الفصل الثالث: التعامل مع المتغيرات و الثوابت

المتغيرات Variables

المتغير هو عبارة عن حاوية (Container) تستخدم لحفظ البيانات الرقمية أو الغير رقمية . و يمكن التعامل معها بأكثر من طريقة كـ ، نقلها من مكان الى آخر ، استبدال و حتى إفراغ قيمتها. و لتسهيل طريقة التعامل مع هذه الحاوية فإننا نقوم بإعطائها اسم خاص بها.

تسمية المتغيرات Naming Variables

عند تسمية المتغيرات يجب الأخذ في الاعتبار عدد من القواعد التي يجب إتباعها:

- تبدأ المتغيرات في PHP بعلامة الدولار $ (dollar sign).
- الذي يلي علامة $ يجب أن يكون إما حرف (letter) أو شرطة سفلية (underscore).
- لا يسمح بالمسافات(space) و علامات الترقيم (punctuation).
- أسماء المتغيرات حساسة لحالة الحرف (Case sensitive).

كما ينصح عند اختيار اسم لمتغير أن يكون الاسم المختار ذا معنى. أي عند النظر الى اسم المتغير نستطيع أن نعرف ما هي البيانات التي سيتم حفظها في ذلك المتغير.

اسناد القيم للمتغيرات Assigning Values to Variables

عملية إعطاء المتغيرات قيم تتم بطريقة سهلة في PHP . كل ما علينا عمله هو استخدام رمز المساواة (=). عند استخدام هذا الرمز يقوم بإسناد القيمة الموجودة في اليسار الى المتغير الموجود في الطرف الايمن.
مثال:

```
$name = " محمد ";
```

في المثال السابق:
$name : هو اسم المتغير
= : علامة المساواة
"محمد" : هي قيمة المتغير
كما يجب أن لا ننسى أن ننهي العبارة بفاصلة منقوطة. المثال التالي يوضح كيفية التعامل مع المتغيرات:

```
<html>
<head><title>Variables</title>
</head>
<body>
<?php
$name = " محمد عبد الله ";
?>
<b> مرحبا يا </b> <h4> <?php echo $name; ?> </h4>
</body></html>
```

كما هو واضح من المثال السابق ستقوم echo بطباعة قيمة المتغير $name.

كما يمكن إسناد قيمة متغير إلى متغير آخر. كما يمكن كذلك أن تكون قيمة متغير ناتجة من تنفيذ عملية حسابية أو ناتجة عن طريق إدخال المستخدم لقيمة المتغير عن طريق نموذج (HTML form). مثال :

PDF created with pdfFactory Pro trial version www.pdffactory.com

```
<?php
// إعطاء متغير قيمة
$now = 2009;

// اسناد قيمة متغير الى متغير آخر
$currentYear = $now;

// قيمة متغير عن طريق عملية حسابية
$lastYear = $currentYear - 1;
echo "<div dir=rtl>";
echo $lastYear;
echo " إنتهى. مرحبا بكم في العام: ";
echo $currentYear;
echo "</div>";
?>
```

لحذف قيمة متغير، نستخدم الدالة : unset( ) . مثال:

```
unset( $name );
```

أنواع البيانات في PHP (Data types)

تعتبر PHP لغة ضعيفة النوع(Weakly typed) ، بمعني أنها لا تهتم كثيرا لنوع البيانات التي سيتم تخزينها داخل المتغير (على خلاف لغات البرمجة الأخرى مثل، java, c++) . أحيانا تعتبر هذه الطريقة مريحة نوعا ما، لكن قد تحدث بعض المشاكل عندما يقوم المستخدم بإدخال البيانات عن طريق نموذج (HTML form). فمثلا قد يحدث أن يدخل المستخدم بيانات رقمية في الوقت الذي تتوقع منه إدخال نص.
لذلك يجب أن نقوم باختبارات البيانات المدخلة عن طريق المستخدم لتجنب حديث أي مشاكل.
على الرغم من أن PHP لا تهتم كثيرا بنوع البيانات ، إلا انها تستخدم أنواع البيانات التالية:

نوع البيانات (Data type)	الوصف (Description)
Integer	بيانات رقمية صحيحة: 2010 ,1,2 بالاضافة الى الارقام الست عشرية (Hexadecimal) مع ضرورة ان تسبق بصفر : 0xFFFFFF
Float(double)	بينات رقمية عشرية: 1.5, 2.34
String	النصوص ، و من الممكن أن تأخذ أي طول. بمعني أنه من الممكن أن يكون طولها صفر (نص فارغ) أو أي طول.
Boolean	هذا النوع من البيانات يأخذ فقط قيمتين: true , false
Array	المصفوفة، عبارة عن متغير لديه الامكانية لتخزين قيم متعددة من أي نوع من أنواع البيانات.
Null	هذا نوع خاص من البيانات يعني أن المتغير لا يحمل أي قيمة.
Object	كائن/شيء ، أحد أنواع البيانات التي ظهرت بعد دعم PHP لتقنية الـ Object oriented.

المثال التالي يقوم بإيضاح كيفية التعامل مع الأنواع السابقة:

PDF created with pdfFactory Pro trial version www.pdffactory.com

```
<?php

$valid_student = true;

$student_id = 181280000;

$student_name = "Ali Mohammad";

$student_gpa = 3.75;

$student_phone=null;

$student_courses = ("cs101","cs102","cs103");

?>
```

إعداد و التحقق من نوع البيانات (Setting and checking data types)

على عكس لغات البرمجة الأخرى (مثل: java , c++) و التي تشترط أن يتم تحديد نوع البيانات مسبقا. PHP تقوم بتحديد نوع البيانات وفقا لمحتوى المتغير. و في حال تغير نوع البيانات لمحتوى المتغير فإن PHP تقوم تلقائيا بتغيير نوع البيانات وفقا للمحتوى الجديد. و للتحقق من نوع البيانات التي يحتويها متغير ما فإنه من الممكن استخدام الدالة : gettype() . مثال:

```
<?php

$valid_student = true;

echo gettype( $valid_student);

$student_id = 181280000;

echo gettype( $student_id);

?>
```

كما يجب الاشارة أن PHP تحتوي على عدد من الدوال التي تستخدم للتحقق من نوع البيانات:

الدالة (Function)	الوصف (Description)
is_bool()	تستخدم للتحقق ما اذا كان المتغير يحتوي على بيانات من نوع Boolean
is_numeric()	تستخدم للتحقق ما اذا كان المتغير يحتوي على بيانات من رقمية
is_int()	تستخدم للتحقق ما اذا كان المتغير يحتوي على أرقام صحيحة
is_float()	تستخدم للتحقق ما اذا كان المتغير يحتوي على أرقام عشرية
is_string()	تستخدم للتحقق ما اذا كان المتغير يحتوي على نص
is_null()	تستخدم للتحقق ما اذا كان المتغير يحتوي على بيانات فارغة
is_array()	تستخدم للتحقق ما اذا كان المتغير عبارة عن مصفوفة

PDF created with pdfFactory Pro trial version www.pdffactory.com

التعامل مع الثوابت Using Constant

على عكس المتغير و الذي من الممكن أن تتغير قيمته في أثناء سير البرنامج، فإن الثابت يحتوي على قيمة ثابتة لا تتغير. لتعريف الثوابت في PHP يجب الأخذ في الاعتبار مايلي:

- إتباع القواعد الخاصة بتعريف المتغير باستثناء علامة $ و التي ليست مطلوبة مع الثوابت.
- استخدام الدالة ( )define.
- ينصح عند تسمية الثوابت بأن تكتب بأحرف كبيرة (Capital letters) .

المثال يوضح كيفية التعامل مع الثوابت:

```
<?php

define("COURSE"," PHP Programming");
define("YEAR",1430);

echo COURSE;
echo "</br>".YEAR;
?>
```

معالجة المتغيرات باستخدام المعاملات Manipulating variables with operators

عند الرغبة في أجراء أي عملية حسابية على قيم المتغيرات فإننا سوف نحتاج لاستخدام المعاملات (Operators). فمثلا لإجراء عملية الجمع لمتغيرين فإننا سنحتاج لاستخدام معامل الجمع (+) و هكذا.
الجدول التالي يوضح أنواع المعاملات الحاسبية و التي من الممكن استخدامها:

المعامل (Operator)	الحسابي	الوصف (Description)
+		الجمع (Add)
-		الطرح (Subtract)
*		الضرب (Multiply)
/		القسمة (Division)
%		خارج القسمة (Reminder)

مثال:

```
<?php

$num1 = 10;
$num2 = 3;

echo $num1 * $num2;
echo "</br>$num1 / $num2";
echo "</br>".$num1 % $num2;
?>
```


PDF created with pdfFactory Pro trial version www.pdffactory.com

معاملات الزيادة و النقصان Increment and Decrement Operators

معاملات الزيادة (++) و معاملات النقصان من ممكن أن تأتي قبل أو بعد المتغير. في حال أتيا قبل المتغير فستتم عملية الزيادة أو النقصان بمقدار (١) قبل إجراء أي عملية على المتغير. و على العكس تماما إذا أتى أيا من المعاملين بعد المتغير سيتم إجراء أي عملية على المتغير و من ثم ستتم عملية الزيادة أو النقصان.
المثال التالي يوضح كيفية استخدام معاملات الزيادة و النقصان:

```
<?php

$num1 = 10;
$num2 = 3;

echo $num1++;
echo "</br>".$num1;
echo "</br>".++$num1;
?>
```

أسبقية المعاملات Operators Precedence

عند التعامل مع المعاملات في PHP، فإننا نتبع نفس القواعد المتبعة مع العمليات الحسابية في علم الحساب.
لذلك عند تنفيذ أي عملية حسابية ، يجب الأخذ في الاعتبار أسبقية المعاملات من حيث التنفيذ. الجدول التالي يوضح ترتيب المعاملات وفقا لأسبقيتها في التنفيذ:

المعامل الحسابي (Operator)	الأسبقية (Precedence)
( )	أعلى
* / %	↑
+ -	أقل

ربط النصوص String Concatenation

لربط النصوص في PHP، نقوم باستخدام اداة الربط (.). المثال التالي يوضح كيفية ربط النصوص باستخدام اداة الربط:

```
<?php

$name = "Ali Abdullah";
$course = " Web Programming";

echo '<h1>Welcome '.$name.' ,you are enrolled in '.$course.' course<h1>';
?>
```

معاملات المقارنة Comparison Operators

تتيح PHP لنا مقارنة المتغيرات أو القيم عن طريق استخدام معاملات المقارنة. الجدول التالي يوضح معاملات المقارنة في PHP:

PDF created with pdfFactory Pro trial version www.pdffactory.com

معامل المقارنة (Comparison Operator)	الوصف (Description)
==	المساواة (Equality)
!=	عدم المساواة (Inequality)
>	أكبر من (Greater than)
<	أقل من (Less than)
>=	أكبر من أو يساوي (Greater than or equal)
<=	أقل من أو يساوي (Less than or equal)
===	التماثل (Identical)
!==	غير متماثل (Not identical)

من الجدول السابق يجب الإشارة إلى أن المقصود بالتماثل (Identical)، هو أن تكون المتغيرات يجب أن تكون متساوية في القيمة و من نفس نوع البيانات (Data type).

علامة (=) المفردة لا تستخدم في عملية المقارنة، تستخدم فقط عند إسناد القيم للمتغيرات.

المعاملات المنطقية Logical Operators

عند التعامل مع التعابير الشرطية المعقدة، سنقابل عدد من المعاملات المنطقية و التي تستخدم عادة عند دمج أكثر من شرط كما سنشاهد لاحقا. الجدول التالي يوضح المعاملات المنطقية في PHP:

المعامل المنطقي (Logical Operator)	الوصف (Description)
&&	و (and)
||	أو (or)
!	النفى

تسمح PHP باستخدام (and و or) بدلا من (&& و ||) ، و هذي تعطي دلالة على مرونة PHP و إتاحتها لكتابة الشفرة بأكثر من طريقة.

المثال التالي يوضح كيفية التعامل مع معاملات المقارنة و المعاملات المنطقية :

```
<?php
// define variables
$price = 100;
$size = 18;

//comparison operators
echo "<br/>".($price == $size);
echo "<br/>".($price != $size);
echo "<br/>".($price >= $size);
echo "<br/>".($price <= $size);
echo "<br/>".($price === $size);

// logical operators
echo "<br/>".($price > 50 && $size < 25);
echo "<br/>".($price > 150 || $size > 75);
echo "<br/>".!($size > 10);
?>
```


PDF created with pdfFactory Pro trial version www.pdffactory.com

أسئلة الفصل

١. قم بكتابة برنامج يقوم بتحويل العملة من دولار امريكي الى ريال سعودي. علما بأن سعر صرف الدولار= ٣,٧٥ ريال .
٢. متى نستخدم المتغيرات و متى نستخدم الثوابت.
٣. ما الفرق بين كلا من: (===) ، (==) و (=).
٤. ما هي التعابير المنطقية المستخدمة في PHP ، و هل هناك طريقة أخرى لكتابتها؟

PDF created with pdfFactory Pro trial version www.pdffactory.com