CHAPTER 2

Variables

Variables are used for storing data, such as numbers or strings, so that they can be used multiple times in a script.

Defining Variables

A variable starts with a dollar sign ($) followed by an *identifier*, which is the name of the variable. A common naming convention for variables is to have each word initially capitalized, except for the first one.

```
$myVar;
```

A value can be assigned to a variable by using the equals sign, or assignment operator (=). The variable then becomes *defined* or *initialized*.

```
$myVar = 10;
```

Once a variable has been defined, it can be used by referencing the variable's name. For example, the value of the variable can be printed to the web page by using `echo` followed by the variable's name.

```
echo $myVar; // "10"
```

Keep in mind that variable names are case sensitive. Names in PHP can include underscore characters and numbers, but they cannot start with a number. They also cannot contain spaces or special characters, and they must not be a reserved keyword.

Data Types

PHP is a loosely typed language. This means that the type of data that a variable can store is not specified. Instead, a variable's data type changes automatically to hold the value that it is assigned.

```
$myVar = 1;   // int type
$myVar = 1.5; // float type
```


M. Olsson, *PHP 7 Quick Scripting Reference*, DOI 10.1007/978-1-4842-1922-5_2

Furthermore, the value of a variable is evaluated differently, depending on the context in which it is used.

```
// Float type evaluated as string type
echo $myVar; // "1.5"
```

Because of these implicit type conversions, knowing the underlying type of a variable is not always necessary. Nevertheless, it is important to have an understanding of the data types that PHP works with in the background. These nine types are listed in Table 2-1.

Table 2-1. *PHP Data Types*

Data Type	Category	Description
int	Scalar	Integer
float	Scalar	Floating-point number
bool	Scalar	Boolean value
string	Scalar	Series of characters
array	Composite	Collection of values
object	Composite	User-defined data type
resource	Special	External resource
callable	Special	Function or method
null	Special	No value

Integer Type

An integer is a whole number. They can be specified in decimal (base 10), hexadecimal (base 16), octal (base 8) or binary (base 2) notation. Hexadecimal numbers are preceded with a `0x`, octal with a `0`, and binary numbers with a `0b`.

```
$myInt = 1234; // decimal number
$myInt = 0b10; // binary number (2 decimal)
$myInt = 0123; // octal number (83 decimal)
$myInt = 0x1A; // hexadecimal number (26 decimal)
```

Integers in PHP are always signed and can therefore store both positive and negative values. The size of an integer depends on the system word size, so on a 32-bit system, the largest storable value is 2^{32-1}. If PHP encounters a larger value, it is interpreted as a float instead.

Floating-Point Type

The float or floating-point type can store real numbers. These can be assigned using either decimal or exponential notation.

```
$myFloat = 1.234;
$myFloat = 3e2; // 3*10^2 = 300
```

The precision of a float is platform dependent. Commonly, the 64-bit IEEE format is used, which can hold approximately 14 decimal digits and a maximum decimal value of 1.8×10^{308}.

Bool Type

The bool type can store a Boolean value, which is a value that can only be either true or false. These values are specified with the `true` and `false` keywords.

```
$myBool = true;
```

Null Type

The case-insensitive constant `null` is used to represent a variable with no value. Such a variable is considered to be of the special null data type.

```
$myNull = null; // variable is set to null
```

Just as with other values, the null value evaluates differently, depending on the context in which the variable is used. If evaluated as a bool, it becomes false; as a number, it becomes zero (0); and as a string, it becomes an empty string ("").

```
$myInt = $myNull + 0;      // numeric context (0)
$myBool = $myNull == true; // bool context    (false)
echo $myNull;              // string context  ("")
```

Default Values

In PHP, it is possible to use variables that have not been assigned a value. Such undefined variables are then automatically created with the null value.

```
echo $myUndefined; // variable is set to null
```

Although this behavior is allowed, it is a good coding practice to define variables before they are used, even if the variables are just set to null. As a reminder for this, PHP issues an error notice when undefined variables are used. Depending on the PHP error reporting settings, this message may or may not be displayed.

```
Notice: Undefined variable: myUndefined in C:\xampp\htdocs\mypage.php on
line 10
```