Chapter 1

Getting Started with PHP

PHP Programming with MySQL
2nd Edition

Lecture 1

Objectives

In this chapter you will:

- Create PHP scripts
- Create PHP code blocks
- Work with variables and constants
- Study data types
- Use expressions and operators

Creating Basic PHP Scripts

Embedded language refers to code that is embedded within a Web page (XHTML document)

PHP code is typed directly into a Web page as a separate section

A Web page containing PHP code must be saved with an extension of .php to be processed by the scripting engine

PHP code is never sent to a client's Web browser; only the output of the processing is sent to the browser

Creating Basic PHP Scripts (continued)

The Web page generated from the PHP code, and XHTML elements found within the PHP file, is returned to the client

A PHP file that does not contain any PHP code should be saved with an **.html** extension

.php is the default extension that most Web servers use to process PHP scripts

Creating PHP Code Blocks

Code declaration blocks are separate sections on a Web page that are interpreted by the scripting engine

There are four types of code declaration blocks:

- Standard PHP script delimiters
- The `<script>` element
- Short PHP script delimiters
- ASP-style script delimiters

Standard PHP Script Delimiters

A **delimiter** is a character or sequence of characters used to mark the beginning and end of a code segment

The standard method of writing PHP code declaration blocks is to use the `<?php` and `?>` script delimiters

The individual lines of code that make up a PHP script are called **statements**

The <script> Element

The **<script> element** identifies a script section in a Web page document

Assign a value of "php" to the **language** attribute of the <script> element to identify the code block as PHP

Short PHP Script Delimiters

The syntax for the short PHP script delimiters is

```
<? statements; ?>
```

Short delimiters can be disabled in a Web server's php.ini configuration file

PHP scripts will not work if your Web site ISP does not support short PHP script delimiters

Short delimiters can be used in XHTML documents, but not in XML documents

ASP-Style Script Delimiters

The syntax for the ASP-style script delimiters is

```
<% statements; %>
```

ASP-style script delimiters can be used in XHTML documents, but not in XML documents

ASP-style script delimiters can be enabled or disabled in the php.ini configuration file

To enable or disable ASP-style script delimiters, assign a value of "On" or "Off " to the `asp_tags` directive in the php.ini configuration file

Understanding Functions

A **function** is a subroutine (or individual statements grouped into a logical unit) that performs a specific task

- To execute a function, you must invoke, or **call**, it from somewhere in the script

A **function call** is the function name followed by any data that the function needs

The data (in parentheses following the function name) are called **arguments** or **actual parameters**

Sending data to a called function is called **passing arguments**

Displaying Script Results

The `echo` and `print` statements are language constructs (built-in features of a programming language) that create new text on a Web page that is returned as a response to a client

The text passed to the `echo` statement is called a "literal string" and must be enclosed in either single or double quotation marks

To pass multiple arguments to the `echo` statement, separate the statements with commas

Displaying Script Results (continued)

Use the `echo` and `print` statements to return the results of a PHP script within a Web page that is returned to a client

The `print` statement returns a value of `1` if successful or a value of `0` if not successful, while the `echo` statement does not return a value

Creating Multiple Code Declaration Blocks

For multiple script sections in a document, include a separate code declaration block for each section

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<?php echo "<p>Output from the first script section.</p>";
?>
<h2>Second Script Section</h2>
  <?php echo "<p>Output from the second script
  section.</p>";?>
</body>
</html>
```


Creating Multiple Code Declaration Blocks (continued)

PHP code declaration blocks execute on a Web server before a Web page is sent to a client

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<p>Output from the first script section.</p>
<h2>Second Script Section</h2>
<p>Output from the second script section.</p>
</body>
</html>
```

Creating Multiple Code Declaration Blocks (continued)

Figure 1-9 Output of a document with two PHP script sections

Creating Multiple Code Declaration Blocks (continued)

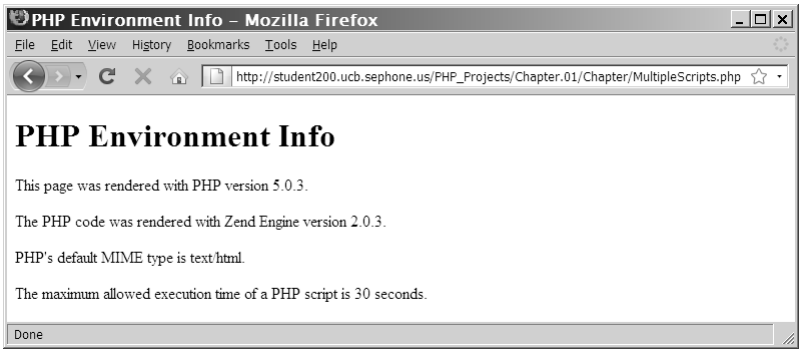

Figure 1-10 PHP Environment Information Web page

Case Sensitivity in PHP

Programming language constructs in PHP are mostly case **insensitive**

```
<?php
echo "<p>Explore <strong>Africa</strong>, <br />";
Echo "<strong>South America</strong>, <br />";
ECHO " and <strong>Australia</strong>!</p>";
?>
```


Adding Comments to a PHP Script

Comments are nonprinting lines placed in code that do not get executed, but provide helpful information, such as:

- The name of the script
- Your name and the date you created the program
- Notes to yourself
- Instructions to future programmers who might need to modify your work

Adding Comments to a PHP Script (continued)

Line comments hide a single line of code
- Add `//` or `#` before the text

Block comments hide multiple lines of code
- Add `/*` to the first line of code
- And `*/` after the last character in the code

Adding Comments to a PHP Script (continued)

```
<?php
/*
This line is part of the block comment.
This line is also part of the block comment.
*/
echo "<h1>Comments Example</h1>"; // Line comments can follow
code statements
// This line comment takes up an entire line.
# This is another way of creating a line comment.
/* This is another way of creating
a block comment. */
?>
```

Using Variables and Constants

The values stored in computer memory are called **variables**

The values, or data, contained in variables are classified into categories known as **data types**

The name you assign to a variable is called an **identifier**

An identifier must begin with a dollar sign ($), may not include a number or underscore as the first character, cannot include spaces, and is case sensitive

Displaying Variables

To display a variable with the `echo` statement, pass the variable name to the `echo`
statement without enclosing it in quotation marks:

```
$VotingAge = 18;
echo $VotingAge;
```

To display both text strings and variables, send
them to the `echo` statement as individual arguments, separated by commas:

```
echo "<p>The legal voting age is ",
$VotingAge, ".</p>";
```

Naming Variables

The name you assign to a variable is called an identifier

The following rules and conventions must be followed when naming a variable:

- Identifiers must begin with a dollar sign ($)
- Identifiers may contain uppercase and lowercase letters, numbers, or underscores (_). The first character after the dollar sign must be a letter.
- Identifiers cannot contain spaces
- Identifiers are case sensitive

Declaring and Initializing Variables

Specifying and creating a variable name is called **declaring the variable**

Assigning a first value to a variable is called **initializing the variable**

In PHP, you must declare and initialize a variable in the same statement:

```
$variable_name = value;
```

Displaying Variables

Figure 1-11 Output from an `echo` statement that is passed text and a variable

Displaying Variables (continued)

The output of variable names inside a text string depends on whether the string is surrounded by double or single quotation marks

Figure 1-12 Output of an `echo` statement that includes text and a variable surrounded by single quotation marks

Modifying Variables

You can modify a variable's value at any point in a script

```
$SalesTotal = 40;

echo "<p>Your sales total is
   $$SalesTotal</p>";

$SalesTotal = 50;

echo "<p>Your new sales total is
$$SalesTotal</p>";
```


Defining Constants

A **constant** contains information that does not change during the course of program execution

Constant names do not begin with a dollar sign ($)

Constant names use all uppercase letters

Use the **`define()`** function to create a constant

```
define("CONSTANT_NAME", value);
```

The value you pass to the `define()` function can be a text string, number, or Boolean value

Working with Data Types

A **data type** is the specific category of information that a variable contains

Data types that can be assigned only a single value are called **primitive types**

Data Type	Description
Integer numbers	The set of all positive and negative numbers and zero, with no decimal places
Floating-point numbers	Positive or negative numbers with decimal places or numbers written using exponential notation
Boolean	A logical value of "true" or "false"
String	Text such as "Hello World"
NULL	An empty value, also referred to as a NULL value

Table 1-1 Primitive PHP data types

Working with Data Types (continued)

The PHP language supports:

- A **resource** data type – a special variable that holds a reference to an external resource such as a database or XML file
- **Reference** or **composite** data types, which contain multiple values or complex types of information
- Two reference data types: **arrays** and **objects**

Working with Data Types (continued)

Strongly typed programming languages require you to declare the data types of variables

Static or strong typing refers to data types that do not change after they have been declared

Loosely typed programming languages do not require you to declare the data types of variables

Dynamic or loose typing refers to data types that can change after they have been declared

Numeric Data Types

PHP supports two numeric data types:

- An **integer** is a positive or negative number and 0 with no decimal places (-250, 2, 100, 10,000)
- A **floating-point number** is a number that contains decimal places or that is written in exponential notation (-6.16, 3.17, 2.7541)
  - **Exponential notation**, or **scientific notation**, is a shortened format for writing very large numbers or numbers with many decimal places (2.0e11)

Boolean Values

A **Boolean value** is a value of `TRUE` or `FALSE`

It decides which part of a program should execute and which part should compare data

In PHP programming, you can only use `TRUE` or `FALSE` Boolean values

In other programming languages, you can use integers such as 1 = `TRUE`, 0 = `FALSE`