PHP

Introduction

- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

19.1 Introduction

- PHP, or PHP: Hypertext Preprocessor, has become the most popular server-side scripting language for creating dynamic web pages.
- PHP is open source and platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports a large number of databases.

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, Mercury, FileZilla, Tomcat etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

19.2 A Simple PHP Program

- The power of the web resides not only in serving content to users, but also in responding to requests from users and generating web pages with dynamic content.
- PHP code is embedded directly into text-based documents, such as HTML, though these script segments are interpreted by a server *before* being delivered to the client.
- PHP script file names end with `.php`.
- In PHP, code is inserted between the scripting delimiters `<?php` and `?>`. PHP code can be placed anywhere in HTML5 markup, as long as the code is enclosed in these delimiters.

19.2 A Simple PHP Program (Cont.)

- Variables are preceded by a `$` and are created the first time they're encountered.
- PHP statements terminate with a semicolon (`;`).
- Single-line comments which begin with two forward slashes (`//`) or a pound sign (#). Text to the right of the delimiter is ignored by the interpreter. Multiline comments begin with delimiter `/*` and end with delimiter `*/`.
- When a variable is encountered inside a double-quoted (`""`) string, PHP interpolates the variable. In other words, PHP inserts the variable's value where the variable name appears in the string.
- All operations requiring PHP interpolation execute on the server before the HTML5 document is sent to the client.
- PHP variables are loosely typed—they can contain different types of data at different times.

PHP Data Types

Type	Description
`int`, `integer`	Whole numbers (i.e., numbers without a decimal point).
`float`, `double`, `real`	Real numbers (i.e., numbers containing a decimal point).
`string`	Text enclosed in either single (`''`) or double (`""`) quotes. [*Note:* Using double quotes allows PHP to recognize more escape sequences.]
`bool`, `boolean`	`true` or `false`.
`array`	Group of elements.
`object`	Group of associated data and methods.
`resource`	An external source—usually information from a database.
`NULL`	No value.

Fig. 19.2 | PHP types.

PHP 5 Syntax

- A PHP script is executed on the server, and the plain HTML result is sent back to the browser.
- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**:
- A PHP file normally contains HTML tags, and some PHP scripting code.
- PHP statements end with a semicolon (;).

PHP Case Sensitivity

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are **NOT case-sensitive**.
- All variable names are **case-sensitive**.

```php
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
```

```php
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

PHP Variables

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

PHP is a Loosely Typed Language

- We did not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

Global and Local Scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

```php
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Global and Local Scope

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```php
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

```php
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP String

- The PHP strlen() function returns the length of a string.
- The PHP str_word_count() function counts the number of words in a string:
- The PHP strrev() function reverses a string:

```php
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

```php
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

```php
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

PHP String(Search For a Specific Text Within a String**)**

- The PHP strpos() function searches for a specific text within a string.
- If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
- The example below searches for the text "world" in the string "Hello world!":

```php
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Replace Text Within a String

- The PHP str_replace() function replaces some characters with some other characters in a string.
- The example below replaces the text "world" with "Dolly":

```php
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP Integer

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- **Rules for integers:**
  - An integer must have at least one digit
  - An integer must not have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

PHP Integer

- In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5985;
var_dump($x);
?>

</body>
</html>
```

Result:

int(5985)

PHP Data Types

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- A Boolean represents two possible states: TRUE or FALSE.
- An array stores multiple values in one single variable.
- Null is a special data type which can have only one value: NULL.

PHP Object

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

```php
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

PHP Resource

- The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.
- A common example of using the resource data type is a database call.

PHP Constants

- A constant is an identifier (name) for a simple value.
- The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- To create a constant, use the define() function.
- Constants are automatically global and can be used across the entire script.

PHP Constants

- **Syntax**
  - define(*name*, *value*, *case-insensitive*)
- **Parameters:**
  - *name*: Specifies the name of the constant
  - *value*: Specifies the value of the constant
  - *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

PHP Constants

- The example below creates a constant with a **case-sensitive** name:

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

- The example below creates a constant with a **case-insensitive** name:

```php
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

PHP Operators

- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators

PHP Arithmetic Operators

- The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

PHP Assignment Operators

- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Conditional Statements

- Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.
- In PHP we have the following conditional statements:
  - **if statement** - executes some code if one condition is true
  - **if...else statement** - executes some code if a condition is true and another code if that condition is false
  - **if...elseif....else statement** - executes different codes for more than two conditions
  - **switch statement** - selects one of many blocks of code to be executed

The PHP switch Statement

- Use the switch statement to **select one of many blocks of code to be executed**.

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

```
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

- Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.
- In PHP, we have the following looping statements:
  - **while** - loops through a block of code as long as the specified condition is true
  - **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
  - **for** - loops through a block of code a specified number of times
  - **foreach** - loops through a block of code for each element in an array

The PHP while Loop

- The while loop executes a block of code as long as the specified condition is true.

```
while (condition is true) {
    code to be executed;
}
```

```php
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

The PHP do...while Loop

- The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

```
do {
    code to be executed;
} while (condition is true);
```

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

```
<?php
$x = 6;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

PHP Functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.
- Besides the built-in PHP functions, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

```
function functionName() {
    code to be executed;
}
```

PHP Functions

- A function name can start with a letter or underscore (not a number).
- Give the function a name that reflects what the function does!
- Function names are NOT case-sensitive.

```php
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

```
<!DOCTYPE html>
<html>
<body>

<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>

</body>
</html>
```

Result:

Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.


Thank you!
Any Questions ?