PHP 2

Numeric Arrays

Rappelez-vous que le premier élément d'un tableau a l'indice 0 et non 1.

```
echo $names[1]; // Outputs "Amy"
```

Nous avons défini un tableau nommé $names qui stocke trois valeurs.
Vous pouvez accéder aux éléments du tableau à l'aide de leurs indices.

```
$names[0] = "David";
$names[1] = "Amy";
$names[2] = "John";
```

Comme alternative, vous pouvez assigner votre index manuellement.

```
$names = array("David", "Amy", "John");
```

Tableaux numériques
Les tableaux numériques ou indexés associent un index numérique à leurs valeurs.
L'index peut être assigné automatiquement (index commence toujours à 0), comme ceci:

```
$name1 = "David";
$name2 = "Amy";
$name3 = "John";
```

Un tableau est une variable spéciale, qui peut contenir plus d'une valeur à la fois.
Si vous avez une liste d'éléments (une liste de noms, par exemple), les stocker dans des variables simples ressemblerait à ceci au dessus. Mais que faire si vous avez 100 noms sur votre liste? La solution: Créez un tableau !

```
<?php
  $myArray[0] = "John";
  $myArray[1] = "<strong>PHP</strong>";
  $myArray[2] = 21;

  echo "$myArray[0] is $myArray[2] and knows $myArray[1]";

  // Outputs "John is 21 and knows PHP"
?>
```

Vous pouvez avoir des entiers, des chaînes et d'autres types de données ensemble dans un tableau.

Associative Arrays

```
$people = array("David"=>"27", "Amy"=>"21", "John"=>"42");
// or
$people['David'] = "27";
$people['Amy'] = "21";
$people['John'] = "42";
```

Les tableaux associatifs sont des tableaux qui utilisent des clés nommées que vous leur assignez.
Il existe deux façons de créer un tableau associatif.
Dans le premier exemple, notez l'utilisation des signes => pour attribuer des valeurs aux clés nommées.

```
<?php
  $people = array("David"=>"27", "Amy"=>"21", "John"=>"42");

  echo $people['Amy'];
  // Outputs 21
?>
```

Utilisez les touches nommées pour accéder aux membres du tableau.

Multi-dimensional Arrays

Les tableaux de plus de trois niveaux de profondeur sont difficiles à gérer.

Un tableau multidimensionnel contient un ou plusieurs tableaux.

La dimension d'un tableau indique le nombre d'indices dont vous auriez besoin pour sélectionner un élément.

- Pour un tableau bidimensionnel, vous avez besoin de deux indices pour sélectionner un élément
- Pour un tableau tridimensionnel, vous avez besoin de trois indices pour sélectionner un élément

```
$people = array(
  'online'=>array('David', 'Amy'),
  'offline'=>array('John', 'Rob', 'Jack'),
  'away'=>array('Arthur', 'Daniel')
);
```

Créons un tableau bidimensionnel qui contient 3 tableaux:

Les tableaux dans le tableau multidimensionnel peuvent être à la fois numériques et associatifs.

```
<?php
  $people = array(
      'online'=>array('David', 'Amy'),
      'offline'=>array('John', 'Rob', 'Jack'),
      'away'=>array('Arthur', 'Daniel')
);

  echo $people['online'][0]; //Outputs "David"
  echo '<br />';
  echo $people['away'][1]; //Outputs "Daniel"

?>
```

Maintenant, le tableau bidimensionnel $people contient 3 tableaux, et il a deux indices: ligne et colonne.
Pour accéder aux éléments du tableau $people, nous devons pointer sur les deux indices.

Include & Require

Les instructions include et require permettent d'insérer le contenu d'un fichier PHP dans un autre fichier PHP, avant que le serveur ne l'exécute.
L'inclusion de fichiers permet d'économiser un peu de travail. Vous pouvez créer un en-tête standard, un pied de page ou un fichier de menu pour toutes vos pages Web. Ensuite, lorsque l'en-tête requiert une mise à jour, vous pouvez mettre à jour le fichier d'inclusion d'en-tête uniquement.

```
<?php
  echo "<h1>Welcome</h1>";
?>
```

Supposons que nous avons un fichier d'en-tête standard appelé header.php.

```
<html>
 <body>

 <?php include 'header.php'; ?>

 <p>Some text.</p>
 <p>Some text.</p>
 </body>
</html>
```

Utilisez l'instruction include pour inclure le fichier d'en-tête dans une page.

```
<html>
 <body>

 <?php include 'header.php'; ?>

 <p>This is a paragraph</p>
 </body>
</html>
```

En utilisant cette approche, nous avons la possibilité d'inclure le même fichier header.php en plusieurs pages.

Result:

Les fichiers sont inclus en fonction du chemin du fichier.
Vous pouvez utiliser un chemin d'accès absolu ou relatif pour spécifier quel fichier doit être inclus.

Utilisez require lorsque le fichier est requis pour l'exécution de l'application.
Utiliser include lorsque le fichier n'est pas requis. L'application doit continuer, même si le fichier n'est pas trouvé.

L'instruction require est identique à include, l'exception étant que, en cas d'échec, il produit une erreur fatale.
Lorsqu'un fichier est inclus à l'aide de l'instruction include, mais que PHP est incapable de le trouver, le script continue à s'exécuter.
Dans le cas de require, le script cesserа l'exécution et produira une erreur.

The If Else Statement

```
if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```

Les instructions conditionnelles exécutent des actions différentes pour différentes décisions.
L'instruction if else est utilisée pour exécuter un certain code si une condition est vraie, et un autre code si la condition est fausse. Vous pouvez également utiliser l'instruction if sans l'instruction else, si vous n'avez pas besoin de faire quoi que ce soit, au cas où la condition est fausse.

```
<?php
  $x = 10;
  $y = 20;
  if ($x >= $y) {
      echo $x;
  } else {
      echo $y;
  }
?>
```

L'exemple ci-dessus affichera le plus grand nombre des deux.

The Elseif Statement

```
if (condition) {
  code to be executed if condition is true;
} elseif (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```

Utilisez l'instruction if ... elseif ... else pour spécifier une nouvelle condition à tester, si la première condition est false. Vous pouvez ajouter autant d'instructions elseif que vous le souhaitez. Il suffit de noter que l'instruction elseif doit commencer par une instruction if.

```
<?php
  $age = 21;

  if ($age<=13) {
      echo "Child.";
  } elseif ($age>13 && $age<19) {
      echo "Teenager";
  } else {
      echo "Adult";
  }
  //Outputs "Adult"
?>
```

Nous avons utilisé l'opérateur ET logique (&&) pour combiner les deux conditions et vérifier pour déterminer si $ age est entre 13 et 19.

The while Loop

Lors de l'écriture de code, vous pouvez vouloir le même bloc de code à exécuter encore et encore. Au lieu d'ajouter plusieurs lignes de code presque égales dans un script, nous pouvons utiliser des boucles pour exécuter une tâche comme celle-ci.

```
while (condition is true) {
  code to be executed;
}
```

La boucle Loop
La boucle while exécute un bloc de code tant que la condition spécifiée est true. Si la condition ne devient jamais fausse, l'instruction continuera à s'exécuter indéfiniment.

```
<?php
  $i = 1;
  while ($i < 7) {
      echo "The value is $i <br />";
      $i++;
  }
?>
```

L'exemple ci-dessus définit d'abord une variable $ i à un ($ i = 1). Ensuite, la boucle while tourne tant que $ i est inférieur à sept ($ i <7). $ I augmentera de 1 chaque fois que la boucle s'exécute ($ i ++);

Résultat ci-dessus

The Do While Loop

Que la condition soit vraie ou fausse, le code sera exécuté au moins une fois, ce qui pourrait être nécessaire dans certaines situations.

```
do {
  code to be executed;
} while (condition is true);
```

La boucle do ... while effectue toujours le bloc de code une fois, vérifie la condition et répète la boucle tant que la condition spécifiée est vraie.

```
<?php
  $i = 1;
  do {
    echo "The number is " . $i . "<br/>";
    $i++;
  } while ($i <= 7);
  //Output:
//The number is 1
//The number is 2
//The number is 3
?>
```

L'exemple ci-dessus va écrire une sortie, puis incrémenter la variable $ i par un. Ensuite, la condition est vérifiée, et la boucle continue à s'exécuter, tant que $ i est inférieur ou égal à 7.

Notez que dans une boucle while, la condition est testée APRÈS l'exécution des instructions dans la boucle. Cela signifie que la boucle while while exécuterait ses instructions au moins une fois, même si la condition est fausse la première fois.

The For Loop

```
for (init; test; increment) {
  code to be executed;
}
```

La boucle for est utilisée lorsque vous savez à l'avance combien de fois le script doit s'exécuter.
Chacune des 3 expressions de paramètre peut être vide ou contenir plusieurs expressions séparées par des virgules.
Dans l'instruction pour, les paramètres sont séparés par des points-virgules.

```
<?php
  for ($a = 0; $a < 6; $a++) {
    echo "Value of a : ", $a , "<br />";
  }
?>
```

L'exemple ci-dessous affiche les nombres de 0 à 5:

Parametres:
Init: Initialise la valeur du compteur de boucle
Test: Évalue chaque fois que la boucle est itérée, continue s'il est évalué à vrai et se termine si elle est évaluée comme fausse
Incrément Augmente la valeur du compteur de boucle

Resultat

The Foreach Loop

```
foreach (array as $value) {
  code to be executed;
  //or
foreach (array as $key => $value) {
  code to be executed;
}
```

La boucle foreach ne fonctionne que sur les tableaux, et est utilisée pour parcourir chaque paire clé / valeur dans un tableau.
Il existe deux syntaxes:

La première forme boucles sur le tableau. A chaque itération, la valeur de l'élément courant est affectée à la valeur $, et le pointeur du tableau est déplacé par un, jusqu'à ce qu'il atteigne le dernier élément du tableau.
La deuxième forme attribuera en outre la clé de l'élément courant à la variable $ key sur chaque itération.

L'exemple suivant à droite illustre une boucle qui génère les valeurs du tableau $ names.

```
<?php
  $names = array("John", "David", "Amy");
  foreach ($names as $name) {
      echo $name."<br />";
  }
// John
// David
// Amy
?>
```

The Switch Statement

```
switch (x) {
  case value1:
    //code to be executed if n=value1
    break;
  case value2:
    //code to be executed if n=value2
    break;

  default:
    // code to be executed if n is different from all labels
}
```

L'instruction switch est une alternative à l'instruction if-elseif-else.
Utilisez l'instruction switch pour sélectionner l'un d'un nombre de blocs de code à exécuter.

Premièrement, notre seule expression, n (le plus souvent une variable), est évaluée une fois. Ensuite, la valeur de l'expression est comparée à la valeur de chaque cas dans la structure. S'il ya une correspondance, le bloc de code associé à ce cas est exécuté. L'utilisation des instructions imbriquées if else entraîne un comportement similaire, mais le commutateur offre une solution plus élégante et optimale.

```
<?php
  $today = "Tue";
  switch ($today) {
    case "Mon":
      echo "Today is Monday.";
      break;
    case "Tue":
      echo "Today is Tuesday.";
      break;
    case "Wed":
      echo "Today is Wednesday.";
      break;
    case "Thu":
      echo "Today is Thursday.";
      break;
    case "Fri":
      echo "Today is Friday.";
      break;
    case "Sat":
      echo "Today is Saturday.";
      break;
    case "Sun":
      echo "Today is Sunday.";
      break;
    default:
      echo "Invalid day.";
  }
  //Output: "Today is Tuesday."
?>
```

Considérez l'exemple suivant, qui affiche le message approprié pour chaque jour.

Le mot-clé break qui suit chaque cas est utilisé pour empêcher le code de s'exécuter automatiquement dans le cas suivant. Si vous oubliez la pause; Declaration, PHP continuera automatiquement à travers les prochaines déclarations de cas, même si l'affaire ne correspond pas.

```
$x=5;
switch ($x) {
  case 1:
    echo "One";
    break;
  case 2:
    echo "Two";
    break;
  default:
    echo "No match";
}

//Outputs "No match"
```

L'instruction par défaut est utilisée si aucune correspondance n'est trouvée. L'instruction par défaut est facultative, elle peut donc être omise.

```
<?php
  $day = 'Wed';
  switch ($day) {
    case "Mon":
      echo "It's first day of the week";
      break;
    case "Tue":
      echo "Working day";
      break;
    case "Wed":
      echo "Working day";
      break;
    case "Thu":
      echo "Working day";
      break;
    case "Fri":
      echo "Friday";
      break;
    default:
      echo "Weekend";
  }
  //Outputs "Working day"
?>
```

Faute de spécifier l'instruction break, PHP continue à exécuter les instructions qui suivent le cas, jusqu'à ce qu'il trouve une pause. Vous pouvez utiliser ce comportement si vous devez arriver à la même sortie pour plus d'un cas. L'exemple ci-dessus aura la même sortie si $ day est égal à 'Tue', 'Wed' ou 'Thu'.

The Break Statement

```
<?php
  $i=1;
  switch ($i) {
    case 1:
      echo "One";
      break;
    case 2:
      echo "Two";
      break;
    case 3:
      echo "Three";
      break;
    default:
      echo "No match";
  }
  //Outputs "One\nThree\nNo match"
?>
```

Comme indiqué dans la leçon précédente, l'instruction break est utilisée pour sortir du commutateur quand un cas est apparié.
Si la pause est absente, le code continue à fonctionner. Par exemple:

L'instruction break arrête le courant pour, foreach, while, do-while ou switch et continue à exécuter le programme sur la ligne à venir après la boucle.
Une instruction break dans la partie externe d'un programme (par exemple, pas dans une boucle de contrôle) arrêtera le script.

Break peut également être utilisé pour arrêter l'exécution de for, foreach, while, do-while structures.

The Continue Statement

```
for ($i=0; $i<10; $i++) {
  if ($i%2==0) {
    continue;
  }
  echo $i . ' ';
}

//Output: 1 3 5 7 9
```

Lorsqu'il est utilisé dans une structure en boucle, l'instruction continue permet de sauter ce qui reste de l'itération de la boucle en cours. Il poursuit ensuite l'exécution à l'évaluation de la condition et passe au début de la prochaine itération.

L'exemple suivant saute les nombres pairs dans la boucle for:

Vous pouvez utiliser l'instruction continue avec toutes les structures en boucle.