Episode 8

Control Structures II: Conditional Statements

Introduction

(Welcome back to another episode of PHP Fundamentals. My name is Julio Quevedo for Framework TV). In the last episode, we took our first look at control structures: Loops. In this episode, we will learn about another kind of control structure: Conditional Statements.

Goals

In this section of the course your goals are:

- ☐ To learn about the **if** statement.
- ☐ To learn about **if . . . else**.
- ☐ To learn about **if . . . elseif . . . else**
- ☐ To learn about **switch**
- ☐ To learn how to use the keywords **break** and **continue**

The if statement

The **if** statement allows you to execute a statement or block of code if a condition is true. Let's take a look at the following code:

```
$temp = 30;

if ($temp < 32) {
    echo "Bundle up, it's cold out there.<br>";
}
echo "Have a great day!<br>";
```

If we examine this code, no pun intended, we have the variable $temp set to 30, and then the **if** statement begins. Inside the parenthesis, we have a condition that evaluates to either true or false. If the condition is true, then the statement inside the curly brackets will run, which is what happens when the value of $temp is 30. Let's now set the value of $temp to 40. When we rerun the code, only the last echo statement is executed. Notice that in either case the echo statement outside of the curly brackets is executed, which means that only the portion inside the brackets is affected by the conditional statement.

if . . . else

Use **if . . . else** when you want a block of code to be executed if a condition is true and another block if the condition is false.
Let's expand our previous example:

```
$temp = 20;

if ($temp < 32) {
    echo "Bundle up, it's cold out there.<br>";
} else {
    echo "Not that cold this morning.<br>";
}
echo "Have a great day!<br>";
```

In this case, if the value of $temp is below 32, one message will appear on the screen. If it is 32 or above, a different message will appear on the screen. As in the previous example, the last echo statement is printed in both cases.

if . . . elseif . . . else

What if instead of having one or two choices, you have a few choices. Then use **if . . . elseif . . . else**. Let's expand our example a little more:

```
$temp3 = 80;

if ($temp3 < 32) {
    echo "Bundle up, it's cold out there.<br>";
} elseif ($temp3 < 55) {
    echo "Not that cold this morning.<br>";
} elseif ($temp3 < 75) {
    echo "It's getting warm out here.<br>";
} else {
    echo "It's a hot morning.<br>";
}
echo "Have a great day!<br>";
```

Changing the value of $temp3 will result in different messages being displayed on the screen. If $temp3 is less than 32, then the first echo statement will be executed. Notice that even though when $temp is less than 32, the other two conditionals are true also, the echo statements corresponding to those conditionals are not executed. Once a true conditional is found all others are skipped, and only the statement outside of the if statement is printed.

Switch and Break

The switch statement works similarly to a series of **if. . . elseif** statements. Let's look at an example:

```
$day = 3;

switch ($day) {
    case 1:
        echo "Today is monday.<br>";
        break;
    case 2:
        echo "Today is tuesday.<br>";
        break;
    case 3:
        echo "Today is wednesday.<br>";
        break;
    case 4:
        echo "Today is thursday.<br>";
        break;
    case 5:
        echo "Today is friday.<br>";
        break;
    case 6:
        echo "Today is saturday.<br>";
        break;
    case 7:
        echo "Today is sunday.<br>";
        break;
    default:
        echo "Have a nice day.<br>";
}
```

The value of $day is 3, so the switch statement uses the **case** keyword to test for that value. The first two tests: **case 1:**, and **case 2:** are false and therefore they are ignored. When it reaches **case 3:**, the next echo statement is executed and then the **break** keyword exits switch.

A critical note on how a switch works: If we remove all the break keywords, then when the first case test is true all the subsequent cases will also be executed. No more testing is done. Let's look at the following code:

```php
$day = 3;

switch ($day) {
    case 1:
        echo "Today is monday.<br>";
    case 2:
        echo "Today is tuesday.<br>";
    case 3:
        echo "Today is wednesday.<br>";
    case 4:
        echo "Today is thursday.<br>";
    case 5:
        echo "Today is friday.<br>";
    case 6:
        echo "Today is saturday.<br>";
    case 7:
        echo "Today is sunday.<br>";
    default:
        echo "Have a nice day.<br>";
}
```

When we run this code, the output will be:

```
Today is Wednesday.
Today is Thursday.
Today is Friday.
Today is Saturday.
Today is Sunday.
Have a nice day.
```

If all the cases are false, or when there a no break keywords, what's in default will execute.

Continue

The **continue** keyword is used to skip the remainder of a loop iteration. Let's take a look at an example:

```php
for ($i = 1; $i <= 20; $i++){
    if (($i % 2) == 0)
        continue;
    echo $i.' ';
}
```

This for loop prints only odd numbers to the screen. As the loop iterates from 1 to 20, the if statement checks if the number is even. If it is, then **continue** is executed, and the rest of the iteration of the loop is skipped, in this case, the echo statement. So when the numbers are even, echo doesn't run.

Debugging Code

Examine the following code. Find and fix the errors.
When free of bugs the following code will output three strings to the screen: The first will be numbers divisible by 5, the second will be numbers divisible by 6 and the third will be numbers divisible by 5 and 6. There are both syntax and logical errors.

```
<?php
$number = 1000;
$div1 = 5;
$div1 = 6
$output1 = '';
$output2 = '';
$output12 = '';

for($i = 1, $i <= number, $i++){
    $flag = 0;
    if(($i % $div1) == 0){
        $output1 .= "$i ";
        $flag++;
    }
    if(($i % $div2) == 0){
        $output2 .= "$i ";
        $flag--;
    }
    if($flag == 3){
        $output12 .= "$i ";
    }
}

?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Debugging</title>
        <style>body{margin: 30px;}</style>
    </head>
    <body>
        <?php
        echo "Divisible by $div1: $output1<br><br>";
        echo "Divisible by $div2: $output2<br><br>";
        echo "Divisible by $div1 and $div2: $output12<br><br>";
        ?>
    </body>
</html>
```

Lab Exercises

1. *Quadratic Formula*. The two real roots of a quadratic equation

$$ax^2 + bx + c = 0$$

can be found using the formulas:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad and \quad r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Where $b^2 - 4ac$ is called the discriminant of the equation.
If the discriminant is positive then there are two roots, if it is zero, then there is only one root, and if it is negative, then there are no roots.

- Write a script that will take three variables: a, b and c. Determine whether the discriminant is positive, zero or negative and based on that find both roots, one root or output a message saying that the equation has no roots.

2. Using the switch statement write a script that will ask for a number from 0 to 6 and output the day of the week. (Sunday = 0, Monday = 1, . . . , Saturday = 6). If the number entered is outside this range output a message saying only numbers from 0 to 6.

3. *Palindrome Number*. A palindrome number is a number that is the same when written backward. Ex: 121 or 303. Write a script that takes a three digit number and determines whether the number is a palindrome.