

Les fondamentaux du web (HTML, CSS, PHP)

Partie 3 – Le PHP

Valentin RIBEZZI

```
_XML', false);  
"5.2", PHP_VERSION, ">")) {  
    greater is required!!!");  
  
ed("pcre")) {  
    requires the pcre extension to php in on  
);  
  
ROOT.'/includes/autoloader.inc.php';  
  
tion  
_ROOT.'/config.php';  
  
I_CONFIG_FILE') || !defined('PSI_DEBUG'  
template("/templates/html/error_config.  
fetch();  
  
out javascript
```

Sommaire

- I. Présentation du PHP
- II. Outils pour le développement
- III. Syntaxe et structures de base
 - A. *Notions de base et inclusion des fichiers*
 - B. *Variables, types de données et opérateurs*
 - C. *Structures de contrôle*
 - D. *Fonctions*

```
_XML', false);  
"5.2", PHP_VERSION, ">")) {  
    greater is required!!!");  
  
ed("pcre")) {  
    requires the pcre extension to php in on  
);  
  
ROOT.'/includes/autoloader.inc.php';  
  
tion  
ROOT.'/config.php';  
  
I_CONFIG_FILE') || !defined('PSI_DEBUG'  
template("/templates/html/error_config.  
fetch();  
  
out javascript
```

01

Les fondamentaux du web (HTML, CSS, PHP)

Partie 3 - Le PHP

Présentation du PHP

Présentation du PHP

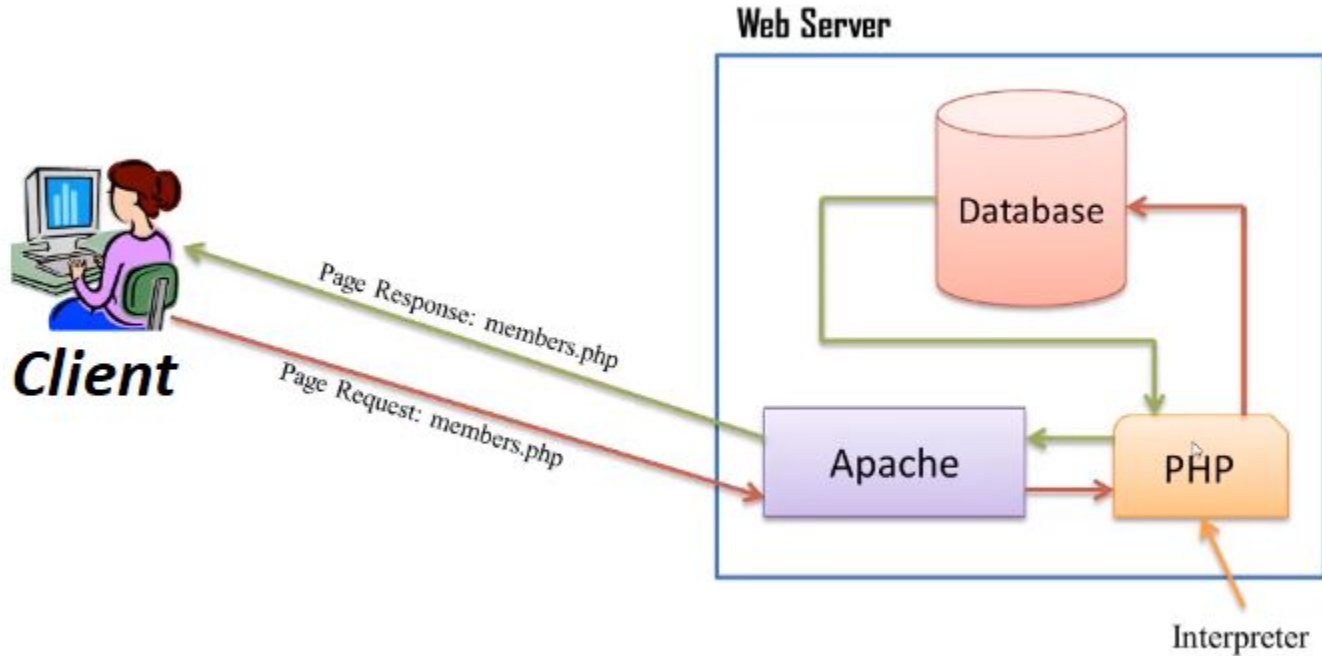
Le PHP, comme le Javascript (que vous verrez plus tard) est un langage de programmation interprété, il n'est pas compilé et doit être exécuté comme un script par un "interprète".

Pour rappel, le langage compilé quant à lui est transformé en code machine lors de sa compilation. Très souvent, le résultat de cette compilation est la création d'un fichier dit "exécutable".

Le PHP permet de rendre des pages "dynamiques", c'est-à-dire que le contenu des pages peut changer en fonction des actions de l'utilisateur (envoi d'un formulaire ou saisie d'information dans un autre formulaire par exemple).

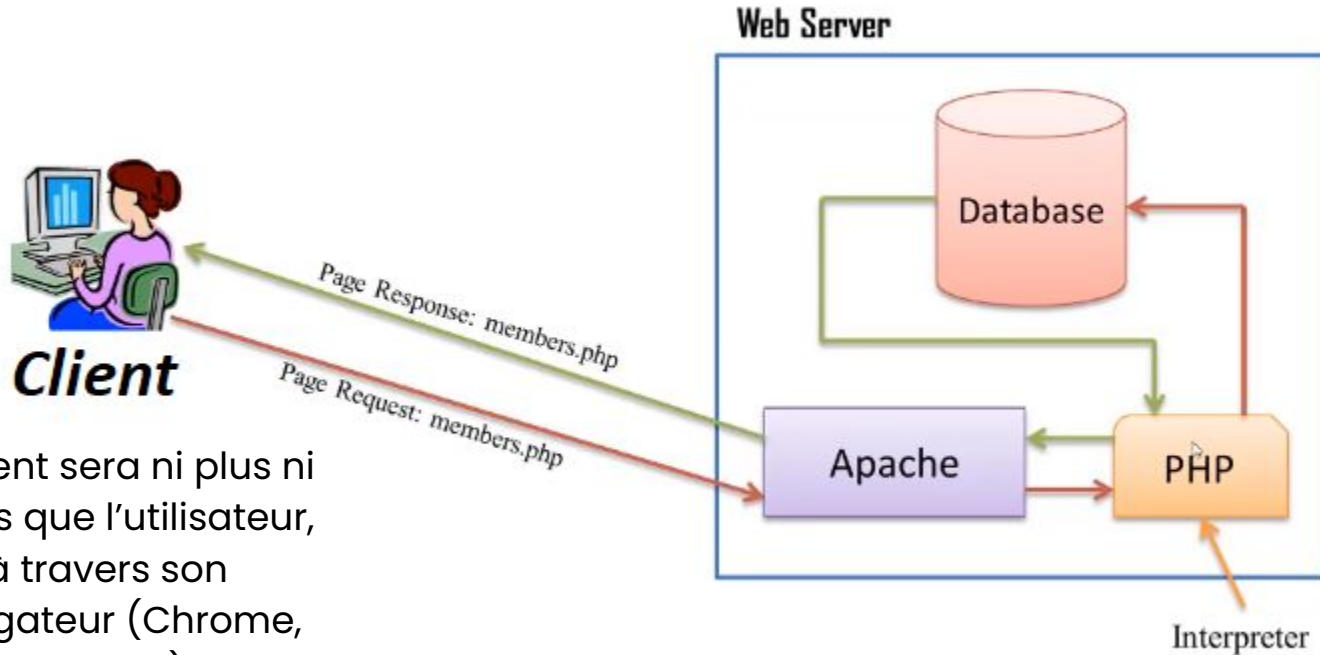
Cela dit, il faut savoir que, par défaut, le navigateur ne sait pas interpréter du PHP ! Voyons pourquoi et comment résoudre ce problème :)

Présentation du PHP



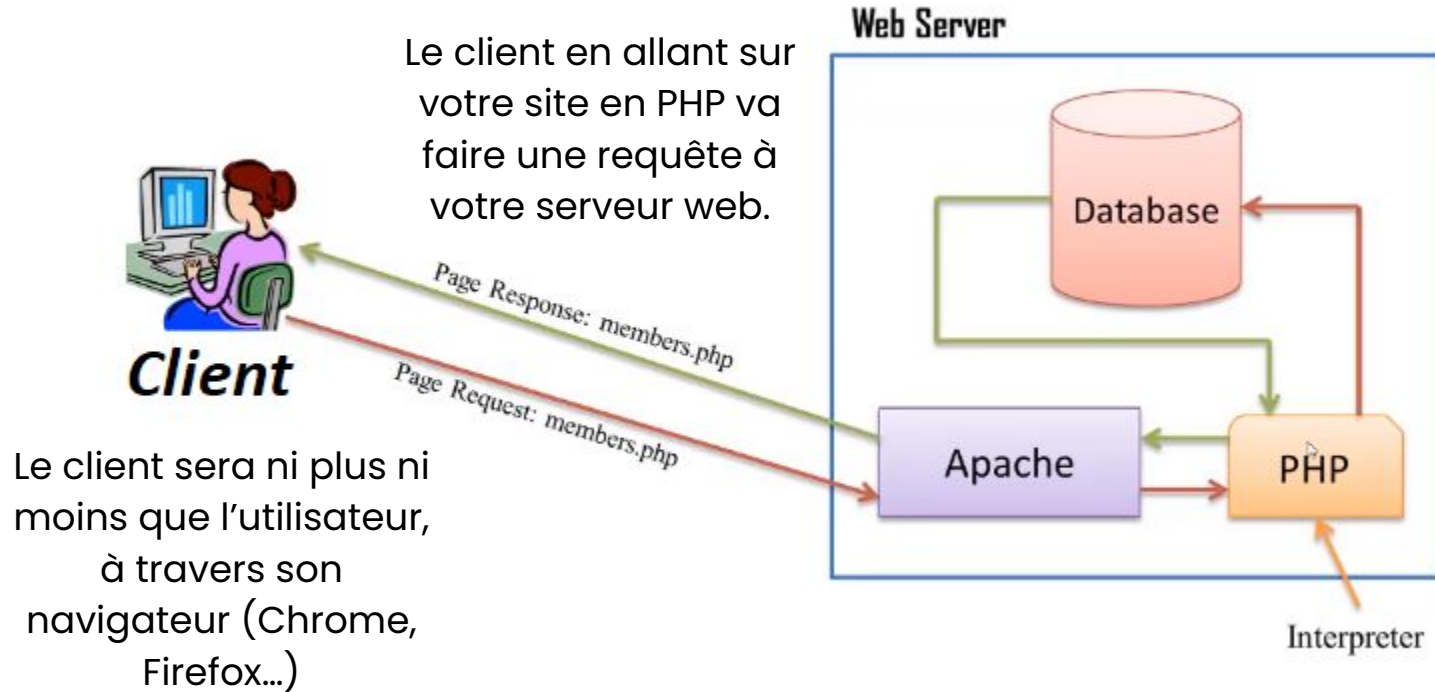
Analysons ce schéma !

Présentation du PHP

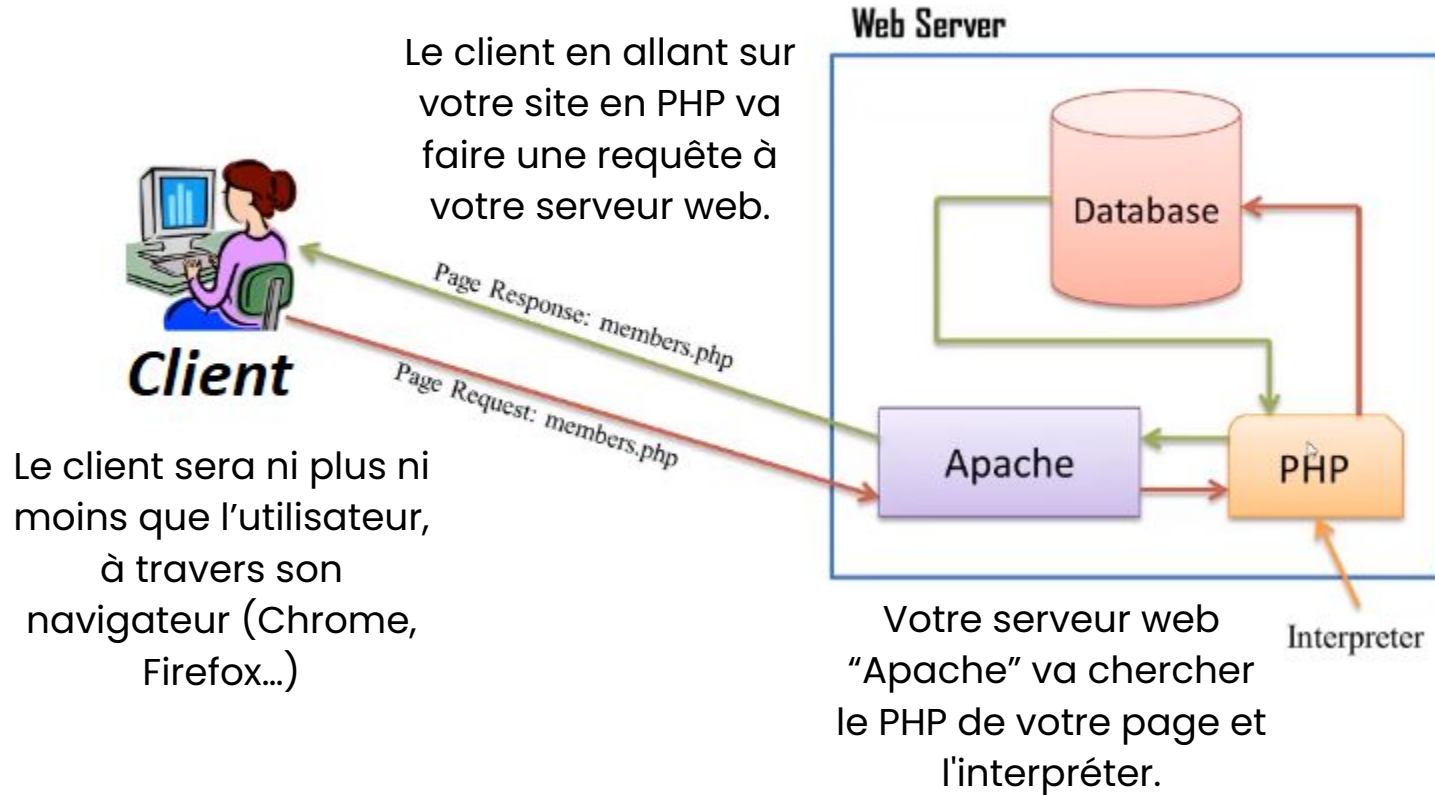


Le client sera ni plus ni moins que l'utilisateur, à travers son navigateur (Chrome, Firefox...)

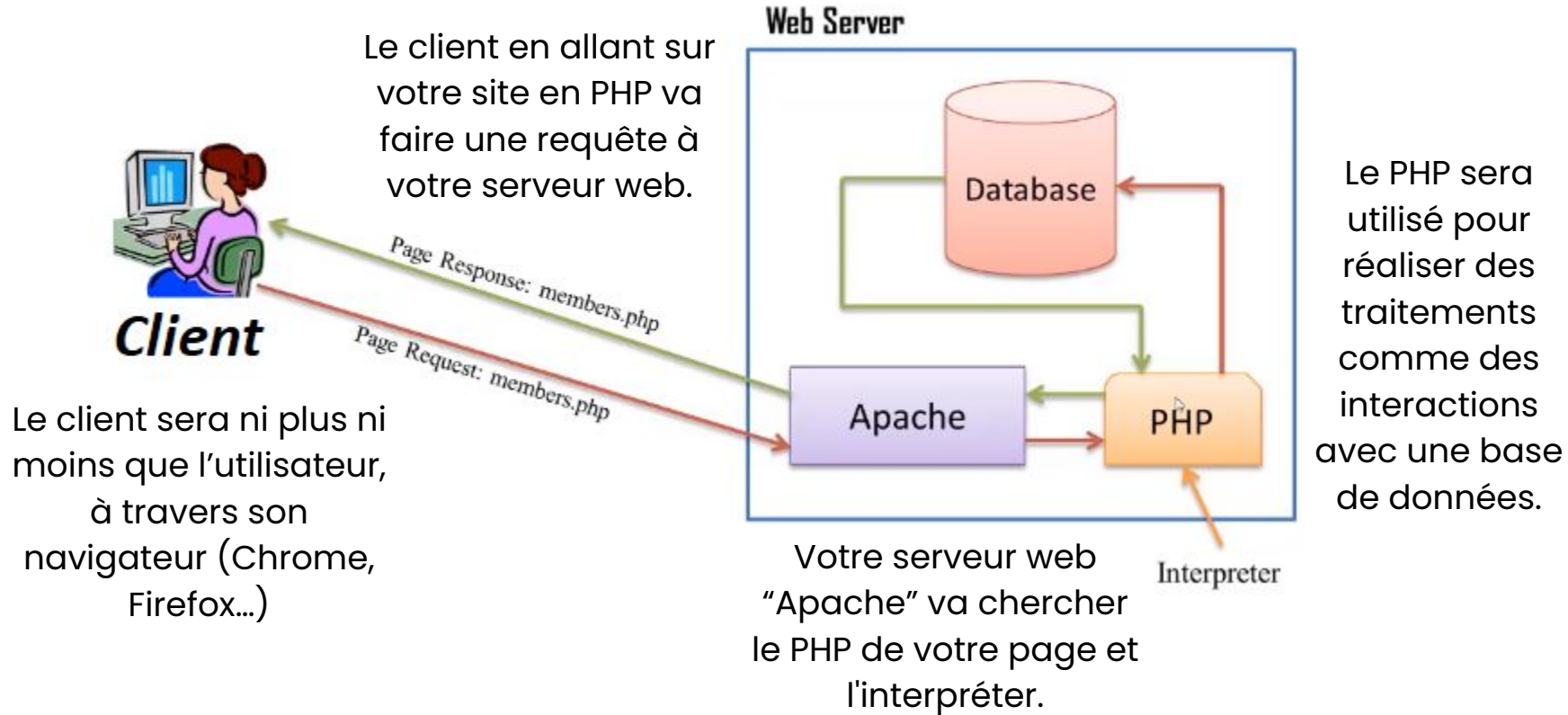
Présentation du PHP



Présentation du PHP



Présentation du PHP



02

Les fondamentaux du web (HTML, CSS, PHP)

Partie 3 - Le PHP

Outils pour le développement

Outils pour le développement



Un éditeur de texte ou IDE

Il servira à écrire les codes PHP qui viendront "s'associer" à nos fichiers HTML et CSS



Un navigateur Web

Il servira à visualiser notre page après nous être connecté à notre serveur web en local.

Outils pour le développement




Le serveur web

Comme dit auparavant, c'est lui qui va être utilisé afin d'interpréter notre code PHP. Il servira aussi à créer des base de données avec MySQL ou MariaDB.

Voyons comment l'utiliser !

Outils pour le développement



Wampserver - Installers, Updates, Addons, Tools

The site don't use cookies, contents no ads, no plotters, no statistics. No information is collected about visitors of the site. The only information collected is that of the hoster and concerns the legal and mandatory log files of access to the site.

Last Wampserver full install version 3.3.5 - Last update 3.3.6 english ▾

WampServer is a Windows-based Web development platform, without Internet access, for dynamic Web applications using the Apache 2.4 server, PHP scripting language and a MySQL and/or MariaDB database. Includes PHPMyAdmin and Adminer for database management. WampServer automatically installs everything you need to intuitively develop Web applications. You can adjust your server without touching its configuration files, using the various left-click and right-click menus of the Tray Menu Manager installed in the taskbar.


[News](#) [Apache 2.4.62 - Wampserver Update 3.3.6 - Tray Menu Manager 3.2.6.6 - PHP 8.2.21, 8.3.9 - MySQL 8.0.38, 8.4.1, 9.0.0](#)

Update page on: 2024-07-18 (Y-M-D) [Page modifications](#) [Installation requirements](#) [Changelog](#)

[Installers](#) [Updates](#) [Applications](#) [Addons](#) [Tools](#) [VC++ Packages](#) [All](#)

Installers Wampserver full install version

Last Wampserver full install version

 **Wampserver 3.3.5 64 bit x64 - Apache 2.4.59 - PHP 7.4.33/8.0.30/8.1.28/8.2.18/8.3.6 - MySQL 8.3.0 - MariaDB 11.3.2 MD5**

Also includes PhpMyAdmin 5.2.1 - Adminer 4.8.1 - PhpSysInfo 3.4.3.
Install Wampserver as an administrator Launch Wampmanager via the Shortcut.


As of May 9, 2023 and in agreement with Maximus23, developer of the Aestan Tray Menu, discontinuation of 32bit support for Wampserver and installation under Windows 7.
From April 1, 2024, support from Windows 10 only

[See 32bit versions](#)

Updates


[All available versions](#)

Wampmanager

 **Wampserver update 3.3.6 64 bit MD5**


Updates only "Wampmanager". It is a cumulative update. It is not necessary to apply the old updates 3.0.4 to 3.3.5 before applying 3.3.6. There will be no changes to your Apache, PHP, MySQL, MariaDB settings and versions used; your local sites and databases will not be affected. This update is necessary to install the latest Apache, PHP, MySQL or MariaDB addons.

xDebug

 **Update xDebug 3.3.2 64 bit MD5**


XDebug update version 3.3.2 for PHP versions 8.0.x to 8.3.x 64 bit already installed.
Can be reinstalled if addition of PHP version.

[Language files](#)

 **Language files MD5**

2024-04-30 - romanian 3.3.5 by Ciprian Murariu
2024-03-30 - romanian 3.3.4 by Ciprian Murariu
2024-02-09 - spanish 3.3.3 by Napolion

Tray Menu Manager (wampmanager.exe)

 **Tray Menu Manager 3.2.6.6 64 bit MD5**

Updated Tray Menu Manager(wampmanager.exe)
+ Correction for 4K display.
+ Scaling correction for different resolutions.
+ 4K centering correction.
+ Windows 11 update correction.
+ Compatible with Windows 24H2 pre-release.
+ Info: 32-bit version no longer supported.
+ Info: Versions below Windows 10 are no longer tested.
+ Code signing.

[Changelog Tray Menu Manager](#)

03

Les fondamentaux du web (HTML, CSS, PHP)

Partie 3 - Le PHP

Syntaxe et structures de base

Syntaxe et structures de base

Notions de base et inclusion des fichiers

Avant toutes choses, il faut savoir qu'il n'est pas possible d'utiliser du PHP si le fichier où il est écrit ne porte pas l'extension **php**

Une fois cela fait, comment construire du **php** pour l'utiliser dans une page web ?

Syntaxe et structures de base

Notions de base et inclusion des fichiers

```
<!DOCTYPE html>
<html lang = "fr">
<head>
  <meta charset = "utf-8" />
  <link rel = "stylesheet" href = "../general.css" />
  <title>echo hello world</title>
</head>
<body>
  <?php
    echo "hello world";
  ?>
</body>
</html>
```

Chaque fois que vous voulez utiliser des instructions **php** dans votre code, vous devrez utiliser `<?php [VOTRE CODE] ?>`

Syntaxe et structures de base

Notions de base et inclusion des fichiers

Une autre manière d'utiliser du code **php** dans une page web est d'intégrer un fichier **php** contenant le code que nous voulons inclure.

Bien entendu, comme pour l'exemple précédent, l'extension doit toujours être **.php**.

```
<?php
include "header.inc.php";
echo "<span class = 'titre'>L'instruction include</span>";
echo "Une " . $fruit . "<br/>"; // affiche : Une + warning (variable indefinie)
include_once "vars.inc.php";
echo "Une " . $fruit . "<br/>"; // affiche : Une pomme
$fruit = "banane";
echo "Une " . $fruit . "<br/>"; // affiche : Une banane
include_once "vars.inc.php";
echo "Une " . $fruit . "<br/>"; // affiche : Une banane
include "vars.inc.php";
echo "Une " . $fruit . "<br/>"; // affiche : Une pomme
//include "x.inc.php"; // warning
//require "x.inc.php"; // fatal error - le script s'arrête
echo "Une " . $fruit . "<br/>"; // affiche : Une pomme
include "footer.inc.php";
?>
```

Syntaxe et structures de base

Variables, types de données et opérateurs

Voyons maintenant comment écrire nos premières instructions !

Commençons par les variables :)

Syntaxe et structures de base

Variables, types de données et opérateurs

Comme dans tous les langages, le **php** utilise des variables, et celles-ci ne sont pas typées !

C'est-à-dire que la valeur de chaque variable peut changer n'importe quand et en n'importe quoi.

Voyons comment cela marche en pratique !

Syntaxe et structures de base

Variables, types de données et opérateurs

Pour déclarer une variable en **php**, on utilise le **\$**.

En plus d'utiliser **\$**, il faut nommer sa variable d'une certaine manière. Il existe plusieurs conventions pour les nommer correctement.

Syntaxe et structures de base

Variables, types de données et opérateurs

| | Case Type | Example |
|----------------------|------------------------------|-------------------------------|
| | Original Variable as String | <code>some awesome var</code> |
| Bosses d'un chameau | Camel Case | <code>someAwesomeVar</code> |
| Serpent rampant | Snake Case | <code>some_awesome_var</code> |
| Brochette de Kebab | Kebab Case | <code>some-awesome-var</code> |
| Ancien langage de p. | Pascal Case | <code>SomeAwesomeVar</code> |
| | Upper Case Snake Case | <code>SOME_AWESOME_VAR</code> |

Syntaxe et structures de base

Variables, types de données et opérateurs

Exemples d'affectation de variables :

\$calcul**A**ddition = 1;

\$calcul**A**ddition**N**umero**D**eux = 2;

Syntaxe et structures de base

Variables, types de données et opérateurs

Il est aussi possible d'affecter une variable par référence !

Lorsqu'on affecte une variable par référence, la nouvelle variable ne fait que référencer (en d'autres termes, "devient un alias de", ou encore "pointe sur") la variable originale. Les deux variables pointent donc vers la même adresse mémoire et sont donc complètement dépendantes l'une de l'autre. La modification de la valeur d'une des variables impactera la valeur de l'autre variable.

Syntaxe et structures de base

Variables, types de données et opérateurs

Exemple d'affectation de variable par référence :

```
$v1 = 1;  
$v2 = $&v1;  
$v1 = 2;
```

Résultat :

```
$v1 = 1 / $v2 = 1  
$v2 = 2 / $v2 = 2
```


Syntaxe et structures de base

Variables, types de données et opérateurs

Il existe aussi des variable dynamiques !

Une variable dynamique est une variable dont le nom est généré dynamiquement (le nom de la variable est lui-même variable !). Une variable dynamique prend la valeur d'une variable et l'utilise comme nom.

```
$var = "calcul";  
$$var = "1 + 2";
```

Résultat :

```
<?php echo $calcul; ?>
```

```
<?php echo ${$var}; ?>
```

Syntaxe et structures de base

Variables, types de données et opérateurs

Il faut aussi savoir que les variables ont des portées,
on ne peut pas les appeler de n'importe où.

Si vous voulez être sûr que votre variable est accessible de l'ensemble de votre page, il faut utiliser le mot clé **global**.

Syntaxe et structures de base

Variables, types de données et opérateurs

Pour finir avec les types de variables, nous avons
les constantes.

Une constante est une sorte de variable dont la valeur est fixe. Une fois qu'une constante est définie, elle ne peut jamais être modifiée ou détruite. La portée d'une constante est globale au script php, même si elle est définie dans le corps d'une fonction.

Syntaxe et structures de base

Variables, types de données et opérateurs

Exemple de constantes :

```
define("DEFAULT_NAME", "Jean");  
define("DEFAULT_MODE", "FPS");
```

À savoir que pour les constantes, la convention utilisée sera "snake-case" en mode UPPERCASE, donc en majuscule.

Syntaxe et structures de base

Variables, types de données et opérateurs

Les chaînes de caractères

Les chaînes de caractères doivent être encadrées par des guillemets simples ' ou par des guillemets doubles ". Dans une chaîne de caractères encadrée par des guillemets simples ', les variables ne sont pas remplacées par leur valeur.

Syntaxe et structures de base

Variables, types de données et opérateurs

Les tableaux

Les tableaux en php présentent une particularité par rapport à ceux de la plupart des autres langages informatiques. On parle ici de tableaux associatifs qui sont des structures qui font correspondre des valeurs à des clés. Une clé peut être une chaîne de caractères ou un entier, une valeur peut être de n'importe quel type. Il est donc possible dans un tableau php de mélanger des valeurs de différents types.

Syntaxe et structures de base

Variables, types de données et opérateurs

Exemple de tableaux :

```
$inventory = array(  
    "magasin" => array(  
        "nom" => "Super Shop",  
        "adresse" => "123 Rue du Commerce, Paris",  
        "telephone" => "01 23 45 67 89"  
    ),  
    "produits" => array(  
        array(  
            "nom" => "Ordinateur portable",  
            "prix" => 799.99,  
            "quantite" => 10,  
            "categorie" => "Electronique",  
            "enStock" => true  
        ),  
        array(  
            "nom" => "Télévision 4K",  
            "prix" => 1199.99,  
            "quantite" => 5,  
            "categorie" => "Electronique",  
            "enStock" => true  
        ),  
        array(  
            "nom" => "Chaise de bureau",  
            "prix" => 89.99,  
            "quantite" => 20,  
            "categorie" => "Mobilier",  
            "enStock" => true  
        ),  
        array(  
            "nom" => "Table en bois",  
            "prix" => 299.99,  
            "quantite" => 2,  
            "categorie" => "Mobilier",  
            "enStock" => false  
        )  
    ),  
    "nbProduits" => 4  
);
```

Syntaxe et structures de base

Variables, types de données et opérateurs

Les opérateurs arithmétiques

Définissons une variable \$a à 21 et \$b à 4.

| Nom | Symbole | Exemple | Explication | Résultat |
|----------------|---------|-----------|-----------------------------|----------|
| Négation | - | -\$a | Opposé de \$a | 21 |
| Addition | + | \$a + \$b | Somme de \$a et \$b | -17 |
| Soustraction | - | \$a - \$b | Différence de \$a et \$b | -25 |
| Multiplication | * | \$a * \$b | Produit de \$a et \$b | -84 |
| Division | / | \$a / \$b | Division de \$a et \$b | -5,25 |
| Modulo | % | \$a % \$b | Reste de \$a divisé par \$b | -1 |

Syntaxe et structures de base

Variables, types de données et opérateurs

Les opérateurs de comparaison

| Nom | Symbole | Exemple | Résultat |
|------------------------|------------------|-------------|---|
| Égal | == | \$a == \$b | true si \$a est égal à \$b après le transtypage |
| Identique | === | \$a === \$b | true si \$a est égal à \$b et qu'ils sont du même type |
| Différent | <> / != / !== | \$a <> \$b | true si \$a est différent de \$b ou bien s'ils ne sont pas du même type |
| Inférieur | < | \$a < \$b | true si \$a est strictement plus petit que \$b |
| Inférieur ou égale | <= | \$a <= \$b | true si \$a est plus petit ou égal à \$b |
| Supérieur | > | \$a > \$b | true si \$a est strictement plus grand que \$b |
| Supérieur ou égal | => | \$a => \$b | true si \$a est plus grand ou égal à \$b |
| Combiné (Depuis PHP 7) | ⇔ | \$a ⇔ \$b | 0 si \$a est égal à \$b, -1 si \$a est inférieur à \$b, +1 si \$a est supérieur à \$b |

Syntaxe et structures de base

Variables, types de données et opérateurs

Les opérateurs d'incrémentation / décrémentation

| Nom | Symbole | Résultat |
|---------------------|--------------------|---|
| Pré-incrémentation | <code>++\$a</code> | Incrémente \$a de 1 puis retourne \$a |
| Pré-décrémentation | <code>--\$a</code> | Décrémente \$a de 1 puis retourne \$a |
| Post-incrémentation | <code>\$a++</code> | Retourne \$a puis on l'incrémente de 1 |
| Post-décrémentation | <code>\$a--</code> | Retourne \$a puis on le décrémente de 1 |

Syntaxe et structures de base

Variables, types de données et opérateurs

Il est possible de combiner certains opérateurs arithmétiques avec l'affectation à des variables.

```
$a = $a + 5;
```

```
$a += 5;
```

```
$a += ++$a;
```

Syntaxe et structures de base

Variables, types de données et opérateurs

Les opérateurs logiques

Les opérateurs logiques permettent de combiner plusieurs tests et renvoient une valeur booléenne (true ou false) en fonction de la valeur des opérandes.

| Nom | Symbole | Exemple | Explication |
|-------------|---------|-------------|--|
| Non | ! | !\$a | true si \$a ne vaut pas true |
| ET logique | && | \$a && \$b | true si \$a et \$b valent true |
| OU logique | | \$a \$b | true si \$a ou \$b valent true |
| ET logique | and | \$a and \$b | true si \$a et \$b valent true |
| OU exclusif | xor | \$a xor \$b | true si \$a ou \$b valent true mais pas les deux en même temps |
| OU logique | or | \$a or \$b | true si \$a ou \$b valent true |

Syntaxe et structures de base

Variables, types de données et opérateurs

Exemple de tests avec des opérateurs logiques :

`$a = false;`

`$b = false;`

`$c = true;`

| Nom | Symbole |
|---|---------|
| <code>\$a and \$b or \$c</code> | true |
| <code>\$a and (\$b or \$c)</code> | false |
| <code>! (\$a xor \$c)</code> | true |
| <code>(\$a \$c) && !\$c</code> | false |

Syntaxe et structures de base

Structures de contrôle

Nous allons maintenant voir les structures de contrôle : **les structures conditionnels** et les **boucles**.

Syntaxe et structures de base

Structures de contrôle

Exécution conditionnelle

L'instruction `if` permet l'exécution d'un bloc instructions seulement dans le cas où une condition est vérifiée (une condition est une expression qui renvoie la valeur booléenne `true` ou `false`).

```
if (condition) {  
    bloc d'instructions  
}
```

Syntaxe et structures de base

Structures de contrôle

L'instruction else permet en plus l'exécution d'un bloc d'instructions dans le cas où la condition n'est pas vérifiée. L'instruction else est facultative et ne peut être présente qu'à la suite de l'instruction if. Les instructions if peuvent être imbriquées indéfiniment les unes dans les autres.

```
if (condition1) {  
    bloc d'instructions1  
} elseif (condition2) {  
    bloc d'instructions2  
} else {  
    bloc d'instructionsN  
}
```


Syntaxe et structures de base

Structures de contrôle

Il est aussi possible de faire une structure conditionnelles en une ligne : cela se nomme
“Opérateur ternaire”

condition ? valeurSiVrai : valeurSiFaux;

Syntaxe et structures de base

Variables, types de données et opérateurs

En **php**, il est possible de tester l'existence d'une variable avec la fonction **isset**.

```
$var1 = isset($var2) ? "coucou" : null;
```

On peut simplifier cette condition grâce à Coalesce, qui existe depuis **php 7**.

```
$var1 = $var2 ?? null;
```

Syntaxe et structures de base

Structures de contrôle

Choix multiple conditionnel

L'instruction switch équivaut à une série d'instructions if elseif.

```
switch ($variable) {  
    case valeur1 :  
        bloc d'instructions  
        break;  
    case valeur2 :  
        bloc d'instructions  
        break;  
    default :  
        bloc d'instructions  
        break;  
}
```

Syntaxe et structures de base

Structures de contrôle

Les boucles

Elles permettent de réaliser un ensemble d'instructions plusieurs fois en fonction d'une condition ou d'expressions.

La boucle "Tant ... que"

La boucle "Pour"

La boucle "Pour chaque"

Syntaxe et structures de base

Structures de contrôle

La boucle “ Tant que ”

La boucle while permet l'exécution d'un bloc d'instructions tant qu'une condition est vérifiée.

```
while (condition) {  
    bloc d'instructions  
}
```

Dans la première version, la condition est évaluée avant chaque passage dans le corps de la boucle. Le bloc d'instructions peut donc n'être jamais exécuté si la condition de la boucle n'est pas vérifiée avant le premier passage.

Syntaxe et structures de base

Structures de contrôle

La boucle “ Tant ... que ”

La boucle while permet l'exécution d'un bloc d'instructions tant qu'une condition est vérifiée.

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

Dans la seconde version, la condition est évaluée après chaque passage dans le corps de la boucle. On est donc certain d'exécuter le bloc d'instructions au moins une fois !

Attention à ce que la condition de la boucle puisse toujours devenir vraie à un moment donné car sinon, on aura une boucle sans fin et il n'y aura pas d'autres solutions pour reprendre la main que de tuer le processus responsable de la boucle sans fin.

Syntaxe et structures de base

Structures de contrôle

La boucle “ Pour ”

La boucle for permet d'exécuter un bloc d'instructions plusieurs fois, comme la boucle **while**.

```
for (expr1; expr2;  
    expr3) {  
    bloc d'instructions  
}
```

Initialisation (**expr1**) : elle est exécutée une seule fois, au début de la boucle, pour définir la valeur de départ.

Condition (**expr2**) : elle est vérifiée avant chaque répétition. Si elle est vraie, la boucle continue ; si elle est fausse, la boucle s'arrête.

Incrémentation (**expr3**) : elle est exécutée après chaque passage, pour mettre à jour la valeur de départ.

Attention, **expr2** doit toujours pouvoir passer à false à un moment donné car sinon, le bloc d'instructions s'exécute indéfiniment (boucle sans fin).

La boucle for est à privilégier par rapport à la boucle while car l'initialisation, la condition de sortie et l'incrément sont présentes sur la même ligne, ce qui améliore la compréhension et la lisibilité du code.

Syntaxe et structures de base

Structures de contrôle

La boucle “ Pour chaque ”

La boucle foreach permet de parcourir séquentiellement tous les éléments d'un tableau.

```
foreach ($tableau as $valeur) {  
    bloc d'instructions  
}
```