PHP variables

Summary: in this lesson, you will learn how to use **PHP variables** to store data in programs.

Define a variable

A variable stores a value of any type, e.g., a string, a number, an array, or an object.

A variable has a name and is associated with a value. To define a variable, you use the following syntax:

```
$variable_name = value;
```

When defining a variable, you need to follow these rules:

- The variable name must start with the dollar sign ($).
- The first character after the dollar sign ($) must be a letter (a-z) or the underscore (_).
- The remaining characters can be underscores, letters, or numbers.

PHP variables are case-sensitive. It means that $message and $Message variables are entirely different.

The following example defines a variable called $title:

```
<?php
$title = "PHP is awesome!";
```

To display the values of variables on a webpage, you'll use the echo construct. For example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>PHP Variables</title>
```

```
</head>
<body>
   <?php
      $title = 'PHP is awesome!';
   ?>
   <h1><?php echo $title; ?></h1>
</body>
</html>
```

If you open the page, you’ll see the following message:

```
PHP is awesome!
```

Another shorter way to show the value of a variable on a page is to use the following syntax:

```
<?= $variable_name ?>
```

For example, the following shows the value of the $title variable in the heading:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>PHP Variables</title>
</head>
<body>
   <?php
                  $title = 'PHP is awesome!';
         ?>

   <h1><?= $title; ?></h1>
</body>
</html>
```

Mixing PHP code with HTML will make the code unmaintainable, especially when the application grows. To avoid this, you can separate the code into separate files. For example:

- index.php – store the logic for defining and assigning value to variables.
- index.view.php – store the code that displays the variables.
- Use the require construct to include the code from the index.view.php in the index.php file.

The following shows the contents of the index.view.php file:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <title>PHP Variables</title>
</head>
<body>
   <h1><?= $title ?></h1>
</body>
</html>
```

And the following shows the contents of the index.php file:

```
<?php

$title = 'PHP is awesome!';
require 'index.view.php';
```

If you open the index.php file on the web browser, you'll see the same output.

By doing this, you separate the code responsible for logic and the code responsible for displaying the file. This is called the **separation of concerns** (SoC) in programming.

Summary

- A variable stores a value, and its name always starts with $ sign.
- Use the separation of concerns principle to separate the PHP logic from HTML.

PHP constants

A constant is simply a name that holds a single value. As its name implies, the value of a constant cannot be changed during the execution of the PHP script.

To define a constant, you use the define() function.
The define() function takes the constant's name as the first argument and the constant value as the second argument. For example:

```
<?php

define('WIDTH','1140px');
echo WIDTH;
```

By convention, constant names are uppercase. Unlike a variable, the constant name doesn't start with the dollar sign($).

By default, constant names are case-sensitive. It means that WIDTH and width are different constants.

It's possible to define case-insensitive constants. However, it's deprecated since PHP 7.3

In PHP 5, a constant can hold a simple value like a number, a string, a boolean value. From PHP 7.0, a constant can hold an array. For example:

```
<?php

define( 'ORIGIN', [0, 0] );
```

Like superglobal variables, you can access constants from anywhere in the script.

The const keyword

PHP provides you with another way to define a constant via the const keyword. Here's the syntax:

```
const CONSTANT_NAME = value;
```

In this syntax, you define the constant name after the const keyword. To assign a value to a constant, you use the assignment operator (=) and the constant value. The constant value can be scalar, e.g., a number, a string, or an array.

The following example uses the const keyword to define the SALES_TAX constant:

```
<?php

const SALES_TAX = 0.085;

$gross_price = 100;

$net_price = $gross_price * (1 + SALES_TAX);

echo $net_price; // 108.5
)
```

The following example uses the const keyword to define the RGB constant that holds an array:

```
<?php

const RGB = ['red', 'green', 'blue'];
```

define vs const

First, the define() is a function while the const is a language construct.

It means that the define() function defines a constant at run-time, whereas the const keyword defines a constant at compile time.

In other words, you can use the define() function to define a constant conditionally like this:

```
<?php

if(condition)
{
   define('WIDTH', '1140px');
}
```

However, you cannot use the const keyword to define a constant this way. For example, the syntax of the following code is invalid:

```
<?php


if(condition)

{

  const WIDTH = '1140px';

}
```

Second, the define() function allows you to define a constant with the name that comes from an expression. For example, the following defines three constants OPTION_1, OPTION_2, and OPTION_3 with the values 1, 2, and 3.

```
<?php

define('PREFIX', 'OPTION');


define(PREFIX . '_1', 1);
define(PREFIX . '_2', 2);
define(PREFIX . '_3', 3);
```

However, you cannot use the const keyword to define a constant name derived from an expression.

Unless you want to define a constant conditionally or use an expression, you can use the const keyword to define constants to make the code more clear.

Note that you can use the const keyword to define constants inside classes.

Summary

- A constant is a name that holds a simple value that cannot be changed during the execution of the script. From PHP 7, a constant can hold an array.
- A constant can be accessed from anywhere in the script.
- Use the define() function or const keyword to define a constant.
- Use the define() function if you want to define a constant conditionally or using an expression.