



Orange Money

A vos smartphones pour voter votre artiste préféré et gagner de Orange

Orange

En sa

Opérateurs et concaténation en PHP

Télécharger le PDF
du cours



Cours complet PHP et MySQL

INTRODUCTION AU COURS PHP ET MYSQL

1. Introduction au cours :
définitions et rôles du PHP et du
MySQL

2. Client et serveur : définitions et
interactions

3. Mise en place de notre
environnement de travail

4. Créer, enregistrer et exécuter un
script PHP

5. Afficher un résultat en PHP avec
une instruction echo ou print

DÉCOUVERTE DES VARIABLES EN PHP

Dans cette nouvelle leçon, nous allons définir ce qu'est un opérateur, établir la liste des types d'opérateurs disponibles en PHP et apprendre à en manipuler certains.

Nous allons également préciser les différences entre l'utilisation des apostrophes ou des guillemets lorsqu'on manipule une valeur de type chaîne de caractères.

Qu'est-ce qu'un opérateur ?

Un opérateur est un symbole qui va être utilisé pour effectuer certaines actions notamment sur les variables et leurs valeurs.

Par exemple, l'opérateur `+` va nous permettre d'additionner les valeurs de deux variables, tandis que l'opérateur `=` va nous permettre d'affecter une valeur à une variable.

La documentation officielle de PHP classe les différents opérateurs qu'on va pouvoir utiliser selon les groupes suivants :

- Les opérateurs arithmétiques ;
- Les opérateurs d'affectation ;
- Opérateurs sur les bits ;
- Opérateurs de comparaison ;
- Opérateur de contrôle d'erreur ;
- Opérateur d'exécution ;
- Opérateurs d'incrément et décrémentation ;
- Les opérateurs logiques ;
- Opérateurs de chaînes ;
- Opérateurs de tableaux ;
- Opérateurs de types ;

Dans cette leçon, nous allons nous concentrer sur les opérateurs arithmétiques, les opérateurs de chaînes et les opérateurs d'affectation.

Nous verrons les autres types d'opérateurs au fil de ce cours lorsque cela fera le plus de sens (c'est-à-dire lorsqu'on en aura besoin).

Les opérateurs de chaînes et la concaténation en PHP

<div>8. Opérateurs et concaténation en PHP</div> <div>LES STRUCTURES DE CONTRÔLE EN PHP</div> <div>9. Présentation des conditions et des opérateurs de comparaison</div> <div>10. Les conditions if, if...else et if...elseif...else</div> <div>11. Créer des conditions robustes avec les opérateurs logiques</div> <div>12. Ecrire des conditions condensées avec les opérateurs ternaire et fusion null</div> <div>13. L'instruction switch en PHP</div> <div>14. Les boucles PHP et les opérateurs d'incrément et de décrément</div> <div>15. Inclure des fichiers dans un autre en PHP avec include et require</div> <div>DÉCOUVERTE DES FONCTIONS EN PHP</div> <div>16. Introduction aux fonctions PHP</div> <div>17. Contrôler le passage des arguments</div> <div>18. Contrôler les valeurs de retour d'une fonction</div> <div>19. La portée des variables en PHP</div> <div>20. Constantes et constantes magiques en PHP</div> <div>LES VARIABLES TABLEAUX EN PHP</div> <div>21. Présentation des tableaux et tableaux numérotés en PHP</div> <div>22. Les tableaux associatifs en PHP</div> <div>23. Les tableaux multidimensionnels en PHP</div> <div>MANIPULER DES DATES EN PHP</div> <div>24. Le timestamp UNIX et la date en PHP</div> <div>25. Obtenir et formater une date en PHP</div> <div>26. Comparer des dates et tester la validité d'une date en PHP</div>	<p>Cet opérateur va s'avérer particulièrement utile lorsqu'on voudra stocker le contenu de plusieurs variables qui stockent des données de type chaîne de caractères ou pour afficher différentes données au sein d'une même instruction <code>echo</code>.</p> <div></div> <p>Pour bien comprendre comment fonctionne l'opérateur de concaténation et son intérêt, il me semble nécessaire de connaître les différences entre l'utilisation des guillemets et des apostrophes lorsqu'on manipule une chaîne de caractères en PHP.</p> <p>Sur ce sujet, vous pouvez retenir que la différence majeure entre l'utilisation des guillemets et d'apostrophes est que tout ce qui est entre guillemets va être interprété tandis que quasiment tout ce qui est entre apostrophes va être considéré comme une chaîne de caractères.</p> <p>Ici, « interprété » signifie « être remplacé par sa valeur ». Ainsi, lorsqu'on inclut une variable au sein d'une chaîne de caractères et qu'on cherche à afficher le tout avec un <code>echo</code> et en utilisant des guillemets, la variable va être remplacée par sa valeur lors de l'affichage.</p> <p>C'est la raison pour laquelle il faut échapper le <code>\$</code> si on souhaite afficher le nom de la variable comme chaîne de caractères plutôt que sa valeur.</p> <p>En revanche, lorsqu'on utilise des apostrophes, les variables ne vont pas être interprétées mais leur nom va être considéré comme faisant partie de la chaîne de caractères.</p> <p>Regardez plutôt l'exemple suivant :</p> <div><pre>1. <!DOCTYPE html> 2. <html> 3. <head> 4. <title>Cours PHP & MySQL</title> 5. <meta charset="utf-8"> 6. <link rel="stylesheet" href="cours.css"> 7. </head> 8. 9. <body> 10. <h1>Titre principal</h1> 11. <?php 12. \$prenom = "Pierre"; 13. \$nom = "Giraud"; 14. \$age = 28; 15. 16. echo "Je m'appelle \$prenom et j'ai \$age ans
"; 17. echo "Je m'appelle {\$prenom} et j'ai {\$age} ans
"; 18. echo 'Je m'appelle \$prenom et j'ai \$age ans
'; 19. 20. \$prez = "Je suis \$prenom \$nom, j'ai \$age ans
"; 21. \$prez2 = "Je suis {\$prenom} {\$nom}, j'ai {\$age} ans
"; 22. \$prez3 = 'Je suis \$prenom \$nom, j'ai \$age ans'; 23. 24. echo \$prez; 25. echo \$prez2; 26. echo \$prez3; 27. ?> 28. <p>Un paragraphe</p></pre></div>
--	---

27. Les variables superglobales PHP	
28. Création et gestion des cookies en PHP	
29. Définir et utiliser les sessions en PHP	
MANIPULER DES FICHIERS EN PHP	
30. Introduction à la manipulation de fichiers en PHP	
31. Ouvrir, lire et fermer un fichier en PHP	
32. Créer et écrire dans un fichier en PHP	Ici, nous déclarons trois variables <code>\$prenom</code> , <code>\$nom</code> et <code>\$age</code> .
33. Autres opérations sur les fichiers en PHP	On essaie ensuite d'afficher du texte avec des <code>echo</code> en incluant nos noms de variables au sein du texte.
UTILISER LES EXPRESSIONS RÉGULIÈRES OU RATIONNELLES EN PHP	
34. Introduction aux expressions rationnelles ou expressions régulières	
35. Les fonctions PCRE PHP	
36. Les classes de caractères des regex	
37. Les métacaractères des regex PHP	Pour notre premier <code>echo</code> , on utilise des guillemets pour entourer le texte. Les variables dans le texte vont être interprétées et c'est leur contenu qui va être affiché.
38. Les options des expressions régulières disponibles en PHP	Notez cependant ici que la syntaxe avec les noms de variables directement au milieu du texte est déconseillée aujourd'hui et qu'on préférera utiliser la syntaxe de de notre deuxième <code>echo</code> qui utilise des accolades pour entourer les variables.
PROGRAMMATION ORIENTÉE OBJET (POO) PHP : CONCEPTS DE BASE	
39. Introduction à la programmation orientée objet PHP : classes, instances et objets	Dans notre troisième <code>echo</code> , on utilise cette fois-ci des apostrophes. Les noms des variables ne vont donc pas être interprétés mais être considérés comme du texte et s'afficher tel quel.
40. Propriétés et méthodes en PHP orienté objet	Finalement, on crée de la même façon trois variables <code>\$prez</code> , <code>\$prez2</code> et <code>\$prez3</code> qui stockent à nouveau du texte au sein duquel on inclut les noms de nos variables.
41. Les méthodes PHP constructeur et destructeur	On <code>echo</code> alors le contenu de nos trois variables. Sans surprise, les variables <code>\$prez</code> et <code>\$prez2</code> stockent le texte donné avec le contenu des variables <code>\$prenom</code> , <code>\$nom</code> et <code>\$age</code> tandis que la variable <code>\$prez3</code> stocke le nom de ces variables plutôt que leurs valeurs.
42. Encapsulation et visibilité des propriétés et méthodes PHP	L'opérateur de concaténation va nous permettre de mettre bout à bout les différentes données tout en faisant en sorte que chaque donnée soit interprétée par le PHP.
43. Classes étendues et héritage en PHP orienté objet	Nous allons l'utiliser pour séparer nos différentes variables des chaînes de caractères autour. Regardez l'exemple suivant pour bien comprendre :
44. Surcharge d'éléments et opérateur de résolution de portée en PHP	<pre>1. <!DOCTYPE html> 2. <html> 3. <head> 4. <title>Cours PHP & MySQL</title> 5. <meta charset="utf-8"></pre>
45. Les constantes de classe en PHP	

47. Les méthodes et les classes abstraites en PHP objet	
48. Les interfaces en PHP orienté objet	
49. Les méthodes magiques en orienté objet PHP	
PROGRAMMATION ORIENTÉE OBJET PHP : NOTIONS AVANCÉES	
50. Le chainage de méthodes en PHP	<pre>10. <h1>Titre principal</h1> 11. <?php 12. \$prenom = "Pierre"; 13. \$nom = "Giraud"; 14. \$age = 28; 15. \$prez = "Je suis " . \$prenom. " " . \$nom. ", j'ai " . \$age. " ans"; 16. \$prez2 = 'Je suis ' . \$prenom. ' ' . \$nom. ', j\'ai ' . \$age. ' ans'; 17. 18. 19. echo "Je m'appelle " . \$prenom. " et j'ai " . \$age. " ans
"; 20. echo 'Je m\'appelle ' . \$prenom. ' et j\'ai ' . \$age. ' ans
'; 21. 22. echo \$prez. '
' . \$prez2; 23. ?> 24. <p>Un paragraphe</p> 25. </body> 26. </html></pre>
51. Les closures et les classes anonymes en PHP objet	
52. L'auto chargement des classes en PHP	
53. Le mot clef final en PHP objet	
54. La résolution statique à la volée ou late static bindings en PHP	
55. Utiliser les traits en orienté objet PHP	
56. L'interface Iterator et le parcours d'objets en PHP	
57. Le passage d'objets en PHP : identifiants et références	
58. Le clonage d'objets et la méthode magique PHP __clone()	
59. La comparaison d'objets PHP	
ESPACES DE NOMS, FILTRES ET GESTION DES ERREURS EN PHP	
60. Les espaces de noms PHP	
61. Présentation des filtres PHP	
62. Filtres de validation, de nettoyage et drapeaux de l'extension PHP Filter	
63. Utilisation pratique des filtres en PHP	
64. Définition et gestion des erreurs en PHP	
65. Déclenchement, capture et gestion des exceptions PHP : try, throw, catch	
INTRODUCTION AUX BASES DE DONNÉES, AU SQL ET À MYSQL	
66. Introduction aux bases de données, au SQL et au MySQL	

Pour concaténer correctement avec l'opérateur de concaténation, la règle est de séparer les différentes variables avec l'opérateur de concaténation (le point) des textes autour. Chaque texte devra être entouré de guillemets ou d'apostrophes selon ce qu'on a choisi.

A ce niveau, il est probable que vous vous demandiez l'intérêt d'utiliser l'opérateur de concaténation qui semble ici compliquer inutilement le code plutôt que simplement des guillemets et des accolades.

Ma réponse va être très simple : ici, vous pouvez utiliser l'une ou l'autre de ces méthodes pour un résultat identique. Cependant, rappelez-vous que l'utilisation d'apostrophes ou de guillemets n'est pas identique au sens où ce qui est entre guillemets va être interprété tandis que la grande majorité de ce qui est entre apostrophes ne le sera pas.

Ainsi, parfois, on voudra utiliser des apostrophes plutôt que des guillemets et dans ce cas, si on souhaite que certaines de nos variables soient interprétées, il faudra utiliser l'opérateur de concaténation.

De manière générale, il est conseillé de toujours utiliser l'opérateur de concaténation lorsqu'on souhaite mettre bout-à-bout plusieurs chaînes de caractères (qui seront

68. Se connecter à une base de données MySQL en PHP
69. Créer une base de données MySQL et une table dans la base
MANIPULER DES DONNÉES DANS DES BASES MYSQL AVEC PDO
70. Insérer des données dans une table MySQL
71. Les requêtes MySQL préparées avec PDO PHP
72. Modifier les données d'une table MySQL ou sa structure
73. Supprimer des données, une table ou une base de données MySQL
74. Sélection simple de données dans une table MySQL en PHP
75. Utiliser des critères de sélection pour sélectionner des données dans une table MySQL
76. Utiliser les fonctions d'agrégation et les fonctions scalaires SQL
JOINTURES, UNION ET SOUS REQUÊTES
77. Présentation des jointures SQL
78. Création de jointures SQL
79. L'opérateur SQL UNION
80. Les opérateurs de sous requête SQL
GESTION DES FORMULAIRES HTML AVEC PHP
81. Rappels sur les formulaires HTML
82. Récupérer et manipuler les données des formulaires HTML en PHP
83. Sécurisation et validation des formulaires en PHP
CONCLUSION DU COURS PHP ET MYSQL
84. Conclusion du cours complet PHP et MySQL

Les opérateurs arithmétiques

Les opérateurs arithmétiques vont nous permettre d'effectuer toutes sortes d'opérations mathématiques entre les valeurs contenues dans différentes variables lorsque ces valeurs sont des nombres.

Le fait de pouvoir réaliser des opérations entre variables va être très utile dans de nombreuses situations. Par exemple, si un utilisateur commande plusieurs produits sur notre site ou plusieurs fois un même produit et utilise un code de réduction, il faudra utiliser des opérations mathématiques pour calculer le prix total de la commande.

En PHP, nous allons pouvoir utiliser les opérateurs arithmétiques suivants :

Opérateur	Nom de l'opération associée
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (reste d'une division euclidienne)
**	Exponentielle (élévation à la puissance d'un nombre par un autre)

Avant d'utiliser les opérateurs arithmétiques, clarifions ce que sont le modulo et l'exponentielle.

Le modulo correspond au reste entier d'une division euclidienne. Par exemple, lorsqu'on divise 5 par 3, le résultat est 1 et il reste 2 dans le cas d'une division euclidienne. Le reste, 2, correspond justement au modulo.

L'exponentielle correspond à l'élévation à la puissance d'un nombre par un autre nombre. La puissance d'un nombre est le résultat d'une multiplication répétée de ce nombre par lui-même. Par exemple, lorsqu'on souhaite calculer 2 à la puissance de 3 (qu'on appelle également « 2 exposant 3 »), on cherche en fait le résultat de 2 multiplié 3 fois par lui-même c'est-à-dire $2*2*2 = 8$.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title>Cours PHP & MySQL</title>
5.     <meta charset="utf-8">
6.     <link rel="stylesheet" href="cours.css">
7.   </head>
8.
```

```
13.      $y = 3;
14.      $z = 4;
15.      echo '$x stocke ' . $x . ', $y stocke ' . $y . ', $z stocke ' . $z .
    '<br>';
16.
17.      $a = $x + 1; // $a stocke 2 + 1 = 3
18.      $b = $x + $y; // $b stocke 2 + 3 = 5
19.      $c = $x - $y; // $c stocke 2 - 3 = -1
20.      echo '$a stocke ' . $a . ', $b stocke ' . $b . ', $c stocke ' . $c .
    '<br>';
21.
22.      $x = $x * $y; // $x stocke désormais 2 * 3 = 6
23.      echo 'La variable $x stocke désormais : ' . $x . '<br>';
24.
25.      $z = $x / $y; // $z stocke désormais 6 / 3 = 2
26.      echo 'La variable $z stocke désormais : ' . $z . '<br>';
27.
28.      $m = 5 % 3; // $m stocke le reste de la division euclidienne de 5
    par 3
29.      echo 'Le reste de la division euclidienne de 5 par 3 est ' . $m .
    '<br>';
30.
31.      $p = $z ** 4; // $p stocke 2^4 = 2 * 2 * 2 * 2 = 16
32.      echo 'La variable $p stocke le résultat de 2 puissance 4 = ' . $p;
33.      ?>
34.      <p>Un paragraphe</p>
35.      </body>
36.      </html>
```

Concernant les règles de calcul, c'est-à-dire l'ordre de priorité des opérations, celui-ci va être le même qu'en mathématiques : l'élévation à la puissance va être prioritaire sur les autres opérations, tandis que la multiplication, la division et le modulo vont avoir le même ordre de priorité et être prioritaires sur l'addition et la soustraction qui ont également le même niveau de priorité.

Si deux opérateurs ont le même ordre de priorité, alors c'est leur sens d'association qui va décider du résultat. Pour les opérateurs arithmétiques, le sens d'association correspond à l'ordre de leur écriture à l'exception de l'élévation à la puissance qui sera calculée en partant de la fin.

Ainsi, si on écrit `$x = 1 - 2 - 3`, la variable `$x` va stocker la valeur -4 (les opérations se font de gauche à droite). En revanche, si on écrit `$x = 2 ** 3 ** 2`, la variable `$x` stockera 512 qui correspond à 2 puissance 9 puisqu'on va commencer par calculer $3 ** 2 = 9$ dans ce cas.

Nous allons finalement, comme en mathématiques, pouvoir forcer l'ordre de priorité en utilisant des couples de parenthèses pour indiquer qu'une opération doit se faire avant toutes les autres :

```

1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Cours PHP & MySQL</title>
5.          <meta charset="utf-8">
6.          <link rel="stylesheet" href="cours.css">
7.      </head>
8.
9.      <body>
10.         <h1>Titre principal</h1>
11.         <?php
12.             $x = 2 + 3 * 4; // $x stocke 14
13.             $y = (2 + 3) * 4; // $y stocke 20
14.             $z = 2 ** 3 - 4 * 4 / 8; // $z stocke 6
15.
16.             echo '$x : ' . $x . ' <br> $y : ' . $y . ' <br> $z : ' . $z;
17.         ?>
18.         <p>Un paragraphe</p>
19.     </body>
20. </html>

```

Ici, `$x` stocke la valeur 14. En effet, la multiplication est prioritaire sur l'addition. On va donc commencer par faire $3 * 4$ puis ajouter 2 au résultat.

La variable `$y` stocke 20. En effet, on utilise des parenthèses pour forcer la priorité de l'addition par rapport à la multiplication.

Finalement, `$z` stocke la valeur 6. En effet, on commence ici par calculer 2 puissance 3 ($2 ** 3 = 8$). Ensuite, on calcule $4 * 4 / 8 = 16 / 8 = 2$ car la multiplication et la division sont prioritaires sur la soustraction. Finalement, on calcule $8 - 2 = 6$.

Notez également que les opérateurs `+` et `-` peuvent également servir à convertir le type de valeur contenue dans une variable vers **Integer** ou **Float** selon ce qui est le plus approprié.

Cette utilisation des opérateurs va pouvoir nous être utile lorsqu'on aura variables contenant des « nombres » stockés sous le type de chaînes de caractères et pour lesquelles on voudra réaliser des opérations mathématiques. Nous aurons l'occasion de rencontrer ce cas plus tard dans ce cours.

```

1.  <!DOCTYPE html>
2.  <html>

```

```
7.      </head>
8.
9.      <body>
10.         <h1>Titre principal</h1>
11.         <?php
12.             $x = "2";
13.             $y = "3.14";
14.
15.             echo '$x stocke la valeur ' . $x . ' de type ' . gettype($x) .
16.             '<br>';
17.             echo '$y stocke la valeur ' . $y . ' de type ' . gettype($y) .
18.             '<br>';
19.
20.             $x = +$x;
21.             $y = -$y;
22.             $z = +"3";
23.
24.             echo '$x stocke la valeur ' . $x . ' de type ' . gettype($x) .
25.             '<br>';
26.             echo '$y stocke la valeur ' . $y . ' de type ' . gettype($y) .
27.             '<br>';
28.             echo '$z stocke la valeur ' . $z . ' de type ' . gettype($z);
29.         ?>
30.         <p>Un paragraphe</p>
31.     </body>
32. </html>
```



Les opérateurs d’affectation et opérateurs combinés

Les opérateurs d’affectation vont nous permettre, comme leur nom l’indique, d’affecter une certaine valeur à une variable.

Nous connaissons déjà bien l’opérateur d’affectation le plus utilisé qui est le signe `=`. Cependant, vous devez également savoir qu’il existe également des opérateurs combinés notamment pour les opérateurs arithmétiques et l’opérateur de concaténation et qui sont les suivants :

Opérateur	Définition
<code>.=</code>	Concatène puis affecte le résultat
<code>+=</code>	Additionne puis affecte le résultat
<code>-=</code>	Soustrait puis affecte le résultat
<code>*=</code>	Multiplie puis affecte le résultat
<code>/=</code>	Divise puis affecte le résultat

Illustrons immédiatement cela et voyons comment se servir de ces opérateurs :

```
1.  <!DOCTYPE html>
2.  <html>
3.    <head>
4.      <title>Cours PHP & MySQL</title>
5.      <meta charset="utf-8">
6.      <link rel="stylesheet" href="cours.css">
7.    </head>
8.
9.    <body>
10.     <h1>Titre principal</h1>
11.     <?php
12.       $a = "Bonjour";
13.       $a .= " le monde"; //$a stocke "Bonjour le monde"
14.       echo '$a stocke : ' . $a. '<br>';
15.
16.       $x = 5;
17.       $x -= 3; //$x stocke désormais 2
18.       echo '$x stocke : ' . $x. '<br>';
19.
20.       $y = 3;
21.       $y **= $x; //$y stocke 3^2 = 3 * 3 = 9
22.       echo '$y stocke : ' . $y;
23.     ?>
24.     <p>Un paragraphe</p>
25.   </body>
26. </html>
```

Ce qu'il faut bien comprendre dans l'exemple précédent est que les opérateurs d'affectation combinés font deux choses à la fois : ils exécutent une opération puis ils affectent une valeur.

Au début, notre variable `$a` stocke `Bonjour`. Ensuite, on utilise l'opérateur d'affectation concaténant `.=` qui va concaténer la valeur à droite avec la valeur contenue dans la variable à gauche avant de lui affecter le résultat.

Ici, on concatène donc la chaîne de caractères `le monde` avec la valeur `Bonjour` et on affecte le résultat (c'est-à-dire les deux chaînes concaténées) dans la variable `$a`. La variable `$a` va donc désormais stocker `Bonjour le monde`.

Par ailleurs, notez que tous les opérateurs d'affectation ont une priorité de calcul égale mais qui est inférieure à celle des opérateurs arithmétiques ou de concaténation.

Lorsque des opérateurs ont des ordres de priorité égaux, c'est le sens d'association de ceux-ci qui va décider du résultat. Pour les opérateurs arithmétiques, on a vu que l'association se faisait par la gauche sauf pour l'élevation à la puissance. Pour les opérateurs d'affectation, l'association se fait par la droite.

```

1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Cours PHP & MySQL</title>
5.          <meta charset="utf-8">
6.          <link rel="stylesheet" href="cours.css">
7.      </head>
8.
9.      <body>
10.         <h1>Titre principal</h1>
11.         <?php
12.             $x = 1;
13.             $y = 2;
14.             $z = 3;
15.             $a = 5;
16.
17.             $x = $z += 2; // $z stocke 5 et $x stocke 5
18.             echo '$x stocke : ' . $x. ' et $z stocke : ' . $z. '<br>';
19.
20.             $y += $z -= 2; // $z stocke 5 - 2 = 3 et $y stocke 2 + 3 = 5
21.             echo '$y stocke : ' . $y. ' et $z stocke : ' . $z. '<br>';
22.
23.             $y /= $z -= 2; // $z stocke 1 et $y stocke 5
24.             echo '$y stocke : ' . $y. ' et $z stocke : ' . $z. '<br>';
25.
26.             $a *= 4 + 2; // $z stocke 30
27.             echo '$a stocke : ' . $a;
28.         ?>
29.         <p>Un paragraphe</p>
30.     </body>
31. </html>

```

Pour notre premier calcul, nous utilisons les deux opérateurs d'affectation `=` et `+=`. L'association va se faire par la droite. On commence donc à ajouter 2 à la valeur de `$z` qui stocke désormais 5 et on stocke la même valeur dans `$x`. Faites bien attention ici : `$x` ne stocke bien évidemment pas la variable `$z` mais seulement la dernière valeur connue de `$z`. Si on modifie ensuite la valeur de `$z`, cela n'a aucun impact sur `$x`.

Les deux exemples suivants utilisent à nouveau deux opérateurs d'affectation. L'association va donc toujours se faire par la droite.

Dans notre dernier exemple, cependant, on utilise à la fois un opérateur d'affectation et un

Laisser un commentaire

Vous devez vous connecter pour publier un commentaire.

[Connexion](#)[Confidentialité](#)[CGV](#)[Sitemap](#)

© Pierre Giraud - Toute reproduction interdite - Mentions légales