Les opérateurs en PHP

Les opérateurs en PHP permettent d'effectuer des opérations sur des variables et des valeurs.

Ils sont essentiels pour manipuler les données et prendre des décisions dans votre code.

1 - Les opérateurs arithmétiques

Ces opérateurs servent à effectuer des calculs mathématiques de base sur des nombres.

Exemples :

```php
 $a = 10;
$b = 3;

echo $a + $b; // Addition : Affiche "13"
echo $a - $b; // Soustraction : Affiche "7"
echo $a * $b; // Multiplication : Affiche "30"
echo $a / $b; // Division : Affiche "3.3333"
echo $a % $b; // Modulo (reste) : Affiche "1"
```

1.1 - L'opérateur d'exponentiation

Introduit en PHP 5.6, cet opérateur permet de calculer une puissance.

```php
 $c = 2 ** 3; // 2 à la puissance 3
echo $c; // Affiche "8"
```

Les opérateurs d'affectation

Utilisés pour assigner des valeurs à une variable. Ils permettent également de combiner l'affectation avec d'autres opérations.

Exemples :

```php
 $x = 5;      // Affectation simple
$x += 2;     // Addition et affectation : $x vaut maintenant 7
$x -= 1;     // Soustraction et affectation : $x vaut maintenant 6
$x *= 3;     // Multiplication et affectation : $x vaut maintenant 18
$x /= 6;     // Division et affectation : $x vaut maintenant 3
$x %= 2;     // Modulo et affectation : $x vaut maintenant 1
```

Les opérateurs de comparaison

Ces opérateurs permettent de comparer deux valeurs et retournent un résultat booléen (`true` ou `false`).

Exemples :

```php
$a = 10;
$b = 20;
$c = '10';

echo $a == $b;   // Égalité : Retourne false
echo $a == $c;   // Égalité : Retourne true (car "10" est converti en 10)
echo $a != $b;   // Différent : Retourne true
echo $a === $c;  // Identité stricte : Retourne false (car types différents : int vs string)
echo $a !== $c;  // Différent strict : Retourne true (car types différents : int vs string)
echo $a < $b;    // Inférieur : Retourne true
echo $a > $b;    // Supérieur : Retourne false
echo $a <= $b;   // Inférieur ou égal : Retourne true
echo $a >= $b;   // Supérieur ou égal : Retourne false
```

3.1 - Opérateur spaceship `<=>`

> Introduit en PHP 7, il compare deux valeurs et retourne `-1`, `0` ou `1` selon leur relation.

```php
echo 10 <=> 20; // Retourne -1
echo 20 <=> 20; // Retourne 0
echo 30 <=> 20; // Retourne 1
```

Les opérateurs logiques

> Utilisés pour combiner plusieurs conditions logiques.

Exemples :

```php
$a = true;
$b = false;

echo $a && $b; // Retourne false (et logique)
echo $a || $b; // Retourne true (ou logique)
echo !$a;      // Retourne false (non logique)
```

Les opérateurs d'incrémentation et de décrémentation

> Ces opérateurs permettent d'augmenter ou de réduire la valeur d'une variable.

Exemples :

```php
$x = 5;

echo ++$x; // Prédécrémentation : Affiche "6"
echo $x++; // Postdécrémentation : Affiche "6", mais $x vaut maintenant "7"
echo --$x; // Préincrémentation : Affiche "6"
echo $x--; // Postincrémentation : Affiche "6", mais $x vaut maintenant "5"
```

Les opérateurs sur les chaînes

Concaténez ou combinez des chaînes en PHP.

Exemples :

```php
$str1 = "Bonjour";
$str2 = " tout le monde";

echo $str1 . $str2; // Concaténation : Affiche "Bonjour tout le monde"
$str1 .= $str2;     // Concaténation et affectation
echo $str1;         // Affiche "Bonjour tout le monde"
```

Les opérateurs ternaires

Simplifient les conditions. Le format est : `(condition) ? valeur_si_vrai : valeur_si_faux`.

Exemple :

```php
$age = 18;
echo ($age >= 18) ? "Majeur" : "Mineur"; // Affiche "Majeur"
```

7.1 - Opérateur null coalescent `??`

Introduit en PHP 7, il retourne la première valeur définie et non nulle parmi les options.

```php
$nom = null;
echo $nom ?? "Anonyme"; // Affiche "Anonyme"
```

7.2 - Opérateur null coalescent et affectation `??=`

Introduit en PHP 7.4, cet opérateur affecte une valeur à une variable uniquement si celle-ci n'est pas encore définie ou qu'elle est `null`.

Exemple :

```php
$user = null;
$user ??= "invité";
echo $user; // Affiche "invité"
```

Dans cet exemple, la variable `$user` n'avait pas de valeur définie, donc elle récupère `"invité"`. Si `$user` contenait déjà une valeur, celle-ci ne serait pas modifiée.

Les opérateurs bit à bit

Manipulent directement les bits d'un nombre, principalement utilisés pour des opérations avancées.

Exemples :

```php
 $a = 6; // En binaire : 110
$b = 3; // En binaire : 011

echo $a & $b; // ET bit à bit : Retourne 2 (010)
echo $a | $b; // OU bit à bit : Retourne 7 (111)
echo $a ^ $b; // OU exclusif : Retourne 5 (101)
echo ~$a;     // Complément à 1 : Retourne -7
echo $a << 1; // Décalage à gauche : Retourne 12 (1100)
echo $b >> 1; // Décalage à droite : Retourne 1 (001)
```

À RETENIR

- Opérateurs arithmétiques :
  - `+` : Addition.
  - `-` : Soustraction.
  - `*` : Multiplication.
  - `/` : Division.
  - `%` : Modulo.
  - `**` : Exponentiation.
- Opérateurs de comparaison :
  - `==` : Égalité.
  - `!=` : Différent.
  - `<` , `>` : Inférieur, supérieur.
  - `<=>` : Comparaison combinée.
- Opérateurs logiques :
  - `&&` : ET.
  - `||` : OU.
  - `!` : NON.
- Opérateurs ternaires et null coalescent :
  - `(condition) ? valeur_vraie : valeur_fausse` .
  - `??` : Retourne la première valeur définie.
  - `??=` : Affecte une valeur si elle est `null` .
- Opérateurs sur les chaînes :
  - `.` : Concaténation.
  - `.=` : Concaténation avec affectation.
- Opérateurs d'incrémentation et décrémentation :
  - `++` : Augmenter de 1.
  - `--` : Réduire de 1.
- Opérateurs bit à bit :
  - `&` : ET bit à bit.
  - `|` : OU bit à bit.
  - `^` : OU exclusif.
  - `~` : Complément à 1.
  - `<<` , `>>` : Décalages.