PHP Variables, Strings, Constants, Arrays, and Functions

by Sophia

WHAT'S COVERED

In this lesson, you will learn about PHP variables, including how to interpolate a variable's data into strings and two methods for creating constant variables. Additionally, you will learn how to create array collections in PHP. You will learn how to create custom-defined PHP functions to serve your processing needs. Finally, you will be introduced to how PHP handles object-oriented concepts.

Specifically, this lesson will cover the following:

1. PHP Variables

PHP is a loosely typed scripting language that interprets a variable's data type by the syntax of the data assigned to it, much like JavaScript. The process of interlacing variables within strings of text using syntax that tells the software to replace the variable with its value is called **interpolation**. To declare a variable, simply provide the variable's name, prefixed with a dollar sign, and then assign it a value. Variable names are case sensitive.

EXAMPLE Declaring PHP variables:

```
<?php
     $firstName = "John";
     $number = 45;
?>
```

When you need to access the value of a variable, you can simply reference the full name of the variable, along with the dollar sign prefix.

EXAMPLE Declaring PHP variables:

```
<?php
     echo "<h1>Welcome, $firstName!</h1>";
?>
```

Unlike many programming or scripting languages, PHP variables do not contain any built-in internal methods; rather, they are available as stand-alone methods. Any method used to manipulate a variable's contents is used by passing the variable itself into the function as an argument. The function will then manipulate the variable as needed and return the results, whereas JavaScript variables contain their own internal methods and will act upon their own data. For instance, a JavaScript string variable contains the *varname*.substr() function, which will return a portion of the original data based on the arguments provided. In PHP, we use the same function but need to pass the string in as the first argument along with the offset value to indicate where to slice the string.

EXAMPLE Using PHP and JavaScript variable methods:

```
<?php   $name = "John Doe";
  echo substr( $name, 3 ); //echoes: n Doe
?>
<script>
  name = "John Doe";
  console.log( name.substr(3) ); //prints: n Doe

</script>
```

KEY CONCEPT

PHP variable names are case sensitive, as is the case with many programming languages. However, built-in operator names and function names are not case sensitive.

The following three commands will work just fine:

EXAMPLE Built-in operators and functions are not case sensitive:

```
<?php
    ECHO suBStR( "John Doe", 3 ); //echoes: n Doe
    eCHo SUBSTR( "John Doe", 3 ); //echoes: n Doe
    Echo sUBStR( "John Doe", 3 ); //echoes: n Doe
?>
```

However, if you try this with the name of a variable, you will encounter an error stating that the variable does not exist.

TERM TO KNOW

Interpolation
The process of interlacing variables within strings of text using syntax that tells the software to replace the variable with its value.

1a. Interpolating String Data

A common operation of any coding language is interpolating variable or object data within a string of text. PHP does provide the ability to interpolate variables directly within a string of text; however, this depends on the type of quotes used.

Double quotes will automatically interpolate variables:

EXAMPLE Comparing single with double quotes for data interpolation:

```
<?php
$name = "John Doe";
echo "Hello $name, Welcome!"; //output: Hello John Doe, Welcome!
echo 'Hello $name, Welcome!'; //output: Hello $name, Welcome!
?>
```

Additionally, if the data were to be concatenated with additional text, as the first part of a compound word, then you would use double quotes and surround the variable name with curly brackets:

EXAMPLE Forcing exact interpolation using curly brackets:

```
<?php
$dir = "upper";
echo "Please use {$dir}case letters when writing your name";
?>
//output: Please use uppercase letters when writing your name
```

Lastly, you can pre-assemble strings and interpolate variable data using what is called a **“heredoc” statement**. The heredoc statement starts like a normal variable assignment statement, but after the assignment operator, there are three less than characters followed by unquoted text and variables. Anything that comes after the three less than characters and before the semicolon will be evaluated and embedded in the variable, including line breaks!

EXAMPLE Using the heredoc statement to produce the output:

```
<?php
$name = "John Doe";
$welcomeMSG = <<< Hello $name, how are you today?
Shall we get started?;
?>
```

Output:
```
Hello John Doe, how are you today?
Shall we get started?
```

TERM TO KNOW

“heredoc” Statement
Special syntax that interprets the text within a code file without processing the code and instead treats it as plain text.

1b. Constant Variables

Constant variables, also just called constants, are variables that cannot be modified once they have been declared and initialized. In programming, constants are ideal for any kind of value or data that will be useful throughout a program and will likely never need to be changed for the life of the application (or at least the current version).

The benefit is that the same identifier can be used through a program without fear of that value ever being changed during the run-time of the application.

In PHP, constants can be declared using one of two methods:

1. The define() method
2. The “const” keyword

Both options have the same effect; however, the const keyword option can be used both outside of and inside of class definitions. The define() function is only usable outside of a **class definition**.

EXAMPLE Declaring constants with define() and const:

```
<?php
     define("MIN_VALUE", 0.0);
     define("MAX_VALUE", 1.0);

     const MIN_DISCOUNT = 0.0;
     const MAX_DISCOUNT = 0.1;
?>
```

KEY CONCEPT

Notice how the identifiers are written in all caps and with underscores separating the words. Because of the nature of constant variables, this is done intentionally to clearly identify them as such. Anytime you are examining code and you notice this naming convention, you should automatically assume that its value is defined somewhere as a constant variable.

EXAMPLE Declaring constants with const inside of a class definition:

```
<?php

    class Example
    {
          const MIN_VALUE = 0.0;
          const MAX_VALUE = 1.0;

          //define("MIN_VALUE", 0.0);   //Not valid inside of class
```

```
        //define("MAX_VALUE", 1.0);  //Not valid inside of class

        public static function getMinValue()
        {
             return self::MIN_VALUE;
        }

        public static function getMaxValue()
        {
             return self::MAX_VALUE;
        }
    }
?>
```

TERMS TO KNOW

Constant Variables
Variables defined in such a manner that their value cannot be changed for the life of the program.

Class Definition
A set of code in object-oriented programming that defines the structure of an object.

---

2. PHP Arrays

Arrays in PHP behave and operate much in the same manner as arrays in other programming or scripting languages. Remember that arrays are used to hold multiple individual values within a single object. To create an array in PHP, you will declare a variable and assign it the value returned from the array() function. The initial elements of the array are included as a comma-separated list within the parentheses.

EXAMPLE Declaring a PHP array:

```
<?php
     $recipe = array( "Banana", 2, "Milk", 0.5 );
?>
```

Just like arrays in other languages, accessing one of the elements in the array is done by placing the 0-based index number between square brackets postfixed to the end of the array's identifier.

EXAMPLE Retrieving an element using the index indicator:

```
<?php
     $recipe = array( "Banana", 4, "Milk", 0.5 );
     echo "1. add $recipe[1] $recipe[0] to a blender.<br>";
     echo "2. add $recipe[3] cups of $recipe[2] to the blender.<br>";
?>
```

```
Output:
1. add 4 Banana to a blender.
2. add 0.5 cups of Milk to the blender.
```

Furthermore, there are a plethora of array methods that can be used to manipulate the elements within the array. However, just like PHP variable methods, these methods are not internal but are stand-alone methods that will affect the array passed in as an argument. Using these methods, we can add new elements, remove elements, sort them, filter them, and much more. Let us add one more ingredient, which consists of two new elements, to the recipe from above.

EXAMPLE Declaring constants with define() and const:

```
<?php
     $recipe = array( "Banana", 4, "Milk", 0.5 );
     echo "1. add $recipe[1] $recipe[0] to a blender.<br>";
     echo "2. add $recipe[3] cups of $recipe[2] to the blender.<br>";
     array_push( $recipe, "Date", 2);
     echo "3. add $recipe[5] $recipe[4] to the blender.<br>";
?>
```

```
Output:
1. add 4 Banana to a blender.
2. add 0.5 cups of Milk to the blender.
3. add 2 Date to the blender.
```

In the array_push(), for example, we provide the original array as the first argument, and the subsequent arguments are the values that will be added to the end of the original array. The new array will now contain the values of ("Banana", 4, "Milk", 0.5, "Date", 2).

KEY CONCEPT

Did you happen to notice the grammatical issue with the word “Banana” in the output of both “1. add 4 Banana . . .” and “3. add 2 Date to the blender.”? It does not match the fact that there are multiple bananas and multiple dates. This was done purposefully to illustrate such issues with interpolating data into a sentence. We could simply update the word to “Bananas,” but what if the value of bananas happened to only be 1? We would have to change “Bananas” back to “Banana.” This is a common issue that can be handled in a couple of ways. In larger application projects, such grammar changes may be handled at the time of input or at the time the data gets used. There will need to be a section of programming logic that detects a word that should be plural and adjusts the word as needed by adding the “s” to the end of the word.

3. Custom-Defined PHP Functions

In PHP, like most languages, developers can define their own **custom-defined PHP functions** that can be reused as needed. Functions are defined using the “function” keyword, followed by the name of the function, the parameter list in parentheses, and the body of code to be executed. Just like typical functions, they will terminate when they reach their final command or a return command. Data can be passed in as arguments and must be provided when the function’s definition includes parameters.

EXAMPLE Defining a custom PHP function:

```
<?php
     function sayHello( $name )
     {
        echo "Hello $name, welcome to the show!";
     }

     sayHello( "John" ); //echoes: Hello John, welcome to the show!
?>
```

It is important to note that in PHP, built-in functions and operator keywords are not case sensitive; however, custom-defined PHP functions, like variable names, are indeed case sensitive.

Functions in PHP can call other functions as well as **recursive function calls**, wherein a function calls itself.

PHP functions do support **default arguments**. These are parameters that are initialized with a value in the parentheses. When the calling code provides a value for the default argument, the value is used instead of the default value. However, a default parameter’s argument can be omitted at the time of the call, in which case the function will still execute but will use the default value.

EXAMPLE Declaring constants with define() and const:

```
<?php
     function alertMessage( $message = "Something went wrong." )
     {
        echo "Error: $message."
     }
     alertMessage("Connection to Database lost"); //echoes: Error: Connection to Database lost.
     alertMessage(); //echoes: Error: Something went wrong.
?>
```

TERMS TO KNOW

Custom-Defined PHP Functions
Callable, named sets of code in PHP, written by the developer.

Recursive Function Calls
When a programming function makes a call to itself.

Default Arguments
When defining a function, default arguments create an optional parameter that, if omitted at the time of the call, will use a default, predetermined value.

4. PHP Classes and Objects

Because PHP supports object-oriented programming, PHP provides us with the ability to define a class structure and use it to create objects in memory. Classes can contain variables, objects, and functions.

When we define a class, we start with the class keyword, the name of the class (typically the first letter is capitalized), and then a set of curly brackets to define its scope.

Then, within the curly brackets, we can define our **class attributes**, also called member variables, and **methods**, also called member functions. Class attributes also can be given **access modifiers** to control access to the attribute values. Setting a class attribute as "**private**" will prevent direct access to the member variable from outside of the class or object instance. Instead, private attributes must be accessed through the use of one of the member functions, often referred to as **accessor methods** (getName(), getValue()) and **mutator methods** (setName(), setValue()). The other access modifiers for class attributes are **public**, in which the value can be accessed and modified from outside of the object, and **protected**, in which only derived subclasses can access the attribute.

EXAMPLE Defining a PHP class:

```
<?php
 class Employee
 {
    public $name;
    protected $pay;
    private $SSN;

    function getPay()
    {
      return $this->pay;
    }
    function setPay( $newPay )
    {
      $this->pay = $newPay;
    }
    function getSSN()
    {
      return $this->SSN;
    }
    function setSSN( $newSSN )
    {
      $this->SSN = $newSSN;
    }
 }

?>
```

The example above defines a basic class for a generic employee object. It includes three attributes, $name, $pay, and $SSN, each with its own access modifier. We also included two sets of accessor and mutator methods for the restricted attributes $pay and $SSN. As you can see from the example, the accessor methods (get) simply return the restricted value from the object, while the mutator (set) expects an argument that is used to update the attribute of the object.

To use the class definition to create an object, we need to use the “new” keyword along with the name of the class but treat it as a function. We can then begin initializing the object’s attributes either using the mutator functions or directly if the attribute is public. Notice that PHP uses the single-line arrow “->” in order to access members of an object.

EXAMPLE Interacting with a PHP object:

```
<?php
     $emp1 = new Employee();
     $emp1->name = "John";
     $emp1->setPay("15.55");
     $emp1->setSSN("123-45-6789");

     echo $emp1->name."<br>";
     echo $emp1->getPay()."<br>";
     echo $emp1->getSSN()."<br>";
?>
```

Finally, what if we want to initialize the attributes at the same time we create the object? We can do this by defining a **constructor function**. A constructor function is a special function that is called into action when a new object is created. This is commonly used to not only create the object but also to initialize the object with default values. The constructor function is created like a normal member function, but it uses the name “__construct()” (the “__” is a double underscore).

The customized constructor can now create an object instance and initialize its attributes with the values from the arguments passed to the constructor function.

EXAMPLE The class constructor function:

```
<?php
 class Employee
 {
    public $name;
    protected $pay;
    private $SSN;

    function __construct($name, $pay, $SSN)
   {
        $this->name = $name;
            $this->pay = $pay;
        $this->SSN = $SSN;
    }

    function getPay()
    {
    / …  /
 }

     $emp1 = new Employee("John Doe", 15.55, "123-45-6789" );
     echo $emp1->name."<br>";
     echo $emp1->getPay()."<br>";
     echo $emp1->getSSN()."<br>";

?>
```

TERMS TO KNOW

Class Attributes
Variables that are assigned to a class definition and will belong to each object instance created. Also called *member variables*.

Methods
Functions assigned to a class definition that will belong to all object instances created. Also called *member functions*.

Access Modifiers
Keyword operators that determine what can access a class's attribute.

Private
An access modifier that prevents access to class attributes from outside of the class itself.

Accessor Methods
Class methods designed to return the value of a private or protected attribute of an object.

Mutator Methods
Class methods designed to modify the value of a private or protected attribute of an object.

Public
An access modifier that allows full access to an object's class attribute from code outside of the class definition.

Protected
An access modifier that restricts access to a class attribute to the class itself and any subclasses derived from the class.

Constructor Function
A specialized function used to customize how a class object is created and prepared at the time of instantiation.

SUMMARY

In this lesson, you learned about **PHP variables**, how to **interpolate their data** into strings, and how to create **constant** immutable variables in PHP. Furthermore, you learned about **PHP arrays** as well as how to **define custom PHP functions**. Finally, you were introduced to PHP's object-oriented concepts, including how to define **PHP classes** and how to use them to instantiate object instances.

TERMS TO KNOW

Access Modifiers
Keyword operators that determine what can access a class's attribute.

Accessor Methods
Class methods designed to return the value of a private or protected attribute of an object.

Class Attributes
Variables that are assigned to a class definition and will belong to each object instance created. Also called *member variables*.

Class Definition
A set of code in object-oriented programming that defines the structure of an object.

Constant Variables
Variables defined in such a manner that their value cannot be changed for the life of the program.

Constructor Function
A specialized function used to customize how a class object is created and prepared at the time of instantiation.

Custom-Defined PHP Functions
Callable, named sets of code in PHP, written by the developer.

Default Arguments
When defining a function, default arguments create an optional parameter that, if omitted at the time of the call, will use a default, predetermined value.

Interpolation
The process of interlacing variables within strings of text using syntax that tells the software to replace the variable with its value.

Methods
Functions assigned to a class definition that will belong to all object instances created. Also called *member functions*.

Mutator Methods
Class methods designed to modify the value of a private or protected attribute of an object.

Private

An access modifier that prevents access to class attributes from outside of the class itself.

Protected

An access modifier that restricts access to a class attribute to the class itself and any subclasses derived from the class.

Public

An access modifier that allows full access to an object's class attribute from code outside of the class definition.

Recursive Function Calls

When a programming function makes a call to itself.

"heredoc" Statement

Special syntax that interprets the text within a code file without processing the code and instead treating it as plain text.