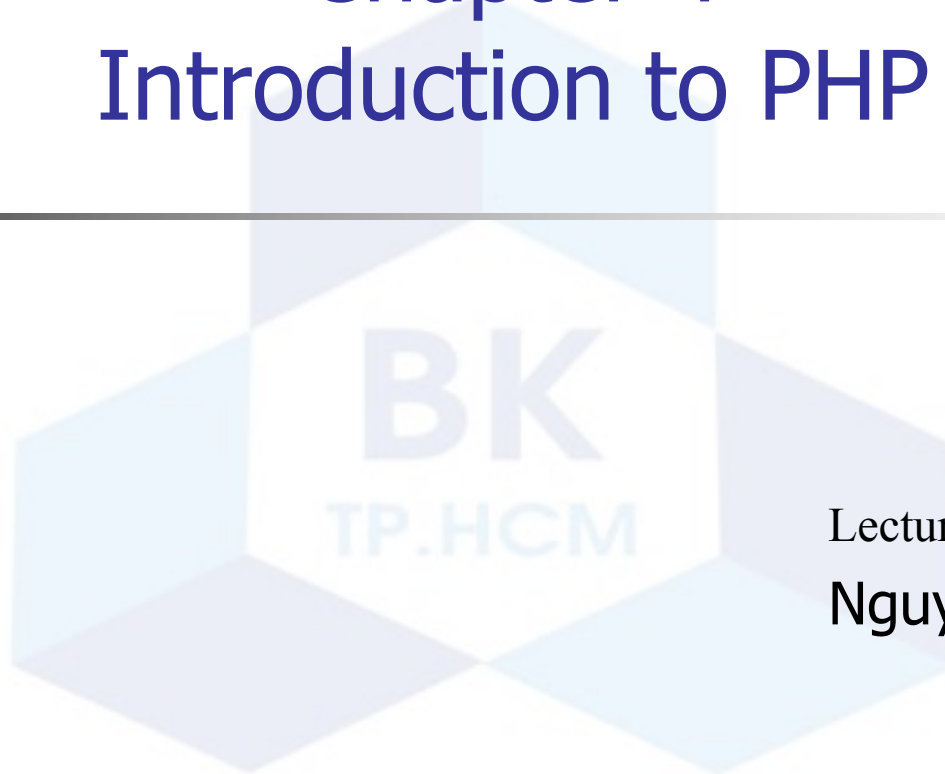


Chapter 4

Introduction to PHP



Lectured by:
Nguyễn Hữu Hiếu

- 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.

Code PHP được chạy trên server (máy chủ), sau đó server xử lý và gửi kết quả là HTML đơn giản về trình duyệt. Người dùng cuối chỉ thấy HTML, không thấy code PHP.

About the PHP Language

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

Cú pháp giống C:

PHP lấy cảm hứng từ ngôn ngữ C, nên nó dùng dấu ngoặc nhọn {} để bao code và dấu chấm phẩy ; để kết thúc câu lệnh. Khoảng trắng (whitespace) không quan trọng, viết sao cũng được miễn đúng cú pháp.

Cú pháp giống Perl:

PHP cũng mượn ý tưởng từ Perl, như dùng dấu \$ trước tên biến (ví dụ: \$tenBien) và hỗ trợ mảng kết hợp (associative arrays) kiểu key-value (ví dụ: \$arr["key"] = "value").

Mở rộng HTML:

PHP cho phép nhúng code trực tiếp vào file HTML. Nghĩa là mày có thể viết HTML bình thường, rồi chèn đoạn code PHP vào giữa để xử lý động, ví dụ: <p>?php echo "Xin chào"; ?>/p>.

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.

Tạo nội dung trang động:

PHP có thể sinh ra nội dung web thay đổi theo tình huống, không cố định như HTML tĩnh.

Làm việc với file trên server:

PHP cho phép tạo, mở, đọc, ghi, xóa, và đóng file trên máy chủ - kiểu như quản lý file từ xa.

Thu thập dữ liệu form:

PHP xử lý thông tin người dùng gửi qua form (ví dụ: tên, email từ form đăng ký).

Gửi và nhận cookies:

PHP có thể lưu trữ và lấy dữ liệu từ cookies để theo dõi người dùng (như nhớ đăng nhập).

Quản lý cơ sở dữ liệu:

PHP thêm, xóa, sửa dữ liệu trong database (như MySQL) dễ dàng.

Hạn chế quyền truy cập:

PHP kiểm soát ai được vào trang nào trên web, ví dụ: chỉ admin mới xem được trang quản lý.

Mã hóa dữ liệu:

PHP có thể mã hóa thông tin để bảo mật, như mã hóa mật khẩu.

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";  
?>
```

Tệp PHP có đuôi .php:

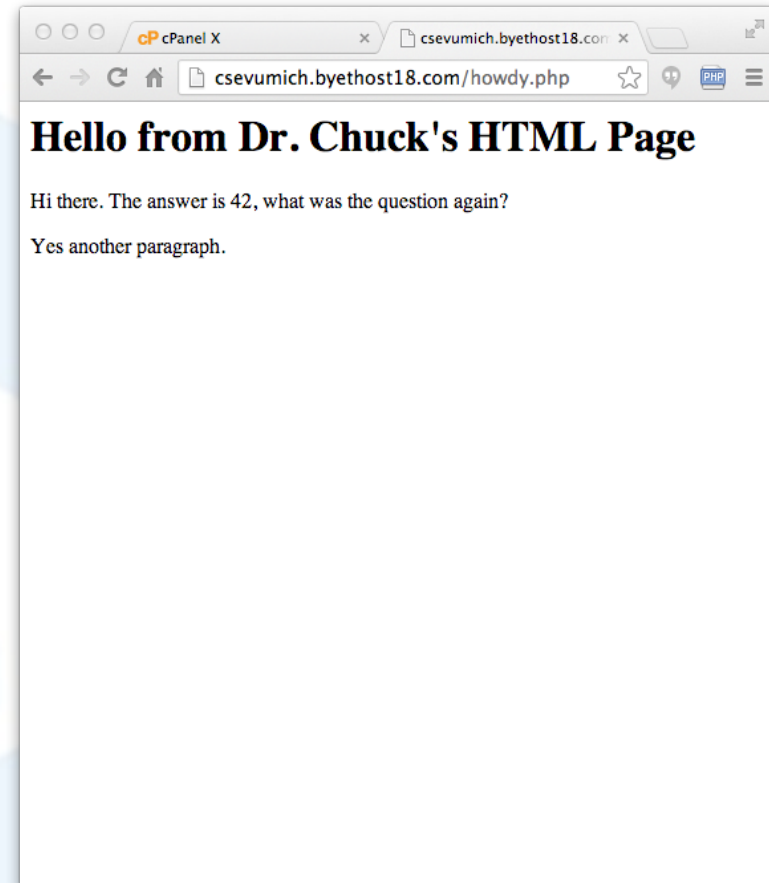
Các tài liệu PHP phải được lưu với phần mở rộng .php (ví dụ: index.php) để server biết đó là file chứa code PHP.

Thẻ PHP để chạy lệnh:

Để bắt đầu code PHP, mà cần thẻ mở `<?php`. Code bên trong thẻ này sẽ được server xử lý. Muốn kết thúc, dùng thẻ đóng `?>`. Nếu không có thẻ đóng, server sẽ hiểu toàn bộ file là PHP.

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>
```



Keywords

1. Keywords (Từ khóa)

Đây là danh sách các từ khóa đặc biệt trong PHP (như `abstract`, `if`, `for`, `class`, `function`, v.v.) mà máy không được dùng làm tên biến, tên hàm, hay bất cứ định danh nào khác. Chúng đã được PHP "đặt trước" để thực hiện các chức năng cụ thể.

Ví dụ: `if` dùng để kiểm tra điều kiện, `function` để định nghĩa hàm.

`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

Variable Names (Tên biến)

Quy tắc đặt tên biến:

Bắt đầu bằng dấu \$, sau đó là chữ cái (A-Z, a-z) hoặc dấu gạch dưới _.

Tiếp theo có thể là bất kỳ chữ cái, số (0-9), hoặc dấu gạch dưới nào.

Phân biệt chữ hoa/thường (case-sensitive), nghĩa là \$abc khác \$ABC.

- 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;
```

Ví dụ hợp lệ:

\$abc = 12; (đúng, bắt đầu bằng \$ và chữ cái)

\$total = 0; (đúng)

\$largest_so_far = 0; (đúng, dùng dấu _)

Ví dụ sai:

abc = 12; (sai, thiếu \$)

\$2php = 0; (sai, không bắt đầu bằng số)

\$bad-punc = 0; (sai, không dùng dấu -, chỉ được dùng _)

<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 aggressive implicit type conversion

<?php

```
$x = "15" + 27;
```

```
echo($x);
```

```
echo("\n");
```

?>

→ 42

Chuyển đổi kiểu dữ liệu tự động (implicit type conversion):

PHP khá "thoáng" trong việc tự động chuyển đổi kiểu dữ liệu khi thực hiện phép tính. Nếu một giá trị nhìn giống số, nó sẽ cố gắng hiểu và tính toán như số, dù nó ban đầu là chuỗi (string).

Arithmetic operators

- They are used to perform mathematics.

Table 3-1. Arithmetic operators

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

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
=	<code>\$j = 15</code>	<code>\$j = 15</code>
<code>+=</code>	<code>\$j += 5</code>	<code>\$j = \$j + 5</code>
<code>--</code>	<code>\$j -= 3</code>	<code>\$j = \$j - 3</code>
<code>*=</code>	<code>\$j *= 8</code>	<code>\$j = \$j * 8</code>
<code>/=</code>	<code>\$j /= 16</code>	<code>\$j = \$j / 16</code>
<code>.=</code>	<code>\$j .= \$k</code>	<code>\$j = \$j . \$k</code>
<code>%=</code>	<code>\$j %= 4</code>	<code>\$j = \$j % 4</code>

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	<code>\$j == 4</code>
!=	Is not equal to	<code>\$j != 21</code>
>	Is greater than	<code>\$j > 3</code>
<	Is less than	<code>\$j < 100</code>
>=	Is greater than or equal to	<code>\$j >= 15</code>
<=	Is less than or equal to	<code>\$j <= 8</code>

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
<code>&&</code>	And	<code>\$j == 3 && \$k == 2</code>
<code>and</code>	Low-precedence and	<code>\$j == 3 and \$k == 2</code>
<code> </code>	Or	<code>\$j < 5 \$j > 10</code>
<code>or</code>	Low-precedence or	<code>\$j < 5 or \$j > 10</code>
<code>!</code>	Not	<code>! (\$j == \$k)</code>
<code>xor</code>	Exclusive or	<code>\$j xor \$k</code>

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

Không cần định nghĩa kiểu biến trước khi sử dụng
Nhưng vẫn phải khai báo biến trước

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

echo:

Là một language construct (cấu trúc ngôn ngữ), không phải hàm, nhưng mà có thể dùng nó như hàm với một tham số nếu muốn.

Đặc biệt:

Không cần dấu ngoặc () nếu chỉ in một thứ.

Nếu không dùng ngoặc, mà có thể truyền nhiều tham số cách nhau bằng dấu phẩy ,.

Ví dụ:

`echo $x;` → In giá trị của `$x`.

`echo $x, "\n";` → In `$x` rồi xuống dòng (nhiều tham số).

print:

Là một hàm thật sự, nhưng dấu ngoặc () là tùy chọn, nên trông nó giống `echo`.

Khác biệt lớn:

Chỉ nhận một tham số duy nhất.

Ví dụ:

`print $x;` → In giá trị của `$x`.

`print "\n";` → In xuống dòng.

So sánh nhanh:

`echo`: Nhanh hơn chút xíu, linh hoạt hơn (nhiều tham số), không trả về giá trị.

`print`: Chỉ một tham số, trả về giá trị 1 (dùng được trong biểu thức), nhưng ít ai dùng tính năng này.

Tóm lại: Dùng `echo` khi muốn in nhanh, nhiều thứ; dùng `print` khi chỉ in một thứ và muốn gọn nhẹ. Hiểu chưa mà? Có gì thắc mắc không?

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;
```

phép nối chuỗi -> đều in ra "...is 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.

```
echo "the number is = " $s
```

=> sai

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 it 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"; }
?>
```

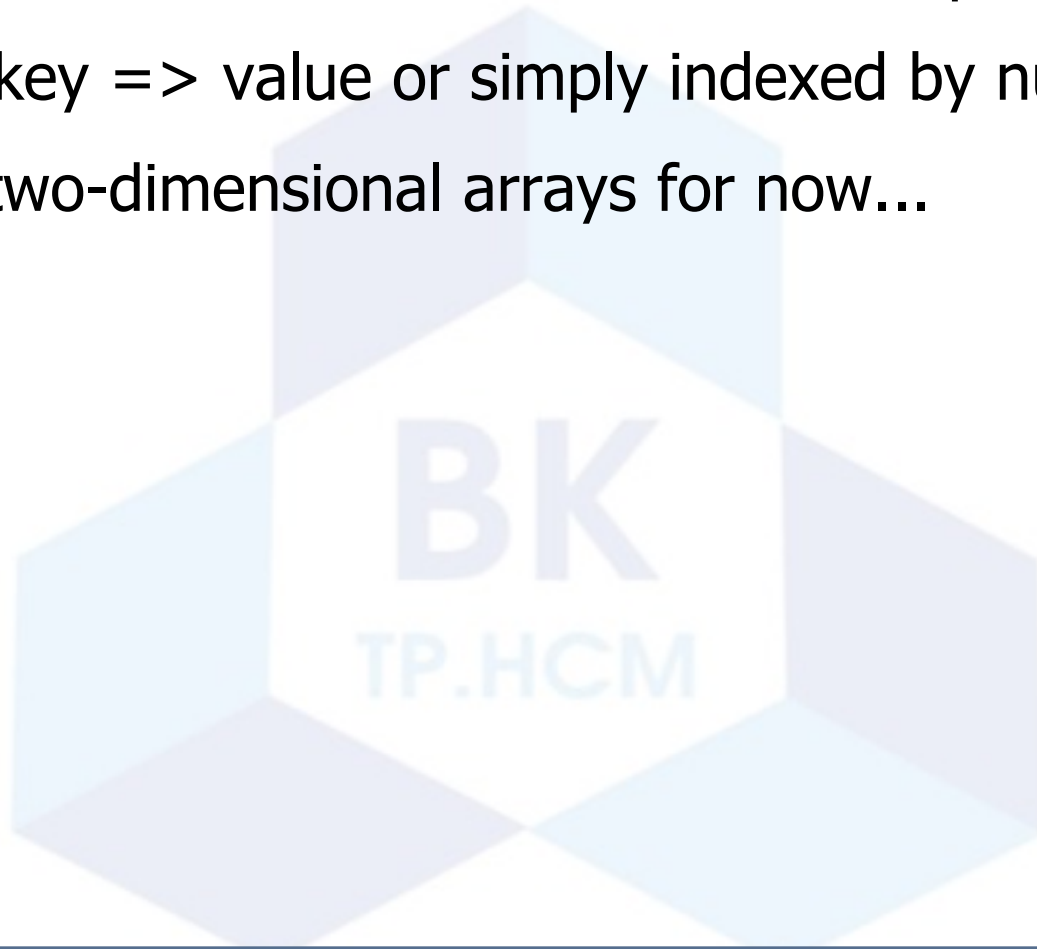
What Style do You Prefer?

```
<?php
    $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";
    }
?>
```

Associative Arrays

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



Associative Arrays (Mảng kết hợp)

Giống Python Dictionary + List, nhưng mạnh hơn:

Trong PHP, mảng (array) vừa giống danh sách (list) có thứ tự, vừa giống từ điển (dictionary) có cặp key-value. Nó linh hoạt hơn Python vì không phân biệt rõ hai kiểu này, dùng chung một cú pháp.

Hai kiểu mảng:

Key => Value (Mảng kết hợp):

Dùng khóa (key) để truy cập giá trị (value), key có thể là chuỗi hoặc số.

Ví dụ:

```
$arr = ["name" => "John", "age" => 25];  
echo $arr["name"]; // In: John
```

Chỉ số số (Indexed array):

Dùng số (0, 1, 2, ...) làm chỉ số, giống mảng thông thường.

Ví dụ:

```
$arr = [10, 20, 30];  
echo $arr[1]; // In: 20
```

Integer Indices

```
<?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
    $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);
?>
```

`print_r()` là một hàm trong PHP dùng để "đổ" (dump) dữ liệu ra màn hình, thường để kiểm tra (debug). Nó hiển thị cấu trúc và nội dung của biến (đặc biệt là mảng) một cách dễ đọc.

Mô tả: In dữ liệu một cách đơn giản, dễ đọc cho con người.

Đặc điểm:

Chỉ hiển thị cấu trúc và giá trị, không kèm thông tin kiểu dữ liệu hay độ dài.

Dùng để xem nhanh nội dung mảng hoặc biến.

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

Mô tả: In dữ liệu chi tiết hơn, bao gồm kiểu dữ liệu và kích thước (verbose hơn).

Đặc điểm:

Hiển thị kiểu (string, int, bool, array, v.v.), độ dài chuỗi, và giá trị.

Hữu ích khi cần kiểm tra kỹ càng.

```
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
```

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

```
?>
```

print_r(\$thing) không in gì vì FALSE không hiển thị rõ.

print_r() được thiết kế để in dữ liệu có cấu trúc (như mảng, đối tượng) hoặc giá trị đơn giản (chuỗi, số).

Khi gặp giá trị FALSE (kiểu boolean), nó không tạo ra đầu ra trực quan vì FALSE trong PHP không có biểu diễn dạng chuỗi rõ ràng khi in trực tiếp bằng print_r().

Kết quả là nó in ra chuỗi rỗng (không thấy gì trên màn hình).

Nếu \$thing = TRUE, print_r() sẽ in 1 (vì TRUE được hiểu là 1).

Nếu \$thing = "hello", nó in hello.

Nhưng với \$thing = FALSE, nó không in gì (hiểu như chuỗi rỗng).

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++)

Initial condition

final condition

running decription

BK
TP.HCM

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"];
```

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

Trong PHP, khi key trong mảng là số (như 5), mà có thể truy cập bằng chuỗi "5" hoặc số 5 đều được, vì PHP tự chuyển đổi kiểu (numeric string thành số).

Kết quả: In ra newwwww.

Ghi chú: \$y["5"] không liên quan gì đến biến \$x = 5, mà chỉ là truy cập key 5 trong mảng.

Hello

newwwww

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>

Dùng nháy đơn hoặc nháy kép:

Chuỗi có thể được bao bởi ' (nháy đơn) hoặc " (nháy kép).

Ví dụ: 'hello' hoặc "hello" đều là chuỗi hợp lệ.

Dấu gạch chéo ngược \ (escape character):

Dùng \ để "thoát" ký tự đặc biệt, cho phép in chúng theo nghĩa đen.

Ví dụ:

"He said \"Hi\"" → In: He said "Hi" (dùng \ " để in dấu nháy kép).

'It\'s me' → In: It's me (dùng \ ' để in nháy đơn).

Chuỗi trải dài nhiều dòng:

Chuỗi có thể viết trên nhiều dòng, và ký tự xuống dòng (\n) sẽ tự động là một phần của chuỗi.

Nháy kép mở rộng biến (variable expansion):

Trong chuỗi dùng ", PHP sẽ thay giá trị của biến trực tiếp vào chuỗi.

```
$x = 5;
```

```
echo "Number is $x"; // In: Number is 5
```

Nhưng với nháy đơn ', biến không được mở rộng:

```
echo 'Number is $x'; // In: Number is $x (nguyên văn)
```

```
<?php
```

```
echo 'this is a simple string';
```

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

Trong PHP, chuỗi được bao quanh bởi dấu nháy đơn
(") chỉ xử lý một vài ký tự escape đặc biệt, cụ thể là:

' để biểu diễn dấu nháy đơn

\\ để biểu diễn ký tự gạch chéo ngược

```
// 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
```

```
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.

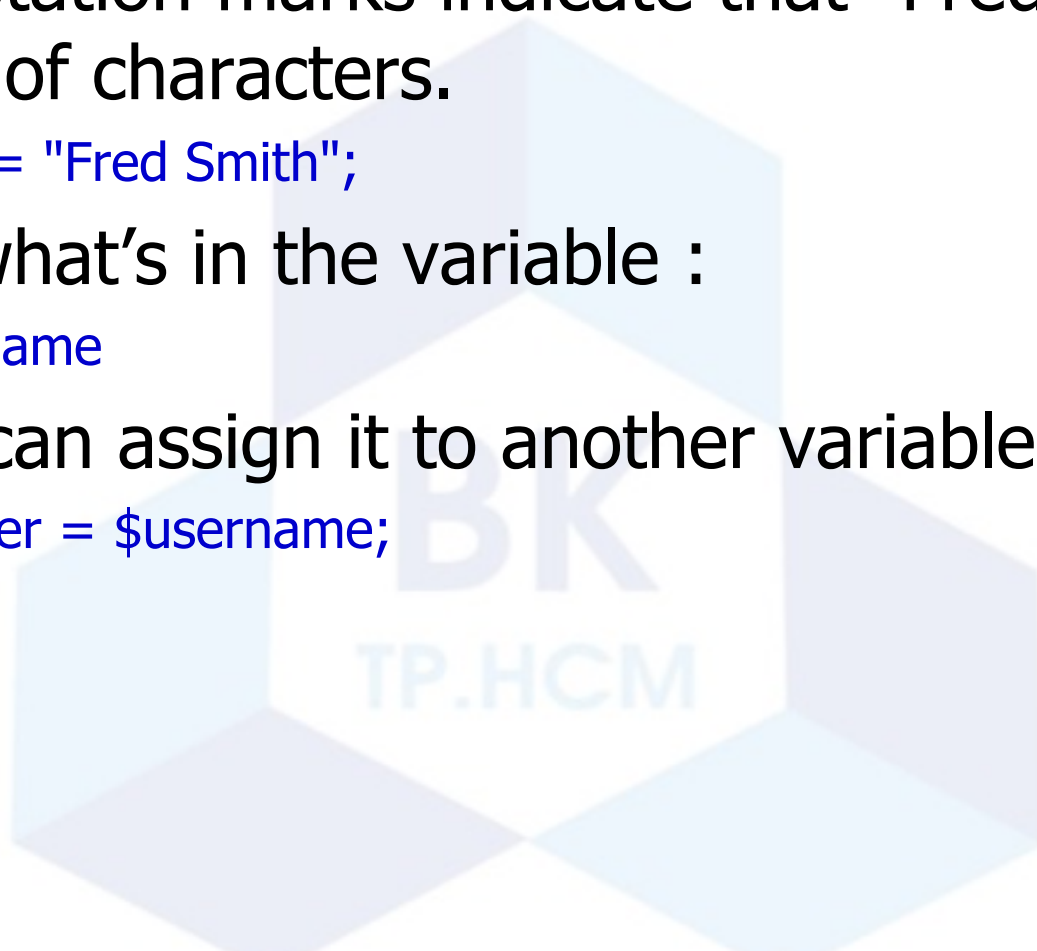
```
$username = "Fred Smith";
```

- 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';`

Nháy đơn = nguyên văn
Nháy kép = gán được biến

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:
- 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'; // Erroneous syntax
```

```
$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.

Nháy đơn chỉ xử lý \' và \\, còn lại nguyên văn hết

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.";
```

```
?>
```

Thay vì sử dụng nhiều echo/print, có thể để trong dấu ngoặc kép hết,
hoặc dùng tag <<< _END

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`

`_END` tag chỉ được đứng một mình một hàng

- 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 (`\`).

Không cần thêm ký tự `\n` để xuống dòng

Khi dùng Heredoc, chỉ cần xuống dòng thật sự là PHP tự hiểu đã có `\n` mà không cần thêm `\n` vào chuỗi.

Có thể dùng dấu nháy đơn `'` và nháy kép `"` thoải mái bên trong Heredoc

Khác với chuỗi bình thường trong PHP (`"` hoặc `''`), Heredoc không yêu cầu thoát (`\`) dấu nháy bên trong nó.

Cont.

```
$text = <<<END
```

```
Hello World!
```

```
END; // ❌ Sai vì có dấu cách trước END;
```

- 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>` Khi nhấn nút "Submit", trình duyệt sẽ gửi dữ liệu của form đến file
welcome.php (được chỉ định trong action="welcome.php").
`<body>` Phương thức gửi là POST (method="post") - nghĩa là dữ liệu sẽ được
gửi đi một cách ẩn (không xuất hiện trên URL).
`<form action="welcome.php" method="post">`

Name: `<input type="text" name="name" />`

Age: `<input type="text" name="age" />`

`<input type="submit" />`

`</form>` Ví dụ PHP xử lý form:
`</body>` `<?php`
`</html>` `// Nhận dữ liệu từ form bằng phương thức POST`
`$name = $_POST['name'];`
`$age = $_POST['age'];`

`echo "Chào, $name! Bạn $age tuổi.";`
`?>`

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

- `$_REQUEST`

`$_REQUEST` là một mảng tự động chứa dữ liệu được gửi từ cả GET, POST, và COOKIE.

Nó có thể được sử dụng để lấy dữ liệu từ form HTML mà không quan tâm form đó dùng `method="get"` hay `method="post"`.

Ví dụ:

```
<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>
```

```
<?php
```

```
$name = $_REQUEST['name'];
```

```
$age = $_REQUEST['age'];
```

```
echo "Chào, $name! Bạn $age tuổi.";
```

```
?>
```

So sánh `$_REQUEST` với `$_POST` và `$_GET`

Biến Lấy dữ liệu từ Khi nào dùng

`$_GET` URL Query String (biến trên URL) Khi không cần bảo mật dữ liệu, như tìm kiếm

`$_POST` Body của HTTP Request Khi cần bảo mật dữ liệu, như đăng nhập

`$_REQUEST` Cả GET, POST, COOKIE Khi không quan tâm form dùng GET hay POST

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>

