

## Table des matières

1. Le PHP .....	2
1.1 Installer PHP .....	2
2. Premier code PHP .....	2
3. Structure du code php .....	3
3.1 Syntaxe des commentaires en PHP .....	3
3.2 Variables en php.....	3
3.3 Fonctions PHP .....	4
3.4 Affichage de variables PHP .....	4
3.5 Opérateurs arithmétiques.....	5
3.6 Opérateurs de chaînes de caractères.....	5
3.7 Test votre code via un serveur .....	5
3.8 Les conditions .....	6
3.9 Les opérateurs .....	6
3.10 Conditions tertiaires .....	7
3.11 Les tableaux.....	7
3.12 Les tableaux associatifs .....	8
3.13 Les boucles.....	8
3.13.1 Boucle for .....	8
3.13.2 Boucle foreach.....	9
3.13.3 Boucle while.....	9
3.14 Les fonctions .....	10

## 1. Le PHP

- La version 1.0 a été créée en 1995 par Rasmus Lerdorf
- Langage procédurale
- Typage faible sur les premières versions

### 1.1 Installer PHP

- Version 7.4 de PHP : <https://windows.php.net/download#php-7.4>
- Télécharger le .zip adéquat en fonction de la version PHP
- Créer un dossier où vous le souhaitez sur votre ordinateur, et renommer le en « php74 »
- Ajouter le lien vers le dossier à votre variable d'environnement
- Ouvrez un invite de commande windows et lancer la commande « php -v »

Vous devriez voir quelque chose comme ça :

```
PHP 7.4.9 (cli)
```

PHP est installé !

## 2. Premier code PHP

- Créer un fichier de nom « test.php », à l'emplacement que vous souhaitez.
- Écrire ces lignes à l'intérieur :

```
<?php  
echo "Hello World!!";  
?>
```

- Puis sauvegarder
- Dans un invite de commande windows, aller au chemin du dossier où vous avez créé le fichier test.php, puis exécutez la commande suivante :

```
php test.php
```

« Hello World !! » doit apparaître dans la console !

## 3. Structure du code php

Balise php ouvrante, permettant d'écrire du code PHP

```
<?php  
  
echo "Hello World!!";  
  
?>
```

Instruction PHP, elle se termine toujours par un « ; »

Balise php fermante, on n'a pas besoin de préciser « php » ici

Vous devez toujours avoir une balise php ouvrante et fermante pour que les instructions PHP soient interprétées.

### 3.1 Syntaxe des commentaires en PHP

```
// Je suis un commentaire en PHP
```

Un commentaire sur une ligne commence par //

```
/*  
 * Moi aussi,  
 * mais je suis multiligne  
 */
```

Un commentaire sur plusieurs lignes commence par « /\* » et se termine par « \*/ », chaque ligne de commentaire à l'intérieur commence par un « \* ».

### 3.2 Variables en php

Une variable php commence TOUJOURS par un « \$ »

```
< ?php  
  
// On assigne la chaine de caractères 'Kevin' à la variable de nom $firstName  
$firstName = 'Kevin' ;  
  
// On assigne l'entier 99 à la variable de nom $age  
$age = 99;  
  
// On assigne le booléen true à la variable $isAlive  
$isAlive = true ;
```

```
// On crée une variable de type « tableau »  
$array = [ ];
```

```
// On crée une variable vide, null  
$null = null ;
```

### 3.3 Fonctions PHP

```
< ?php  
  
// On assigne la chaîne de caractères 'Kevin' à la variable de nom $firstName  
$firstName = 'Kevin';  
  
echo strlen($firstName) ;
```

Affiche en texte le résultat du strlen, strlen permet de donner la taille d'une chaîne de caractère, ici elle affichera 5.

Il existe de nombreuses fonctions PHP, vous les trouverez ici :

<https://www.php.net/manual/fr/indexes.functions.php>

### 3.4 Affichage de variables PHP

```
< ?php  
  
// On assigne la chaîne de caractères 'Kevin' à la variable de nom $firstName  
$firstName = 'Kevin';  
  
// Affiche une chaîne de caractère  
echo $firstName;  
  
// Affiche une chaîne de caractère  
print $firstName;  
  
// Affiche le contenu d'une variable sous forme de chaîne de caractère  
print_r($firstName);  
  
// Affiche les informations d'une variable  
var_dump($firstName);
```

## 3.5 Opérateurs arithmétiques

```
$result = 2 + 5;

$result = 5 - 5;

$result = 5 * 5;

$result = 8 / 2;

$result = 8 % 2;

// Ces deux lignes sont identiques
$result = $result + 5;

$result += 5;

$result *= 2;

$result++;

$result--;
```

## 3.6 Opérateurs de chaînes de caractères

```
<?php

$hello = "Hello";
$world = "world";

// Concaténation
$result = $hello." ".$world;

// Concaténation aussi
$result .= "!";

echo $result;
```

## 3.7 Test votre code via un serveur

Lancez un terminal windows, allez dans le dossier où vous avez votre fichier php à exécuter et faire la commande suivante :

```
php -S localhost:8000
```

Une URL s'affiche, souvent : <http://localhost:8000>

Ouvrez un navigateur web et copier/coller l'url dedans, puis ajoutez « **/test.php** »

Vous devriez voir une page web avec « Hello world !! » d'afficher !

### 3.8 Les conditions

```
<?php

if ($result >= 50) {
    echo "Your result is at least the average.";
} else if ($result >= 90) {
    echo "Your result is very good.";
} else {
    echo "Your result is not bad.";
}
```

### 3.9 Les opérateurs

```
if ($result == 50)

if ($result > 50)

if ($result < 50)

if ($result >= 50)

if ($result <= 50)

if ($result > 80 && $result < 20)

if ($result >= 50 || $result == 0)

if (!$result < 50)
```

## 3.10 Conditions tertiaires

```
<?php

$result = 52;

if ($result > 50) {
    echo "Your result is above 50!";
} else {
    echo "Your result is under 50...";
}

// Ternaire équivalente à la condition ci-dessus
echo ($result > 50) ? "Your result is above 50!" : "Your result is under 50...";
```

La partie avant le ‘?’ est la condition la première partie après est le résultat si la condition est valide et la partie après les ‘:’ est l’autre résultat en cas contraire.

## 3.11 Les tableaux

```
$array = [];

// Initialise un tableau avec des valeurs
$array = ['test', 15];

// Affiche la valeur à l'index 0 soit test
echo $array[0];

// Ajoute une valeur au tableau à l'index suivant, soit l'index 3
$array[] = 12;

// Assigne une valeur à l'index 1 (remplace la valeur 15 par 17)
$array[1] = 17;

// Affiche le tableau
print_r($array);

Array (
    [0] => test
    [1] => 17
    [2] => 12
)
```

- On peut stocker n’importe quel type d’élément dans un tableau
- L’index est numérique et commence à 0

## 3.12 Les tableaux associatifs

```
// Initialise un tableau avec des valeurs et des clés
$array = [
    'anglais' => 15,
    'math' => 12,
    0 => 13.5
];

// Affiche la valeur pour la clé 'math' soit 12
echo $array['math'];

// Ajoute une valeur au tableau à pour la clé 'espagnol'
$array['espagnol'] = 18;

// Affiche le tableau
print_r($array);

Array (
    [anglais] => 15
    [math] => 12
    [0] => 13.5
    [espagnol] => 18
)
```

- On peut définir nous-même les valeurs des clés-valeur (K,V) pour les tableaux
- Les clés sont de types string ou int
- On suit la donc la logique « clé » => valeur

## 3.13 Les boucles

### 3.13.1 Boucle for

Ça sert à répéter une ou plusieurs opérations, ici on veut afficher les nombres de 1 à 10.

```
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

Initialisation

condition de la boucle

« autre opération » effectuée à chaque passage



### 3.13.2 Boucle foreach

Très utile pour les parcours de tableaux

```
$array = [  
    'math' => 15,  
    'english' => 10  
];  
  
foreach ($array as $key => $value) {  
    echo $key. " : " . $value;  
}
```

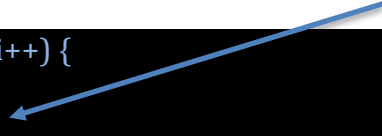
### 3.13.3 Boucle while

```
$x = 10;  
  
while ($x < 10) {  
    $x++;  
}  
echo $x;
```

Boucle « Tant que », elle continue tant que la condition est réalisée.  
(Attention aux boucles infinies !)

Il est aussi possible d'arrêter manuellement une boucle :

```
for ($i = 0; $i < 10; $i++) {  
    if ($i === 5) {  
        break;  
        // arrête la boucle  
    }  
    echo $i;  
}
```



Si \$i atteint la valeur « 5 » alors la boucle s'arrêtera et l'écho ne sera pas fait.

## 3.14 Les fonctions

Une fonction se déclare toujours avec le mot clé « function » suivi d'un nom et de deux parenthèses.

```
// Déclaration de la fonction "hello"
function hello() : void {
    echo "Hello world";
}

// Appel de la fonction "hello"
hello();
```

- On peut déclarer une fonction avant ou après l'appel
- Les noms de fonction doivent être écrit en camel-case

Une fonction peut aussi prendre des arguments :

```
// Déclaration de la fonction "getHello"
function getHello($name) : string {
    return "Hello ".$name;
}

// Appel de la fonction "getHello"
$hello = getHello("Théau");
echo $hello;
```

- L'argument se met entre les parenthèses de la fonction, une fonction peut prendre plusieurs arguments, tous séparés par des virgules
- Une fonction peut retourner une valeur, celle-ci se fait avec le mot clé « return »

Une fonction peut aussi avoir un argument avec une valeur par défaut, c'est-à-dire que si l'argument n'est pas renseigné, il prend la valeur par défaut :

```
// Déclaration de la fonction "getHello"
function getHello($name = "world") {
    return "Hello ".$name;
}

// Appel de la fonction "getHello"
$hello = getHello();
echo $hello;
```

Une fonction n'a pas accès aux variables à l'extérieur, c'est-à-dire qu'il ne peut pas la modifier :

```
$word = 'test';

// Déclaration de la fonction "toUpper"
function toUpper($word) {
    $word = strtoupper($word);
}

// Appel de la fonction "toUpper"
toUpper($word);
echo $word;
```

Que vaut « \$word » ?

```
$word = 'test';

// Déclaration de la fonction "toUpper"
function toUpper(&$word) {
    $word = strtoupper($word);
}

// Appel de la fonction "toUpper"
toUpper($word);
echo $word;
```

Il faut passer la variable par référence afin de permettre à la fonction de la modifier.

Que vaut « \$word » ?