

**PHP**

**Clevory Training**

# Plan de formation

---



- Introduction
- Intégration PHP et HTML



- Fonctionnalités de base de PHP
- Les chaînes de caractères et les tableaux



- Les fonctions
- Les formulaires



- Les cookies
- Les sessions

Aymen SELLAOUTI

- Docteur en informatique
- Enseignant universitaire
- Créateur de contenu (PHP, Symfony, Angular, NestJs) et Fondateur de la chaine Youtube Techwall (+20000 abonnés)



<https://www.youtube.com/c/TechWall>

- Formateur en Fullstack Javascript (Angular, Nest JS)
- Consultant PHP Symfony et Fullstack Javascript

# PHP

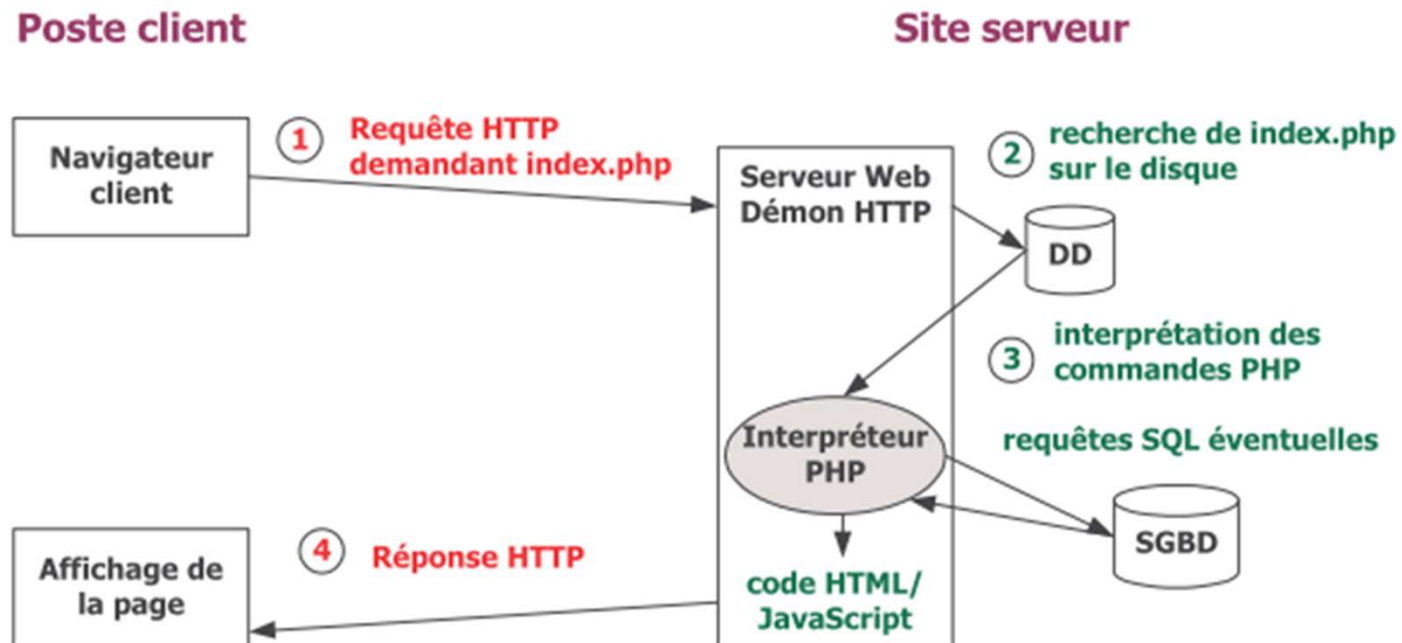
## Intro

### INTRO

- PHP (PHP Hypertext Preprocessor) est un langage de script open source côté serveur.
- Langage interprété
- S'exécute coté serveur
- Le code est intégré au code source de la page HTML
- Permet de rendre les pages HTML Dynamique
- Utilise le protocole HTTP.

# PHP

## Intro



# PHP


## Intro

- L'interpréteur PHP lit un fichier source puis génère un flux de sortie avec les règles suivantes:
- Toute ligne située à l'extérieur d'un bloc PHP ( entre `<?php` et `?>`) est utilisé inchangée dans le flux de sortie
- Le code PHP est interprété et on récupère des résultats qu'il intègre au flux de sortie
- S'il y a des erreurs des messages d'erreurs sont intégrés dans le flux de sortie

# PHP

## Intro

- Pour Installer PHP, vous pouvez installer un serveur comme XAMP ou WAMP (<https://www.apachefriends.org/fr/download.html>)

 XAMPP pour Windows 8.0.28, 8.1.17 & 8.2.4

Version		Code de vérification	Taille
8.0.28 / PHP 8.0.28	Contenu	md5 sha1	Télécharger (64 bit) 144 Mb
8.1.17 / PHP 8.1.17	Contenu	md5 sha1	Télécharger (64 bit) 148 Mb
8.2.4 / PHP 8.2.4	Contenu	md5 sha1	Télécharger (64 bit) 149 Mb

XAMPP Control Panel v3.3.0 [ Compiled: Apr 6th 2021 ]

XAMPP Control Panel v3.3.0

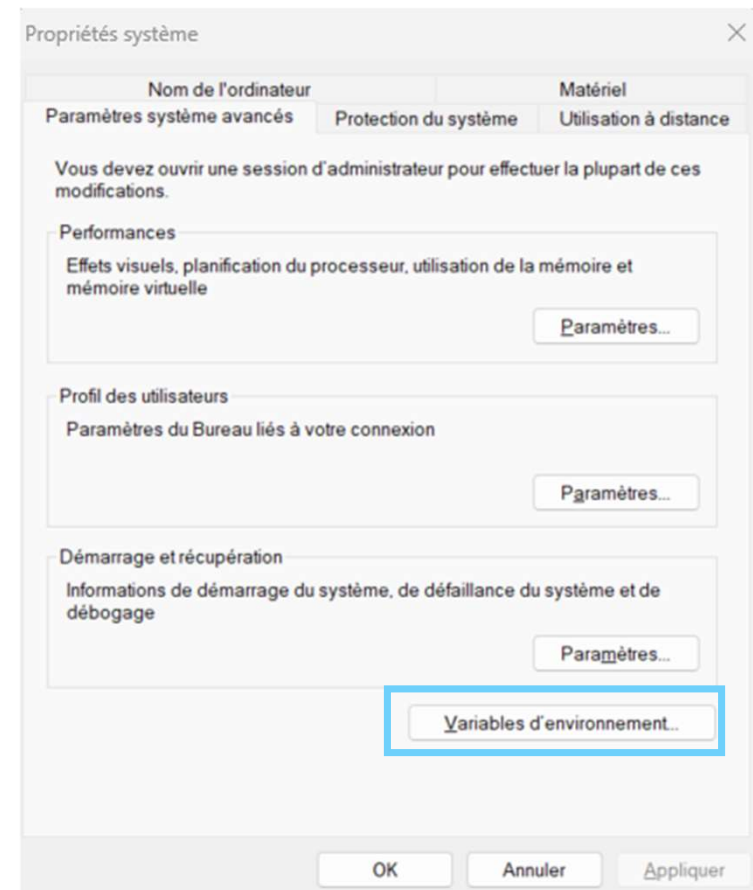
Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache			Start Admin Config Logs
<input type="checkbox"/>	MySQL			Start Admin Config Logs
<input type="checkbox"/>	FileZilla			Start Admin Config Logs
<input type="checkbox"/>	Mercury			Start Admin Config Logs
<input type="checkbox"/>	Tomcat			Start Admin Config Logs

16:42:46 [main] there will be a security dialogue or things will break! So think about running this application with administrator rights!  
16:42:46 [main] XAMPP Installation Directory: "c:\xampp816"  
16:42:46 [main] Checking for prerequisites  
16:42:48 [main] All prerequisites found  
16:42:48 [main] Initializing Modules  
16:42:48 [main] Starting Check-Timer  
16:42:48 [main] Control Panel Ready

# PHP

## Intro

- XAMPP vous offre une panoplie d'outils y compris PHP. Cependant vous pouvez installer PHP séparément sans passer par aucune de ces logiciels.
- Une fois installé, vous devez **ajouter php en tant que variable d'environnement**.
- Pour ce faire, **cherchez « variable d'environnement »** et vous allez avoir l'interface suivante :
- Sélectionner le bouton **Variables d'environnement**

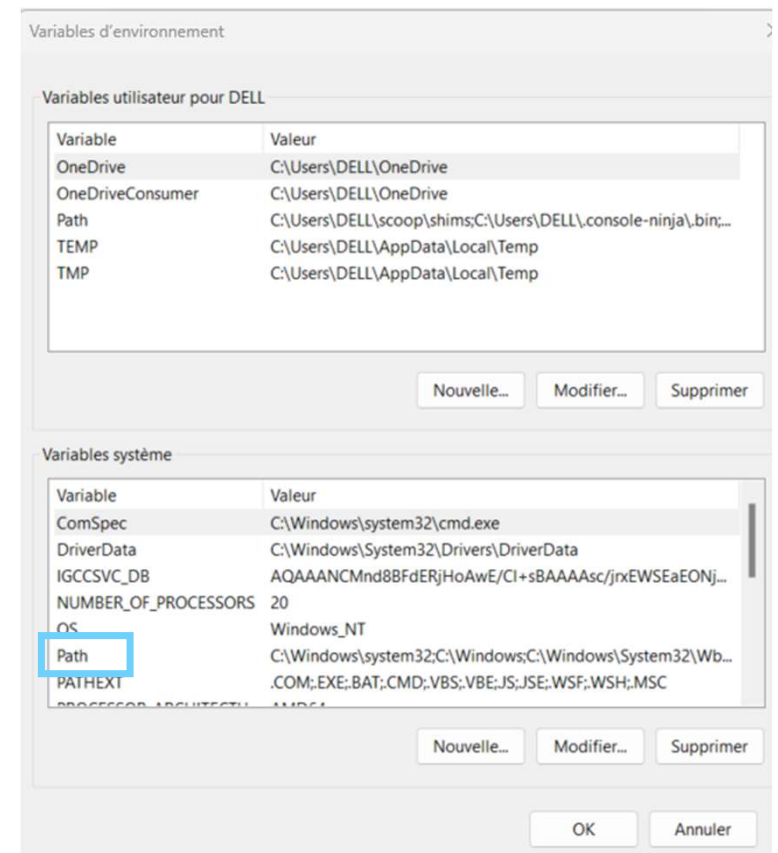




# PHP

## Intro

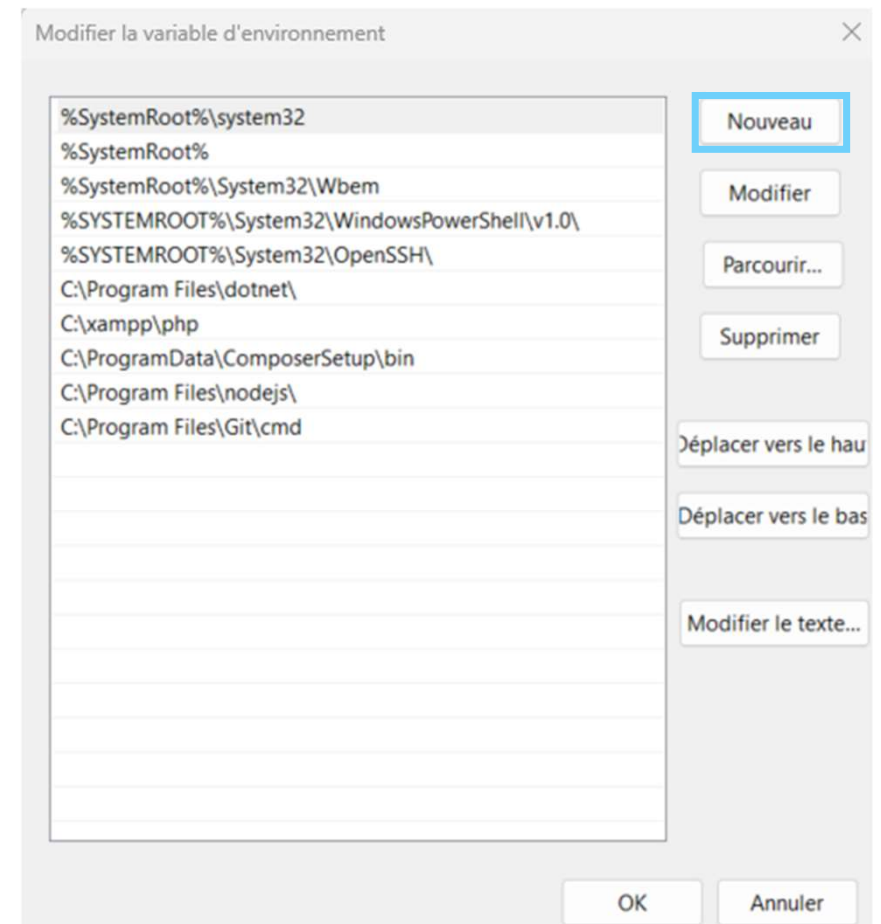
- Allez dans la section Variables système et sélectionner la variable **Path**



# PHP

## Intro

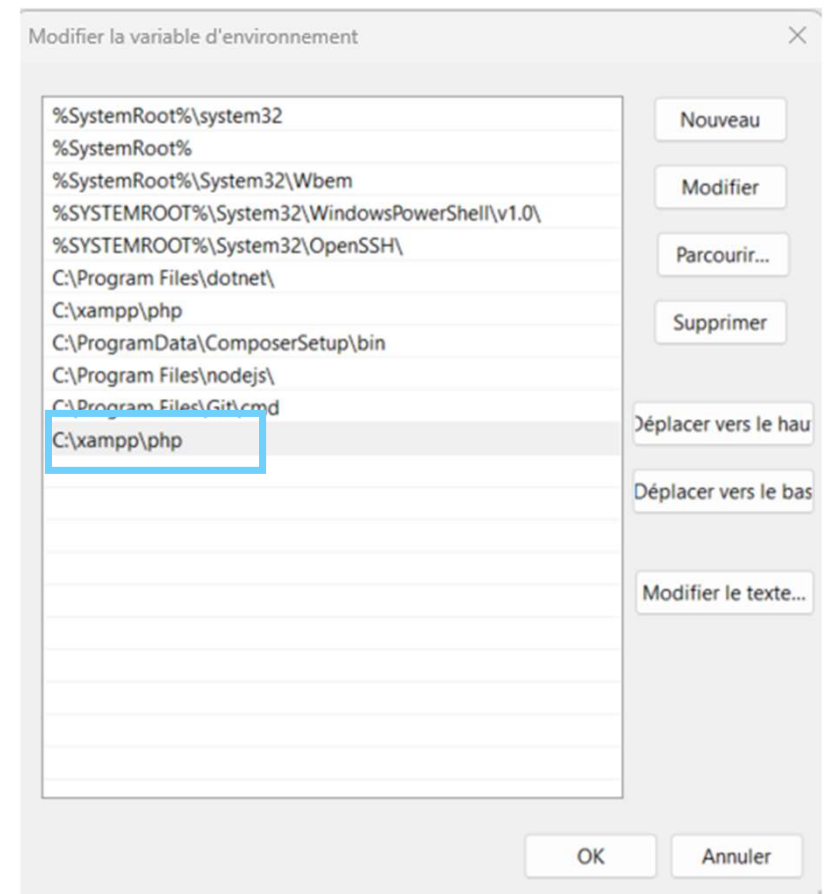
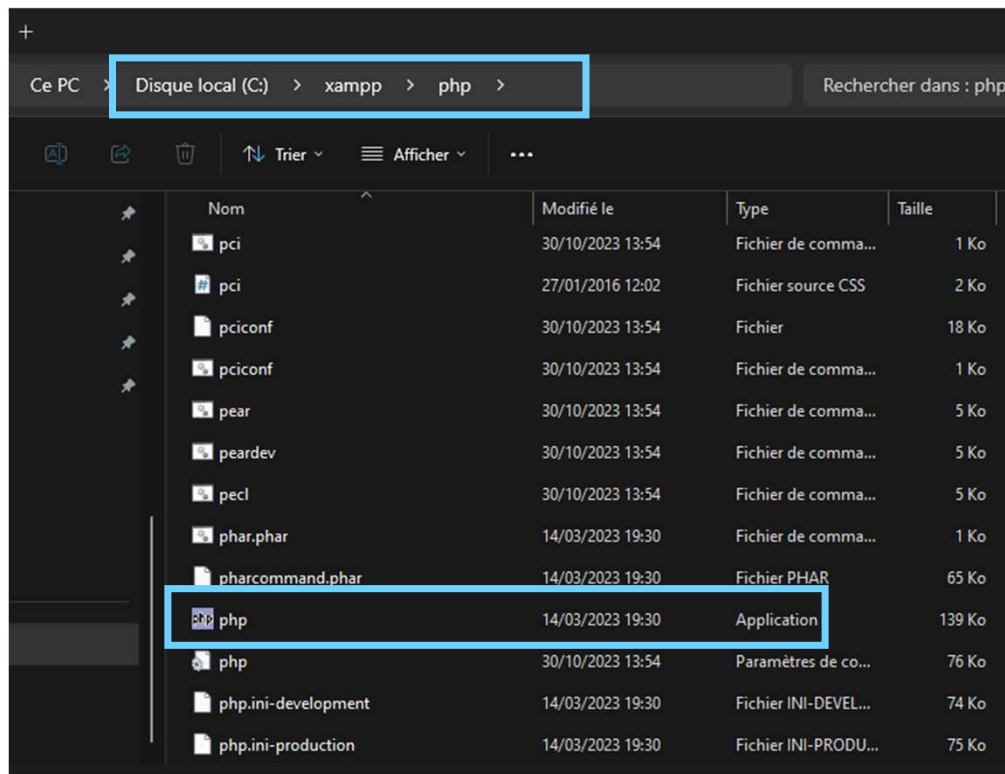
➤ Sélectionnez **Nouveau**



# PHP

## Intro

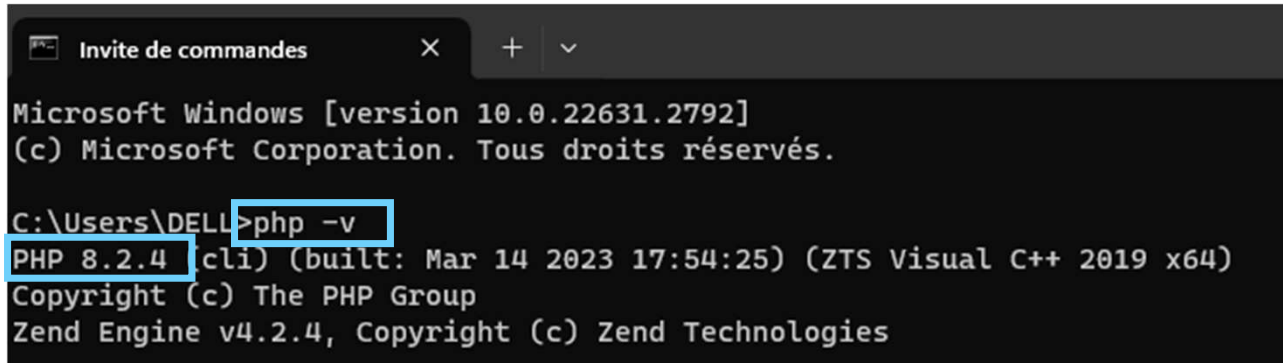
- Ajouter le chemin du **dossier php de XAMPP**, dans ce dossier se trouvera **l'exécutable php**



# PHP

## Intro

- Pour vérifier que **PHP est visible dans votre système**, ouvrez un **Terminal** et tapez la commande **php -v**



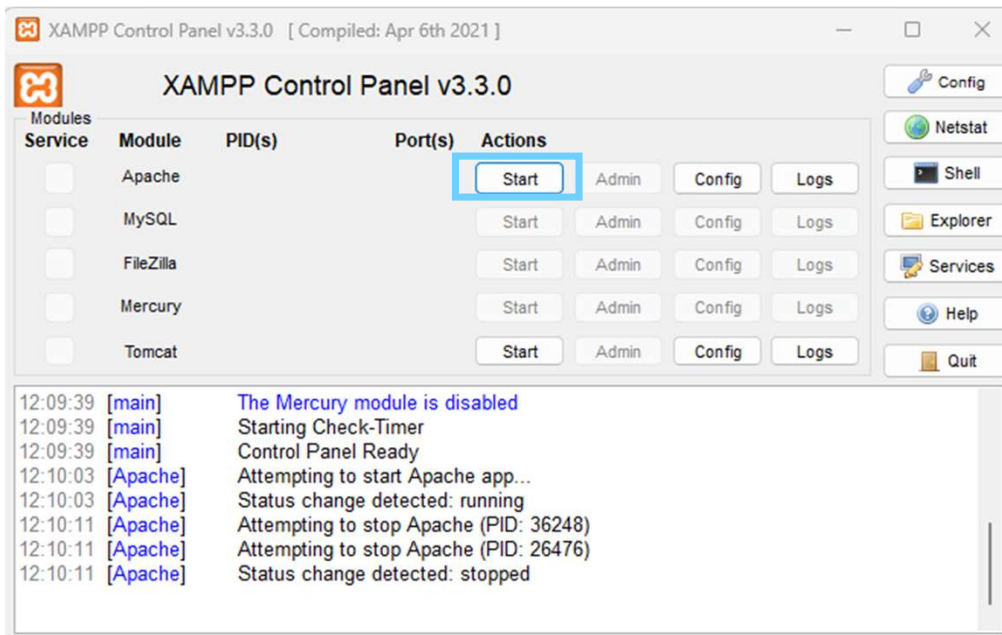
```
Invite de commandes
Microsoft Windows [version 10.0.22631.2792]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\DELL>php -v
PHP 8.2.4 (cli) (built: Mar 14 2023 17:54:25) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.4, Copyright (c) Zend Technologies
```

# PHP

## Lancez votre serveur Web

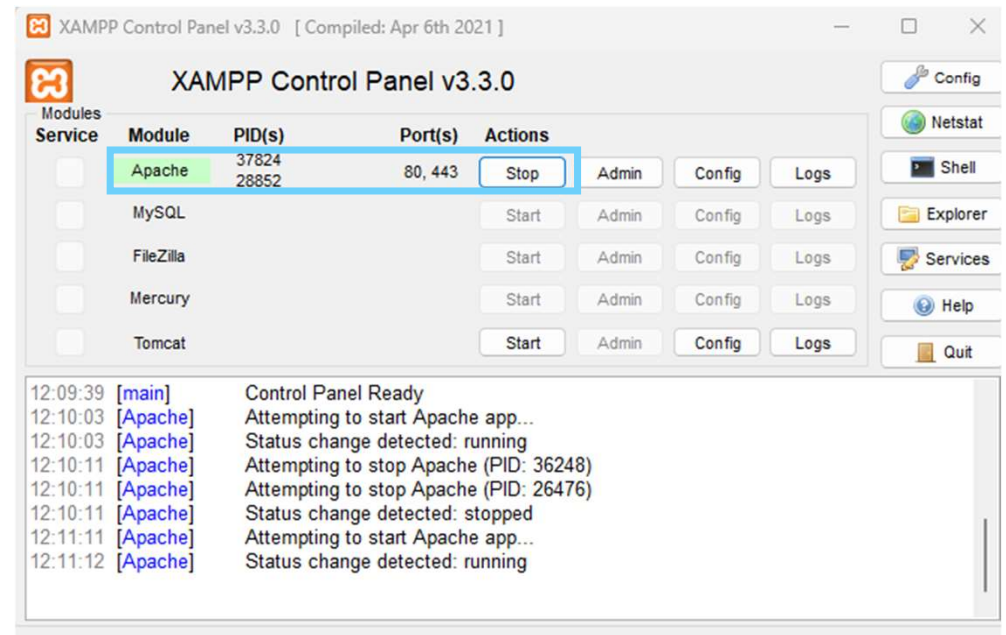
- Afin d'exécuter votre code PHP, vous pouvez utiliser XAMPP qui vous offre le serveur Apache



XAMPP Control Panel v3.3.0 [ Compiled: Apr 6th 2021 ]

Service	Module	PID(s)	Port(s)	Actions	Admin	Config	Logs
<input type="checkbox"/>	Apache			<b>Start</b>	Admin	Config	Logs
<input type="checkbox"/>	MySQL			Start	Admin	Config	Logs
<input type="checkbox"/>	FileZilla			Start	Admin	Config	Logs
<input type="checkbox"/>	Mercury			Start	Admin	Config	Logs
<input type="checkbox"/>	Tomcat			Start	Admin	Config	Logs

12:09:39 [main] The Mercury module is disabled  
12:09:39 [main] Starting Check-Timer  
12:09:39 [main] Control Panel Ready  
12:10:03 [Apache] Attempting to start Apache app...  
12:10:03 [Apache] Status change detected: running  
12:10:11 [Apache] Attempting to stop Apache (PID: 36248)  
12:10:11 [Apache] Attempting to stop Apache (PID: 26476)  
12:10:11 [Apache] Status change detected: stopped



XAMPP Control Panel v3.3.0 [ Compiled: Apr 6th 2021 ]

Service	Module	PID(s)	Port(s)	Actions	Admin	Config	Logs
<input type="checkbox"/>	Apache	37824 28852	80, 443	<b>Stop</b>	Admin	Config	Logs
<input type="checkbox"/>	MySQL			Start	Admin	Config	Logs
<input type="checkbox"/>	FileZilla			Start	Admin	Config	Logs
<input type="checkbox"/>	Mercury			Start	Admin	Config	Logs
<input type="checkbox"/>	Tomcat			Start	Admin	Config	Logs

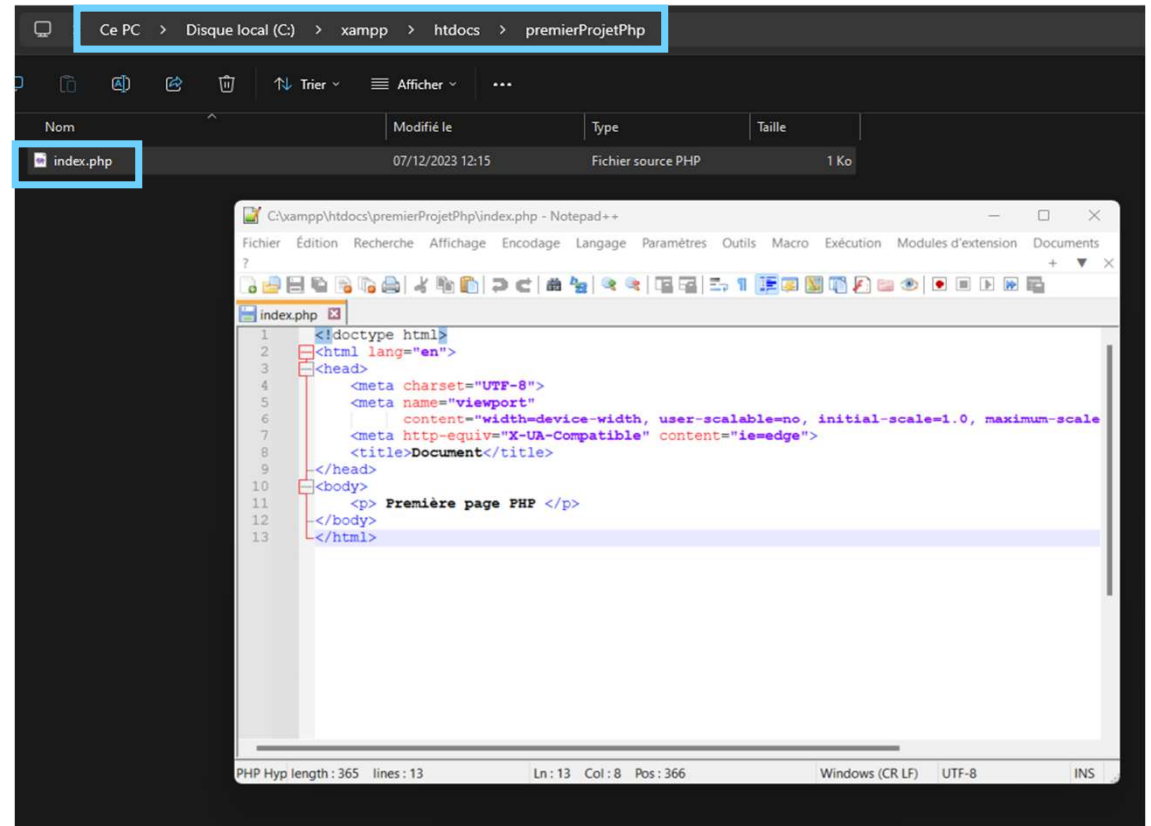
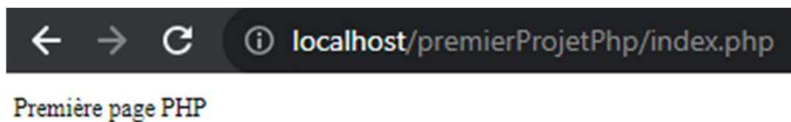
12:09:39 [main] Control Panel Ready  
12:10:03 [Apache] Attempting to start Apache app...  
12:10:03 [Apache] Status change detected: running  
12:10:11 [Apache] Attempting to stop Apache (PID: 36248)  
12:10:11 [Apache] Attempting to stop Apache (PID: 26476)  
12:10:11 [Apache] Status change detected: stopped  
12:11:11 [Apache] Attempting to start Apache app...  
12:11:12 [Apache] Status change detected: running

# PHP

## Lancez votre serveur Web

- Allez dans le dossier htdocs sous votre dossier xampp
- Créer un dossier first
- Créer une page index.php
- Mettez y du code HTML standard
- Maintenant accéder au lien suivant

<http://localhost/nomDossier/nomPage.php>



# PHP

## Lancez votre serveur Web

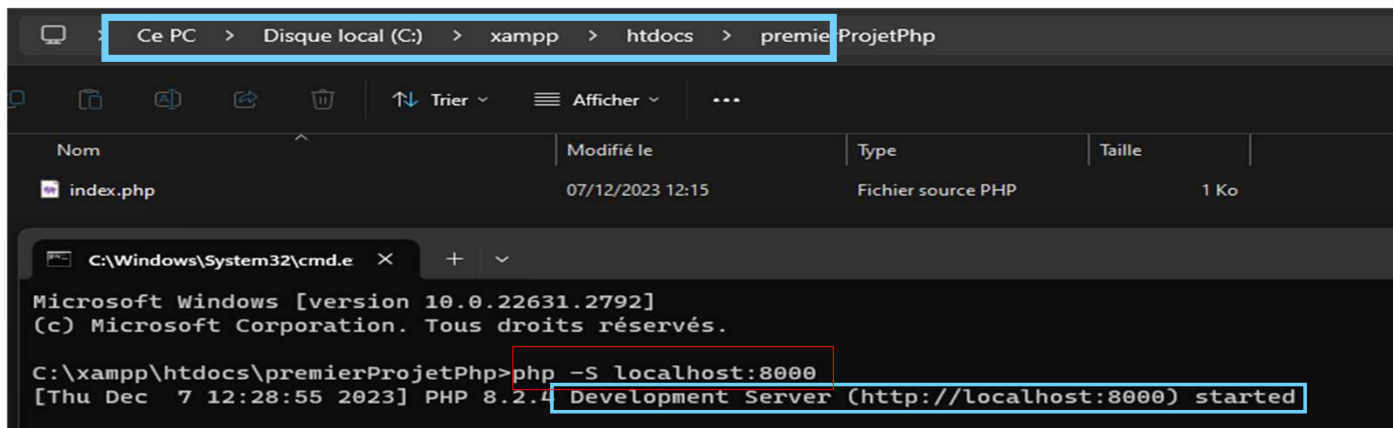
- Vous pouvez aussi lancer un serveur local directement en utilisant php
- Ouvrez le **terminal dans le dossier de votre projet php** et tapez **php -S localhost:8000** (8000 est un numéro de port logiciel libre dans votre projet, avec php on utilise généralement 8000).

The image shows a Windows File Explorer window and a Command Prompt window. The File Explorer window displays the directory structure: Ce PC > Disque local (C:) > xampp > htdocs > premierProjetPhp. The file index.php is listed with a modification date of 07/12/2023 12:15 and a size of 1 Ko. The Command Prompt window shows the command `php -S localhost:8000` being executed, resulting in the message: [Thu Dec 7 12:28:55 2023] PHP 8.2.4 Development Server (http://localhost:8000) started.

# PHP

## Lancez votre serveur Web

- Maintenant et comme vous pouvez le lire sur le terminal, votre serveur est à l'écoute sur le lien localhost:8000

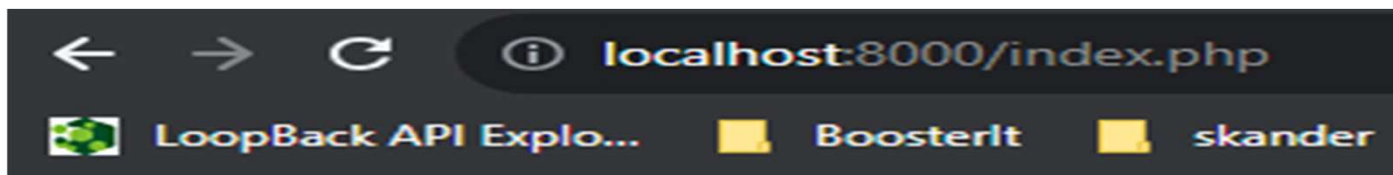


```
Ce PC > Disque local (C:) > xampp > htdocs > premierProjetPhp
```

Nom	Modifié le	Type	Taille
index.php	07/12/2023 12:15	Fichier source PHP	1 Ko

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [version 10.0.22631.2792]
(c) Microsoft Corporation. Tous droits réservés.

C:\xampp\htdocs\premierProjetPhp>php -S localhost:8000
[Thu Dec 7 12:28:55 2023] PHP 8.2.4 Development Server (http://localhost:8000) started
```



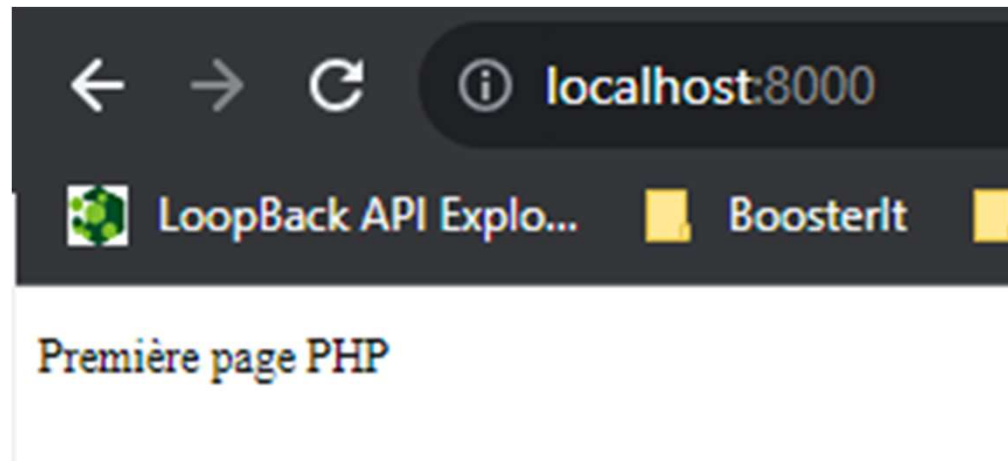
Première page PHP



# PHP

## Lancez votre serveur Web

- Si vous ne mentionnez pas le nom de la page, le serveur ira chercher une page index, s'il la trouve, il l'affiche, sinon il affiche une erreur de type 404.



# PHP

## Intégration PHP et HTML

### Intégration PHP et HTML

#### Principe:

- Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- L'intégration nécessite l'utilisation de balises

Avec le style php: `<?php` ligne de code PHP `?>`

avec le style des ASP : `<%` ligne de code ASP `%>`

# PHP

## Intégration PHP et HTML

### Intégration PHP et HTML

On peut intégrer autant de bloc PHP que nous voulons, il suffit d'ouvrir et de fermer le bloc avec `<?php` et `?>`.

```
<?php
//ligne de code PHP Bloc1
?>
<html>
<head>
  <title>
    <?php
    //ligne de code PHP Bloc 2
    ?>
  </title>
</head>
<body>
  //ligne de code HTML
  <?php
  //ligne de code PHP Bloc 3
  ?>
  //ligne de code HTML
  ....
  <?php
  //ligne de code PHP Bloc 4
  ?>
</body>
</html>
```

# PHP

## Intégration PHP et HTML

### Intégration PHP et HTML

#### Forme d'une page PHP

Intégration « indirecte »

Inclure un fichier PHP dans un fichier HTML : include()

Fichier Principal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
include "information.php" ;
?>
</body>
</html>
```

Fichier à inclure : information.php

```
<?php
    echo « Bonjour je suis FLEN » ;
?>
```

# PHP

## Intégration PHP et HTML

### Intégration PHP et HTML

#### Forme d'une page PHP

Intégration « indirecte »

Inclure un fichier PHP dans un fichier HTML : `include()`

Il existe deux fonctions permettant d'inclure un fichier dans un autre :

- `require`
- `include`

Ces deux fonctions jouent le même rôle cependant la différence réside dans leur gestion des erreurs. En effet, si une erreur arrive lors du chargement, `include` génère un avertissement, et le script continue de s'exécuter. Par contre, `require()` génère une erreur fatale et bloque l'exécution du script.

Il existe aussi les variantes `require_once` et `include_once` qui vérifie si le fichier a été déjà ajouté avant si oui il ne l'ajoute plus.

# Fonctionnalités de base de PHP

- Commentaires : /\* Bloc de commentaires \*/

// Ligne de commentaire

- Insensible à la casse pour **les fonctions** et **pas pour les noms des variables**
- Toutes les variables ont un nom précédé par \$
- Variables non typées à la déclaration (l'affectation détermine le type de la variable)

# Fonctionnalités de base de PHP

## Sensibilité à la casse

### Sensibilité à la casse

Lorsqu'on parle de sensibilité à la casse, on parle de faire la différence entre minuscule et majuscule.

PHP est

➤ Sensible à la casse pour :

- variables
- constants
- clé des tableaux
- attributs de classes
- constantes de classes

# Les variables

## Définition

Les variables peuvent être considérées comme des conteneurs de données.

Les noms de variables commencent avec \$.

Exemple : \$firstname = "aymen";

\$x=2782;

Le nom d'une variable contient uniquement **des caractères alphanumériques** et le tiret bas (A-z, 0-9, et \_).

Le nom de la variable ne peut commencer que par une lettre ou le tiret bas.

PHP est **un langage faiblement typé**. Ceci signifie que la variable en PHP épouse le type de son contenu. Si cette variable contient la valeur 5 alors son type est entier. Si on lui affecte ensuite la chaîne "bonjour" elle aura pour type chaîne. L'affectation d'une valeur à une variable détermine son type.

**Exemple :**

**\$x = 2; // \$x is integer**

**\$x = "hello"; // \$x is string**



# Les variables

## Affichage des variables

- Plusieurs fonctions permettent d'afficher une variable en **PHP**. La plus connue est **echo** qui n'est pas vraiment une fonction et ne se comporte pas comme telle : <http://php.net/manual/fr/function.echo.php>
- Echo permet d'afficher une chaîne de caractères ou tout type convertible en chaîne de caractères.

### Petit détour pour les chaînes :

- Lorsque vous utilisez `"`, vous pouvez insérer directement des variables dans votre chaîne de caractères.
- Si vous avez une variable `$x = "aymen"`, alors `echo "Bonjour $x";` sera affiché : Bonjour aymen.
- En effet, php va chercher dans la chaîne la présence de variable et va les interpréter avant d'afficher la chaîne.
- On peut aussi écrire `echo " Bonjour {$x}";`
- Les guillemets simples annulent quant à elles le remplacement des variables.
- `echo ' Bonjour $x';` sera affiché : Bonjour \$x.
- Dans le cas de guillemets simples, nous concaténons avec l'opérateur `'.`
- `echo 'Bonjour'.$x; //` Bonjour aymen
- Il existe d'autres façons d'afficher des variables avec `print` ou `printf`.
- Faites une petite recherche et testez ces différentes façons.

# Les variables

## Les types

Les principaux types de PHP (non exhaustifs) sont :

- **Les chaînes de caractères** (String en anglais) qui permettent de stocker du texte. Elles sont écrites entre guillemets ou entre apostrophes.
- **Les nombres**
- **Les réels**
- **Les booléens** : qui a pour valeur true ou false et qui permet d'évaluer une expression

# Les variables

## Bonnes pratiques

- Le nom de votre variable doit être significatif. Eviter les abréviations comme pers pour personne.
- Une variable peut être une left value ou lvalue. Ceci s'applique lorsque cette variable se trouve à gauche d'une affectation. Par exemple dans l'instruction `$x = 2;` la variable `$x` est une lvalue, elle joue le rôle d'un récipient dans lequel on va stocker le contenu de la partie droite.
- Lorsque la variable se trouve à droite d'une opération, elle est interprétée et on la remplace par son contenu.

### Exemple:

`$x= 2; //lvalue $x est un récipient`

`$z=$x + 3; // Ici $z est une lvalue, par contre $x est remplacé par sa  
// valeur 2 et on aura finalement dans $z la valeur 5.`

# Les variables

## Existence

- Afin de vérifier l'existence d'une variable, PHP nous offre la fonction **isset**, cette fonction vérifie qu'une variable existe et est non null

```
<?php
$y = null;

if (!isset($x)) {
    echo "\$x n'existe pas" . PHP_EOL;
}
if (!isset($y)) {
    echo "\$y est null";
}
```

- Vous pouvez aussi utiliser l'opérateur coalescent **??** Qui vérifie si un élément existe et non null et qui peut être chaîné

```
$y = null;
$m = null;
$z = 5;
echo $m ?? $y ?? $z;
```

# Les variables

## Les variables dynamiques

- Une **variable dynamique** est une variable qui permet d'avoir **un nom dynamique d'une variable**.
- Elle prend la valeur d'une variable et l'utilise comme nom d'une autre variable.
- Pour déclarer une variable dynamique on utilise **\$\$**.
- Exemple :  
`$ndv = 'varDyn'`  
`$$ndv = 'contenuVarDyn';`
- Quels sont les variables présentes dans ce script ?
- Ici nous avons deux variables **\$ndv** qui contient la chaine **varDyn** et **\$varDyn** qui contient la chaine **contenuVarDyn**.

# Les variables

## Portée des variables

### ➤ Variable locale

Visible uniquement à l'intérieur d'un contexte d'utilisation

### ➤ Variable globale

Visible dans tout le script

Accessible localement avec l'instruction `global` ou avec la variable globale `$GLOBALS`

### Exemple :

```
<?php
$globalVar1 = 1;
$globalVar2 = 2;
function somme() {
    global $globalVar1, $globalVar2;
    $globalVar2 = $globalVar1 +
    $globalVar2;
}
somme();
echo $ globalVar2;
```

```
<?php
$globalVar1 = 1;
$globalVar2 = 2;
function somme() {
    $GLOBALS['globalVar2'] =
    $GLOBALS['globalVar1'] +
    $GLOBALS['globalVar2'];
}
somme();
echo $ globalVar2;
?>
```

# Les variables

## Fonctions de manipulation de variables

### ➤ Fonctions de vérifications de variables

**floatval()** : Convertit une chaîne en nombre à virgule flottante

**empty()** : Détermine si une variable est vide

**gettype()** : Retourne le type de la variable (boolean, integer, double(float), string, array, object, null, unknown type)

**intval()** : Retourne la valeur numérique entière équivalente d'une variable

**is\_Type()** : ( ) , is\_bool(), is\_double(), is\_float(), is\_int(), is\_integer, is\_long(), is\_obj, is\_array(), is\_real(), is\_numeric(), is\_string() vérifie si la variable est de type **Type**.

**isset()** : Détermine si une variable est définie et est différente de NULL

**settype(var, newType)** : Affecte un type à une variable

**strval()** : Récupère la valeur d'une variable, au format chaîne

**unset()** : Détruit une variable

**var\_dump()** : Affiche les informations d'une variable

# Les variables

## Les variables superglobales

### ➤ Variables d'environnement Client

Variable	Description
<code>\$_SERVER["HTTP_HOST"]</code>	Nom d'hôte de la machine du client (associée à l'adresse IP)
<code>\$_SERVER["HTTP_REFERER"]</code>	URL de la page qui a appelé le script PHP
<code>\$_SERVER["HTTP_ACCEPT_LANGUAGE"]</code>	Langue utilisée par le serveur
<code>\$_SERVER["CONTENT_TYPE"]</code>	Type de données contenu présent dans le corps de la requête. Il s'agit du type MIME des données
<code>\$_SERVER["REMOTE_ADDR"]</code>	L'adresse IP du client appelant le script
<code>\$_SERVER["PHP_SELF"]</code>	Nom du script PHP



# Les variables

## Les variables superglobales

### ➤ Variables d'environnement Serveur

Variable	Description
<code>\$_SERVER["SERVER_NAME"]</code>	Le nom du serveur
<code>\$_SERVER["HTTP_HOST"]</code>	Nom de domaine du serveur
<code>\$_SERVER["SERVER_ADDR"]</code>	Adresse IP du serveur
<code>\$_SERVER["SERVER_PROTOCOL"]</code>	Nom et version du protocole utilisé pour envoyer la requête au script PHP
<code>\$_SERVER["DATE_GMT"]</code>	Date actuelle au format GMT
<code>\$_SERVER["DATE_LOCAL"]</code>	Date actuelle au format local
<code>\$_SERVER["\$DOCUMENT_ROOT"]</code>	Racine des documents Web sur le serveur

### ➤ `phpinfo(INFO_VARIABLES );` permet d'afficher les variables d'environnement

# Les constantes

## introduction

- Une constante est un identifiant (un nom) qui représente une valeur simple.
- Une constante ne peut jamais être modifiée durant l'exécution du script.
- Par convention, les constantes sont toujours en majuscules.
- Pour définir une constante dans un script et hors d'une classe PHP on utilise la syntaxe suivante :

```
define("nomDeLaCostante", "valeurDeLaConstante");
```

### Exemple :

```
define("jeSuisUneConstante", "bonjour");
```

# Les constantes

## Les Constantes prédéfinies

PHP propose aussi un grand nombre de constante prédéfinies. Etant données que le nombre est très grand, vous pouvez lister ces variables en utilisant la fonction : `get_defined_constants()` qui liste l'ensemble de ces fonctions.

```
<?php  
    print_r(get_defined_constants());  
?>
```

Nous pouvons voir par exemple la constante `__FILE__` qui informe sur le nom du fichier en cours d'exécution.

`PHP_Version` qui donne la version PHP utilisée ou encore `PHP_OS` qui informe sur l'OS utilisé dans le serveur.

<https://www.php.net/manual/en/function.get-defined-constants.php>

# Fonctionnalités de base de PHP

## Opérations élémentaires

Exemple	Nom	Résultat
+\$a	Identité	Conversion de $\$a$ vers <a href="#">int</a> ou <a href="#">float</a> , selon le plus approprié.
-\$a	Négation	Opposé de $\$a$ .
$\$a + \$b$	Addition	Somme de $\$a$ et $\$b$ .
$\$a - \$b$	Soustraction	Différence de $\$a$ et $\$b$ .
$\$a * \$b$	Multiplication	Produit de $\$a$ et $\$b$ .
$\$a / \$b$	Division	Quotient de $\$a$ et $\$b$ .
$\$a \% \$b$	Modulos	Reste de $\$a$ divisé par $\$b$ .
$\$a ** \$b$	Exponentielle	Résultat de l'élévation de $\$a$ à la puissance $\$b$ . Introduit en PHP 5.6.

# Les structures de contrôles

## Les structures conditionnelles

- L'instruction if  
if (condition réalisée) { liste d'instructions }
- L'instruction if ... Else  
if (condition réalisée)  
{  
    liste d'instructions  
}  
  
else {  
    autre série d'instructions  
}
- L'instruction if ... elseif ... Else  
if (condition réalisée) {liste d'instructions}  
elseif (autre condition ) {autre série d'instructions }  
else (dernière condition réalisée) { série d'instructions }
- Opérateur ternaire  
(condition) ? instruction si vrai : instruction si faux

# Les structures de contrôles

## Les structures conditionnelles

```
if ($temperature < 0) {  
    echo 'il fait très froid';  
} elseif ($temperature < 10) {  
    echo 'la température est gérable';  
} elseif ($temperature < 26) {  
    echo 'Il commence à faire chaud';  
} else {  
    echo 'il fait chaud';  
}
```

# Les structures de contrôles

## Les structures conditionnelles

➤ Une nouvelle variante proposée par PHP7 de La condition ternaire `?:` est l'opérateur `??`, il permet généralement d'affecter des **valeur par défaut** à une variable si la première affectation ne fonctionne pas.

➤ Syntaxe : `$var = $newV1 ?? $newV2 ?? ... ?? $newVn`

Prenons cet exemple

➤ `$maVariable = isset($newVal1)? $newVal1 : $newVal2;`

Ce code va vérifier si `$newVal1` existe et non null, c'est le travail de la fonction `isset`. Si elle existe alors il l'affectera à `$maVariable` sinon il affectera la variable `$newVal2`.

Afin de raccourcir ce traitement qui est assez récurrent on utilise la syntaxe suivante :

`$var = $newVal1 ?? $newVal2;`

On peut enchaîner plusieurs fois et le raisonnement reste le même.

# Les structures de contrôles

## Les structures conditionnelles

L'instruction **switch** permet de compresser un code avec if et elseif. Ici on teste sur la valeur de Variable et selon (case) la valeur qu'elle prend on exécute un bloc d'instructions.

**Switch ne teste que l'égalité.**

Le mot clé break permet d'arrêter le traitement une fois les instructions du bloc sélectionné se terminent.

Sans le mot clé break, toutes les instructions qui suivent seront exécutées.

```
switch (Variable) {  
    case Valeur1:  
        Liste d'instructions;  
        break;  
    case Valeur1:  
        Liste d'instructions.  
        break;  
    case Valeurs...:  
        Liste d'instructions;  
        break;  
    default:  
        Liste d'instructions;  
}
```

```
$x=2;  
switch ($x) {  
    case 1 : echo 'Lundi';  
        break;  
    case 3 : echo 'Mercredi';  
        break;  
    default :  
        echo 'Autre jour';  
}
```



# Les structures de contrôles

## Les structures itératives

➤ Lorsque un traitement se répète plusieurs fois, il est nécessaire d'utiliser les structures itératives.

➤ La boucle for

La syntaxe de la boucle for est la même qu'en C ou en Javascript.

```
for (initialisation; Condition ; incrémentation) {
```

```
    Block d'instructions;
```

```
}
```

Exemple : Affichage des chiffres de 1 à 10, chacun dans une ligne

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

```
    echo $i. '<br>';
```

```
}
```

```
for($i = 0; $i < 10; $i += 2):  
    echo "$i <br>";  
endfor;
```

# Les structures de contrôles

## Les structures itératives

### ➤ Les boucles while et do While

Elles permettent de répéter le bloc d'instruction tant que la condition est vraie.

```
➤ While (condition) {  
    bloc d'instructions ;  
}
```

```
➤ do {  
    bloc d'instructions ;  
} While (condition);
```

➤ La différence entre ces deux boucles est que la première peut n'exécuter aucune instruction. Pour la seconde au moins une instruction est exécutée.

```
while ($x%2 != 0) {  
    echo $x. ' n'est pas divisible par deux<br>';  
    $x=($x*3) - 1;  
}
```

# Les chaines de caractères

## Fonctions prédéfinies

➤ Une chaîne de caractère est une série de caractères.

### **Quelques fonctions prédéfinies pour la gestion de chaînes de caractères**

Pour les détails sur ces fonctions et d'autres fonctions de chaînes consultez le Manuel PHP :

<http://php.net/manual/fr/ref.strings.php>

[chr](#) — Retourne un caractère à partir de son code ASCII

[count\\_chars](#) — Retourne des statistiques sur les caractères utilisés dans une chaîne

[echo](#) — Affiche une chaîne de caractères

[explode](#) — Coupe une chaîne en segments et prends en paramètres le délimiteur et la chaîne.

[fprintf](#) — Écrit une chaîne formatée dans un flux

[htmlentities](#) — Convertit tous les caractères éligibles en entités HTML

[htmlspecialchars\\_decode](#) — Convertit les entités HTML spéciales en caractères

[htmlspecialchars](#) — Convertit les caractères spéciaux en entités HTML

[implode](#) — Rassemble les éléments d'un tableau en une chaîne

[lcfirst](#) — Met le premier caractère en minuscule

[ltrim](#) — Supprime les espaces (ou d'autres caractères) de début de chaîne

[money\\_format](#) — Met un nombre au format monétaire

[nl2br](#) — Insère un retour à la ligne HTML à chaque nouvelle ligne

[ord](#) — Retourne le code ASCII d'un caractère

[print](#) — Affiche une chaîne de caractères

# Les chaînes de caractères

## Fonctions prédéfinies

[printf](#) — Affiche une chaîne de caractères formatés  
[rtrim](#) — Supprime les espaces (ou d'autres caractères) de fin de chaîne  
[sprintf](#) — Retourne une chaîne formatée  
[sscanf](#) — Analyse une chaîne à l'aide d'un format  
[str\\_getcsv](#) — Analyse une chaîne de caractères CSV dans un tableau  
[str\\_repeat](#) — Répète une chaîne  
[str\\_replace](#) — Remplace toutes les occurrences dans une chaîne  
[str\\_shuffle](#) — Mélange les caractères d'une chaîne de caractères  
[str\\_split](#) — Convertit une chaîne de caractères en tableau  
[str\\_word\\_count](#) — Compte le nombre de mots utilisés dans une chaîne  
[strcmp](#) — Comparaison binaire de chaînes  
[strip\\_tags](#) — Supprime les balises HTML et PHP d'une chaîne  
[stripos](#) — Recherche la position de la première occurrence dans une chaîne, sans tenir compte de la casse  
[stripslashes](#) — Supprime les antislashes d'une chaîne  
[strstr](#) — Trouve la première occurrence dans une chaîne  
[stristr](#) — Version insensible à la casse de strstr  
[strlen](#) — Calcule la taille d'une chaîne  
[strncmp](#) — Comparaison binaire des n premiers caractères

# Les chaînes de caractères

## Fonctions prédéfinies

[strpos](#) — Cherche la position de la première occurrence dans une chaîne

[strrchr](#) — Trouve la dernière occurrence d'un caractère dans une chaîne

[strrev](#) — Inverse une chaîne

[stripos](#) — Cherche la position de la dernière occurrence d'une chaîne contenue dans une autre, de façon insensible à la casse

[strrpos](#) — Cherche la position de la dernière occurrence d'une sous-chaîne dans une chaîne

[strtok](#) — Coupe une chaîne en segments

[strtolower](#) — Renvoie une chaîne en minuscules

[strtoupper](#) — Renvoie une chaîne en majuscules

[strtr](#) — Remplace des caractères dans une chaîne

[substr\\_compare](#) — Compare deux chaînes depuis un offset jusqu'à une longueur en caractères

[substr\\_count](#) — Compte le nombre d'occurrences de segments dans une chaîne

[substr\\_replace](#) — Remplace un segment dans une chaîne

[substr](#) — Retourne un segment de chaîne

[trim](#) — Supprime les espaces (ou d'autres caractères) en début et fin de chaîne

[ucfirst](#) — Met le premier caractère en majuscule

[ucwords](#) — Met en majuscule la première lettre de tous les mots

# Les chaînes de caractères

## Les nouveautés de PHP8

### Nouveautés dans PHP8

- **str\_contains** : détermine si une chaîne contient une sous chaîne en prenant en considération la casse.
- **str\_starts\_with** : détermine si une chaîne commence par une sous chaîne en prenant en considération la casse.
- **str\_ends\_with** : détermine si une chaîne se termine par une sous chaîne en prenant en considération la casse.

# Les tableaux

## Introduction

➤ Un tableau est une succession d'éléments de **différents types**

➤ Deux méthodes permettent de créer un tableau :

➤ En affectant à une variable un tableau en utilisant les **[]**

```
$tab = []; // crée un tableau vide
```

```
$tab = [1,'abc']; // crée un tableau avec deux éléments
```

➤ En utilisant la fonction **array()**

```
$tab1 = array(); // crée un tableau vide
```

```
$tab1 = array(1,'abc'); // crée un tableau avec deux éléments
```

➤ Les éléments d'un tableau peuvent **pointer vers d'autres tableaux**

➤ **L'index** d'un tableau en PHP commence de **0**

➤ On **ne définit pas la taille** du tableau

➤ Pour **ajouter/modifier** un élément dans un tableau on utilise la syntaxe suivante : **tab[indice] = valeur.**

➤ Pour supprimer un élément d'un tableau on utilise la fonction **unset** qui prend en paramètre l'élément **supprimer.**

➤ La fonction **count()** pour avoir le nombre d'éléments d'un tableau

➤ Il existe deux types de tableaux

➤ Tableaux indicés

➤ Tableau associatif

# Les tableaux

## Les tableaux indicés

- Tableaux indicés : Les éléments sont accessibles par leur index qui commence par 0 et en utilisant les []. Exemple t[5] affiche l'élément d'indice 5.

### Exemple

```
$cartes =array(1,2,3,4,5,6,7,8,9,10,11,12,13);
```

### Parcours

```
echo "Affichage avec for : <br>;
```

```
for ($i=0;$i<count($cartes);$i++){  
    echo "carte de valeur : ". $cartes[$i]. "<br>;  
}
```

```
echo "Affichage avec foreach : <br>;
```

```
foreach($cartes as $carte){  
    echo "carte de valeur : ". $carte. "<br>;  
}
```



# Les tableaux

## Les tableaux associatifs

➤ Tableau **associatif** : L'indice de chaque élément est une chaîne de caractère.

### Exemple

```
$cartes = array(  
    « As »=>1, « Dous »=>2, « Tris »=>3, « Quatro »=>4, « Chinka »=>5,  
    « six »=>6, « Sept »=>7, « huit »=>8, « neuf »=>9, « Dix »=>10, « Valet »=>11,  
    « Dame »=>12, « Roi »=>13,  
);
```

### Parcours

```
foreach($cartes as $indCarte => $Carte) {  
    echo "La carte ". $indCarte . " de valeur : " . $Carte . "<br>";  
}
```

<http://php.net/manual/fr/language.types.array.php>

# Les tableaux

## Quelques fonctions sur les tableaux

- L'une des fonctionnalités la plus utile dans un tableau est la recherche. Nous pouvons chercher sur deux éléments :
  - Les **clés** avec la fonction `array_key_exists` qui prend en paramètre la clé et le tableau et retourne un booléen indiquant si le tableau contient cette clé ou pas.
  - Les **valeurs** avec la fonction `in_array` qui prend aussi en paramètre l'élément à chercher et le tableau et retourne un booléen indiquant si le tableau contient cet élément ou pas.
  - Nous pouvons aussi récupérer la clé d'un élément dans un tableau s'il existe avec la fonction `array_search`. Si la valeur n'existe pas dans le tableau la fonction renvoi **false**.

# Les tableaux

## Quelques fonctions sur les tableaux

- `$tableau = array_count_values($variable)` retourne un tableau comptant le nombre d'occurrences des valeurs d'un tableau.
- `$tableau = array_diff($var_1, $var_2, ..., $var_N)` retourne dans un tableau contenant les valeurs différentes entre deux ou plusieurs tableaux.
- `$tableau = array_intersect($var_1, $var_2, ..., $var_N)` retourne un tableau contenant les enregistrements communs aux tableaux entrés en argument.
- `$tableau = array_merge($var_1, $var_2, ..., $var_N)` enchaîne des tableaux entrés en argument afin d'en retourner un unique.

# Les tableaux

## Quelques fonctions sur les tableaux

- `$tableau = array_merge_recursive($var_1, $var_2, ..., $var_N)` enchaîne des tableaux en conservant l'ordre des éléments dans le tableau résultant.
- `sort($var)` : tri les valeurs du tableau selon le code ASCII  
Le tableau `initial` est `modifié et non récupérables` dans son ordre original  
Pour les tableaux `associatifs` les `clés seront perdues` et remplacées par un indice créé après le tri.
- `rsort ($var)` tri en ordre inverse des codes ASCII.
- `asort ($var)` trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs
- `arsort ($var)` la même action mais en ordre inverse des codes ASCII
- `natcasesort ($var)` effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)

# Les tableaux

## Quelques fonctions sur les tableaux

➤ `sort($var, $flag)` : tri les valeurs du tableau selon type introduit par `$flag`. Le tableau initial est **modifié et non récupérable** dans son ordre original.

Pour les tableaux **associatifs** les **clés seront perdues** et remplacées par un indice créé après le tri.

Voici quelques exemples de la valeur que peut avoir `$flag` :

➤ **`SORT_REGULAR`** (la valeur par défaut donc si vous ne mettez rien c'est cette valeur qui est prise en considération) : compare les éléments normalement (ne modifie pas les types)

➤ **`SORT_NUMERIC`** : compare les éléments numériquement

➤ **`SORT_NATURAL`** - compare les éléments comme des chaînes de caractères en utilisant l'ordre naturel comme le fait la fonction

<http://php.net/manual/fr/function.sort.php>

# Les tableaux

## Quelques fonctions sur les tableaux

### ➤ Tri personnalisé des tableaux associatif

➤ Sachant que le trie utilise le code ASCII pour comparé les éléments, on peut personnaliser la fonction de comparaison de la manière suivante.

```
function compare($a, $b) {  
    if ($a == $b) {  
        return 0;  
    }  
    return ($a < $b) ? -1 : 1;  
}  
  
// Tableau à trier  
$array = array('a' => 4, 'b' => 8, 'c' => -1,  
    'd' => -9, 'e' => 2, 'f' => 5, 'g' => 3, 'h' => -4);  
print_r($array);  
  
// Trie et affiche le tableau résultant  
uasort($array, 'compare');
```

# Les dates

## Gestion des dates avec PHP

➤ Plusieurs fonctions de gestion des dates.

`getdate()` : Retourne un tableau associatif contenant les informations de date et d'heure du timestamp lorsqu'il est fourni, sinon, le timestamp de la date/heure courante locale.

<http://php.net/manual/fr/function.getdate.php>

`date()` : Retourne une date sous forme d'une chaîne, au format donné par le paramètre format, fournie par le paramètre timestamp ou la date et l'heure courantes si aucun timestamp n'est fourni. En d'autres termes, le paramètre timestamp est optionnel et vaut par défaut la valeur de la fonction `time()`.

<http://php.net/manual/fr/function.date.php>

`mktime()` : Retourne le timestamp d'une date

`mktime(`

`int $hour,`  
`?int $minute = null,`  
`?int $second = null,`  
`?int $month = null,`  
`?int $day = null,`  
`?int $year = null`

`): int|false`

<https://www.php.net/manual/fr/function.mktime.php>

```
$now = date("l d/m/Y");  
$birthday = date("l d/m/Y", mktime(12, 10, 50, 7, 2, 1982));  
  
echo "<br> now: $now <br>";  
echo "<br> birthday: $birthday<br>";
```

# Les dates

## Gestion des dates avec PHP

- d - The day of the month (from 01 to 31)
- D - A textual representation of a day (three letters)
- j - The day of the month without leading zeros (1 to 31)
- l (lowercase 'l') - A full textual representation of a day
- N - The ISO-8601 numeric representation of a day (1 for Monday, 7 for Sunday)
- S - The English ordinal suffix for the day of the month (2 characters st, nd, rd or th. Works well with j)
- w - A numeric representation of the day (0 for Sunday, 6 for Saturday)
- z - The day of the year (from 0 through 365)
- W - The ISO-8601 week number of year (weeks starting on Monday)
- F - A full textual representation of a month (January through December)
- m - A numeric representation of a month (from 01 to 12)
- M - A short textual representation of a month (three letters)
- n - A numeric representation of a month, without leading zeros (1 to 12)
- t - The number of days in the given month
- L - Whether it's a leap year (1 if it is a leap year, 0 otherwise)
- o - The ISO-8601 year number
- Y - A four digit representation of a year
- y - A two digit representation of a year
- a - Lowercase am or pm
- A - Uppercase AM or PM
- B - Swatch Internet time (000 to 999)



# Les dates

## Gestion des dates avec PHP

- g - 12-hour format of an hour (1 to 12)
- G - 24-hour format of an hour (0 to 23)
- h - 12-hour format of an hour (01 to 12)
- H - 24-hour format of an hour (00 to 23)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds, with leading zeros (00 to 59)
- u - Microseconds (added in PHP 5.2.2)
- e - The timezone identifier (Examples: UTC, GMT, Atlantic/Azores)
- I (capital i) - Whether the date is in daylight savings time (1 if Daylight Savings Time, 0 otherwise)
- O - Difference to Greenwich time (GMT) in hours (Example: +0100)
- P - Difference to Greenwich time (GMT) in hours:minutes (added in PHP 5.1.3)
- T - Timezone abbreviations (Examples: EST, MDT)
- Z - Timezone offset in seconds. The offset for timezones west of UTC is negative (-43200 to 50400)
- c - The ISO-8601 date (e.g. 2013-05-05T16:34:42+00:00)
- r - The RFC 2822 formatted date (e.g. Fri, 12 Apr 2013 12:01:05 +0200)
- U - The seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)

# Les fonctions

## Définition

### ➤ Déclaration et appel d'une fonction

```
Function nom_fonction($arg1, $arg2, ...$argn)
{
    déclaration des variables ;
    bloc d'instructions ;
    // en cas de retour de valeur
    return $resultat ;
}
```

### ➤ Fonction avec nombre d'arguments inconnu

- **func\_num\_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
- **func\_get\_arg(\$i)** : retourne la valeur de la variable située à la position \$i dans la liste des arguments passés en paramètres.
  - Ces arguments sont numérotés à partir de 0

# Les fonctions

## Définition

➤ Exemple fonction avec nombre d'arguments inconnu

Ecrire une fonction qui calcule le produit des paramètres qui lui ont passé en argument. Le 0 n'est pas comptabilisé

```
<?php
function produit()
{
    $nbarg = func_num_args();
    $prod=1;
    for ($i=0 ; $i <$nbarg ; $i++)
    {
        $prod *= func_get_arg($i);
    }
    return $prod;
}
echo "le produit est : ", produit (2, 7, 82, 8, 11, 2016), "<br />" ;
?>
```

# Les fonctions

## Les nouveautés PHP7

Avec PHP 7 un nouvel opérateur est apparu comme alternative de la fonction `func_get_args()`. C'est l'opérateur `...`

Reprenons la fonction somme et ajoutons cet opérateur :

```
function somme(...$args){}
```

Nous récupérons maintenant directement dans le tableau que nous avons appelé `$args` l'ensemble des paramètres passé à la fonction sans avoir à utiliser l'ancienne méthode.

# Les fonctions

## Passage de paramètre par référence

### ➤ Passage de paramètre par référence

- Pour passer une variable par référence, il faut que son nom soit précédé du symbole & (exemple &\$a) lors de la définition.

Ecrire une fonction `sommeProduit` qui calcule la somme et le produit de deux entiers passés en paramètre

```
<?
function sommeProduit(&$som, &$prod, $x, $y)
{
    $som=$x+$y;
    $prod=$x*$y;
}
$s=0;
$p=0;
sommeProduit($s,$p,5,2);

echo " somme = $s et prod = $p";
?>
```

### ➤ L'appel récursif

- PHP admet les appels récursifs de fonctions

# Les fonctions

## Typage des paramètres

Depuis PHP 7, il est possible de définir un typage des paramètres et surtout un typage scalaire.

Les types possible sont :

1. int
2. float
3. string
4. bool
5. array
6. Nom de classe
7. Nom d'une interface

La fonction somme par exemple devient la suivante :

```
function somme(int $x, int $y) {  
    return $x + $y;  
}
```

# Les fonctions

## Typage

Il existe deux types de typages en PHP :

### 1. Typage Faible

Dans ce type de typage, en cas où le type de la variable ne correspond pas, un transtypage est alors automatiquement exécuté. Par exemple si on reprend l'exemple de la fonction somme

```
function somme(int $x, int $y){return $x + $y;}
```

Supposons que nous avons l'appel suivant : somme(2.5,3) alors la fonction s'exécutera avec les valeurs 2 et 3. 2 étant le transtypage de 2.5.

### 2. Typage Fort

Appelé aussi typage strict. Ce typage refuse tout paramètre dont le type diffère du type attendu. Appliqué lorsqu'on ajoute en haut du script avec l'appel suivant : declare(strict\_types=1).

On peut aussi avoir un typage sur la valeur de retour.

**Exemple :**

```
int function somme(int $x, int $y){return $x + $y;}
```

# Les fonctions

## Les paramètres par défaut

On peut ajouter des paramètres par défaut aux paramètres de la fonction afin de les prendre en considération en cas où l'utilisateur ne passe pas les paramètres attendus.

Exemple

```
int function somme(int $x=0, int $y=0) {  
    return $x + $y;  
}
```

Dans cet exemple, si on exécute cet appel `somme()`;  
La fonction n'ayant pas reçu de paramètres va utiliser les paramètres par défaut. `$x` et `$y` prendront les valeurs 0. Donc la fonction retournera 0.



# Les fonctions

## Appel dynamique

- Dans certains cas, vous voulez exécuter une fonction dont le nom n'est pas forcément connu à l'avance par le programmeur du script
- L'appel **dynamique** d'une fonction s'effectue en suivant le nom d'une variable contenant le nom de la fonction par des parenthèses

**<?php**

```
$datejour = getdate();
```

```
$heure = $datejour["hours"];
```

```
$minute=$datejour["minutes"];
```

```
function bonjour(){
```

```
    global $heure;
```

```
    global $minute;
```

```
    echo "<b> BONJOUR A VOUS IL EST : $heure H $minute </b> ";
```

```
}
```

```
function bonsoir (){
```

```
    global $heure;
```

```
    global $minute;
```

```
    echo "<b> BONSOIR A VOUS IL EST : $heure H $minute <br />";
```

```
}
```

```
if ($heure <= 17) {
```

```
    $salut = "bonjour";
```

```
}
```

```
else $salut="bonsoir";
```

```
//appel dynamique de la fonction
```

```
$salut();
```

```
?>
```

# Les formulaires

## Introduction

- Les formulaires sont un maillon essentiel et incontournable permettant l'interaction entre un site ou une application web et ses utilisateurs.
- Revenons un peu sur les éléments de bases à maîtriser dans un formulaire pour permettre cet échange.
- La balise permettant de créer un formulaire et la balise form :
- `<form>` `</form>` nous permet donc de délimiter le formulaire. Les attributs associés à cette balise sont :
  - **action** : c'est le fichier dans le serveur qui va traiter les informations saisies.
  - Remarque : Si vous voulez que le fichier qui contient le formulaire soit celui qui le traite, garder le champ **action vide** ou utiliser la variable super globale `$_SERVER` et accéder à la variable `PHP_SELF` : `$_SERVER["PHP_SELF"]`.

# Les formulaires

## Introduction

- **method** : qui prend ou la valeur **post** ou la valeur **get** et qui détermine la d'envoi des données vers le serveur. La méthode **get**, qui est la méthode par défaut, présente l'inconvénient d'ajouter les données du formulaire à l'adresse URI du fichier qui les traite, ce qui les rend visibles par le visiteur. Cet inconvénient peut être exploité pour passer des données à un script dès son appel. De plus, il existe une limite à la longueur des URI et donc à la quantité de données à transmettre. Ces problèmes ne se retrouvent pas avec la valeur **post**.
- **name** : nom du formulaire, utile surtout en JS.
- **enctype** : identifie le type d'encodage des données transmises au serveur. Généralement ce champ n'est pas ajouté et il prend donc sa valeur par défaut ("**application/x-www-form-urlencoded**"). En cas d'envoi de fichier du client vers le serveur (upload image par exemple), l'encodage doit être "**multipart/form-data**".

# Les formulaires

## Récupérer les variables envoyées par un formulaire

- Afin de récupérer une variable envoyée à travers un formulaire, nous utilisons les variables globales `$_POST` et `$_GET` selon la méthode utilisée lors de l'envoi et le nom de la variable envoyée (le contenu de l'attribut `name` qui est **OBLIGATOIRE**).
- Lorsque vous utilisez un champ `file` pour envoyer des fichiers vers le serveur, vous devez utiliser la variable `$_FILES` qui utilise les mêmes règles que `post` et `get` pour l'accès à une donnée.
- Ces variables sont des tableaux associatifs.
- **Exemple :**
  - Pour récupérer la variable de nom « nom » et envoyée par `post` on utilise :  
`$_POST['nom'];`

# Les formulaires

## Récupérer les variables envoyées par un formulaire

- Soit la page HTML suivante qui permet d'envoyer le nom de l'utilisateur à la page Bonjour.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<form action="bonjour.php" method="post">
  <input type="text" name="nom">
  <input type="submit" value="envoyer">
</form>
</body>
</html>
```

# Les formulaires

## Récupérer les variables envoyées par un formulaire

- La page « `bonjour.php` » va recevoir les données envoyées par la page `bonjour.html` dans la variable `$_POST`. Ensuite elle va dire bonjour au nom envoyé par le formulaire

```
<html>
<head>
  <title>Bonjour</title>
</head>

<body>
  Bonjour <?= $_POST[ 'nom' ] ?>
</body>
</html>
```

# Exercice

## Récupérer les variables envoyées par un formulaire

- Modifier le code du formulaire en changeant la méthode de post vers get.
- Envoyer votre requête et vérifier ce qui se passe au niveau de l'URL.

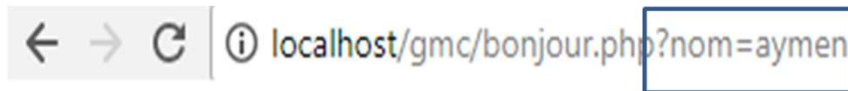
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <form action="bonjour.php" method="get">
    <input type="text" name="nom">
    <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

<http://localhost/gmc/bonjour.php?nom=aymen>

# Exemple

## Récupérer les variables envoyées par une URL

- Si vous avez seulement modifié la méthode post vous allez avoir cet affichage



Bonjour

**Notice:** Undefined index: nom in C:\xampp\htdocs\gmc\bonjour.php on line 18

- La première constatation est que l'URL contient le paramètre envoyé. C'est la spécificité du **get** et ça ne marche pas qu'avec les formulaire en effet vous pouvez créer votre URL et y injecter des données en utilisant cette syntaxe.
- La seconde constatation est l'erreur qui apparait et c'est normal vu que vous n'avez pas modifié la variable super globale que vous utilisez. Vous devez utiliser maintenant `$_GET` et non `$_POST`.
- Ajoutez un champ prénom et vérifiez votre URL pour savoir comment injecter plusieurs paramètres à une URL.



# Formulaire

## Récupérer les variables envoyées par un formulaire

- Ne faites jamais confiance aux données envoyées par l'utilisateur. Vérifier les données côté serveur même si vous les avez validé avec du code JS côté client.
- L'une des failles les plus connues est la faille XSS (cross-site scripting). Elle consiste à injecter du code HTML contenant du code JS dans vos pages afin qu'il soit exécuté par les visiteurs du site.
- Pour éviter un tel problème, deux solutions peuvent être utilisées :
  - Echapper le code HTML reçu en utilisant la fonction `htmlspecialchars` qui convertit les caractères spéciaux en entités HTML (<http://php.net/manual/fr/function htmlspecialchars.php>).
  - Supprimer le code HTML avec `strip_tags` qui supprime les balises HTML et PHP d'un texte (<http://php.net/manual/fr/function strip-tags.php>).

# Files

## Récupérer les variables envoyées par un formulaire

- Nous allons maintenant ajouter un champ de type file qui permet d'ajouter une copie de la carte d'adhérent du demandeur de la commande afin d'assurer une certaine crédibilité de la commande.
- Le traitement de ce type de fichier est particulier, en effet on ne transfère plus des types scalaires mais des fichiers qui peuvent être exécutés dans votre serveur.
- On récupère donc du côté du serveur les informations sur cet élément à travers la variable super globale `$_FILES` qui contient les informations sur le fichier.

# Récupérer les variables envoyées par un formulaire

## Gestion des fichiers

- Pour pouvoir envoyer des fichiers dans un formulaire il faut ajouter `enctype="multipart/form-data"` dans la balise form.
- Afin de récupérer les propriétés du fichier à uploader, on utilise la variable globale `$_FILES` au lieu de `$_POST` ou `$_GET`. Cette variable offre plusieurs informations sur le fichier dont l'information concernant son emplacement.
- Pour copier un fichier dans le serveur on utilise la fonction `copy` qui prend en paramètre le chemin source suivi de la destination.

<https://www.php.net/manual/fr/function.copy.php>

# FILES

## Récupérer les variables envoyées par un formulaire

Supposons que le nom du champ file est fichier, l'accès à la variable `$_FILES['fichier']` retourne un tableau associatif contenant l'ensemble des informations sur le fichier uploadé.

- **Name** : le nom du fichier sur le poste client.
  - **Type** : Type MIME du fichier (permet de contrôler le type des fichiers à accepter)
  - **Size** : taille en octet du fichier
  - **Tmp\_name** : nom du fichier temporaire. En effet, le fichier est mis temporairement dans un fichier défini par la directive "upload\_tmp\_dir" du fichier php.ini. Si vous n'enregistrez pas le fichier, il sera perdu.
- 
- Si vous voulez enregistrer le fichier dans votre serveur vous pouvez utiliser la fonction
  - **move** (`path_fich_temp`, `path_fich_src`):
  - Vous pouvez utiliser **file\_exists** qui Vérifie si un fichier ou un dossier existe en cas de besoin.

# FILES

## Récupérer les variables envoyées par un formulaire

- Si vous voulez permettre un upload multiple, il vous suffit d'utiliser la même logique que pour les checkbox cad que le nom de votre champ va être suivi de `[]` pour l'informer de stocker les informations dans un tableau.
- **Remarque : Préparer un dossier pour y mettre vos uploads.**

# Redirection

- Généralement, vous allez être appelé à faire des redirections dans vos scripts, imaginez par exemple un script d'authentification ou en cas de succès vous voulez rediriger votre utilisateur vers sa page d'accueil.
- Pour ce faire, PHP nous offre la méthode header
- header prend en premier paramètre une chaîne de caractère, qui et pour spécifier une redirection doit être composé du mot clé Location suivi de : puis la page vers laquelle vous voulez rediriger votre utilisateur

```
header('location:home.php');
```

# Les cookies

## Introduction

- Lorsque nous voulons **conserver des informations tout au long de la navigation d'un utilisateur sur le site**, de conserver **ses habitudes**, des informations utiles ect, on utilise les **cookies et les sessions**.
- Un cookie est un **fichier texte** enregistré **côté client**.
- Il permet d'enregistrer des informations utiles sur et pour le client qui sont généralement utilisées pour ses prochaines visites
- Pour des raisons de sécurité, les cookies **ne peuvent être lus** que par les pages du **serveur créateur** du cookie en question.
- La **date d'expiration** des cookies est définie par le **serveur** web qui les a créés.
- Les cookies disponibles sont importés par PHP sous forme de variables identifiées **sous les noms utilisés par ces cookies**
- La variable globale du serveur **\$\_COOKIES** enregistre tous les cookies qui ont été définis.

# Les cookies

## Exemple d'utilisation

- Mémorisation des paniers dans les applications d'e-commerce pour une prochaine visite
- Identification des utilisateurs
- Des pages web individualisées
- Afficher des menus personnalisés
- Afficher des pages adaptées aux utilisateurs en fonction de leurs précédentes visites.



# Les cookies

## Manipulation

➤ `bool setcookie (string $name, string $value, int $expire = , string $path , string $domain , bool $secure = false , bool $httponly = false )`

- Tout les arguments sauf `$name` ne sont pas obligatoire
- `$name` : nom du cookie
- `$value` : contenu du cookie, n'y stocker pas de mot de passes ou des propriétés critiques ou importantes
- `$expire` : Le temps après lequel le cookie expire. C'est un timestamp Unix, donc, ce sera un nombre de secondes depuis l'époque Unix (1 Janvier 1970). Il faut donc fixer cette valeur à l'aide de la fonction `time()` qui permet d'avoir le timestamp actuel en y ajoutant le nombre de secondes après lequel on veut que le cookie expire. Si le paramètre est omis, le cookie n'est valable que le temps de la connexion du visiteur sur le site *Exemple* : `time()+60*60*24*365` fera expirer le cookie dans 1 an.
- `Secure` : est une valeur de type `boolean` qui vaut `TRUE` si le cookie doit être transmis par une connexion sécurisée (`https://*`) et `FALSE` dans le cas contraire. C'est la valeur par défaut.

**Remarque** : Vous pouvez écrire plusieurs valeurs sous un même nom de cookie en utilisant la notation à crochets des tableaux. Cependant le nom ne prend pas de " Exemple : `setcookie("client[pays]");`

# Les cookies

## Accéder à un cookie

- Pour accéder à un cookie, il faut se souvenir qu'il est stocké dans la variable super globale `$_COOKIES` ce qui fait qu'on lui accède comme n'importe quelle variable dans un tableau associatif via son nom.
- Exemple `$_COOKIES['nom']` permet d'accéder au cookie nom.

# Les cookies

## Supprimer un cookie

- Il n'existe pas une fonction dédiée à la suppression d'un cookie.
- Deux solutions peuvent être utilisées :
  - Renvoyer le cookie grâce à la fonction `setcookie()` en spécifiant uniquement le `nom du cookie à supprimer`.
  - Envoyer un cookie dont la `date d'expiration est passée` en spécifiant par exemple `time()-1`

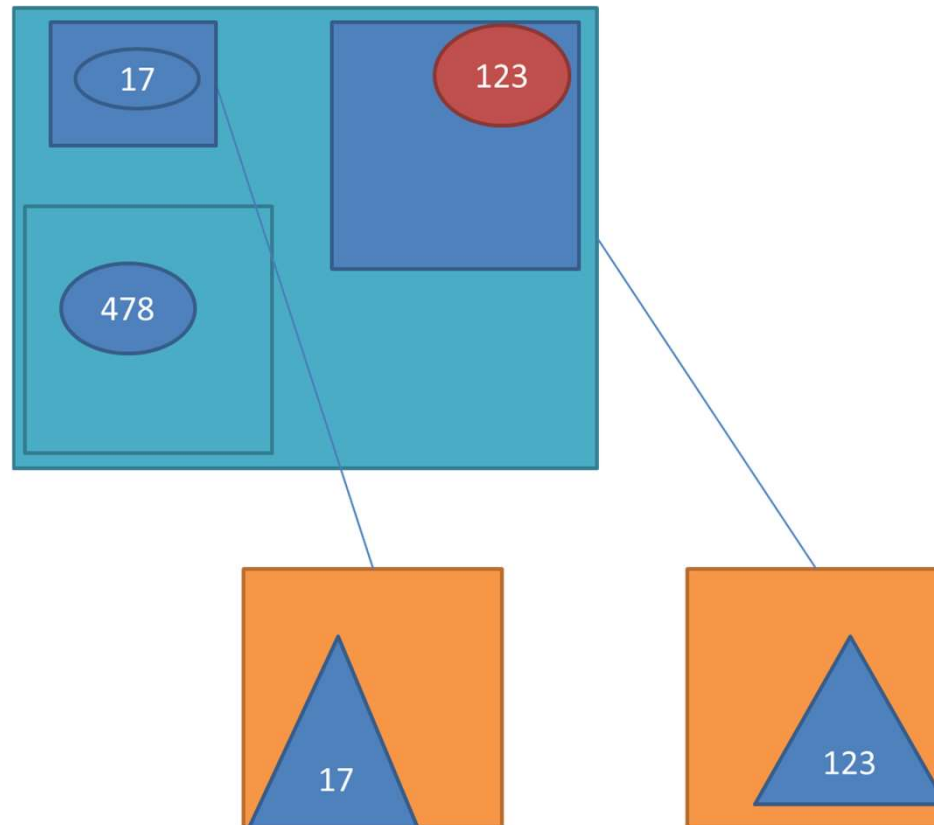
# Les sessions

## Introduction

- Mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
- Stocké côté serveur.
- Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web
- Des données spécifiques à un visiteur pourront être transmises de page en page afin d'adapter personnellement les réponses d'une application PHP
- Chaque session est identifiée par un numéro d'identification dénommé identifiant de session (SID)

# Les sessions

## Introduction



# Les sessions

## Fonctionnement

- Afin de gérer vos sessions, un répertoire est créé sur le **serveur** à l'emplacement désigné par le fichier de configuration **php.ini**, afin de recueillir les données de la nouvelle session.

```
session.save_path = "c:/wamp/tmp";
```

- Le fichier php.ini peut également préciser la durée de vie d'une session en seconde par **session.gc\_maxlifetime**

```
session.gc_maxlifetime = 1800
```

# Les sessions

## Utilité

### ➤ Sécurisé l'accès à vos pages

- Conserver le user authentifié
- Restreindre certaines fonctionnalités à des rôles particulier
- Passer un « token » pour vérifier qu'on suit un même process.

### ➤ Conserver des données le long de la visite d'un utilisateur

- Ces coordonnées
- Le contenu d'un panier pour un site de e-commerce

# Les sessions

## Création

- Afin de **créer une session** on utilise la fonction **session\_start()**
- Cette fonction **doit** être appelée à **chaque début de page avant le code HTML.**
- Si une session existe déjà, cette fonction va charger les données de la session dans le tableau `$_SESSION`. En effet elle va aller récupérer la session qui est sérialisée au niveau du serveur et va la désérialiser et irriguer la variable `$_SESSION`.
- Si la session n'existe pas :
  - Elle créera une session Vide.
  - Un cookie `PHPSESSID` sera créé avec l'identifiant de la session

<https://www.php.net/manual/fr/function.session-start.php>



# Les sessions

## Manipuler la session

- Pour récupérer ou ajouter une variable de session utiliser la variable **superglobale \$\_SESSION**

```
// Affecter une variable à la session
$_SESSION['nbVisite'] = 5;
// Récupérer une variable de la session
echo "J'ai visité la page : {$_SESSION['nbVisite']} fois";
```

# Les sessions

## Suppression

- Pour supprimer les variables de sessions utiliser la fonction `session_unset()`
- Cette fonction détruit toutes les variables d'une session
- Vous pouvez directement vider la variable `$_SESSION`
- Il est conseillé de passer par la syntaxe suivante : `$_SESSION = []`

<https://www.php.net/manual/fr/function.session-unset.php>

The background is a dark blue overlay on a photograph of a business meeting. Several hands are visible, some holding pens and pointing at documents. The documents contain various charts, including bar charts, pie charts, and line graphs. Overlaid on the image are several geometric elements: a solid white diagonal line from the top left to the middle right, a dashed white line from the top left to the middle left, a solid white diagonal line from the bottom right to the middle left, a light blue plus sign in the upper right, a light blue plus sign in the lower left, and a light blue lightning bolt in the bottom right corner.

**MERCI.**