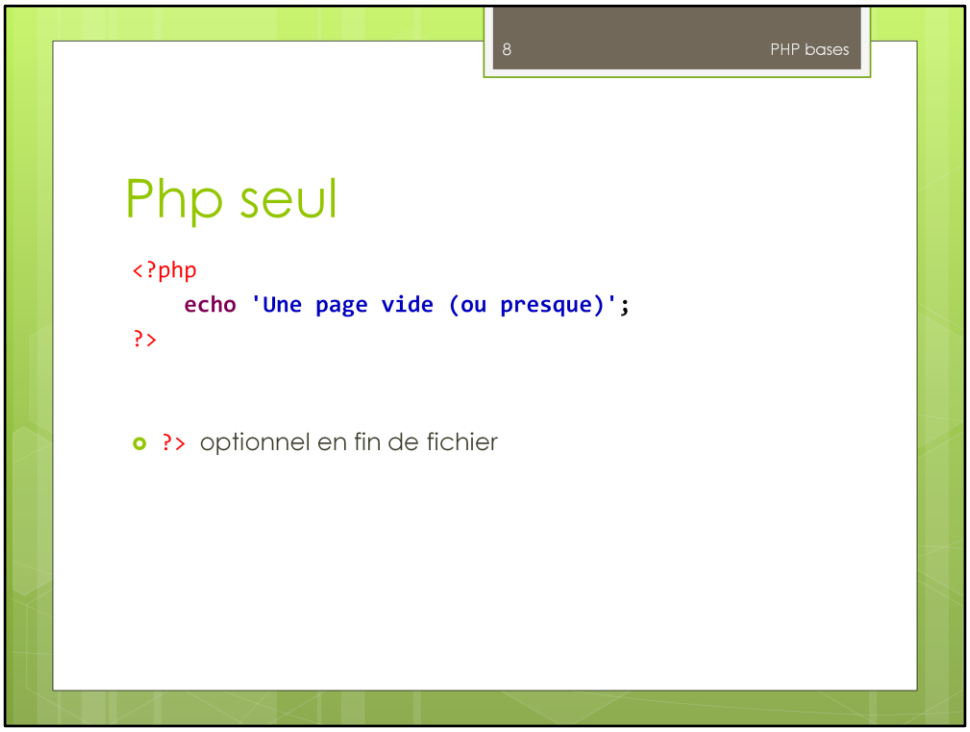
```
<!DOCTYPE html>
<ht<?php echo 'm'; ?>l lang="fr">
	<head>
		<title>Première page</title>
		<meta charset="utf-8"/>
	</head>
	<body>
		Une page vide (ou presque)
	</body>
</html>
```

Php & CSS

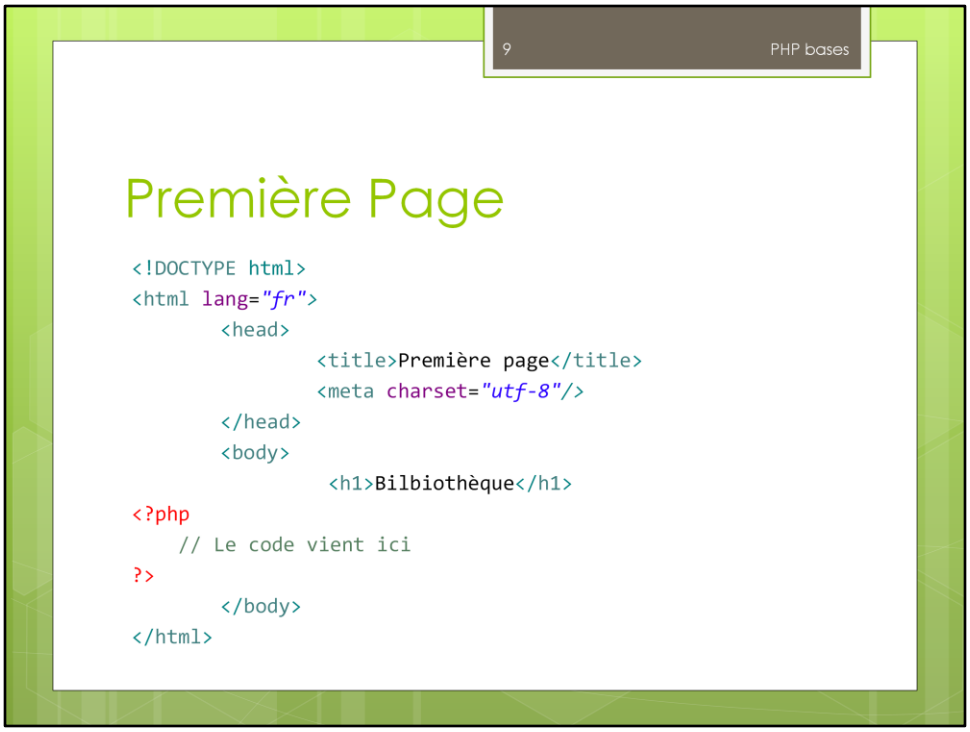
```
@CHARSET "ISO-8859-1";

body {
        background-color: 'lightgrey';
}
```

Php & CSS

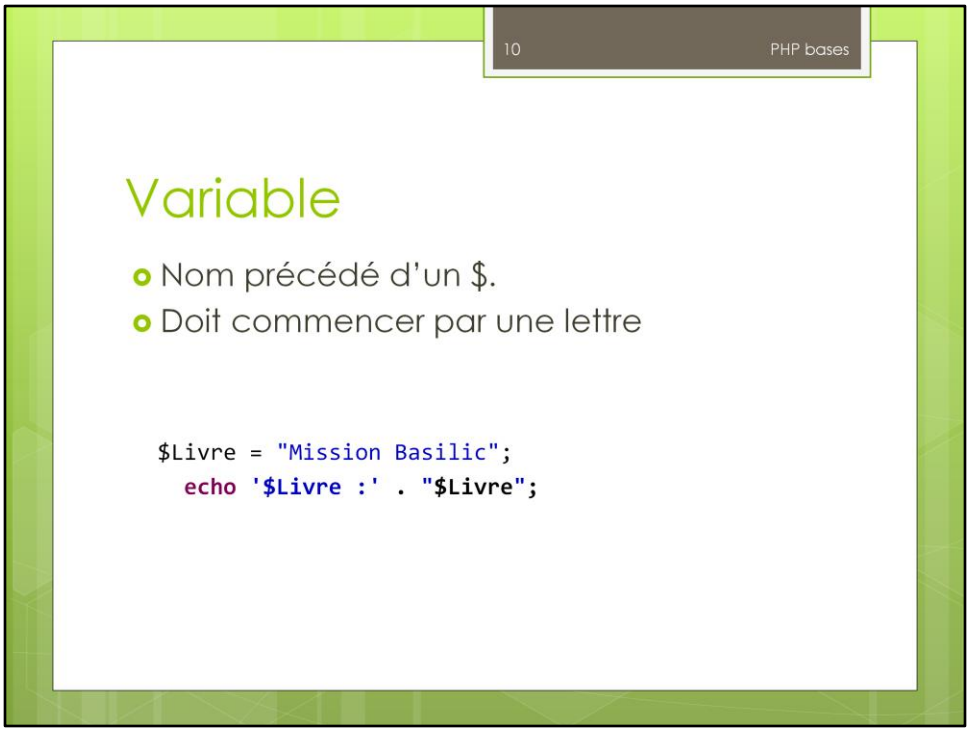
```
@CHARSET "ISO-8859-1";

body {
<?php echo 'Une background-color: \'lightgrey\'; '; ?>
}
```



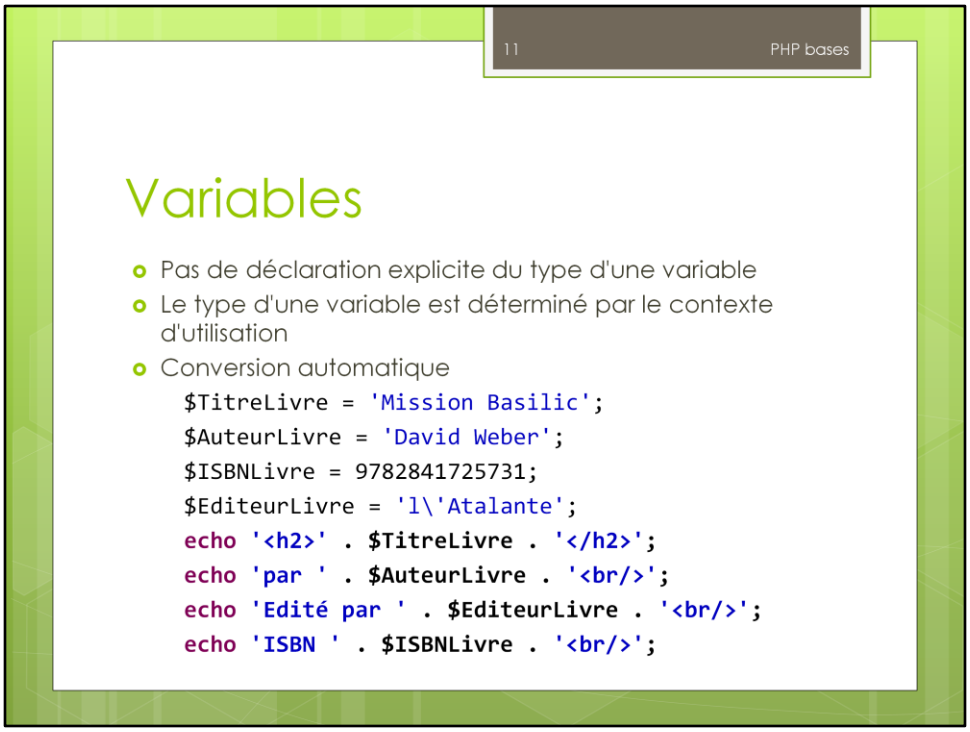
?> optionnel car en fin de fichier, le script est considéré comme terminé. S'il est présent les caractères suivants sont considérés comme du contenu réel. Cela peut rajouter des caractères gênants dans une page HTML.
De même avant le <?php en début de fichier, un caractère non imprimable (BOM notamment) peut poser de gros soucis notamment sur l'envoi des entêtes de fichiers (headers)



On gardera cette page comme contour de tous les scripts suivants, sauf énoncé autrement.



Le nom des variable peut être décrit par l'expression régulière : [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*



Ici, $ISBNLivre est un entier, mais il devient un string quand il doit être concaténé pour echo.

Booléen

- Peut prendre la valeur `true` ou `false`
- Valeurs considérées comme fausses :
  - Le booléen `false` lui-même
  - L'entier `0`
  - Le nombre décimal `0.0`
  - La chaîne de caractères vide et la chaîne de caractères `"0"`
  - Le tableau vide
  - La valeur spéciale `NULL`
  - **Toutes** les autres valeurs sont considérées comme `true`
- Opérateurs de conversion : `(bool)`, `(boolean)`

Entiers

- Un type entier est un entier naturel
- Les entiers peuvent être spécifiés en base 10, 8 ou 16
- Les entiers peuvent être optionnellement précédés par le signe plus ou moins (+ ou –) :

```
$a = 1234; // Nombre entier en base 10.
$a = -123; // Nombre entier négatif.
$a = 0123; // Nombre entier en base 8, octale (83 en base 10).
$a = 0x12; // Nombre entier en base 16, hexadécimale (18 en base 10).
```

- Conversion :
  - Opérateurs de conversion : `(int)`, `(integer)`, `intval()`
  - Lors de la conversion entre un nombre décimal et un entier, le nombre sera arrondi :
    - À la valeur inférieure s'il est positif, à la supérieure s'il est négatif

Décimaux

- Aussi appelés :
  - *Double*
  - *Float*
  - Nombres réels :

```
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
```

- Conversion en nombre décimal :
  - Opérateurs de conversion : `(real)`, `(double)` et `(float)`
  - Fonctions `floatval()` et `doubleval()`

Chaîne de caractères

- Séquence de caractères
- Une chaîne peut être spécifiée de trois manières :
  - Guillemets doubles "
  - Guillemets simples '
  - Syntaxe *heredoc* (plus de détails : documentation PHP) :

```
$a = "hello world";
$b = '$a : ';
$c = <<<EOT
    Hello world
EOT;
```

Opérateurs unaires

- Signes : +, –
- Négation : !
- Incrémentation/décrémentation (pré et post) : ++, --
- Conversion : `(int)`, `(bool)`, etc.
- Clonage : `clone`
- Contrôle d'erreur : `@`

```
$a = -0.5;
--$a;
$b = (bool) $a;
$c = !$b;
```

Opérateurs unaires

- Arithmétiques : +, -, *, /, %
- Concaténation de chaîne : .
- Affectations : =, +=, .=, *=, etc.
- Comparaisons : ==, >, >=, <, !=, <>, ===, !==, etc.
- Logiques : &&, ||, xor, etc.
- Binaires : &, |, ~, ^, <<, >>

```
$a = 5; $b = 10;
$c = '$a : ' . $a;  // Pas de caractère '+' !
$d = ($a == $b) && ((0 != $b % $a) || ($b < 20));
```

Tableaux

```
    $Avis1 = array (4, 'Bob', 'Super', 'Il est super,
surtout la page 17');
    $Avis2 = array ('note' => 2, 'auteur' => 'Bob',
'titre' => 'En fait non', 'avis' => 'j\'ai lu la page
57, elle est pourrie');
    $Avis3[] = 3;
    $Avis3['auteur'] = 'Bob';
    $Avis3[] = 'Oh, finalement';
    $Avis3['avis'] = 'c\'est pas si mal';
```

- unset ()
- count ()

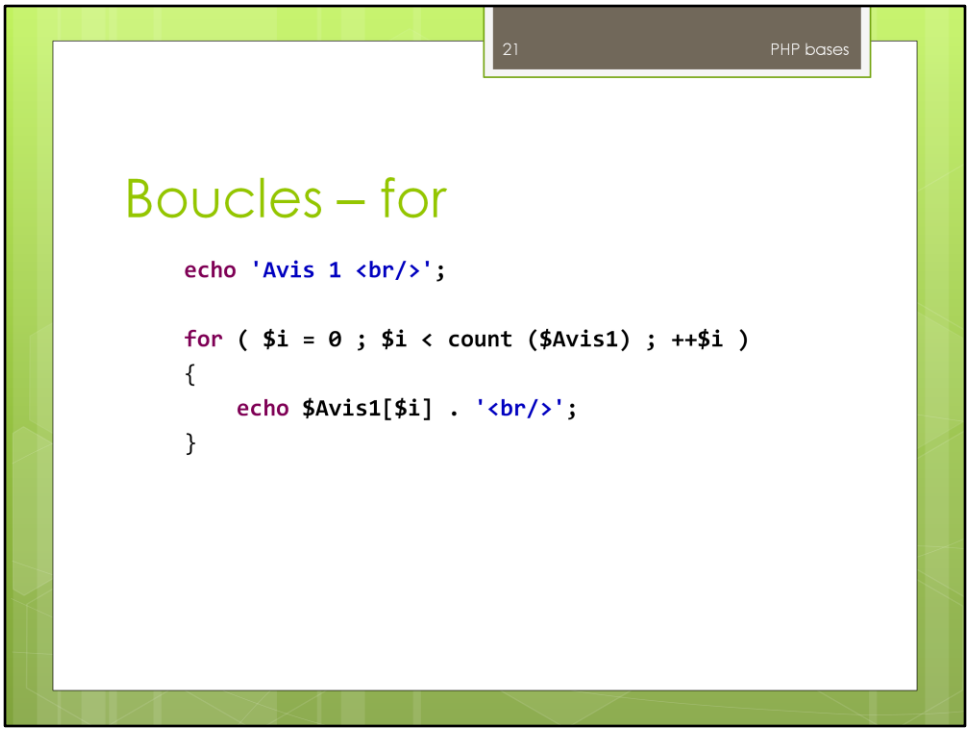
Unset : désallouer
Count : compter le nombre d'éléments.

Boucles - while

```
echo 'Avis 1 <br/>';
$i = 0;

while ( $i < count ($Avis1) )
{
    echo $Avis1[$i] . '<br/>';
    ++$i;
}
```

Boucles – do…while

```
echo 'Avis 1 <br/>';
$i = 0;

do
{
    echo $Avis1[$i] . '<br/>';
    ++$i;
}
while ( $i < count ($Avis1));
```



Pour $Avis1 cela est formidable, mais pour $Avis2 et $Avis3, où les index sont soit associatifs, soit mélangés. Du coup, le parcours par index est impossible.

Boucles – foreach

```php
echo 'Avis 1 <br/>';
foreach ( $Avis1 as $Key => $Value )
{
    echo $Key . ' ' . $Value . '<br/>';
}

echo 'Avis 3 <br/>';
foreach ( $Avis3 as $Value )
{
    echo $Value . '<br/>';
}
```

Boucles

```
$Avis = array ($Avis1, $Avis2, $Avis3);

echo 'Avis<table border=1>';
foreach ( $Avis as $a )
{
    echo '<tr>';
    foreach ( $a as $Key => $Value )
        echo '<td>' . $Value . '</td>';
    echo '</tr>';
}
echo '</table>';
```

Conditions – if, else if, else

```php
echo 'Avis<table border=1>';
foreach ( $Avis as $a )
{
    if ( $a['note'] >= 4 )
        ++$a['note'];
    else if ( $a['note'] > 2 )
        $a['avis'] = 'Avis supprimé';
    else
        $a['avis'] = 'Avis modéré';

    echo '<tr>';
    foreach ( $a as $Key => $Value )
        echo '<td>' . $Value . '</td>';
    echo '</tr>';
}
echo '</table>';
```

Conditions – switch, case

```
echo 'Avis<table border=1>';
foreach ( $Avis as $a ) {
    switch ( $a['note'] ) {
        case 2 :
            $a['avis'] = 'Avis supprimé';
            break;
        case 4 :
            ++$a['note'];
            break;
        default :
            $a['avis'] = 'Avis modéré';
    }
    echo '<tr>';
    foreach ( $a as $Key => $Value )
        echo '<td>' . $Value . '</td>';
    echo '</tr>';
}
echo '</table>';
```

Fonctions – Appel

```
$Avis[] = array ('note' => 4, 'auteur' => 'Bob',
'titre' => 'Super',
'avis'  => 'Il est super, surtout la page 17');
$Avis[] = array ('note' => 2, 'auteur' => 'Bob',
'titre' => 'En fait non',
'avis'  => 'j\'ai lu la page 57, elle est pourrie');
$Avis[] = array ('note' => 3, 'auteur' => 'Bob',
'titre' => 'Oh, finalement',
'avis'  => 'c\'est pas si mal');
```

Fonctions – Création

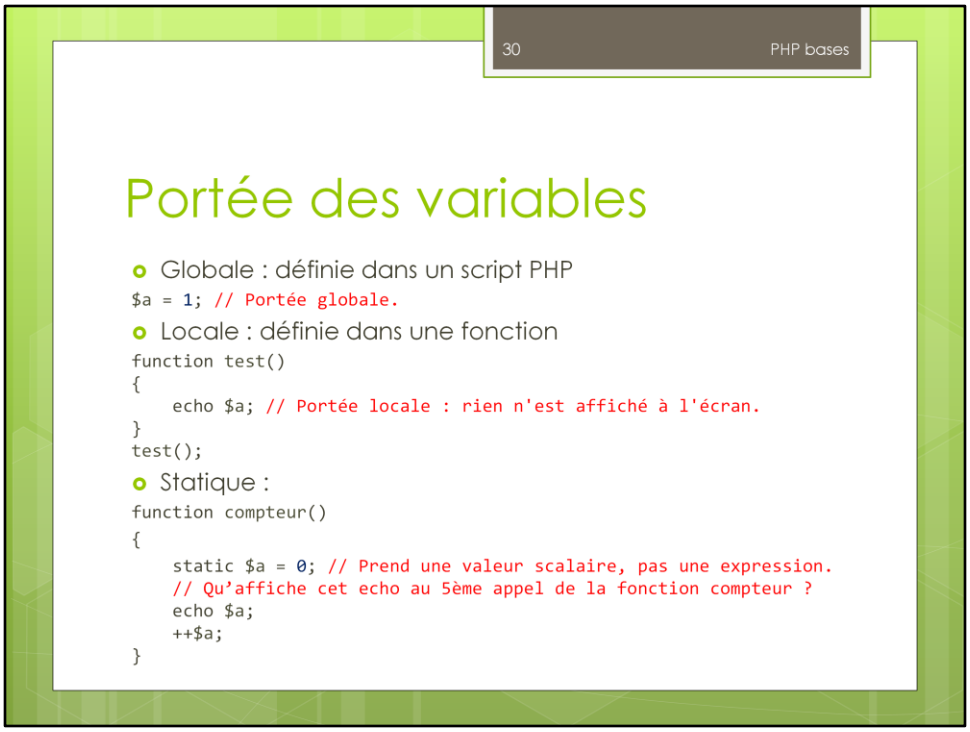
```
function AfficherLivre ()
{
    $TitreLivre = 'Mission Basilic';
 . . .
    echo '</table>';
} // AfficherLivre ()


AfficherLivre ();
```

Fonctions – Arguments

```php
function AfficherAvis ( $Avis )
{
    echo 'Avis<table border=1>';
 . . .
    echo '</table>';
} // AfficherAvis ( $Avis )


AfficherAvis ( $Avis );
```

Fonctions – Retour

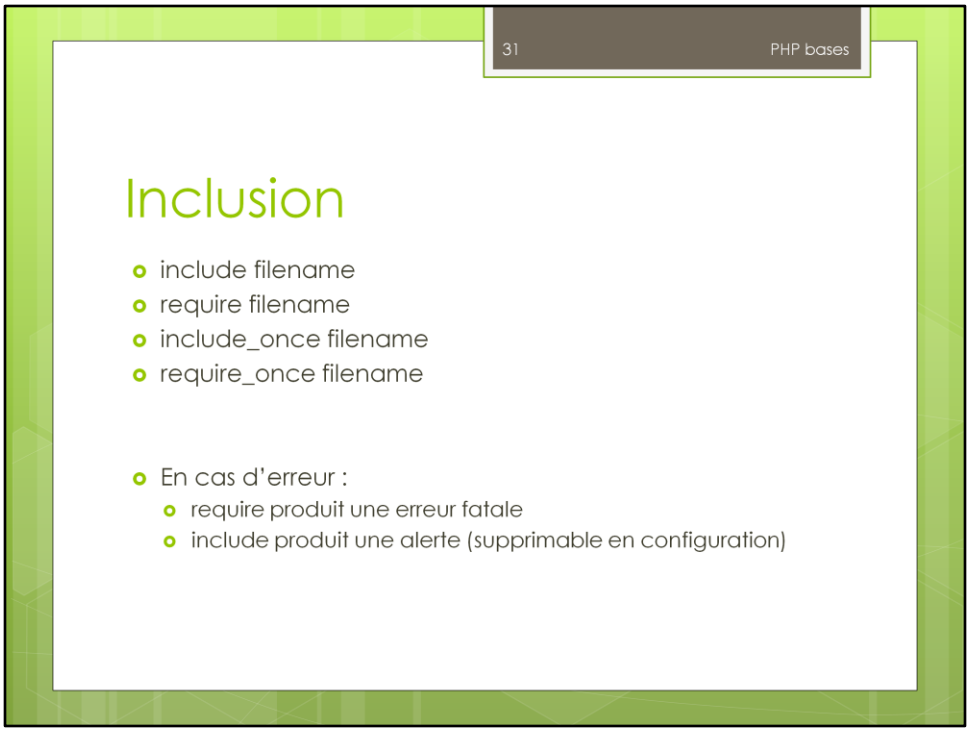
```php
function AfficherAvis ( $Avis )
{
    $NbAvis = 0;
    echo 'Avis<table border=1>';
 . . .
            case 4 :
                ++$a['note'];
                ++$NbAvis;
                break;
 . . .
    echo '</table>';
    return $NbAvis;
} // AfficherAvis ( $Avis )


$NbAvisReels = AfficherAvis ( $Avis );
```



Vous pouvez noter que ce concept diffère un petit peu du langage C dans lequel une variable globale est automatiquement accessible dans les fonctions, à moins d'être redéfinie localement dans la fonction. Cela peut poser des problèmes si vous redéfinissez des variables globales localement. En PHP, une variable globale doit être déclarée à l'intérieur de chaque fonction afin de pouvoir être utilisée dans cette fonction.

La valeur d'affectation d'une variable statique doit être obligatoirement scalaire, pas une expression.

Qu'affiche cet echo au 5ème appel de la fonction compteur ? 4 !

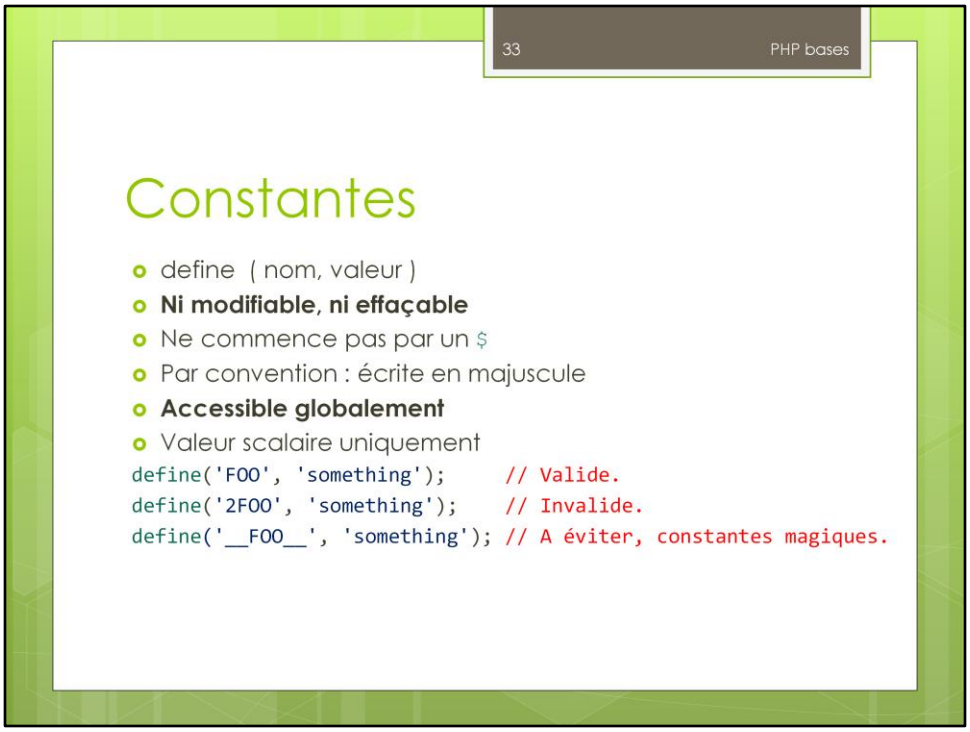


Attention les version _once sont nettement plus lentes.

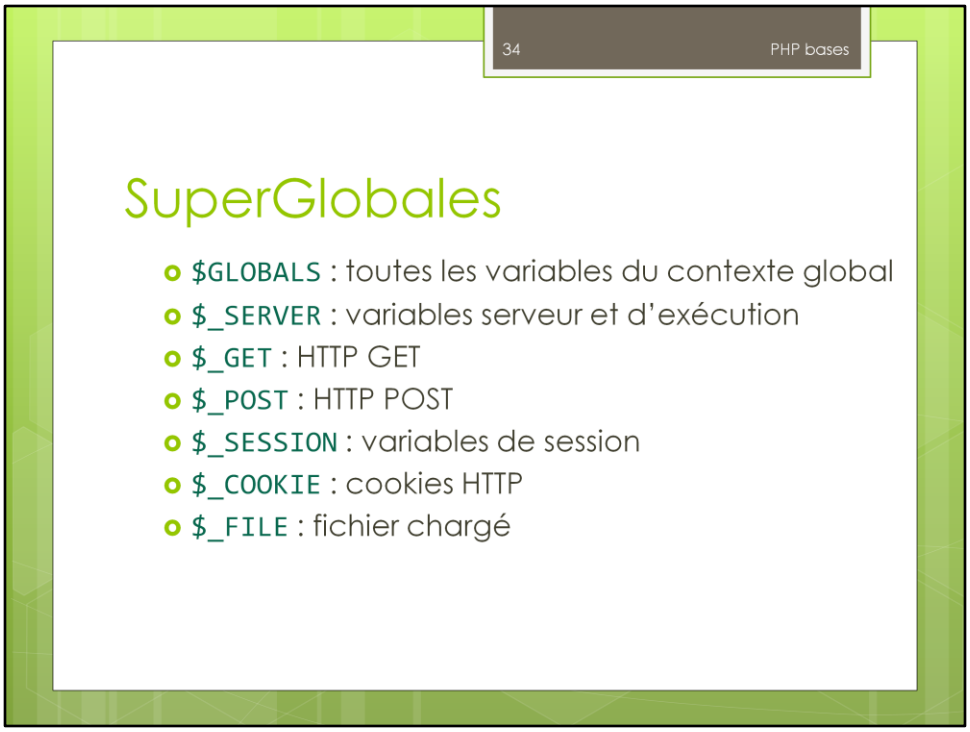
Inclusion

```
<!DOCTYPE html>
<html lang="fr">
        <head>
                <title>Première page</title>
                <meta charset="utf-8"/>
        </head>
        <body>
                <h1>Bilbiothèque</h1>
<?php
        require '__Fonctions.php';
        afficherLivre ();
?>
        </body>
</html>
```

Attention les version _once sont nettement plus lentes.



__ … __ correspond généralement à des noms de constantes réservées dites constantes magiques : __FILE__, __LINE__, __DIR__, __FUNCTION__, __CLASS__, __TRAIT__, __METHOD__, __NAMESPACE__



Superglobales : Disponibles partout sans besoin de déclarer global $_VAR. Il s'agit de tableaux associatifs.

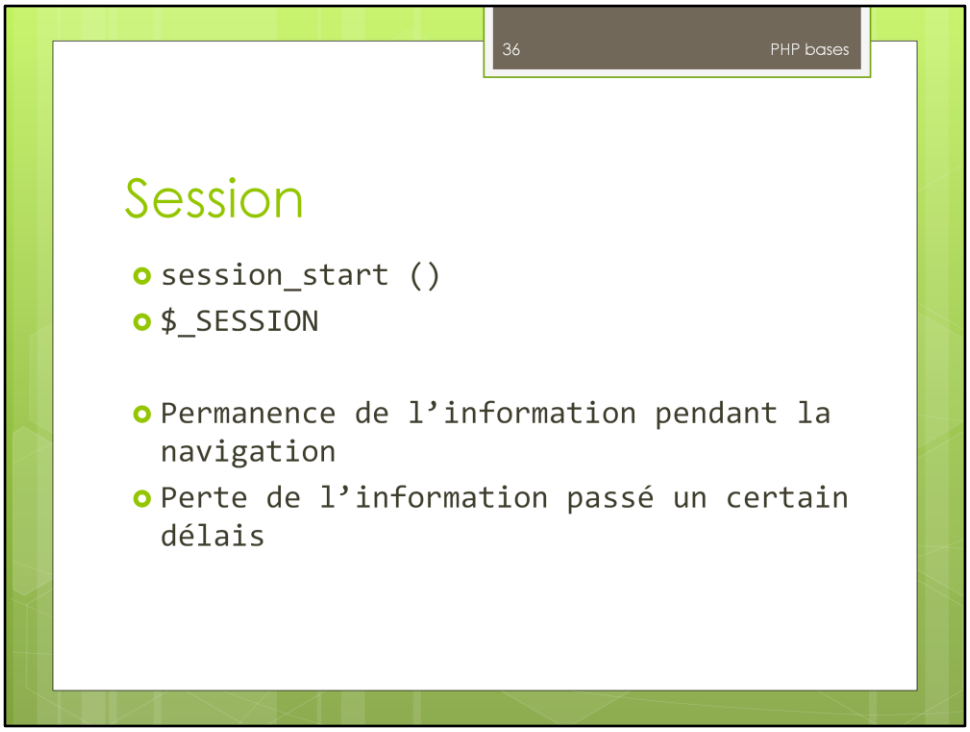
$_SERVER - Variables de serveur et d'exécution :
- $_SERVER['SERVER_NAME'] : nom du serveur hôte sur lequel est exécuté le script. ;
- $_SERVER['SERVER_PROTOCOL'] : nom et révision du protocole de communication : HTTP/1.0 ;
- $_SERVER['REQUEST_METHOD'] : méthode de requête utilisée pour accéder à la page : 'GET','HEAD', 'POST' ou 'PUT'.

$_FILE : ce tableau contient les variables fournies par le navigateur lors d'un chargement de fichier par le client.

Exercice : Ajouter au fichier précédent l'affichage du contenu de $_GET, $_POST, $_SERVER et $_SESSION

SuperGlobales

```php
foreach ( $_GET as $Key => $Value )
    echo 'GET : ' . $Key . ' => ' . $Value . '<br/>';

foreach ( $_POST as $Key => $Value )
    echo 'POST : ' . $Key . ' => ' . $Value . '<br/>';

foreach ( $_SERVER as $Key => $Value )
    echo 'SERVER : ' . $Key . ' => ' . $Value . '<br/>';

foreach ( $_SESSION as $Key => $Value )
    echo 'SESSION : ' . $Key . ' => ' . $Value . '<br/>';
```



Transformer le fichier Formulaire.html en Formulaire.php
Ajouter un session_start ();
Ajouter une donnée dans $_SESSION

Dans le fichier de travail,
Ajouter un session_start (); au début de la première zone php.

Crédits

Auteur

- Christophe Delagarde

Sources

- Site officiel PHP (http://www.php.net/)
- Mickaël Martin Nevot (http://www.mickael-martin-nevot.com/)
- W3Schools (http://www.w3schools.com/php/default.asp)
- Open Classrooms (https://openclassrooms.com/courses?q=php)