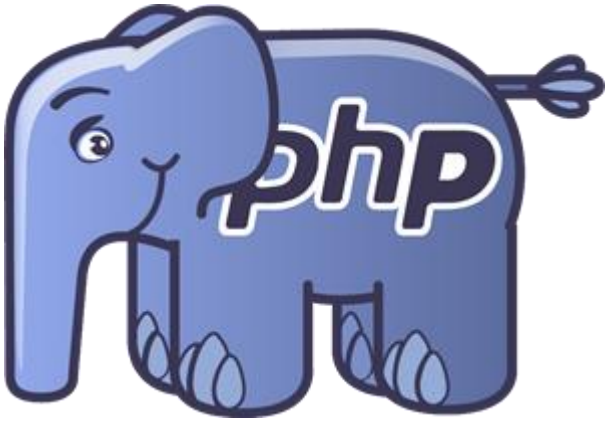
Les bases de PHP

Fonctionnement, syntaxe de base

CONTENU



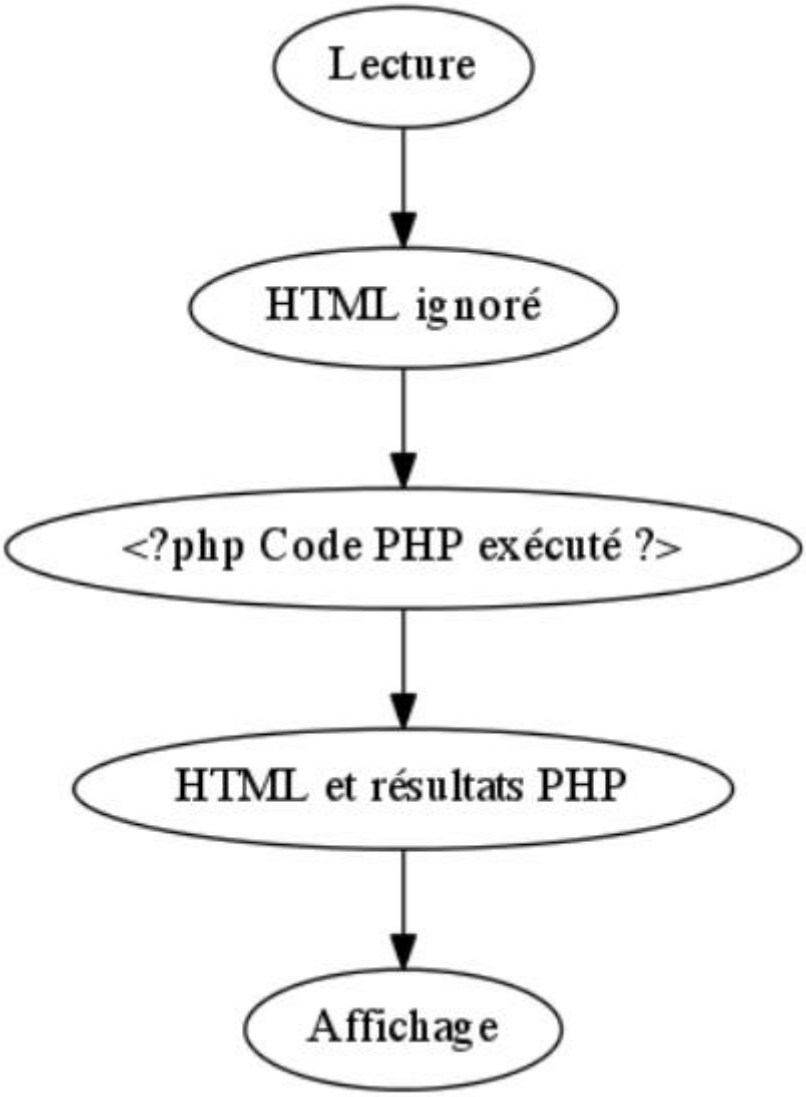
https://www.php.net

L'INTERPRÉTEUR PHP

L'interpréteur PHP lit un fichier avec l'extension **.php** puis génère un flux de sortie avec les règles suivantes :

- Toute ligne située à l'extérieur d'un bloc PHP (entre < ?php et ?>) est recopiée inchangée dans le flux de sortie.
- Le code PHP est interprété et génère éventuellement des résultats intégrés eux aussi au flux de sortie.
- Les erreurs éventuelles donnent lieu à des messages d'erreurs qu'on retrouve également dans le flux de sortie.

Une page html pure sauvegardée avec l'extension **.php** ne sera donc pas modifiée par PHP et sera renvoyée telle quelle…



EXEMPLE SIMPLE

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Bonjour depuis PHP </title>
</head>
<body>
<?php echo 'Bonjour généré dynamiquement en PHP !'; ?>
</body>
</html>
```

La portion de code située entre les balises **<?php** et **?>** est interprétée par PHP et produira un affichage dans le flux de sortie. Tout le code HTML situé à l'extérieur de ces balises est ignoré par PHP et sera retourné tel quel.

Résultat en sortie :

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Bonjour depuis PHP </title>
</head>
<body>
Bonjour généré dynamiquement en PHP !</body>
</html>
```

Rappel: Pour que le code soit interprété par PHP, le nom du fichier doit se terminer par l'extension **.php** (index.php, user.php…).

SYNTAXE DE BASE

L'objectif de PHP est généralement :

1) Effectuer un traitement (calculs, chargement de fichiers ou de données…).
2) Afficher le résultat du traitement dans le flux de sortie.

/!\ Une instruction PHP se termine toujours par un point-virgule **;**

L'instruction **echo** permet de déclencher un affichage dans le flux de sortie.

Exemple

```
<?php
echo "Bonjour tout le monde !"; // Affiche le texte entre guillemets
?>
```

Le texte situé entre les guillemets sera affiché dans le flux de sortie.

Un texte situé entre des guillemets est une **chaine de caractères** ou **string**.

VARIABLES

Une variable est un symbole qui permet de stocker une valeur en mémoire.
Pour faire simple, un symbole est un nom de variable qui est un "raccourci" vers une valeur stockée en mémoire.

En PHP le nom d'une variable est précédé du caractère **$**.

Exemple

```php
<?php
$identifiant = 1; // Déclaration d'une variable ayant pour valeur "1".
?>
```

Ci-dessus, La valeur **1** est stockée en mémoire et il est possible d'y accéder à nouveau en utilisant la variable (le symbole) associée.

Exemple

```php
<?php
$identifiant = 1; // Déclaration d'une variable ayant pour valeur "1".
echo $identifiant; // Affichage de la valeur stockée dans la variable $identifiant
?>
```

POURQUOI "VARIABLE" ?

Comme son nom l'indique, la valeur d'une variable peut… varier dans le temps.
Tout au long de l'exécution du programme, la valeur des variables créées peut changer. Vous comprenez maintenant le nom 😉.

Exemple

```php
<?php
$nom = "Mike"; // Déclaration d'une variable ayant pour valeur "Mike".
echo $nom; // Affiche "Mike"

$nom = "Paul"; // La valeur de la variable change et devient "Paul".
echo $nom; // Affiche "Paul"
?>
```

/!\ L'instruction **echo** permet uniquement d'afficher la valeur des types simples (texte, nombres…) et des objets implémentant la méthode __toString(). Tenter d'afficher une donnée d'un autre type résultera une erreur. Ceci sera abordé dans les cours suivants.

TYPES DE VARIABLES

Une variable stocke une valeur d'un certain type. Un type représente le format de la donnée stockée.

Principaux types en PHP :

Type	Format	Description	Affichage avec echo
bool	Booléen	2 valeurs possibles : **true** ou **false** (vrai ou faux)	**Oui** (true affiche 1, false n'affiche rien)
int	Nombre entier	1 nombre entier positif ou négatif	**Oui**
float	Nombre réel	1 nombre à décimales positif ou négatif	**Oui**
string	Chaine de caractères	Du texte, html, xml...	**Oui**
array	Tableau	Un tableau stocke plusieurs valeurs	**Non**, pas directement
object	Objet	Une instance d'une classe quelconque	**Oui** si méthode __toString() implémentée

Exemple

```php
<?php
// Déclaration d'une variable de type "bool" ayant pour valeur "true" (vrai).
$booleen = true;

// Déclaration d'une variable de type "int" ayant pour valeur "42".
$nombre = 42;

// Déclaration d'une variable de type "float" ayant pour valeur "13.37".
$reel = 13.37;

// Déclaration d'une variable de type "string" ayant pour valeur "Mike".
$chaine = "Mike";

// Déclaration d'une variable de type "array" contenant 3 éléments "Joe" "Jack" et "Cindy"
$tableau = [ "Joe", "Jack", "Cindy" ];

// Déclaration d'une variable de type "object" ayant pour valeur une instance de la classe DateTime.
$objet = new DateTime("2020-01-30");

?>
```

Vous remarquerez que contrairement à certaines langages dits fortement typés, le type de chaque variable n'est pas précisé.

PHP est à l'origine un langage faiblement typé.

Le typage fort est cependant possible (et recommandé) depuis PHP7 et est implémenté au fil des versions.

Les cours suivants, traitant du modèle objet de PHP aborderont cette approche.

CHAINES DE CARACTÈRES

Généralement et après avoir effectué ses différents traitements, PHP affichera le contenu de chaines de caractères.
Notez bien que PHP permet de générer et d'afficher la plupart des types "texte".

Types textes communs utilisés dans les applications Web et leurs Type MIME :

Nom	Type MIME	Description
TEXTE	text/plain	Texte simple
HTML	text/html	Document HTML
CSS	text/css	Feuille de style CSS
JS	application/javascript	Script Javascript
CSV	text/csv	Tableau de données au format CSV
XML	application/xml	Document XML
JSON	application/json	Document JSON

Exemple

```
<?php

// déclaration d'une variable "$html" ayant pour valeur du code HTML.
$html = "<p>Bonjour tout le monde</p>";

// Affichage du résultat
echo $html;
```

CONCATÉNATION

La concaténation désigne l'action de mettre bout à bout plusieurs chaines de caractères.
Le principe est de séparer les différentes chaines de caractères par l'opérateur de concaténation.

En PHP, l'opérateur de concaténation est le caractère point "**.**".

Exemple

```
<?php

$nom = "DEV";
$prenom = "Mike";

// Affichage du résultat
echo "Bonjour" . $prenom  . $nom;
```

Dans l'exemple ci-dessus, La chaine "Bonjour" , la variable "$nom" et la variable "$prenom" sont séparées par un "**.**" .
C'est la concaténation.

Le code précédent affichera : "Bonjour MikeDEV".

EXPRESSIONS

Une expression est un segment de code effectuant un traitement et retournant une valeur.

Exemple

```php
<?php

// "1 + 3" est une expression (un calcul simple). le résultat du calcul est retourné et stockée dans la variable "$calcul".
$calcul = 1 + 3;

echo $calcul; // Affichage du résultat (4)
```

EXPRESSION BOOLÉENNE

Une expression booléenne est une expression dont le résultat sera… une valeur booléenne (true ou false).
Une expression booléenne utilise les **opérateurs de comparaison** et les **opérateurs logiques**.

Opérateurs de comparaison PHP :

Symbole	Nom	Exemple	Signification
==	est égal à	**$a == $b**	La valeur de $a est-elle égale à la valeur de $b ?
===	est strictement égal à	**$a === $b**	Le type et la valeur de $a sont-ils identiques au type et à la valeur de $b ?
!=	est différent de	**$a != $b**	La valeur de $a est-elle différente de la valeur de $b ?
!==	est strictement différent de	**$a !== $b**	Le type ou la valeur de $a sont-ils différents du type ou de la valeur de $b ?
<	est strictement inférieur à	**$a < $b**	La valeur de $a est-elle strictement inférieure à la valeur de $b ?
<=	est inférieur ou égal à	**$a <= $b**	La valeur de $a est-elle inférieure ou égale à la valeur de $b ?
>	est strictement supérieur à	**$a > $b**	La valeur de $a est-elle strictement supérieure à la valeur de $b ?
>=	est supérieur ou égal à	**$a >= $b**	La valeur de $a est-elle supérieure ou égale à la valeur de $b ?

Opérateurs logiques PHP :

Symbole	Nom	Exemple	Signification
&&	ET	**($a == $b) && ($a > $c)**	La valeur de $a est-elle égale à la valeur de $b ? ET La valeur de $a est-elle supérieure à la valeur de $c ?
||	OU	**($a == $b) \|\| ($a < $c)**	La valeur de $a est-elle égale à la valeur de $b ? OU La valeur de $a est-elle inférieure à la valeur de $c ?

Exemple

```php
<?php

$a = 1;
$b = 2;
$c = 5;

$test1 = ($a < $b);  // vrai
$test2 = ($a < $b) && ($a > $c);  // faux
$test3 = ($a < $b) || ($a > $c);  // vrai
```

BLOC DE CODE

Un bloc de code est un ensemble d'instructions rassemblées entre accolades **{ }**.
Un bloc de code peut être considéré comme une expression complexe (contenant plusieurs instructions).

Exemple

```php
<?php
// bloc de code

{
    $a = 1;
    $b = 2;
    $c = 5;
}
```

Les blocs de code sont généralement utilisés pour définir le code à exécuter après une instruction conditionnelle, une boucle ou lors de l'appel d'une fonction.

STRUCTURES CONDITIONNELLES

Une structure conditionnelle permet de définir quel bloc de code exécuter selon le résultat d'une expression booléenne.

IF / ELSE

Exemple

```php
<?php

$a = 1;
$b = 2;

// structure conditionnelle
// Si la valeur de $a est supérieure à la valeur de $b
// on exécute le premier bloc situé après l'instruction "if"
// sinon, on exécute le bloc de code situé après l'instruction "else"

if($a > $b)
{
    echo "A est supérieur à B";
}
else
{
    echo "B est supérieur à A";
}
```

IF / ELSEIF / ELSE

Exemple

```php
<?php

$a = 1;
$b = 2;

// Si la valeur de $a est supérieure à la valeur de $b
// on exécute le premier bloc situé après l'instruction "if"
// sinon si $a est égal à $b, on exécute le 2ème bloc après le "elseif"
// sinon, on exécute le bloc de code situé après l'instruction "else"

if($a > $b)
{
    echo "A est supérieur à B";
}
elseif($a == $b)
{
    echo "A est égal à B";
}
else
{
    echo "B est supérieur à A";
}
```

SWITCH / CASE

L'instruction switch évalue une expression et, selon le résultat obtenu et le cas associé, exécute les instructions correspondantes.

Exemple

```php
<?php
$a = 1;

switch($a)
{
    case 1:
        echo "Gagné, A est égal à 1";
        break;
    case 2:
        echo "Gagné, A est égal à 2";
        break;
    default:
        echo "Perdu, A ne devrait pas être égal à " . $a;
        break;
}
```

La valeur de $a est évaluée.

Si $a vaut 1, le code sous le cas 1 (case 1) est exécuté.

L'instruction **break** qui suit permet de de sortir du bloc switch courant après exécution du cas 1.

C'est-à-dire que si le cas 1 est exécuté, les autres cas seront ignorés.

Si $a vaut 2, le code sous le cas 1 est ignoré et le code du cas 2 est exécuté.

Si la valeur de $a ne correspondant à aucun cas, le cas par défaut (default) est exécuté.

/! Pour les structures conditionnelles, prévoyez **toujours** un cas par défaut :

- **else** pour les instructions if/elseif/else.
- **default** pour les instructions switch/case.

--- FIN DU DOCUMENT ---