PHP If...Else Statements

In this tutorial you'll learn how to write decision-making code using if...else...elseif statements in PHP.

PHP Conditional Statements

Like most programming languages, PHP also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time. This means, you can create test conditions in the form of expressions that evaluates to either `true` or `false` and based on these results you can perform certain actions.

There are several statements in PHP that you can use to make decisions:

- The **if** statement
- The **if...else** statement
- The **if...elseif....else** statement
- The **switch...case** statement

We will explore each of these statements in the coming sections.

The if Statement

The *if* statement is used to execute a block of code only if the specified condition evaluates to true. This is the simplest PHP's conditional statements and can be written like:

```
if(condition){
    // Code to be executed
}
```

The following example will output "Have a nice weekend!" if the current day is Friday:

Example **Run this code »**

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
}
?>
```

The if...else Statement

You can enhance the decision making process by providing an alternative choice through adding an *else* statement to the *if* statement. The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false. It can be written, like this:

```
if(condition){
    // Code to be executed if condition is true
} else{
    // Code to be executed if condition is false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!"

Example **Run this code »**

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} else{
    echo "Have a nice day!";
}
?>
```

The if...elseif...else Statement

The *if...elseif...else* a special statement that is used to combine multiple *if...else* statements.

```
if(condition1){
    // Code to be executed if condition1 is true
} elseif(condition2){
    // Code to be executed if the condition1 is false and condition2 is
true
} else{
    // Code to be executed if both condition1 and condition2 are false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday, otherwise it will output "Have a nice day!"

Example **Run this code »**

```php
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} elseif($d == "Sun"){
    echo "Have a nice Sunday!";
} else{
    echo "Have a nice day!";
}
?>
```

You will learn about PHP switch-case statement in the next chapter.

The Ternary Operator

The ternary operator provides a shorthand way of writing the *if...else* statements. The ternary operator is represented by the question mark (`?`) symbol and it takes three operands: a condition to check, a result for `true`, and a result for `false`.

To understand how this operator works, consider the following examples:

Example **Run this code »**

```php
<?php
if($age < 18){
    echo 'Child'; // Display Child if age is less than 18
} else{
    echo 'Adult'; // Display Adult if age is greater than or equal to
18
}
?>
```

Using the ternary operator the same code could be written in a more compact way:

Example **Run this code »**

```php
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

The ternary operator in the example above selects the value on the left of the colon (i.e. 'Child') if the condition evaluates to true (i.e. if `$age` is less than 18), and selects the value on the right of the colon (i.e. 'Adult') if the condition evaluates to false.

Tip: Code written using the ternary operator can be hard to read. However, it provides a great way to write compact if-else statements.

The Null Coalescing Operator PHP 7

PHP 7 introduces a new null coalescing operator (`??`) which you can use as a shorthand where you need to use a ternary operator in conjunction with `isset()` function.

To uderstand this in a better way consider the following line of code. It fetches the value of `$_GET['name']` , if it does not exist or `NULL` , it returns 'anonymous'.

Example **Run this code »**

```php
<?php
$name = isset($_GET['name']) ? $_GET['name'] : 'anonymous';
?>
```

Using the null coalescing operator the same code could be written as:

Example **Run this code »**

```php
<?php
$name = $_GET['name'] ?? 'anonymous';
?>
```

As you can see the later syntax is more compact and easy to write.