

Développer un site web dynamique en PHP

Module 2 – Les bases du langage



Objectifs

- Découvrir le langage PHP
- Appréhender la syntaxe pour les bases du langage
- Savoir utiliser les fonctions proposées dans PHP
- Utiliser des expressions rationnelles

La documentation

- <http://php.net/manual/fr/index.php>

The screenshot shows the PHP documentation website. The top navigation bar includes links for 'php', 'Downloads', 'Documentation', 'Get Involved', and 'Help', along with a search box. A green banner for 'LaravelConf Taiwan 2017' is visible. The breadcrumb trail indicates the current location: 'Manuel PHP > Référence des fonctions > Traitement du texte > Chaîne de caractères > Fonctions sur les chaînes de caractères'. The main content area features a 'Change language' dropdown set to 'French', with links to 'Edit' and 'Report a Bug'. The function 'echo' is highlighted, with a note that it is available in PHP 4, PHP 5, and PHP 7. The description states: 'echo — Affiche une chaîne de caractères'. A sidebar on the right lists other functions in the 'Fonctions sur les chaînes de caractères' category: 'addslashes', 'bin2hex', 'chop', 'chr', and 'chunk_split'.

php Downloads Documentation Get Involved Help Search

LaravelConf Taiwan 2017

Manuel PHP > Référence des fonctions > Traitement du texte > Chaîne de caractères
> Fonctions sur les chaînes de caractères

« crypt explode »

Change language: French Edit Report a Bug

echo

(PHP 4, PHP 5, PHP 7)
echo — Affiche une chaîne de caractères

Fonctions sur les chaînes de caractères

- addslashes
- bin2hex
- chop
- chr
- chunk_split

Les balises PHP

- `<?php ... ?>`
 - Exemple :

```
<?php
echo '<h1>Hello World !</h1>';
echo '<p>Bonjour ' . 'Valérie ' . ' !</p>';
?>
```
- `<?= ... ?>`
 - Exemple :
`<?= 'Bonjour Mélanie !' ?>` équivalent à `<?php echo 'Bonjour Mélanie !'; ?>`
- `<?php ...`
 - Si le fichier ne contient que des instructions PHP

Les commentaires

- // ou # jusqu'à la fin de la ligne
- /* jusqu'à */

- Exemple :

```
<?php
```

```
// Eventuellement cette syntaxe
```

```
echo('Bonjour Olivier !'); /* syntaxe comme si écho était une fonction */
```

```
# autre commentaire
```

Les instructions

- Séparées par ;

- Exemple :

```
<?php  
echo '<h1>Hello World !</h1>';  
echo '<p>Bonjour ' . 'Valérie ' . ' !</p>';
```

Les bases du langage

Premier lancement de l'IDE NetBeans

Démonstration



Les variables – les règles de nommage

- Commence par \$
- Distingue minuscules et majuscules
- Pas de chiffre juste après le \$
- Caractères autorisés : lettres, chiffres et _



\$Nom ≠ \$nom

Les variables – la portée

- Exemple :

```
<?php
$titre_page = 'Titre de la page ...'; // titre de la page
?>
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>
      <?php echo $titre_page; ?>
    </title>
    <link href="../../../style/style.css" rel="stylesheet">
  </head>
```

...



Pas de déclaration
des variables

Les variables – l'assignation par valeur ou référence

- Assignation par valeur

- Exemple :

```
$prenom1 = 'Maxime';  
$prenom2 = $prenom1;  
$prenom1 = 'Vincent';  
echo '$prenom1 = ' . $prenom1 . ' et $prenom2 = ' . $prenom2 . '<br>;'
```



\$prenom1 = Vincent et \$prenom2 = Maxime

- Assignation par référence &

- Exemple :

```
$nom1 = 'Duchemin';  
$nom2 = &$nom1;  
$nom2 = 'Lepont';  
echo '$nom1 = ' . $nom1 . ' et $nom2 = ' . $nom2 . '<br>;'
```



\$nom1 = Lepont et \$nom2 = Lepont

Les variables dynamiques

- Nom de variable dans une variable \$\$

```
$une_variable = 10;  
$nom_variable = 'une_variable';  
echo '<code>$une_variable</code> = ' . $une_variable . '<br>';  
echo '<code>$nom_variable</code> = ' . $nom_variable . '<br>';  
echo '<code>$$nom_variable</code> = ' . $$nom_variable . '<br>';
```



```
$une_variable = 10  
$nom_variable = une_variable  
$$nom_variable = 10
```

~~\$\$nom_variable~~

Variables – quelques fonctions utiles

- Affichage des informations d'une variable
 - `var_dump()`
- Test d'existence d'une variable
 - `isset()`
- Test si une variable est vide
 - `empty()`
- Suppression d'une variable
 - `unset()`

Variables – quelques fonctions utiles

```
$une_variable = 10;  
var_dump($une_variable);  
echo '<p>La variable <code>$une_variable</code> est-elle définie ? <p>';  
var_dump(isset($une_variable));  
echo '<p>La variable <code>$une_variable</code> est-elle vide ?</p>';  
var_dump(empty($une_variable));  
echo '<p>Suppression de la variable <code>$une_variable</code><br>';  
unset($une_variable);  
echo 'La variable <code>$une_variable</code> est-elle définie ? </p>';  
var_dump(isset($une_variable));
```



```
int 10  
La variable $une_variable est-elle définie ?  
boolean true  
La variable $une_variable est-elle vide ?  
boolean false  
Suppression de la variable $une_variable  
La variable $une_variable est-elle définie ?  
boolean false
```

Les variables – le changement de type

```
$nom = 'Pablo';  
var_dump($nom);  
$nom = 3;  
var_dump($nom);
```



```
string 'Pablo' (length=5)  
int 3
```



Une variable peut
changer de type
au cours de son existence

Les variables – le transtypage

- Le transtypage explicite

```
$x = '1abc';  
settype($x, 'integer');  
echo '1abc converti en entier ' . $x . '<br>';  
$y = 'abc1';  
settype($y, 'integer');  
echo 'abc converti en entier ' . $y . '<br>';
```



1abc converti en entier 1
abc converti en entier 0

Les variables – le transtypage

- Le transtypage implicite

```
$nom = '2 fingers up';  
var_dump($nom);  
$nom = 5 * $nom;  
var_dump($nom);
```



```
string '2 fingers up' (length=12)  
int 10
```



Pas d'erreur
ni d'avertissement

Les constantes – les règles de nommage

- Règles de nommage
 - Pas de caractère \$
 - Ne commence pas par un chiffre
 - Caractères autorisés : lettres, chiffres et _
 - Distingue minuscules et majuscules
- Convention de nommage
 - Tout en majuscule

UNE_AUTRE_CONSTANTE

Les constantes – la déclaration

- Déclaration
 - `const`
 - `define()`
- Test d'existence
 - `defined()`

Les constantes – un exemple d'utilisation

```
define('CONSTANTE', 'php en version ' . phpversion());  
const UNE_AUTRE_CONSTANTE = 'PHP 7';  
echo '<p><code>CONSTANTE</code> = ' . CONSTANTE . '<br></p>';  
echo '<code>UNE_AUTRE_CONSTANTE</code> = ' . UNE_AUTRE_CONSTANTE . '<br>';  
var_dump(defined('UNE_AUTRE_CONSTANTE'));
```



CONSTANTE = php en version 7.0.10
UNE_AUTRE_CONSTANTE = PHP 7
boolean true

Le type `boolean`

- 2 valeurs possibles (`true` et `false`)
- Insensible à la casse : `true` = `TRUE` = `True`...
- Opérateurs booléens :
 - Et `AND` `&&`
 - Ou `OR` `||`
 - Ou exclusif `XOR`
 - Non `!`
- Équivalence à la valeur `False`
 - `0` (entier ou réel)
 - `''` (chaîne vide) ou `'0'`
 - `[]` (tableau vide)
 - `null`

Les bases du langage

Les types `int` et `float`

- Entier
 - `int`
- Nombre à virgule
 - `float` (uniquement)

Les types `int` et `float`

- Opérateurs arithmétiques

- | | | |
|---------------------|----|----|
| • Addition | + | += |
| • Soustraction | - | -= |
| • Multiplication | * | *= |
| • Division | / | /= |
| • Modulo | % | %= |
| • Exponentielle | ** | |
| • Incrémentation | ++ | |
| • Décrémententation | -- | |

Le type `string`

- Simples cotes `' '`
 - Pas d'interprétation
 - Caractère d'échappement `\`
 - `\\` pour le back slash, `\'` pour l'apostrophe
- Doubles cotes `" "`
 - Interprétation des caractères de mise en forme (`\n`, `\r`, `\t`...)
 - Interprétation des variables
 - Caractère d'échappement `\`
 - `\\` pour le back slash, `\"` pour le guillemet, `\$` pour le dollar
- Opérateur de concaténation `.`

Opérateurs de comparaison

- Égal (valeur uniquement) ==
- Identique (valeur et type) ===
- Différent (valeur uniquement) != <>
- Différent (valeur et type) !==
- Supérieur >
- Inférieur <
- Supérieur ou égal >=
- Inférieur ou égal <=

Le tableau numérique (indices ordonnés consécutifs)

Indice	Valeur
0	'zéro'
1	'un'
2	'deux'
3	'trois'

```
$n[] = 'zéro';  
$n[] = 'un';  
$n[] = 'deux';  
$n[] = 'trois';  
var_dump($n);
```



```
array (size=4)  
    0 => string 'zéro' (length=5)  
    1 => string 'un' (length=2)  
    2 => string 'deux' (length=4)  
    3 => string 'trois' (length=5)
```

Le tableau numérique (indices non ordonnés, non consécutifs)

Indice	Valeur
20	'vingt'
30	'trente'
10	'dix'

```
$m[20] = 'vingt';  
$m[30] = 'trente';  
$m[10] = 'dix';  
var_dump($m)
```



```
array (size=3)  
    20 => string 'vingt' (length=5)  
    30 => string 'trente' (length=6)  
    10 => string 'dix' (length=3)
```

Le tableau associatif

Indice	Valeur
'dix-sept'	17
'quarante-deux'	0x2A
'trois virgule cinq'	3.5

```
$a['dix-sept'] = 17;  
$a['quarante-deux'] = '0x2A';  
$a['trois virgule cinq'] = 3.5;  
var_dump($a);
```



```
array (size=3)  
    'dix-sept' => int 17  
    'quarante-deux' => string '0x2A' (length=4)  
    'trois virgule cinq' => float 3.5
```

Le tableau mixte

Indice	Valeur
0	'zéro'
1	'un'
5	'cinq'
6	'six'
'un'	1
7	'sept'
-1	'moins un'

```
$nb[] = 'zero';  
$nb[] = 'un';  
$nb[5] = 'cinq';  
$nb[] = 'six';  
$nb['un'] = 1;  
$nb[] = 'sept';  
$nb[-1] = 'moins un';
```



```
array (size=7)  
    0 => string 'zero' (length=4)  
    1 => string 'un' (length=2)  
    5 => string 'cinq' (length=4)  
    6 => string 'six' (length=3)  
    'un' => int 1  
    7 => string 'sept' (length=4)  
    -1 => string 'moins un' (length=8)
```

Le tableau multidimensionnel

Indice	Valeur	
'FRANCE'	Indice	Valeur
	0	'Paris'
	1	'Lyon'
	2	'Nantes'
'ITALIE'	Indice	Valeur
	0	'Rome'
	1	'Venise'

```
$villes_france[] = 'Paris';  
$villes_france[] = 'Lyon';  
$villes_france[] = 'Nantes';  
var_dump($villes_france);
```

```
$villes_italie[] = 'Rome';  
$villes_italie[] = 'Venise';  
var_dump($villes_italie);
```



```
array (size=3)  
    0 => string 'Paris' (length=5)  
    1 => string 'Lyon' (length=4)  
    2 => string 'Nantes' (length=6)  
array (size=2)  
    0 => string 'Rome' (length=4)  
    1 => string 'Venise' (length=6)
```

Le tableau multidimensionnel

Indice	Valeur	
'FRANCE'	Indice	Valeur
	0	'Paris'
	1	'Lyon'
	2	'Nantes'
'ITALIE'	Indice	Valeur
	0	'Rome'
	1	'Venise'

```
$villes['FRANCE'] = $villes_france;  
$villes['ITALIE'] = $villes_italie;  
var_dump($villes);
```



```
array (size=2)  
  FRANCE =>  
    array (size=3)  
      0 => string 'Paris' (length=5)  
      1 => string 'Lyon' (length=4)  
      2 => string 'Nantes' (length=6)  
  'ITALIE' =>  
    array (size=2)  
      0 => string 'Rome' (length=4)  
      1 => string 'Venise' (length=6)
```

La syntaxe des tableaux avec `array()` ou `[]`

Indice	Valeur
0	'zéro'
1	'un'
5	'cinq'
6	'six'
'un'	1
7	'sept'
-1	'moins un'

```
$nombres = array('zéro', 'un', 5 => 'cinq', 'six', 'un' => 1, 'sept', -1 => 'moins un');  
$nombres = ['zéro', 'un', 5 => 'cinq', 'six', 'un' => 1, 'sept', -1 => 'moins un'];
```

array (size=7)

```
0 => string 'zero' (length=4)  
1 => string 'un' (length=2)  
5 => string 'cinq' (length=4)  
6 => string 'six' (length=3)  
'un' => int 1  
7 => string 'sept' (length=4)  
-1 => string 'moins un' (length=8)
```



La syntaxe des tableaux avec `array()` ou `[]`

Indice	Valeur	
'FRANCE'	Indice	Valeur
	0	'Paris'
	1	'Lyon'
	2	'Nantes'
'ITALIE'	Indice	Valeur
	0	'Rome'
	1	'Venise'

```
$villes = array('FRANCE' => array('Paris', 'Lyon', 'Nantes'), 'ITALIE' => array('Rome', 'Venise'));  
$villes = ['FRANCE' => ['Paris', 'Lyon', 'Nantes'], 'ITALIE' => ['Rome', 'Venise']];
```



```
array (size=2)  
    FRANCE' =>  
        array (size=3)  
            0 => string 'Paris' (length=5)  
            1 => string 'Lyon' (length=4)  
            2 => string 'Nantes' (length=6)  
    'ITALIE' =>  
        array (size=2)  
            0 => string 'Rome' (length=4)  
            1 => string 'Venise' (length=6)
```


L'accès aux éléments du tableau

```
echo '<code>$nombres[1]</code>' = '. $nombres[1] . '<br>';  
echo '<code>$nombres['un']</code>' = '. $nombres['un'] . '<br>';  
  
echo '<code>$villes['FRANCE']</code>' = ' . $villes['FRANCE']<br>';  
echo '<code>$villes['ITALIE']</code>' = ' . $villes['ITALIE']<br>';
```



```
$nombres[1] = un  
$nombres['un'] = 1  
$villes['FRANCE'][0] = Paris  
$villes['ITALIE'][1] = Venise
```

Les tableaux – quelques fonctions utiles

- Création d'un tableau à partir d'une chaîne de caractères

- explode()

```
$liste = 'bleu, vert, orange, rouge, blanc';  
var_dump($liste);
```

```
$couleur = explode(',', $liste);  
var_dump($couleur);
```



```
string 'bleu, vert, orange, rouge, blanc'  
(length=32)  
array (size=5)  
    0 => string 'bleu' (length=4)  
    1 => string 'vert' (length=4)  
    2 => string 'orange' (length=6)  
    3 => string 'rouge' (length=5)  
    4 => string 'blanc' (length=5)
```

- Création d'une chaîne de caractères à partir d'un tableau

- implode()

```
var_dump(implode('|', $couleur));
```



```
string 'bleu|vert|orange|rouge|blanc' (length=28)
```

La structure de contrôle `if`

- Forme instruction unique

```
if ($a > $b)  
    echo 'a est plus grand que b';
```

- Forme bloc

```
if ($a > $b) {  
    echo 'a est plus grand que b';  
    $b = $a;  
}
```

La structure de contrôle `if else`

- Forme instruction unique

```
if ($a > $b)
    echo 'a est plus grand que b';
else
    echo 'a est plus petit ou égal à b';
```

- Forme bloc

```
if ($a > $b) {
    echo 'a est plus grand que b';
    $b = $a;
} else {
    echo 'a est plus petit ou égal à b';
}
```

La structure de contrôle `if elseif else`

```
if ($a > $b) {  
    echo 'a est plus grand que b';  
} elseif ($a == $b) {  
    echo 'a est égal à b';  
} else {  
    echo 'a est plus petit que b';  
}
```

La structure de contrôle `if: endif;`

- Forme alternative

```
<?php
if ($a > $b) :
    echo '<p>a est plus grand que b</p>';
endif;
?>
```

```
<?php if ($a > $b): ?>
<p>a est plus grand que b</p>
<?php endif; ?>
```

- Équivalent à

```
<?php
if ($a > $b)
    echo '<p>a est plus grand que b</p>';
?>
```

La structure de contrôle if: elseif: else: endif;

- Forme alternative

```
<?php
if ($a > $b):
    echo '<p>a est plus grand que b</p>';
elseif ($a == $b):
    echo '<p>a est égal à b</p>';
else:
    echo '<p>a est plus petit que b</p>';
endif;
?>
```

```
<?php if ($a > $b): ?>
<p>a est plus grand que b</p>
<?php elseif ($a == $b): ?>
<p>a est égal à b</p>
<?php else: ?>
<p>a est plus petit que b </p>
<?php endif; ?>
```

```
<?php
if ($a > $b) {
    echo '<p>a est plus grand que b</p>';
} elseif ($a == $b) {
    echo '<p>a est égal à b</p>';
} else {
    echo '<p>a est plus petit que b</p>';
}
?>
```

Opérateur ternaire ? :

- Condition ? Valeur_si_vrai : Valeur_si_faux

```
<?php  
$nb = 7;
```

```
echo 'il y a ' . $nb . ' élément';  
if ($nb > 1)  
    echo 's';  
echo '<br>';
```

```
// version plus courte grâce à l'opérateur ternaire ? :  
echo 'il y a ' . $nb . ' élément' . ($nb > 1 ? 's' : '') . '<br>';
```


La structure de contrôle switch

- **Forme bloc**

```
switch ($i) {  
    case 1:  
        echo '<p>un</p>';  
        break;  
    case 2:  
        echo '<p>deux</p>';  
        break;  
    case 3:  
        echo '<p>trois</p>';  
        break;  
    default:  
        echo '<p>ni un ni deux ni trois</p>';  
        break;  
}
```

La structure de contrôle switch

- Forme alternative

```
<?php switch ($i) :  
    case 1: ?>  
<p>un</p>  
<?php      break;  
    case 2: ?>  
<p>deux</p>  
<?php      break;  
    case 3: ?>  
<p>trois</p>  
<?php      break;  
    default: ?>  
<p>ni un ni deux ni trois</p>  
<?php      break;  
endswitch; ?>
```

La structure de contrôle switch

- Plusieurs cas ensemble

```
switch ($i) {  
    case 4:  
        echo 'est un nombre pair et de plus ';  
    case 1:  
    case 9:  
        echo 'est un carré parfait';  
        break;  
    case 2:  
    case 6:  
    case 8:  
    case 10:  
        echo 'est un nombre pair';  
        break;  
    default:  
        echo 'est ni pair, ni un carré parfait';  
        break;  
}
```

La structure de contrôle while

- Forme instruction unique

```
while ($i <= 10)
    echo '<p>passage n°' . $i++ . '</p>';
```

- Forme bloc

```
while ($i <= 10) {
    echo '<p>passage n°' . $i++ . '</p>';
}
```

- Forme alternative

```
<?php while ($i <= 10) : ?>
<p>passage n° <?= $i++ ?></p>
<?php endwhile; ?>
```

La structure de contrôle `do while`

- Forme bloc (unique syntaxe possible)

```
do {  
    echo '<p>passage n°' . $i++ . '</p>';  
} while ($i < 10);
```

La structure de contrôle for

- Forme instruction unique

```
for ($i = 1; $i <= 10; $i++)  
    echo '<p>passage n°' . $i . '</p>';
```

- Forme bloc

```
for ($i = 1; $i <= 10; $i++) {  
    echo '<p>passage n°' . $i . '</p>';  
}
```

- Forme alternative

```
<?php for ($i = 1; $i <= 10; $i++) : ?>  
<p>passage n°<?= $i ?></p>  
<?php endfor; ?>
```

La structure de contrôle foreach (valeur)

- Forme instruction unique

```
foreach ($tableau as $valeur)
    echo $valeur.'<br>';
```

- Forme bloc

```
foreach ($tableau as $valeur) {
    echo $valeur.'<br>';
}
```

- Forme alternative

```
<?php foreach ($tableau as $valeur) : ?>
<?= $valeur ?><br>
<?php endforeach; ?>
```

La structure de contrôle foreach (clef/valeur)

- Forme instruction unique

```
foreach ($tableau as $clef => $valeur)
    echo $clef.' -> '. $valeur.'<br>';
```

- Forme bloc

```
foreach ($tableau as $clef => $valeur) {
    echo $clef.' -> '. $valeur.'<br>';
}
```

- Forme alternative

```
<?php foreach ($tableau as $clef => $valeur) : ?>
<?= $clef ?> -> <?= $valeur ?><br>
<?php endforeach; ?>
```


L'inclusion d'un fichier

- `include`

- Fichier `identite.php`

```
<?php  
$prenom = 'Claire';
```

- Fichier `include.php`


```
<?php  
include './identite.php';  
echo 'Bonjour ' . $prenom . ' !';
```





Bonjour Claire !

L'inclusion d'un fichier

- include
 - Avertissement si le fichier n'est pas trouvé

 Warning: include(./inexistant.php): failed to open stream: No such file or directory in C:\wamp64\www\CoursPHP\1-bases\include_require\include2.php on line 2				
Call Stack				
#	Time	Memory	Function	Location
1	0.0010	368120	{main}()	...index.php:0
2	0.0010	372680	afficheCode()	...index.php:18
3	0.0020	373696	require('C:\wamp64\www\CoursPHP\1-bases\include_require\include2.php')	...afficheCode.php:15

 Warning: include(): Failed opening './inexistant.php' for inclusion (include_path='.:C:\php\pear') in C:\wamp64\www\CoursPHP\1-bases\include_require\include2.php on line 2				
Call Stack				
#	Time	Memory	Function	Location
1	0.0010	368120	{main}()	...index.php:0
2	0.0010	372680	afficheCode()	...index.php:18
3	0.0020	373696	require('C:\wamp64\www\CoursPHP\1-bases\include_require\include2.php')	...afficheCode.php:15

 Notice: Undefined variable: prenom in C:\wamp64\www\CoursPHP\1-bases\include_require\include2.php on line 3				
Call Stack				
#	Time	Memory	Function	Location
1	0.0010	368120	{main}()	...index.php:0
2	0.0010	372680	afficheCode()	...index.php:18
3	0.0020	373696	require('C:\wamp64\www\CoursPHP\1-bases\include_require\include2.php')	...afficheCode.php:15

Bonjour !

Fichier include2.php

```
<?php
include './inexistant.php';
echo 'Bonjour ' . $prenom . ' !';
```



L'inclusion d'un fichier

- require

- Fichier identite.php

```
<?php  
$prenom = 'Claire';
```

- Fichier require.php

```
<?php  
require './identite.php';  
echo 'Bonjour ' . $prenom . ' !';
```



Bonjour Claire !


L'inclusion d'un fichier

- require

- Fichier require2.php

```
<?php
require './inexistant.php';
echo 'Bonjour ' . $prenom . ' !';
```

- Erreur fatale si le fichier n'est pas trouvé

 Fatal error: require(): Failed opening required './inexistant.php' (include_path='.:C:\php\pear') in C:\wamp64\www\CoursPHP\1-bases\include_require\require2.php on line 2				
Call Stack				
#	Time	Memory	Function	Location
1	0.0010	370656	{main}()	...\index.php:0
2	0.1680	376648	afficheCode()	...\index.php:28
3	0.1690	377664	require('C:\wamp64\www\CoursPHP\1-bases\include_require\require2.php')	...\afficheCode.php:15



L'inclusion d'un fichier

- `include_once` **et** `require_once`
 - Ces instructions incluent le fichier si cela n'a pas encore été fait

- Fichier `identiteConst.php`

```
<?php  
define('NOM', 'Albert');
```

- Fichier `require3.php`

```
<?php  
require './identiteConst.php';  
require './identiteConst.php';  
echo 'Bonjour ' . NOM . ' !';
```

(!) Notice: Constant NOM already defined in C:\wamp64\www\CoursPHP\1-bases\include_require\identiteConst.php on line 2				
Call Stack				
#	Time	Memory	Function	Location
1	0.0010	370704	{main}()	...\index.php:0
2	0.1830	376696	afficheCode()	...\index.php:31
3	0.1850	378104	require('C:\wamp64\www\CoursPHP\1-bases\include_require\require3.php')	...\afficheCode.php:15
4	0.1850	378952	require('C:\wamp64\www\CoursPHP\1-bases\include_require\identiteConst.php')	...\require3.php:3
5	0.1850	379016	define()	...\identiteConst.php:2

Bonjour Albert !



L'inclusion d'un fichier

- `include_once` **et** `require_once`
 - Incluent le fichier si cela n'a pas encore été fait

- Fichier `identiteConst.php`

```
<?php  
define('NOM', 'Albert');
```

- Fichier `require_once.php`

```
<?php  
require_once './identiteConst.php';  
require_once './identiteConst.php';  
echo 'Bonjour ' . NOM . ' !';
```



Bonjour Albert !

Quelques fonctions utiles de PHP

- Le changement de la casse
 - `strtoupper()`
 - Passage en majuscules
 - `strtolower()`
 - Passage en minuscules
 - `ucwords()`
 - Passage de la première lettre de chaque mot en majuscule

Quelques fonctions utiles de PHP

- Exemples

```
<?php
$x = 'bIeNvEnUe à EnI éCoLe InFoRmAtIqUe';
echo "<p><code>strtolower('$x')</code> = " . strtolower($x) . '<br>';
echo "<code>strtoupper('$x')</code> = " . strtoupper($x) . '<br>';
echo "<code>ucwords('$x')</code> = " . ucwords($x) . '<br>';
echo "<code>ucwords(strtolower('$x'))</code> = " . ucwords(strtolower($x)
) . ' </p>';
```

strtolower('bIeNvEnUe à EnI éCoLe InFoRmAtIqUe') = **bienvenue à eni école informatique**
strtoupper('bIeNvEnUe à EnI éCoLe InFoRmAtIqUe') = **BIENVENUE à ENI ÉCOLE INFORMATIQUE**
ucwords('bIeNvEnUe à EnI éCoLe InFoRmAtIqUe') = **BleNvEnUe à EnI éCoLe InFoRmAtIqUe**
ucwords(strtolower('bIeNvEnUe à EnI éCoLe InFoRmAtIqUe')) = **Bienvenue à Eni école Informatique**



Quelques fonctions utiles de PHP

- La mise en forme de valeurs numériques

- `sprintf()`

- Formate une chaîne de caractères

```
echo sprintf('%02d/%02d/%04d', 1, 1, 1981);
```



01/01/1981

- `number_format()`

- Transforme un nombre en une chaîne de caractères formatée

```
$x = 1234.567;  
echo number_format($x) . '<br>';  
echo number_format($x, 1) . '<br>';  
echo number_format($x, 2, ',', ' ') . '</p>';
```



1,235
1,234.6
1 234,57

Quelques fonctions utiles de PHP

- Recherche dans une chaîne de caractères
 - `strpos()`
 - `strrpos()` Dernière occurrence
 - `stripos()` Sans tenir compte de la casse
 - `strripos()` Dernière occurrence et sans tenir compte de la casse

Quelques fonctions utiles de PHP

- Recherche dans une chaîne de caractères

```
<?php
$mail = 'henri.desmoulins3@eni.fr';
$position = strpos($mail, '@');
if ($position === false) {
    echo "'@' est introuvable dans $mail<br>";
} else {
    echo "'@' est à la position $position dans $mail<br>";
}
```



'@' est à la position 17 dans henri.desmoulins3@eni.fr

Quelques fonctions utiles de PHP

- Recherche dans une chaîne de caractères

```
<?php
$mail = 'henri.desmoulins3@eni.fr';
$position = strpos($mail, 'h');
if ($position === false) {
    echo "'h' est introuvable dans $mail<br>";
} else {
    echo "'h' est à la position $position dans $mail<br>";
}
```



Rappelez-vous que
0 est équivalent à false
Il faut donc utiliser
l'opérateur ===



'h' est à la position 0 dans henri.desmoulins3@eni.fr

Quelques fonctions utiles de PHP

- Recherche dans une chaîne de caractères

```
<?php
$mail = 'henri.desmoulins3@eni.fr';
$position = strpos($mail, 'w');
if ($position === false) {
    echo "'w' est introuvable dans $mail<br>";
} else {
    echo "'w' est à la position $position dans $mail<br>";
}
```



'w' est introuvable dans henri.desmoulins3@eni.fr

Quelques fonctions utiles de PHP

- Manipulation de dates

- mktime()

- Retourne le timestamp Unix, c'est-à-dire le nombre de secondes écoulées entre le 1^{er} janvier 1970 et la date

```
<?php
```

```
echo '<p>Création du timestamp pour le 10 avril 2017 à 11h45 et 30 secondes<br>';
```

```
echo '<code>mktime(11, 45, 30, 4, 10, 2017)</code> = ' . mktime(11, 45, 30, 4, 10, 2017) . '</p>';
```

```
echo '<p><code>mktime(0, 0, 0, 1, 1, 1970)</code> = ' . mktime(0, 0, 0, 1, 1, 1970) . '</p>';
```



Création du timestamp pour le 10 avril 2017 à 11h45 et 30 secondes

`mktime(11, 45, 30, 4, 10, 2017) = 1491824730`

`mktime(0, 0, 0, 1, 1, 1970) = 0`

Quelques fonctions utiles de PHP

- Manipulation de dates

- date()

- Retourne une chaîne de caractères au format souhaité

```
<?php
echo '<p><code>date("d/m/Y H:i:s", mktime(11, 45, 30, 4, 10, 2017))</code> = ' .
    date("d/m/Y H:i:s", mktime(11, 45, 30, 4, 10, 2017)) . '<br>';
echo 'Unix a fêté sa milliardième seconde le ' . date("d/m/Y H:i:s", 1000000000) . '</p>';
echo '<p>Date du jour au format JJ/MM/AAAA : ' . date('d/m/Y') . '<p>';
```



```
date("d/m/Y H:i:s", mktime(11, 45, 30, 4, 10, 2017)) = 10/04/2017 11:45:30
Unix a fêté sa milliardième seconde le 09/09/2001 01:46:40
Date du jour au format JJ/MM/AAAA : 13/09/2017
```

Quelques fonctions utiles de PHP

- Modification des paramètres de php.ini pour ce script

- ini_set()

```
<?php
ini_set('date.timezone', 'Australia/Sydney');
echo '<p>A Sydney, il est ' . date("H:i:s") . '</p>';

ini_set('date.timezone', 'Europe/Paris');
echo '<p>En France il est ' . date("H:i:s") . '</p>';

ini_set('date.timezone', 'America/Buenos_Aires');
echo '<p>En Argentine, il est ' . date("H:i:s") . '</p>';
```



A Sydney, il est 20:04:33
En France il est 12:04:33
En Argentine, il est 07:04:33

Les fonctions

- Les mêmes règles de nommage que les variables (sans le \$)
- Pas de distinction entre fonction et procédure
- Les fonctions n'ont pas besoin d'avoir été définies avant leur utilisation

Les fonctions

- La création d'une fonction

```
<?php
function somme($valeur1, $valeur2) {
    return $valeur1 + $valeur2;
}
```

- L'appel d'une fonction

```
<?php
$a = 82;
echo $a . ' + 35 = ' . somme($a, 35);
```



82 + 35 = 117

Les fonctions

- Le passage d'un paramètre par valeur



Modification
d'une copie

```
<?php
function multiplieValeur($valeur, $multiplicateur) {
    $valeur *= $multiplicateur;
    echo 'Valeur multipliée dans la fonction : ' . $valeur . '<br>';
}

$valeur = 10;
echo 'Valeur avant la fonction : ' . $valeur . '<br>';
multiplieValeur($valeur, 2);
echo 'Valeur après la fonction : ' . $valeur . '<br>';
```



Valeur avant la fonction : 10
Valeur multipliée dans la fonction : 20
Valeur après la fonction : 10

Les fonctions

- Le passage d'un paramètre par référence &



Modification
de l'original

```
<?php
function multiplieReference(&$valeur, $multiplicateur) {
    $valeur *= $multiplicateur;
    echo 'Valeur multipliée dans la fonction : ' . $valeur . '<br>';
}

$valeur = 10;
echo 'Valeur avant la fonction : ' . $valeur . '<br>';
multiplieReference($valeur, 2);
echo 'Valeur après la fonction : ' . $valeur . '<br>';
```



Valeur avant la fonction : 10
Valeur multipliée dans la fonction : 20
Valeur après la fonction : 20

Les fonctions

- Le passage d'un paramètre : les valeurs par défaut

```
<?php
define('QUATRE', 4);

function produit($valeur1 = QUATRE, $valeur2 = 2) {
    return $valeur1 * $valeur2;
}

echo '<p><code>produit(5, 3)</code> : 5 × 3 = ' . produit(5, 3) . '<br>';
echo '<code>produit(5)</code> : 5 × 2 = ' . produit(5) . '<br>';
echo '<code>produit()</code> : 4 × 2 = ' . produit() . '</p>';
```



```
produit(5, 3) : 5 × 3 = 15
produit(5) : 5 × 2 = 10
produit() : 4 × 2 = 8
```

Les fonctions

- Les variables statiques

```
<?php
function variableStatique() {
    $variable = 0;
    static $variable_statique = 0;

    $variable++;
    echo "Variable : $variable <br>";

    $variable_statique++;
    echo "Variable statique : $variable_statique <br>";
}

variableStatique();
variableStatique();
variableStatique();
```



Variable : 1
Variable statique : 1
Variable : 1
Variable statique : 2
Variable : 1
Variable statique : 3

Les fonctions

- Les fonctions variables

```
<?php  
function somme($valeur1, $valeur2) {  
    return $valeur1 + $valeur2;  
}
```

```
<?php  
$addition = 'somme';  
echo ucfirst($addition) . 'Somme 2 + 2 = ' . $addition(2, 2), '<br>';
```



Somme 2 + 2 = 4

Les fonctions

- Le passage d'une fonction en paramètre



Multiplication $5 \times 8 = 40$

Somme $5 + 8 = 13$

Division $5 / 8 =$ La fonction "division" n'existe pas !

```
<?php
function operation($valeur1, $valeur2, $fonction) {
    if (is_string($fonction) && !function_exists($fonction)) {
        $retour = 'La fonction "' . $fonction . '" n\'existe pas !';
    } else {
        $retour = $fonction($valeur1, $valeur2);
    }
    return $retour;
}

echo 'Multiplication 5 × 8 = ' . operation(5, 8, 'produit') . '<br>';
echo 'Somme 5 + 8 = ' . operation(5, 8, 'somme') . '<br>';
echo 'Division 5 / 8 = ' . operation(5, 8, 'division'), '<br>';
```


Les fonctions

- Les fonctions anonymes

```
<?php
$resultat = operation(5, 8, function($valeur1, $valeur2) {
    return $valeur1 - $valeur2;
});
echo 'Soustraction 5 - 8 = ' . $resultat . '<br>';

$soustraction = function($valeur1, $valeur2) {
    return $valeur1 - $valeur2;
};
echo 'Soustraction 2 - 2 = ' . $soustraction(2, 2), '<br>';
```



Soustraction 5 - 8 = -3
Soustraction 2 - 2 = 0

Les fonctions

- Les fonctions typées

```
<?php
function resteDivisionEntiere(int $dividende, int $diviseur): int {
    var_dump($dividende);
    return $dividende % $diviseur;
}

echo '<p><code>resteDivisionEntiere(17, 5)</code> = ' . resteDivisionEntiere(17, 5). '</p>';

// il effectue une conversion implicite vers un nombre entier
echo '<p><code>resteDivisionEntiere(17.8, 5)</code> = ' . resteDivisionEntiere(17.8, 5). '</p>';

// erreur car l'argument passé n'est pas du bon type (la conversion implicite ne fonctionne pas)
echo '<p><code>resteDivisionEntiere(\'a\', 5)</code> = ' . resteDivisionEntiere('a', 5). '</p>';
```



```
int 17
resteDivisionEntiere(17, 5) = 2
int 17
resteDivisionEntiere(17.8, 5) = 2
```

Les fonctions

- L'utilisation d'une variable globale

```
<?php
$a = 67600;

function util_var_globale() {
    global $a;
    echo 'valeur de variable globale a : ' . $a . '<br>';
    $GLOBALS['b'] = 'test';
}

util_var_globale();
echo 'valeur de variable globale b : ' . $b;
```



valeur de variable globale a : 67600
valeur de variable globale b : test

Les expressions rationnelles

- Vérification complexe au sein d'un texte
 - Exemple : est-ce que le mot de passe a au moins 6 caractères, au moins un chiffre et une majuscule ?
- Remplacement complexe
 - Exemple : passer en majuscule les mots commençant par un 'a' et ayant entre 3 et 7 caractères !
- Utilise la syntaxe PCRE (Perl Compatible Regular Expressions)

Les bases du langage

Les expressions rationnelles

- Vérification
 - `preg_match()`
- Remplacement
 - `preg_replace()`

Les expressions rationnelles

- Les caractères

- Tout caractère se désigne lui-même sauf les caractères spéciaux (point, parenthèses, crochets, accolades, tiret, backslash...)
- Pour les caractères spéciaux, il faut les échapper avec un backslash
 - Exemple : le motif `\[a\]` correspond à `a` entre crochets
- `.` Remplace n'importe quel caractère
 - Exemple : le motif `e.i` correspond à un mot ayant un `e` suivi d'un caractère quelconque suivi d'un `i`.

- Le début et la fin

- `^` représente le début de la chaîne
 - Exemple : le motif `^a` représente une chaîne commençant par `a`.
- `$` représente la fin de la chaîne
 - Exemples : le motif `b$` représente une chaîne terminant par `b`.
le motif `^e..$` représente les chaînes de trois caractères commençant par un `e`.

Les expressions rationnelles

- Les quantifieurs
 - $\{n\}$: ce qui précède doit être répété n fois
 - Exemple : $10\{9\}$ représente le chiffre 1 suivi de 9 fois le chiffre 0.
 - $\{min, max\}$: ce qui précède doit être répété entre min et max fois.
 - Exemple : $G\{2, 7\}g1e$ représente un G puis entre 2 et 7 o et enfin g1e.
 - $\{min, \}$: ce qui précède est répété au minimum min fois.
 - $?$: équivalent à $\{0, 1\}$ c'est-à-dire optionnel
 - $+$: équivalent à $\{1, \}$ c'est-à-dire « au moins une fois »
 - $*$: équivalent à $\{0, \}$

Les expressions rationnelles

- Les ensembles

- [...]: l'un des caractères parmi
 - Exemple : `[ab]` représente le caractère `a` ou le caractère `b`.
- [^...]: tout sauf ces caractères
 - Exemple : `[^ab]` représente n'importe quel caractère sauf les caractères `a` ou `b`.
- [-]: l'un des caractères parmi la plage de valeurs
 - Exemple : `[a-z]` représente une lettre minuscule.

Les expressions rationnelles

- Les groupes

- (...)

- regroupe les éléments à l'intérieur comme un seul élément
 - capture les éléments
 - Exemple : `(très){0,2}puissant` représente « **puissant** », « **très puissant** » et « **très très puissant** ».
Cela capture la chaîne « **très** » si elle est présente au moins une fois

- (? :...)

- regroupe les éléments sans capturer le groupe
 - Exemple : `(?:très){1,2}([a-z]+)` capturera uniquement « **puissant** », si l'expression analysée est « **très très puissant** ».

- (?:=...)

- regroupe les éléments sans capturer le groupe sans consommer les caractères

Les expressions rationnelles

- Les classes abrégées

- `\d` : un chiffre (équivalent à `[0-9]`)
- `\D` : un caractère qui n'est pas un chiffre (équivalent à `[^0-9]`)
- `\w` : n'importe quel caractère alphanumérique ou underscore (équivalent à `[a-zA-Z0-9_]`)
- `\W` : ni un caractère alphanumérique et ni underscore (équivalent à `[^a-zA-Z0-9_]`)
- `\t` : une tabulation
- `\n` : un saut de ligne
- `\r` : un retour chariot
- `\s` : un espace vide (espace, tabulation, retour à la ligne, saut de page...)
- `\S` : tout sauf un espace vide

Les expressions rationnelles

- Les délimiteurs

- Entourent l'expression rationnelle
- N'importe quel caractère peut être utilisé
 - # par exemple dans ce cours

```
#expressionRationnelle#options
```

- Les options

- Se positionnent après le second délimiteur
- Exemples :
 - i : insensibilité à la casse
 - m : considère la chaîne comme une seule ligne
 - s : le méta caractère . remplace n'importe quel caractère y compris les nouvelles lignes

Les bases du langage

Création d'une expression rationnelle

Démonstration



Les expressions rationnelles : preg_match ()

```
<?php
$motif = '#^([0-9]{1,2})/([0-9]{1,2})/([0-9]{4})$#';

// Tableau contenant les chaines à tester :
$dates[] = '1/4/2018'; // OK
$dates[] = '10/04/2018'; // OK
$dates[] = '10/04/18'; // KO car l'année n'est pas sur 4 chiffres

echo '<ul>';
foreach ($dates as $date) {
    echo '<li>Test de ' . $date . ' : ';
    $ok = (preg_match($motif, $date, $resultat) > 0);
    if ($ok) {
        echo 'est valide :<ul><li>jour : ' . $resultat[1] . '</li><li>mois : ' . $resultat[2] .
            '</li><li>année : ' . $resultat[3] . '</li></ul></li>';
    } else {
        echo 'n\'est pas valide';
    }
}
echo '</ul>';
```



Test de 1/4/2018 : est valide :
jour : 1
mois : 4
année : 2018
Test de 10/04/2018 : est valide :
jour : 10
mois : 04
année : 2018
Test de 10/04/18 : n'est pas valide

Les expressions rationnelles : preg_replace()

```
<?php
$motif = '#^([0-9]{1,2})/([0-9]{1,2})/([0-9]{4})$#';
$savant = '01/04/2014';
$apres = preg_replace($motif, '$3-$2-$1', $savant);
echo $savant . ' > ' . $apres;
```



01/04/2014 > 2014-04-01