Episode 5

Expressions and Operators

Introduction

In this episode, we are going to talk about Expressions and Operators. We are going to learn what an expression is and we are going to learn what different kinds of operators there are to combine one or more expressions to create a new one. We are also going to learn that there are operators that can only be applied to certain data types. And finally, we'll learn about operator precedence.
I hope you have fun in this episode. Let's get started.

Goals

In this section of the course your goals are:

- ☐ To learn what an expression is.
- ☐ To learn the different kinds of PHP opearators which include:
  - ☐ Arithmetic
  - ☐ Assignment
  - ☐ Increment and Decrement
  - ☐ Logical
  - ☐ Comparison
- ☐ To learn about operator precedence.

Expressions

What is an expression? In PHP an expression is anything that has a value. It can be a variable, a constant or even a function or an array, about which we will learn later. Let's look at some examples:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Expressions</title>
</head>
<body>
</body>
</html>
```

Inside the body we are going to add the opening and closing php tags, and let's add the following:

```
echo 22.76;
echo "<br>";
```

The simplest form of an expression is a single value.
In this case the number 22.76.

An expression can also be a variable:

```
$age = 3;
echo $age;
echo "<br>";
```

In the second line we have an expression in the form of the variable $age with a value of 3.

What about a string:

```
echo "Eric";
echo "<br>";
```

Notice that we are outputting two values to the screen. The first one is the string Eric and the second is the string that the browser interprets as a line break.
In fact, we have done this for all three examples.

Let's take a look at a more complex example:

```
$friend1 = "Hallie";
$friend2 = "Tiffany";
$friend3 = "MacKenzie";
$friend4 = "Leah";
```

We have created four string variables. Each has a value, which means that each of them is an expression.
Now we are going to combine them into a single expression.

```
$singleExpression = "My name is Eric, I am $age years old.<br>
                     I have many friends: $friend1, $friend2,<br>
                     $friend3, $friend4, Jim and Vinnie.<br>";

echo $singleExpression;
```

Here is the file with all the examples:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Expressions</title>
</head>
<body>
    <?php
    // The simplest expression is a single value -- 22.76
    echo 22.76;
    echo "<br>";

    // An expression can also be a variable
    $age = 3;
    echo $age;
    echo "<br>";

    // Or a string.
    echo "Eric";
    echo "<br>";

    // Many expressions...
    $friend1 = "Hallie";
    $friend2 = "MacKenzie";
    $friend3 = "Tiffany";
    $friend4 = "Leah";

    // ...can be combined into one
    $singleExpression = "My name is Eric, I am $age years old.<br>
                         I have many friends: $friend1, $friend2,<br>
                         $friend3, $friend4, Jim and Vinnie.<br>";

    echo $singleExpression;

    ?>
</body>
</html>
```

That last example is one way to combine several expressions into one.

Operators

We can also use operators to combine more than one expression into one, or to change the value of a single expression. Let's look at some examples:

Arithmetic Operators

The operators that you'll be most familiar with are the arithmetic operators:
There is the addition operator:

```
$sum = 5 + 7;
$sum2 = 5.3 + 3.4 + 4.5;
echo $sum, "<br>", $sum2, "<br><br>";
```

The subtraction operator:

```
$sub = 4 - 10;
$sub2 = 34.56 - 22.76;
echo $sub, "<br>", $sub2, "<br><br>";
```

The multiplication operator:

```
$mult = 5 * 7;
$mult2 = 5.3 * 3.4 * 4.5;
echo $mult, "<br>", $mult2, "<br><br>";
```

The division operator:

```
$div = 8 / 2;
$div2 = 34.56 / 22.76;
$div3 = 7 / 5;
echo $div, "<br>", $div2, "<br>", $div3, "<br><br>";
```

Notice that the division operator yields a float type when a division between two integers doesn't give an exact result.
And the modulus operator:

```
$mod = 11 % 3;
$mod2 = 5.5 % 2.5;
echo $mod, "<br>", $mod2, "<br><br>";
```

The modulus operator yields the remainder of a division between two integers. It is usually not used with floats. But to illustrate what happens when you use two floats values, those values are converted to integers and then the operation takes place.
Finally we have the exponentiation operator, which consists of two asterisks.

```
$exp = 2 ** 3;
$exp2 = 5.5 ** 2.5;
echo $exp, "<br>", $exp2, "<br><br>";
```

Assignment Operators

We have been using the assignment operator from the beginning and it is just the equal sign:

```php
$var = 3;
echo $var, "<br>";
```

Whatever is to the right of the equal sign is evaluated and assigned to the variable or constant on the left.

Increment and Decrement Operators

Operator	Name	Result
++$var	Pre-Increment	Increments $var by one then uses it
$var++	Post-Increment	Uses $var then increments it by one
- -$var	Pre-Decrement	Decrements $var by one then uses it
$var- -	Post-Decrement	Uses $var then decrements it by one

To show how this works let's start by giving $inc a value of 10.
Next, we're going to use the pre-increment operator on $inc inside an echo statement.
What happens here is that $inc is incremented to 11 and then it is used by the echo statement.
Next, we'll use the post-increment operator on $inc. When we output it to the screen, we see that we still get 11.
What happened here is that $inc was first used by echo and only after was its value incremented to 12.
If we output $inc once again, this time without an increment operator, we see that its value is in fact 12.

```php
$inc = 10;
echo ++$inc, "<br>";
echo $inc++, "<br>";
echo $inc, "<br>";
```

The decrement operator works in the same manner.

Logical Operators

Name	Example	Result
And	$x and $y	**TRUE** only if both $x and $y are **TRUE**
And	$x && $y	**TRUE** only if both $x and $y are **TRUE**
Or	$x or $y	**TRUE** if either $x or $y is **TRUE**
Or	`$x || $y`	**TRUE** if either $x or $y is **TRUE**
Xor	$x xor $y	**TRUE** if either $x or $y is **TRUE**, but not both
Not	!$x	**TRUE** only if $x is **FALSE**

Logical operators are used usually used in control structures like loops and conditional statements both of which we will learn later.
Here are some examples:

```
echo "TRUE and TRUE : ";
var_dump(true and true);
echo "<br>";
echo "TRUE and FALSE : ";
var_dump(true and false);
echo "<br>";
echo "FALSE and FALSE : ";
var_dump(false and false);
echo "<br><br>";

echo "TRUE or TRUE : ";
var_dump(true or true);
echo "<br>";
echo "TRUE or FALSE : ";
var_dump(true or false);
echo "<br>";
echo "FALSE or FALSE : ";
var_dump(false or false);
echo "<br><br>";

echo "TRUE xor TRUE : ";
var_dump(true xor true);
echo "<br>";
echo "TRUE xor FALSE : ";
var_dump(true xor false);
echo "<br>";
echo "FALSE xor FALSE : ";
var_dump(false xor false);
echo "<br><br>";

echo "!TRUE : ";
var_dump(!true);
echo "<br>";
echo "!FALSE : ";
var_dump(!false);
echo "<br><br>";
```

Comparsion Operators

Most people should also be familiar with comparison operators from middle school math:

Name	Example	Result
Equal	$x == $y	**TRUE** if $x and $y have same value

Name	Example	Result
Not Equal	$x != $y	**TRUE** if $x and $y don't have same value
Greater Than	$x > $y	**TRUE** if $x is greater than $y
Less Than	$x < $y	**TRUE** if $x is less than $y
Greater Than or Equal to	$x >= $y	**TRUE** if $x is equal to or greater than $y
Less Than or Equal to	$x <= $y	**TRUE** if $x is equal to or less than $y
Identical	$x === $y	**TRUE** if $x and $y have the same value and are of the same type

Comparison operators are also usually used in control structures. Here are some examples:

```php
$com1 = 5;
$com2 = 10;

echo "5 is equal to 10 : ";
var_dump($com1 == $com2);
echo "<br>";

echo "5 is not equal to 10 : ";
var_dump($com1 != $com2);
echo "<br>";

echo "5 is greater than 10 : ";
var_dump($com1 > $com2);
echo "<br>";

echo "5 is less than 10 : ";
var_dump($com1 < $com2);
echo "<br>";

echo "5 is greater than or equal to 10 : ";
var_dump($com1 >= $com2);
echo "<br>";

echo "5 is less than or equal to 10 : ";
var_dump($com1 <= $com2);
echo "<br><br>";

$com3 = 7;
$com4 = "7";

echo "7 is equal to '7' : ";
var_dump($com3 == $com4);
echo "<br>";

echo "7 is identical  to '7' : ";
var_dump($com3 === $com4);
echo "<br><br>";
```

String Concatenation Operator

There is one more operator that we are going to talk about: The string concatenation operator. Let's look at an example:

```php
$name = "Eric ";
$desc = "is very friendly.";
```

We have two string variables, let's put them together using the concatenation operator:

```
$sent = $name . $desc;
echo $sent;
```

The $sent variable is now a single string.

Operator Precedence

Operator precedence means that PHP knows when to use an operator before using a different one. Here are some examples:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Expressions</title>
    <style> body {margin: 30px;}</style>
</head>
<body>
    <?php
    // ----------------------------
    // ** ++ and -- are higher than
    // * / and %
    // ----------------------------
    //*
    $a = 4 * 5 ** 2;
    echo $a;
    echo "<br><br>";
    //*/

    // -------------------------
    // * / and % are higher than
    // + - and .
    // -------------------------
    //*
    $b = 4 * 5 + 6;
    echo $b;
    echo "<br><br>";
    //*/

    // -------------------------
    // + - and . are higher than
    // > >= < and <=
    // -------------------------
    //*
    $c = 5 + 3 < 5 + 2;
    var_dump($c);
    echo "<br><br>";
    //*/
```

```
    // -------------------------
    // > >= < and <= are higher than
    // and or xor && ||
    // -------------------------
    //*
    $d = (5 > 3 and 4 > 2);
    var_dump($d);
    echo "<br><br>";
    //*/

    ?>
</body>
</html>
```

Here is a table with the operators that we have looked at so far. Those higher on the table have higher precedence. You can override operator precedence using parenthesis.

Operator
** ++ - -
!
* / %
+ - .
> >= < <=
== != ===
&& `||` and or xor

Debugging Code

Examine the following code. Find and fix the errors.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Debugging Episode 5</title>
    <style> body{margin: 30px;} </style>
</head>
<body>
    <?php
    echo &sum = 5 + 4;
    echo "<br>";

    &diff = 56.3 - 3.23
    echo $dif;
    echo "<br>";

    $mult1 = 4 - * 6;
    $mult2 = -4 * 5;
    echo $mult1 / $mult2;
    echo "<br>";

    // I want the result to be 18
    $res = 2 * 3 + 6;
    echo $res;
    echo "<br>";

    // I want the result to be 120
    $res2 = 3 * 4 + 2 * 5;
    echo $res2;
    echo "<br>";

    // There is something missing
    $x = (44.5 * 0.55 - 50.2 * 5.9) / (3.4 * 0.55 - 50.2 * 2.1;
    echo $x;

    ?>
</body>
</html>
```

Lab Exercises

The following exercises are meant to help you practice your coding skills. They need to be submitted for you to get a certification. As stated before coding is not a spectator sport, the more you practice (and make mistakes) the more you will learn.

1. Write a script that displays the result of

$$9.5 * 4.5 - 2.5 * 3$$
$$45.5 - 3.5$$

2. Pi can be computed using the following formula:

$$\pi = 4 * \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} \cdots\right)$$

   Write a script that calculates the result of:

$$4 * \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13}\right)$$

3. Given the following $2X2$ system of linear equation:

$$ax + by = e$$
$$cx + dy = f$$

   You can use Cramer's rule to solve it.

$$x = \frac{ed - bf}{ad - bc} \qquad y = \frac{af - ec}{ad - bc}$$

   Write a script that solves the following equation and displays the values for x and y.

$$2.2x + 5.6y = 45.4$$
$$1.3x + 7.1y = 34.3$$

In the next episode, we'll talk about forms. See you then.