Chapter 4
Introduction to PHP

Lectured by:
Nguyễn Hữu Hiếu

PHP

- PHP is a server scripting language, and is a powerful tool for making dynamic and interactive Web pages quickly
- PHP (recursive acronym for PHP: Hypertext Preprocessor)
- PHP is a widely-used, and free.
- PHP runs over different operating systems such as Windows, Linux, Mac Os and Unix.
- PHP scripts are executed on the server, and the plain HTML result is sent back to the browser.

About the PHP Language

- Syntax is inspired by C
  - Curly braces, semicolons, no signficant whitespace
- Syntax inspired by perl
  - Dollar signs to start variable names, associative arrays
- Extends HTML to add segments of PHP within an HTML file.

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 form data.
- PHP can send and receive cookies.
- PHP can add, delete, modify data in your database.
- PHP can restrict users to access some pages on your website.
- PHP can encrypt data.

Introduction

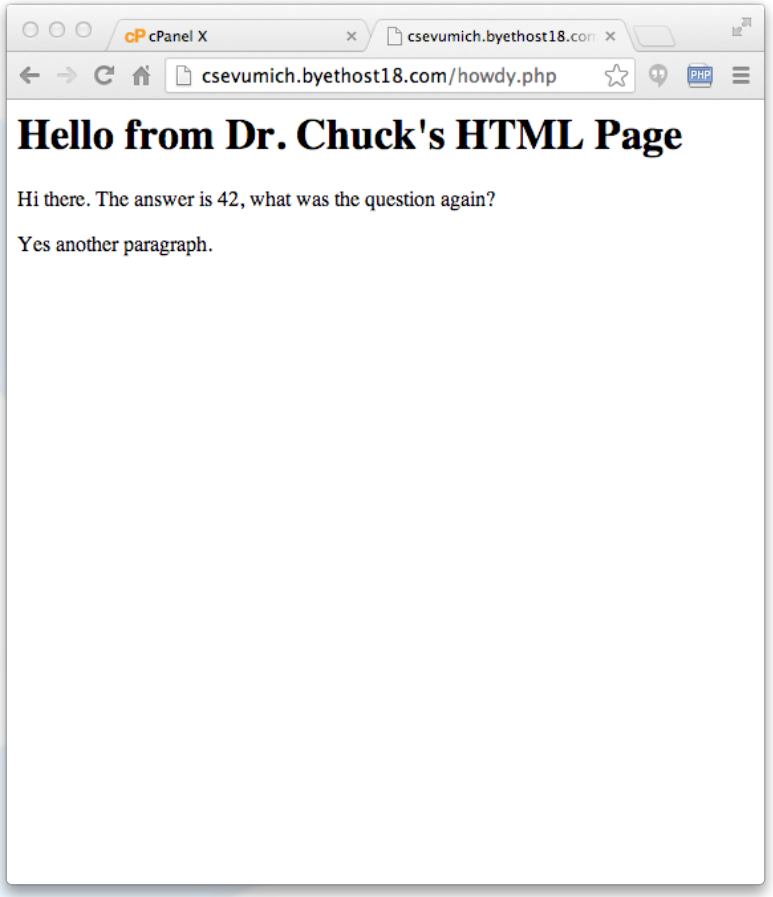
- Documents end with the extension *.php*
- To trigger the PHP commands, you need <?php tag.
  and they finish only when the closing part ?> is encountered

Example:

```
<?php
  echo "Hello world";
?>
```

Example

```
<h1>Hello from Dr. Chuck's HTML Page</h1>
<p>
<?php
    echo "Hi there.\n";
    $answer = 6 * 7;
    echo "The answer is $answer, what ";
    echo "was the question again?\n";
?>
</p>
<p>Yes another paragraph.</p>
```



Key Words

abstract and array() as break case catch class clone const continue declare default do else elseif end declare endfor endforeach endif endswitch endwhile extends final for foreach function global goto if implements interface instanceof namespace new or private protected public static switch $this throw try use var while xor

http://php.net/manual/en/reserved.php

Variable Names

- Start with a dollar sign ($) followed by a letter or underscore, followed by any number of letters, numbers, or underscores
- Case matters

```
$abc = 12;
$total = 0;
$largest_so_far = 0;
```

```
abc = 12;
$2php = 0;
$bad-punc = 0;
```

http://php.net/manual/en/language.variables.basics.php

Variable naming rules

- Variable names must start with a letter of the alphabet or the _ (underscore) character.
- Variable names can contain only the characters a-z, A-Z, 0-9, and _ (underscore).
- Variable names may not contain spaces. If a variable must comprise more than one word, the words should be separated with the _ (underscore) character (e.g., $user_name).
- Variable names are case-sensitive. The variable $High_Score is not the same as the variable $high_score.

Variable Name Weirdness

- Things that look like variables but are missing a dollar sign can be confusing

```
$x = 2;
$y = x + 5;
print $y;
```

5

Print $x // 2

```
$x = 2;
y = $x + 5;
print $x;
```

Parse error

Parse error: syntax error, unexpected '=' in **C:\xampp\htdocs\vtes.php** on line 2

Expressions

- Completely normal like other languages (+ - / *)
- More agressive implicit type conversion

```
<?php
    $x = "15" + 27;
    echo($x);
    echo("\n");
?>
```

Arithmetic operators

- They are used to perform mathematics.

Table 3-1. Arithmetic operators

Operator	Description	Example
+	Addition	$j + 1
-	Subtraction	$j - 6
*	Multiplication	$j * 11
/	Division	$j / 4
%	Modulus (division remainder)	$j % 9
++	Increment	++$j
--	Decrement	--$j

Assignment operators

- These operators are used to assign values to variables.
- Strings have their own operator, the period (.), detailed in the section "String concatenation"

Table 3-2. Assignment operators

Operator	Example	Equivalent to
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j - 3
*=	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Comparison operators

- Comparison operators are generally used inside a construct such as an if statement in which you need to compare two items.

Table 3-3. Comparison operators

Operator	Description	Example
==	Is **equal** to	$j == 4
!=	Is **not equal** to	$j != 21
>	Is **greater than**	$j > 3
<	Is **less than**	$j < 100
>=	Is **greater than or equal** to	$j >= 15
<=	Is **less than or equal** to	$j <= 8

Logical operators

- For example, you might say to yourself, "If the time is later than 12 PM and earlier than 2 PM, then have lunch." In PHP,
- if ($hour > 12 && $hour < 14) dolunch();

Table 3-4. Logical operators

Operator	Description	Example
&&	**And**	$j == 3 && $k == 2
and	Low-precedence **and**	$j == 3 **and** $k == 2
||	**Or**	$j < 5 || $j > 10
or	Low-precedence **or**	$j < 5 **or** $j > 10
!	**Not**	! ($j == $k)
xor	**Exclusive or**	$j **xor** $k

Variable Assignment

- The syntax to assign a value to a variable is always:

  variable = value

  Ex. $x += 10;
- **Variable incrementing and decrementing**
- Adding or subtracting 1 operation
- **Prefix form: ++$x; --$y;**

  ```
  Ex: if (++$x == 10) echo $x;
  ```
- This tells PHP to *first* increment the value of $x and then test whether it has the value 10 and, if so, output its value.
- **Postfix form: ++$x; --$y;**

  ```
  Ex: if ($y-- == 0) echo $y;
  ```
- Suppose $y starts out as 0 before the statement is executed. The comparison will return a TRUE result, but $y will be set to −1 after the comparison is made.

  So what will the echo statement display: 0 or −1?

Variable Typing

- In php, variables do not have to be declared before they are used.
- PHP always converts variables to the type required by their context when they are accessed.
- you can create a multiple-digit number and extract the *n*th digit from it, simply by assuming it to be a string.

```
<?php
$number = 12345 * 67890; //=838102050
echo substr($number, 3, 1);//(number,position, no.of char)
?>
```

- $number is a numeric variable.
- But the PHP function substr asks for one character to be returned from $number
- PHP turns $number into a nine-character string.
- so that substr can access it and return the character, which in this case is **1**.

Cont.

- The same goes for turning a string into a number
- The variable $pi is set to a string value automatically turned into a floating-point in the third line by the equation for calculating a circle's area
- Example: *Automatically converting a string to a number*

```
<?php
$pi = "3.1415927";
$radius = 5;
echo $pi * ($radius * $radius);
?>
```

Output

- echo is a language construct - can be treated like a function with one parameter. Without parenthesis, it accepts multiple parameters.
- print is a function - only one parameter but parenthesis are optional so it can look like a language construct

```php
<?php
    $x = "15" + 27;
    echo $x;
    echo("\n");
    echo $x, "\n";
    print $x;
    print "\n";
    print($x);
    print("\n");
?>
```

Echo Commands

- echo command used in a number of different ways to output text from the server to your browser.
- In some cases, a string literal has been output.

```
echo  "welcome in php";
```

- In others, strings have first been concatenated.

```
$x=7;
echo "the number is = ".$x;
```

- or variables have been evaluated.

```
$x=7;
echo "the number is = $x";
```

- It was Also shown the output spread over multiple lines.

The Difference Between the echo and print Commands

- an alternative to echo that you can use: print.
- The two commands are quite similar to each other, but print is an actual function that takes a single parameter, whereas echo is a PHP language construct.
- Echo is faster than print in general text output, because is not being a function—it doesn't set a return value.

Example:

```
$b=7; $a=1;
$b ? print "TRUE" : print "FALSE";// TRUE
$a>=$b ? print "TRUE" : print "FALSE";// FALSE
```

Conditional - if

- Logical operators (== != < > <= >= && || !)
- Curly braces

```
<?php
    $ans = 42;
    if ( $ans == 42 )
    {
       print "Hello world!\n";
    }
    else {
            print "Wrong answer\n";
    }
?>
```

→ `Hello World!`

Whitespace does not matter

```
<?php
    $ans = 42;
    if ( $ans == 42 ) {
       print "Hello world!\n";
    } else {
       print "Wrong answer\n";
    }
?>
```

```
<?php $ans = 42; if ( $ans == 42 ) { print
"Hello world!\n"; } else { print "Wrong answer\n"; }
?>
```

What Style do You Prefer?

```php
<?php
    $ans = 42;
    if ( $ans == 42 ) {
       print "Hello world!\n";
    } else {
       print "Wrong answer\n";
    }
?>
```

```php
<?php
    $ans = 42;
    if ( $ans == 42 )
    {
       print "Hello world!\n";
    }
    else
    {
       print "Wrong answer\n";
    }
?>
```

Associative Arrays

- Like Python Dictonaries+Lists - but more powerful
- Can be key => value or simply indexed by numbers
- Ignore two-dimensional arrays for now...

Integer Indices

```php
<?php
    $stuff = array("Hi", "There");
    echo $stuff[1], "\n";
?>
```

```
There
```

Integer Indices

```
<?php
    $stuff = array();
    $stuff[] = "Hello";
    $stuff[] = "World";

    echo $stuff[1], "\n";
?>
```

```
World
```

Integer Indices

```
<?php
    $stuff = array();
    $stuff[2] = "Hello";
    $stuff[9] = "World";

    echo $stuff[9], "\n";
?>
```

```
World
```

Key / Value

```php
<?php
    $stuff = array("name" => "Chuck",
                   "course" => "SI664");
    echo $stuff["course"], "\n";
?>
```

```
SI664
```

Dumping an Array

- The function print_r() dumps out PHP data - it is used mostly for debugging

```
<?php
   $stuff = array("name" => "Chuck",
                  "course" => "SI664");
    print_r($stuff);
?>
```

```
Array
(
    [name] => Chuck
    [course] => SI664
)
```

Dumping an Array

- The function print_r() dumps out PHP data - it is used mostly for debugging

```
<?php
    $stuff = array();
    $stuff[2] = "Hello";
    $stuff[9] = "World";
    print_r($stuff);
?>
```

```
Array
(
    [2] => Hello
    [9] => World
)
```

var_dump .vs. print_r

```php
<?php
   $stuff = array("name" => "Chuck",
                  "course" => "SI664");
    var_dump($stuff);
?>
```

```
array(2) {
  ["name"]=>
  string(5) "Chuck"
  ["course"]=>
  string(5) "SI664"
}
```

http://stackoverflow.com/questions/3406171/php-var-dump-vs-print-r

var_dump() is more verbose

```php
<?php
    $thing = FALSE;
     echo("One\n");
     print_r($thing);
     echo("Two\n");
     var_dump($thing);
?>
```

```
One
Two
bool(false)
```

http://stackoverflow.com/questions/3406171/php-var-dump-vs-print-r

PHP Looping

- **while**
  - loops repeat until final condition is reached

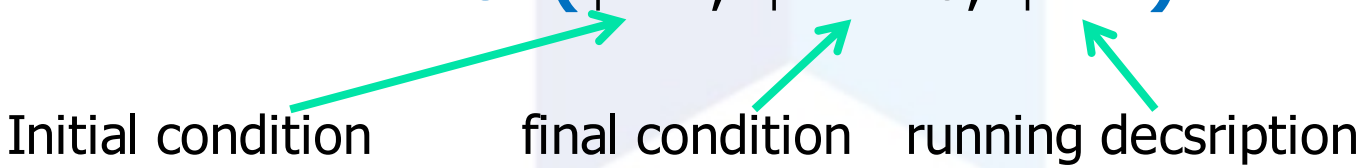
```
$i =1;
while ($i<=10)
      {
        echo $i;
        $i++;
      }
```

- **do…while**
  - kind of reversed while function
  - **Do {** code to be executed;**}**
  - **While(**final condition**);**

PHP Looping

- **for**
  - Repeats the specific part of code so many times we choose

```
for ($i=1; $i<=10; $i++)
```



Looping Through an Array

```
<?php
    $stuff = array("name" => "Chuck",
                   "course" => "SI664");
    foreach($stuff as $k => $v ) {
       echo "Key=",$k," Val=",$v,"\n";
    }
?>
```

```
Key=name Val=Chuck
Key=course Val=SI664
```

Variable Name Weirdness

- Things that look like variables but are missing a dollar sign as an array index are unpredictable....

```
$x = 5;
$y = array("x" => "Hello");
print $y[“x”];
```

Hello

```
$x = 5;
$y = array(
        "x" => "Hello",
        5 => "newwwwww”
);
print $y[“5”];
```

newwwwww

Strings

- String literals can use single quotes or double quotes
- The backslash (\) is used as an "escape" character
- Strings can span multiple lines - the newline is part of the string
- In double-quoted strings variable values are expanded

http://php.net/manual/en/language.types.string.php

```php
<?php
echo 'this is a simple string';

echo 'You can also have embedded newlines in
strings this way as it is
okay to do';

// Outputs: Arnold once said: "I'll be back"
echo 'Arnold once said: "I\'ll be back"';

// Outputs: This will not expand: \n a newline
echo 'This will not expand: \n a newline';

// Outputs: Variables do not $expand $either
echo 'Variables do not $expand $either';
?>
```

```php
<?php
echo "this is a simple string\n";

echo "You can also have embedded newlines in
strings this way as it is
okay to do";

// Outputs: This will expand:
//          a newline
echo "This will expand: \na newline";

// Outputs: Variables do 12
$expand = 12;
echo "Variables do $expand\n";
?>
```

comments

```
<?php
echo 'This is a test'; // This is a c++ style comment
    /* This is a multi line comment
       yet another line of comment */
    echo 'This is yet another test';
    echo 'One Final Test'; # This is a shell-
    style comment
?>
```

http://php.net/manual/en/language.basic-syntax.comments.php

String variables

- The quotation marks indicate that “Fred Smith” is a *string* of characters.

```
$username = "Fred Smith";
```

- to see what’s in the variable :

```
echo $username
```

- Or you can assign it to another variable

```
$current_user = $username;
```

String concatenation

- String concatenation uses the period (.) operator to append one string of characters to another.

```
Ex: echo "You have " . $msgs . "
messages.";
```

- Assuming that the variable $msgs is set to the value 5, the output will be: `You have 5 messages.`
- you can append one string to another using .= like this:

```
Ex: $bulletin .= $newsflash;
```

String types

- PHP supports two types of strings that are denoted by the type of quotation mark that you use.
- If you wish to assign a literal string, preserving the **exact contents**, you should use the single quotation mark (**apostrophe**), like this:

```
Ex: $info = 'Preface variables with a $
like this: $variable';
```

Cont. String types

- In this case, every character within the single-quoted string is assigned to $info.
- If you had used double quotes, PHP would have attempted to evaluate $variable as a variable.
- when you *want* to include the value of a variable inside a string, you do so by using a double-quoted string:

```
Ex: echo "There have been $count
presidents of the US";
```

- this syntax also offers a simpler form of concatenation This is called *variable substitution*.

Escaping characters

- Sometimes a string needs to contain characters with special meanings that might be interpreted incorrectly.
- For example, the following line of code will not work:

```
$text = 'My sister's car is a Ford'; // Erroneous syntax
```

- Because the apostrophe encountered in the word *sister's* will tell the PHP parser that the end of the string has been reached.
- To correct this, you can add a backslash

```
$text = 'My sister\'s car is a Ford';
```

Cont. Escaping characters

- Examples:

```
$text = "My Mother always said \"Eat your greens\".";
```

- you can use escape characters to insert various special characters into strings, such as tabs, newlines, and carriage returns.
- These are represented by \t, \n, and \r.

```
$heading = "Date\tName\tPayment";
```

- These special backslash-preceded characters work only in double-quoted strings.
- In single-quoted strings, the preceding string would be displayed with the ugly \t sequences instead of tabs.
- Within single-quoted strings, only the escaped apostrophe(\') and the escaped backslash itself (\\) are recognized as escaped characters.

Multiple-Line Commands

- There are times when you need to output quite a lot of text from PHP
- using several echo (or print) statements would be time-consuming
- To overcome this, PHP offers two conveniences:
- The first is just to put multiple lines between quotes

```
<?php
$author = "Alfred E Newman";
echo "This is a Headline
This is the first line.
This is the second.
Written by $author.";
?>
```

Cont.

- Variables can also be assigned, as:

```
<?php
$author = "Alfred E Newman";
$text = "This is a Headline
This is the first line.
This is the second.
Written by $author.";
?>
```

- Second alternative multiline echo statement, PHP offers a multiline sequence using the **<<<** operator.
- commonly referred to as *here-document* or *heredoc* for short.

Cont.

- This is a way of specifying a string literal, preserving the line breaks and other whitespace (including indentation) in the text.

```
<?php
$author = "Alfred E Newman";
echo <<<_END
This is a Headline
This is the first line.
This is the second.
- Written by $author.
_END;
?>
```

- this code tell PHP to output everything between the two **_END** tags as if it were a double-quoted string.
- This means it's possible,for a developer to write entire sections of HTML directly into PHP code and then just replace specific dynamic parts with PHP variables.

Cont. _END

- enclosing _END tag *must* appear right at the start of a new line
- and must be the *only* thing on that line, no comment and no spaces are allowed.
- Once you have closed a multiline block, you are free to use the same tag name again.
- Remember: using the <<<_END..._END; heredoc construct, you don't have to add \n line-feed characters to send a line feed—just press Return and start a new line.
- Also, unlike either a double-quote- or single-quote delimited string, you are free to use all the single and double quotes you like within a heredoc, without escaping them by preceding them with a backslash (\).

Cont.

- Example: shows how to use the same syntax to assign multiple lines to a variable.

```
<?php
$author = "Alfred E Newman";
$out = <<<_END
This is a Headline
This is the first line.
This is the second.
- Written by $author.
_END;
?>
```

- The variable $out will then be populated with the contents between the two tags.
- If you were appending rather than assigning, you also could have used .= in place of = to append the string to $out.

PHP Functions

- All function starts with **function($*parameter*)**
- Requirements for naming functions are same as these for variables
- The { mark opens the function code, while } mark closes it
- It can have either defined or no parameter

- More than 700 built-in functions available

PHP Forms and User Input

- Used to gain information from users by means of HTML
- Information is worked up by PHP

```
<html>
<body>
<form action="welcome.php" method="post">
    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />
    <input type="submit" />
</form>
</body>
</html>
```

The $_GET Variable

- Used to collect values from a form
- Displays variable names and values are in the URL
  - http://www.w3schools.com/welcome.php?name=jo&age=39
- Can send limited amount of information (max. 100 characters)

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?> <br />
You are <?php echo $_GET["age"]; ?> years old
</body>
</html>
```

The $_POST variable

- Used to collect values from a form
- Information from a form is invisible
  - http://www.w3schools.com/welcome.php
- No limits on the amount of information to be send

```
<html>
  <body>
    Welcome <?php echo $_POST["name"]; ?><br />
    You are <?php echo $_POST["age"]; ?> years old.
  </body>
</html>
```

Summary

- This is a sprint through the language features of PHP

Tài Liệu Tham Khảo

- [1] Stepp,Miller,Kirst. Web Programming Step by Step.(1st Edition, 2009) Companion Website: http://www.webstepbook.com/
- [2] W3Schools, http://www.w3schools.com/html/default.asp