

# PHP : introduction

**Achref El Mouelhi**

Docteur de l'université d'Aix-Marseille  
Chercheur en programmation par contrainte (IA)  
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`



# Plan

- 1 Introduction
- 2 Installations
  - WAMP
  - IDE
- 3 Avant de commencer
- 4 Premier projet web PHP
- 5 Premier script PHP
  - Écriture dans le terminal
  - Arguments d'un script
  - Lecture depuis le terminal
- 6 Projet web PHP avec Docker
- 7 Commentaires

## PHP

- Initialement pour **P**ersonal **H**ome **P**age ensuite pour **PHP** : **H**ypertext **P**reprocessor
- Langage de programmation open-source
  - orienté-objet et procédural
  - interprété
  - faiblement typé
- Créé en 1994 par **Rasmus Lerdorf** dans le cadre d'un projet personnel pour gérer les visiteurs de son site web
- Syntaxe très proche du **C** (procédural), **C++** (procédural, orienté-objet) et **Java** (orienté-objet)

## PHP, pourquoi ?

- Langage de haut niveau (pas de gestion de mémoire, pas d'allocation dynamique, pas de pointeur... comme en **C** et **C++**)
- Facile à apprendre et à utiliser
- Nombreuses documentations, supports vidéos, plusieurs exemples sur internet
- Énorme communauté : un des langages les plus utilisés dans le monde
- Permettant de développer rapidement des programmes portables : **Windows**, **Mac OS**, **Linux**...
- Compatible avec une large gamme de serveurs web : **Apache**, **Nginx**, **IIS**

## Quel type d'application ?

- Des applications utilisables en ligne de commande (scripting)
- La partie serveur pour
  - applications web
  - applications mobiles
  - sites web dynamiques
  - Services web

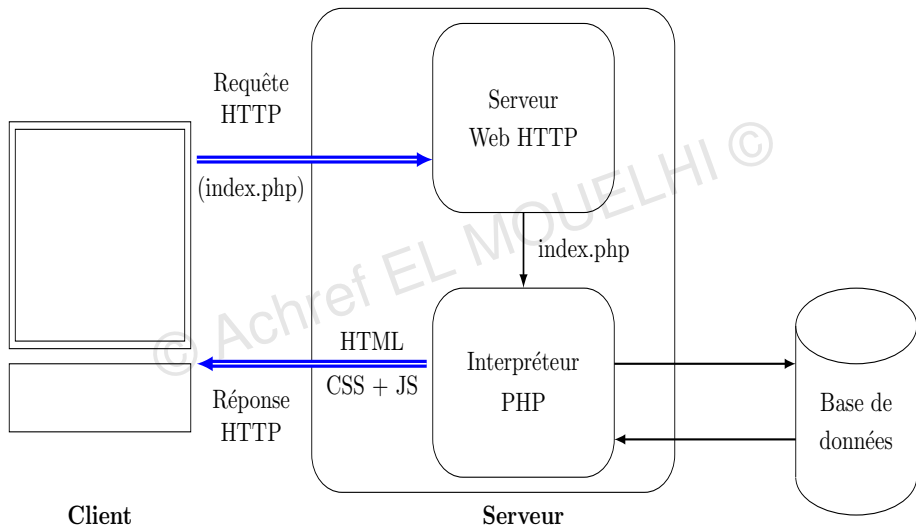
## Quelques technologies concurrentes à PHP

- **JSP** (**J**ava **S**erver **P**ages) : contenu dynamique écrit en **Java**
- **ASP.NET** (**A**ctive **S**erver **P**ages) : contenu dynamique pouvant être écrit dans tous les langages de la plateforme **.NET** (**C#**, **VB.NET**...)

## Dates de sorties de quelques versions de PHP

- 8 juin 1995 : Version 1.0.
- 1 novembre 1997 : Version 2.0.
- 6 juin 1998 : Version 3.0.
- 22 mai 2000 : Version 4.0.
- 13 juillet 2004 : Version 5.0.
- 3 décembre 2015 : Version 7.0.
- 1 décembre 2016 : Version 7.1.
- 30 novembre 2017 : Version 7.2.
- 6 décembre 2018 : Version 7.3.
- 21 novembre 2019 : Version 7.4.
- 26 novembre 2020 : Version 8.0.
- 25 novembre 2021 : Version 8.1.
- 8 décembre 2022 : Version 8.2.
- 23 novembre 2023 : Version 8.3.
- 21 novembre 2024 : Version 8.4.

## PHP





### De quoi on a besoin (le minimum) ?

- Un éditeur de texte (**Bloc-notes**, **Notepad++**, **Sublime Text...**)
- Un serveur web (**Apache**)
- Un interpréteur du langage **PHP**
- Et probablement un **SGBD** (généralement **MySQL**)

## WAMP, pourquoi ?

**WAMP** intègre :

- Un serveur web (**Apache**)
- Un interpréteur du langage **PHP**
- Et un système de gestion de base de données (**MySQL** et **MariaDB**)

## Avant d'installer **WAMP**

Il faut aller à <https://wampserver.aviatechno.net/>,  
télécharger **All VC Redistribuable Packages (x86\_x64) (32 & 64bits)**  
**MD5** puis décompresser et tout installer.

## Première utilisation de **WAMP**

- Démarrer **WAMP**
- Cliquer sur **WAMP** dans la barre de démarrage et choisir Redémarrer les services
- Si l'icône de **WAMP** n'est pas en vert, aller vérifier  
<http://forum.wampserver.com/read.php?1,88043>

### Pour intégrer **PHP 8.2** avec **WAMP 3.3.3**

- Aller à <https://wampserver.aviatechno.net/> et télécharger puis installer **Wampserver update 3.3.3 MD5**
- Aller à <https://wampserver.aviatechno.net/> et télécharger puis installer **PHP 8.3.0 64 bit x64 MD5**
- Aller à <https://wampserver.aviatechno.net/> et télécharger puis installer **MySQL 8.2.0 64 bit x64 MD5**
- Aller à <https://wampserver.aviatechno.net/> et télécharger puis installer **Apache 2.4.58 64 bit x64 MD5**
- Redémarrer les services de **WAMP**

### Quelques éléments dans le menu de démarrage de **WAMP**

- localhost : page de démarrage de **WAMP**
- phpMyAdmin : page web permettant la gestion des bases de données **MySQL**
- Répertoire `www` : emplacement des projets **PHP** sur le disque dur
- ...

## Utiliser un Environnement de développement intégré (IDE) ?

- Console auto-intégrée
- Auto-complétion
- Auto-compilation
- Coloration syntaxique
- Meilleure structuration du projet

## Quelques **IDE** pour **PHP**

- **Visual Studio Code**
- Web Storm
- PHP Storm
- Eclipse
- ...



## Visual Studio Code (ou VSC), pourquoi ?

- Gratuit
- Pouvant s'adapter selon le langage de programmation
- Extensible via l'installation de quelques centaines d'extensions

© Achret L

## Visual Studio Code (ou VSC), pourquoi ?

- Gratuit
- Pouvant s'adapter selon le langage de programmation
- Extensible via l'installation de quelques centaines d'extensions

## VSC : téléchargement

<https://code.visualstudio.com/download>

## Quelques raccourcis pour VSC

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `ctrl` `:`
- Pour faire une sélection multiple : `Ctrl` `f2`
- Pour dupliquer une sélection : `Ctrl` `d`
- Pour placer le curseur dans plusieurs endroits différents : `Alt`
- Pour consulter la liste des raccourcis : `Ctrl` `k` + `Ctrl` `s`

## Extensions **VSC** pour **PHP**

- **PHP Intelephense** pour l'auto-complétion et le formatage du code.
- **PHP Debug** pour le débogage.

**FIG (Framework Interoperability Group)**

- Un groupe de travail
- Fondé par les développeurs des frameworks **PHP** majeurs (**Symfony**, **Laravel**, **Zend**)
- Gérant les normes et recommandations pour améliorer l'interopérabilité entre les frameworks et bibliothèques **PHP**
- Objectif : créer des standards communs afin d'éviter la fragmentation de l'écosystème **PHP**

## PSR (PHP Standards Recommendations)

- Un ensemble de recommandations standardisées définies par le **PHP-FIG**
- 27 **PSR** publiés, couvrant divers aspects du développement **PHP**
- Objectif : un code **PHP** plus standardisé, modulaire et compatible entre frameworks et bibliothèques.

## Les plus importantes **PSR**

- **PSR-1 / PSR-12** (Style de code)
- **PSR-4** (Autoloading)
- **PSR-5** : Standard en cours pour la documentation **PHPDoc**
- **PSR-7 / PSR-15** (Requêtes, réponses, middlewares)
- **PSR-3** (Logging)
- **PSR-6 / PSR-16** (Cache)
- **PSR-11** (Dépendances)

## Règles de nommage en **PHP**

- Pour les classes : **Pascal case**
- Pour les fonctions et variables : **Snake case**
- Pour les attributs et méthodes : **Camel case** ou **Snake case**
- Pour les constantes : **All Caps Snake case**
- Pour les noms de projets : **Kebab case**



## Règles de nommage en **PHP**

- Pour les classes : **Pascal case**
- Pour les fonctions et variables : **Snake case**
- Pour les attributs et méthodes : **Camel case** ou **Snake case**
- Pour les constantes : **All Caps Snake case**
- Pour les noms de projets : **Kebab case**

## Remarque

Certains anciens éléments **PHP** prédéfinis ne respectent pas ces règles.

Quelques recommandations **PSR**

- **PSR-12** recommande 120 caractères max pour plus de lisibilité.
- **PSR-2** impose l'accolade sur la ligne suivante (pour fonctions et classes).
- **PSR-12** exige une ligne vide après la déclaration de `namespace` et les `use`.
- **PSR-12** précise que les opérateurs logiques (`&&`, `||`, `.`) doivent être précédés et suivis d'un espace.
- Les listes d'arguments longues doivent être sur plusieurs lignes avec indentation.
- **PSR-1** impose que tous les fichiers **PHP** commencent par `<?php` ou `<?=' sans balises de fermeture ?> si ce n'est que du PHP.`

## Le code **PHP**

- Les fichiers contenant un code **PHP** doivent avoir l'extension `.php`
- Un bloc de code **PHP** est situé entre les deux balises `<?php ... ?>`
- Une page **PHP**  $\equiv$  page **HTML** contenant une ou plusieurs balises **PHP**.
- **PHP** est sensible à la casse (sensitive-case).

## Instructions

- Chaque instruction se termine par ;
- Il est possible d'écrire plusieurs instructions sur une même ligne (**mais** ce n'est pas une bonne pratique)

## Comment créer un projet sous **VSC** ?

- Créez un répertoire `premier-cours-php` dans le répertoire `www`
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `premier-cours-php`
- Dans `premier-cours-php`, créez un fichier `index.php`

# PHP

Ajouter le contenu suivant dans `index.php`

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
  <title>Cours PHP</title>
</head>

<body>
  <?php
    echo "Hello world";
  ?>
</body>

</html>
```

## Fonctions d'affichage : `echo` et `print`

- utilisables avec et sans parenthèses.
- `echo` n'a pas de valeur de retour.
- `echo` sans parenthèse peut accepter plusieurs paramètres séparés par des virgules.
- `print` retourne toujours 1, donc elle peut être utilisée dans des expressions.
- `print` n'accepte qu'un seul paramètre.
- `echo` est légèrement plus rapide que `print`.

## Comment voir le résultat de cette page ?

- Ouvrir un navigateur
- Aller à `localhost/premier-cours-php/index.php`



# PHP

Ajouter le contenu suivant dans un fichier `script.php`

```
<?php
    echo "Hello world";
?>
```

# PHP

**Pour exécuter le script précédent, lancez la commande suivante depuis un terminal**

```
php chemin/vers/script.php
```

© Achref EL MOU

# PHP

Pour exécuter le script précédent, lancez la commande suivante depuis un terminal

```
php chemin/vers/script.php
```

## Remarque

Si la commande `php` n'est pas reconnue, alors pensez à rajouter le chemin `c:\wamp64\php\bin` aux variables de chemin de **Windows**.

# PHP

**Pour exécuter un script PHP en lui envoyant des paramètres**

```
php chemin/vers/script.php param1 param2 param3
```

© Achref EL MOUËL

# PHP

Pour exécuter un script PHP en lui envoyant des paramètres

```
php chemin/vers/script.php param1 param2 param3
```

Pour afficher les arguments envoyés (code de `script.php`)

```
<?php
    echo "Hello world";
    print_r($argv);
?>
```

# PHP

Pour lire une valeur saisie dans le terminal, on utilise `readline`

```
<?php
    $nombre = readline("Entrez un nombre : ");
    echo "Vous avez saisi : $nombre";
?>
```

## Comment créer un projet sous **VSC** ?

- Créez un répertoire `premier-cours-php` dans le répertoire `www`
- Lancez **VSC** et allez dans `File > Open Folder...` et choisissez `premier-cours-php`
- Dans `premier-cours-php`, créez un fichier `index.php`, `Dockerfile` et `docker-compose.yml`

# PHP

Ajouter le contenu suivant dans `index.php`

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
  <title>Cours PHP</title>
</head>

<body>
  <?php
    echo "Hello world";
  ?>
</body>

</html>
```



## Le contenu suivant dans Dockerfile

```
# Utiliser l'image officielle de PHP avec Apache
FROM php:8.2-apache
```

```
# Copier le code source dans le conteneur
COPY . /var/www/html/
```

```
# Exposer le port 80 pour accéder à l'application
EXPOSE 80
```

**Le contenu suivant dans** `docker-compose.yml`

```
version: "3.8"
services:
  php:
    build: .
    ports:
      - "8080:80"
    volumes:
      - ../var/www/html
```

**Pour démarrer les conteneurs, lancez la commande suivante depuis un terminal**

```
docker-compose up -d
```

© Achref EL MOU

# PHP

**Pour démarrer les conteneurs, lancez la commande suivante depuis un terminal**

```
docker-compose up -d
```

Comment voir le résultat de cette page ?

- Ouvrir un navigateur
- Aller à `localhost:8080`

## En **PHP**

Trois types de commentaires

## Commentaire sur une seule ligne (C++-style)

```
// commentaire
```

© Achref EL MOUELHI ©

## Commentaire sur une seule ligne (C++-style)

```
// commentaire
```

### Raccourcis VSC

- Pour commenter : `Ctrl` + `k` ensuite `Ctrl` + `c`
- Pour décommenter : `Ctrl` + `k` ensuite `Ctrl` + `u`
- Ou aussi : `Ctrl` + `:` pour commenter et décommenter

**On peut aussi utiliser # pour les commentaires mono-lignes**

```
# commentaire
```

© Achref EL MOUËL



On peut aussi utiliser # pour les commentaires mono-lignes

```
# commentaire
```

### Remarque

Depuis **PHP 8**, la séquence # [text] est utilisée pour les attributs.

## Commentaire sur plusieurs lignes (C-style comment)

```
/* le commentaire  
   la suite  
   et encore la suite  
*/
```

© Achref EL M.

## Commentaire sur plusieurs lignes (C-style comment)

```
/* le commentaire  
   la suite  
   et encore la suite  
*/
```

### Raccourci VSC

Pour commenter ou décommenter : Alt + Shift + a

## Commentaire pour documentation

```
/**  
@author Achref El Mouelhi  
*/
```