Phần mềm mã nguồn mở

Ngôn ngữ lập trình PHP

ThS. Nguyễn Kim Hưng
jackyhung12345@gmail.com

Nội dung

- Giới thiệu PHP
- Cơ chế hoạt động của WebServer
- Cú pháp & Quy ước trong PHP

Giới thiệu về PHP – Lịch sử phát triển

- **PHP :** Rasmus Lerdorf in 1994 (được phát triển để phát sinh các form đăng nhập sử dụng giao thức HTTP của Unix).
- **PHP 2 (1995) :** Chuyển sang ngôn ngữ script xử lý trên server. Hỗ trợ CSDL, Upload File, khai báo biến, mảng, hàm đệ quy, câu điều kiện, …
- **PHP 3 (1998) :** Hỗ trợ ODBC, đa hệ điều hành, giao thức email (SNMP, IMAP), bộ phân tích mã PHP (parser) của Zeev Suraski và Andi Gutmans
- **PHP 4 (2000) :** Trợ thành một thành phần độc lập cho các webserver. Parse đổi tên thành Zend Engine. Bổ sung các tính năng bảo mật cho PHP
- **PHP 5 (2005) :** Bổ sung Zend Engine II hỗ trợ lập trình HĐT, XML, SOAP cho Web Services, SQLite
- Phiên bản mới nhất của PHP là version PHP 5.5.10 *(www.php.n*

Giới thiệu về PHP – PHP là gì ?

- **PHP** viết tắt của **P**HP **H**ypertext **P**reprocessor
- Là ngôn ngữ server-side script, tương tự như ASP, JSP, … thực thi ở phía WebServer
- Tập tin PHP có phần mở rộng là .php
- Cú pháp ngôn ngữ giống ngôn ngữ C & Perl

Giới thiệu về PHP – Ưu điểm 1

- PHP được sử dụng làm
  - Server Side Scripting
  - CommandLine Scripting (cron – Linux, Task Scheduler – Windows, Text Processing)
  - Xây dựng ứng Desktop – PHP GTK

Giới thiệu về PHP – Ưu điểm 2

- **Đa môi trường (Multi-Platform)**
  - Web Servers: Apache, Microsoft IIS, Caudium, Netscape Enterprise Server
  - Hệ điều hành: UNIX (HP-UX, OpenBSD, Solaris, Linux), Mac OSX, Windows NT/98/2000/XP/2003/vista
  - Hệ QTCSDL: Adabas D, dBase,Empress, FilePro (read-only), Hyperwave, IBM DB2, Informix, Ingres, InterBase, FrontBase, mSQL, Direct MS-SQL, MySQL, ODBC, Oracle (OCI7 and OCI8), Ovrimos, PostgreSQL, SQLite, Solid, Sybase, Velocis,Unix dbm

Giới thiệu về PHP – Ưu điểm 3

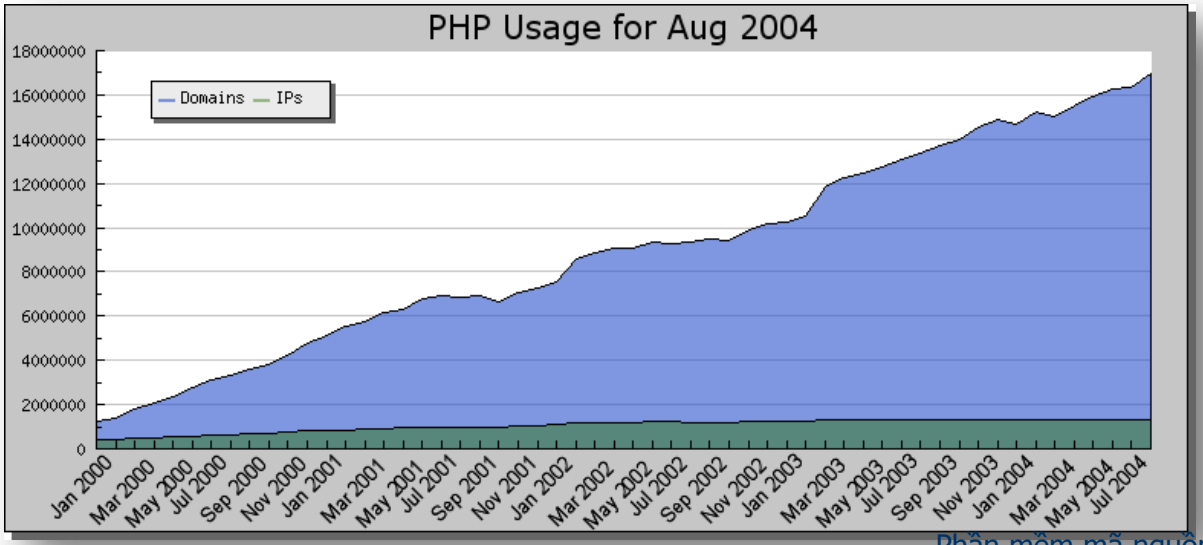
- **Miễn phí**

	PHP
Software	Free
Platform	Free (Linux)
Development Tools	Free (PHP Coder, jEdit, …)

Giới thiệu về PHP – Ưu điểm 4

- Được sử dụng rộng rãi trong môi trường phát triển web
  - 20,917,850 domains (chiếm hơn 32% tên miền website)
  - 1,224,183 IP addresses

    (04/2007 Netcraft Survey – *http://www.php.net/usage.php*)



Giới thiệu về PHP – Một số website lớn

PHP at Yahoo!

http://www.yahoo.com

The Internet's most trafficked site

Joomla! ...because open source matters

Portal

mambo power in simplicity

Portal

moodle

Course Management System

MediaWiki

Wiki

SUGARCRM COMMERCIAL OPEN SOURCE

Customer Relationship Management

zen cart the art of e-commerce

e-Commerce

XOOPS powered by you

Portal

phpbb creating communities

Bulletin Board

Dragonfly

Content Management System

MANTIS

Help Desk

Giới thiệu về PHP – Cần gì để chạy PHP ?

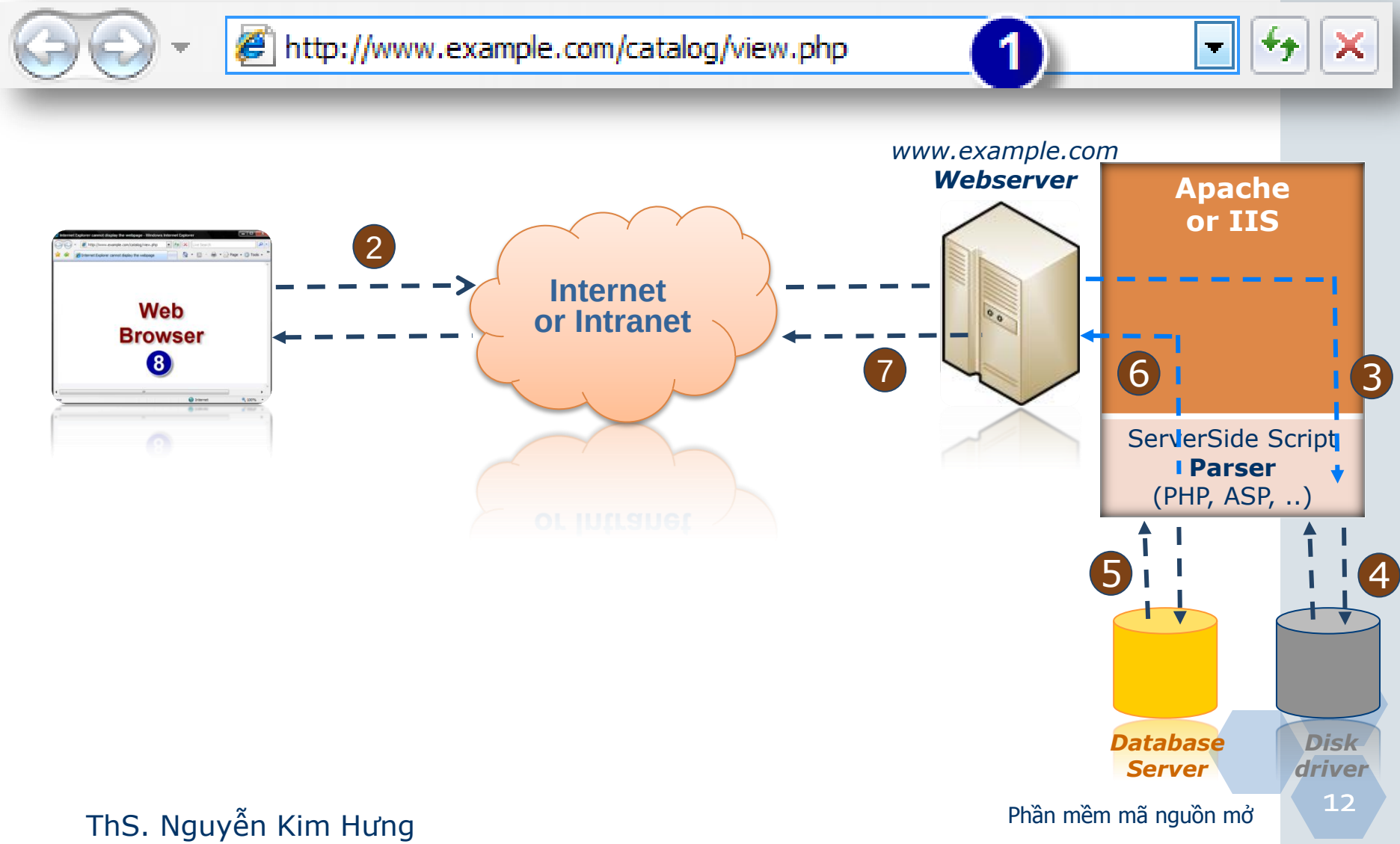
- **Download PHP**
  - Download PHP for free here: http://www.php.net/downloads.php
- **Download MySQL Database**
  - Download MySQL for free here: http://www.mysql.com/downloads/index.html
- **Download Apache Server**
  - Download Apache for free here: http://httpd.apache.org/download.cgi

→ Download WAMP,LAMP

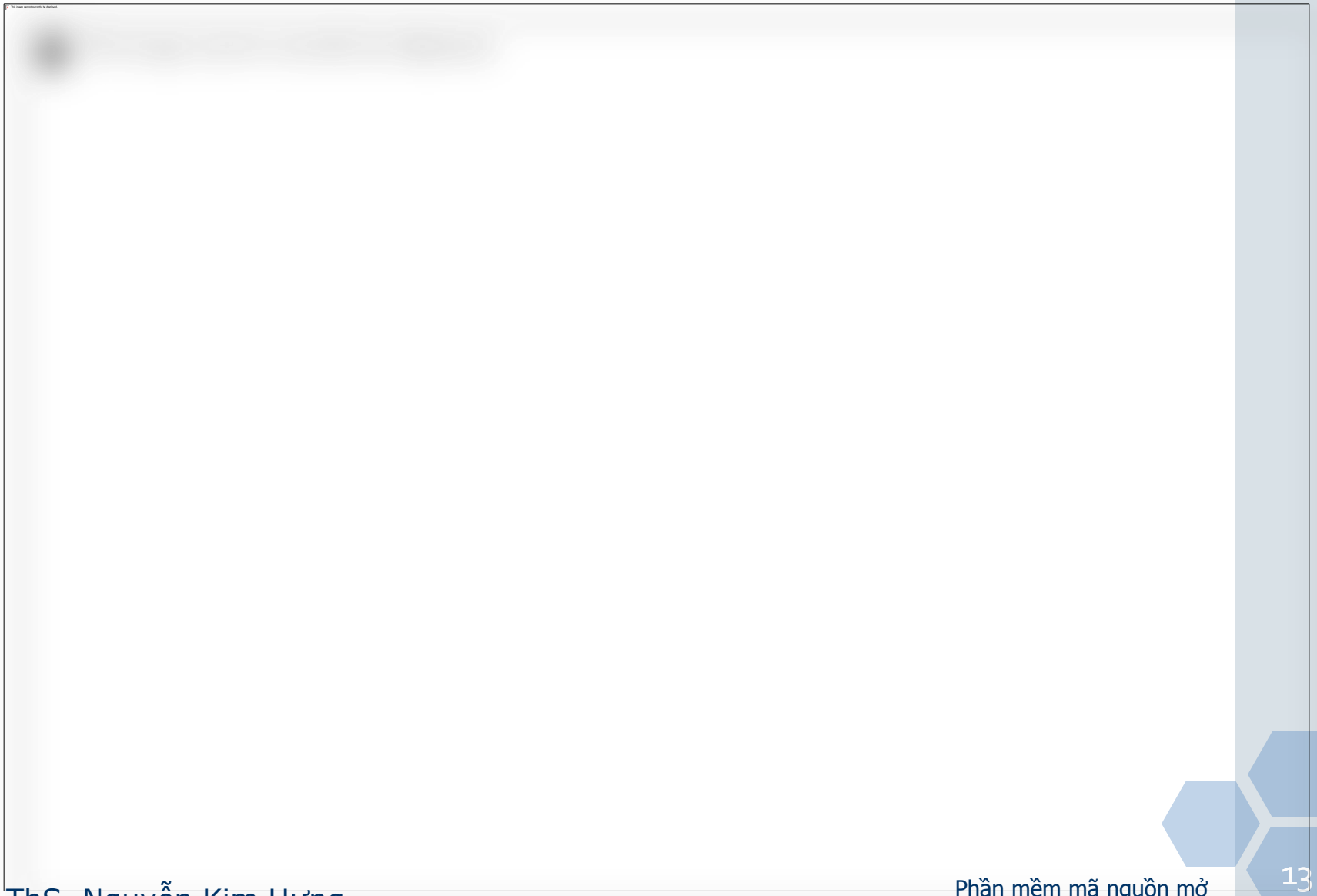
Nội dung

- Giới thiệu PHP
- Cơ chế hoạt động của WebServer
- Cú pháp & Quy ước trong PHP

Cơ chế hoạt động của WebServer

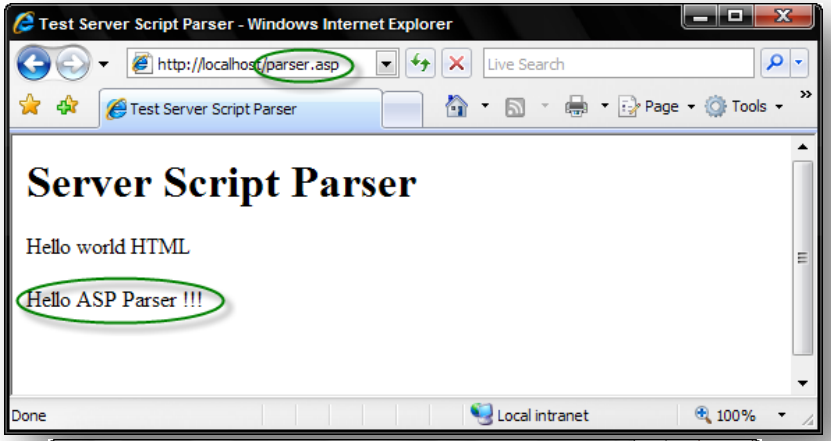


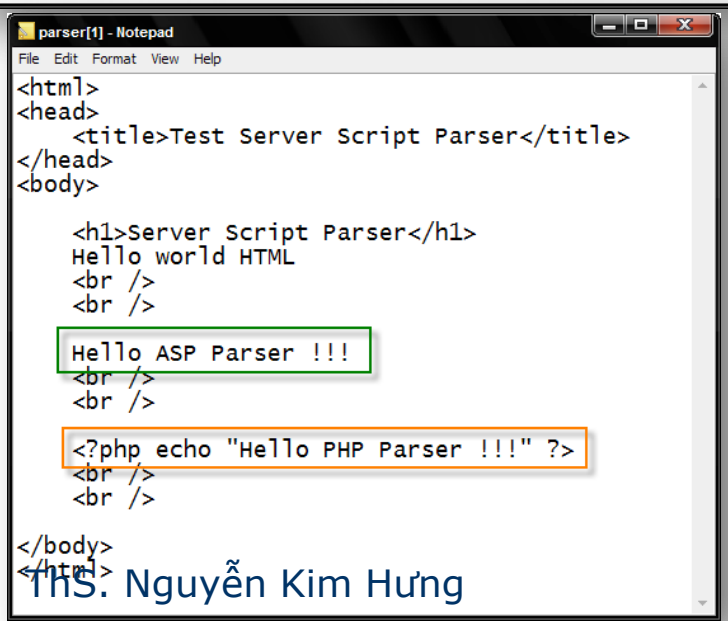
Cơ chế hoạt động của WebServer



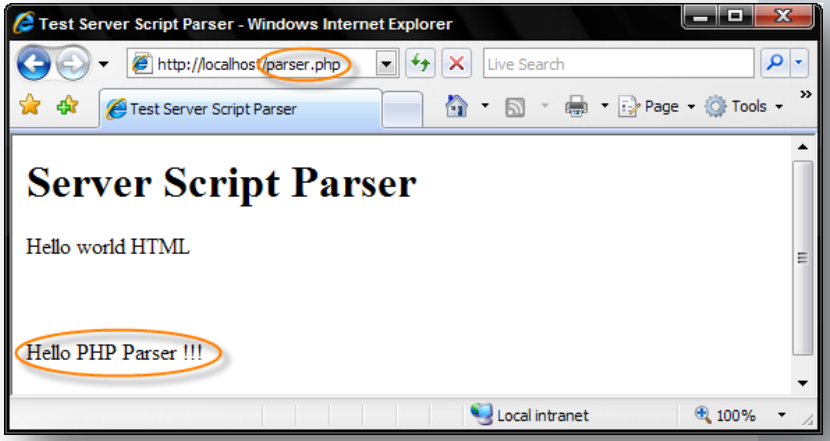
Cơ chế hoạt động của WebServer

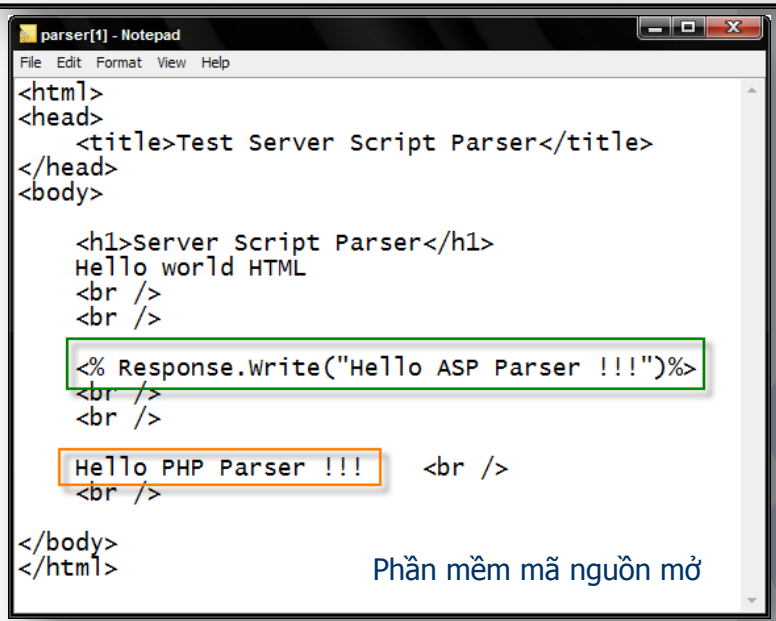
❖ Parser.asp





❖ Parser.php





Nội dung

- Giới thiệu PHP
- Cơ chế hoạt động của WebServer
- Cú pháp & Quy ước trong PHP

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Phạm vi biến
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

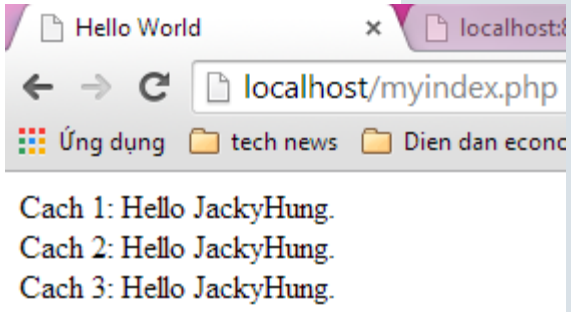
Quy ước

- Mã lệnh PHP được đặt trong các cặp thẻ sau :

Thẻ mở	Thẻ đóng
<?	?>
<?php	?>
<script language="php">	<script>

myindex.php | info.txt | feed_new_v1.xml | hosts | select product.txt

```
1  <html>
2  <head>
3      <title>Hello World</title>
4  </head>
5  <body>
6      <?
7          echo 'Cach 1: Hello JackyHung. <br>';
8      ?>
9
10     <?php
11         echo 'Cach 2: Hello JackyHung. <br>';
12     ?>
13
14     <script language="php">
15         echo 'Cach 3: Hello JackyHung. <br>';
16     </script>
17 </body>
18 </html>
```



Quy ước

- Tất cả các câu lệnh php đều cách nhau bởi dấu ";"
- Khối (nhiều) lệnh được đặt trong cặp { }
- Không phân biệt khoảng trắng, Tab, xuống dòng trong câu lệnh

```
<?php print "Hello"; print " World!"; ?>
```

```
<?php
    Print "Hello"                    ;
    print " World!";
?>
```

- Ghi chú : Theo cú pháp ghi chú của C++ & Perl

```
// Đây là ghi chú
# Đây là ghi chú
/* Đây là ghi
   chú nhiều dòng*/
```

Ví dụ

```
1
2  <?
3  Echo "PHP is simple"; //day la vi du ve code PHP
4  /* Voi cu phap nay chung ta
5  Co the  chu thich 1 cum ma lenh */
6  ?>
7
```

```
1  <?php
2  Echo "Hello word";
3  Printf"<br><font color=red>Who Are You ?</font>";
4  ?>
5
```

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

Khai báo biến

$ten_bien = value;

- Không khai báo kiểu dữ liệu
- Biến tự động được khởi tạo ở lần đầu tiên gán giá trị cho biến
- Tên biến :
  - **Có thể bao gồm các Ký tự (A..Z, a..z), Ký số (0..9), _, $**
  - **Không được bắt dầu bằng ký số (0..9)**
  - **Không chứa ký tự trắng (space, tab)**
  - **Phân biệt chữ hoa – chữ thường**

Ví dụ :

$size $my_drink_size $_drinks
$drink4you

$$2hot4u $drink-size x

Khai báo biến

Variable variables

- Cho phép thay đổi tên biến
- Ví dụ:

```
$varname = “my_variable”;
$$varname = “xyz”; // $my_variable = “xyz”
```

Hằng số - Constants

- Ví dụ:

```
define(“MY_CONST”, 10);
echo MY_CONST;
```

Ví dụ sử dụng biến

- Gán giá trị cho biến

```
<?php
    $qty = 30;
    $price = 20;
    $total = $qty * $price;
    echo "Tong tien :" . $total;
?>
```

- Thay đổi biến

```
<?php
    $qty = "soluong";
    echo "qty:" . $qty ."<br>";
    $$qty = 40;
    echo "so luong :" . $soluong;
?>
```

Chú ý: Toán tử "." dùng để nối chuỗi

Phạm vi của biến

- Có ba mức phạm vi:
  - Biến hàm: được khai báo và sử dụng cục bộ trong phạm vi hàm
  - Biến toàn cục (không nằm trong hàm): được khai báo và sử dụng bên trong một script, mặc định là không thể sử dụng bên trong các hàm.
  - Biến siêu toàn cục:

    Có thể sử dụng ở mọi nơi, không thể định nghĩa bởi người dùng.

Một số biến siêu toàn cục

- $GLOBALS
- $_SERVER
- $_GET, $_POST
- $_SESSION, $_COOKIE
- $_REQUEST
- $_ENV
- $php_errormsg

Biến $GLOBAL

- PHP coi 1 biến có một giới hạn. Để xác định một biến toàn cục (global) có tác dụng trong một hàm , ta cần khai báo lại. Nếu không giá trị của biến sẽ được coi như là biến cục bộ.

- Ví dụ

```
<?
    $a = 1;
    $b = 2;

    Function Sum ()
        {
            global $a, $b;
            $b = $a + $b;
        }

    Sum ();
    echo $b;
?>
```

Biến $GLOBAL

- Một cách khác để dùng biến toàn cục trong 1 hàm là ta dùng mảng $GLOBAL của PHP

- Ví dụ

```
<?
    $a = 1;
    $b = 2;

    Function Sum ()
    {
        $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
    }

    Sum ();
    echo $b;
?>
```

Biến $REQUEST

- Lấy các giá trị của GET, POST, COOKIE … theo thứ tự GPCES (Get, Post, Cookie, Enviroment, Server)
- Tuy nhiên , các phần tử trong mảng REQUEST là hoàn toàn độc lập với các phần tử trong mảng GET , POST vvv... Bạn có thể thay thế bằng giá trị khác với mảng REQUEST như giá trị trong GET,POST thì không đổi.
- Ví dụ:

```
<?
    $_POST['username'] = "cottonbelly";
    $_GET['username'] = "snoopy0877";

    echo $_POST['username']; // sẽ in ra : cottonbelly
    echo $_GET['username']; // sẽ in ra : snoopy0877
    echo $_REQUEST['username']; // sẽ in ra : snoopy0877

    $_REQUEST['username'] = "lambada";

    echo $_POST['username']; // sẽ in ra : cottonbelly
    echo $_GET['username']; // sẽ in ra : snoopy0877
    echo $_REQUEST['username']; // sẽ in ra : lambada thay vì snoopy0877
?>
```

Tuổi thọ của biến

- Biến được tạo ra khi được gán giá trị lần đầu và tồn tại trong suốt quá trình thực thi script
- Mỗi lần script được thực thi là biến được tạo ra độc lập với các lần thực thi khác của cùng script đó

Hằng

- Định nghĩa:
  - ***define*** (‘tên_hằng’, giá trị)
  - Giá trị hằng chỉ được dùng các kiểu dữ liệu cơ bản
  - Bắt buộc định nghĩa trước khi dùng
- Quy ước về cách đặt tên:
  - ***Giống*** cách đặt tên biến
  - Không sử dụng ký hiệu ***$***
  - Thường đặt tên bằng chữ ***in hoa***

Ví dụ

```php
<?
$a= 100 // biến a ở đây có giá trị là 100.
$a= "PHP is easy" // Biến a ở đây có giá trị "PHP Is easy".
Biena=123 //Có lỗi vì bắt đầu 1 biến phải có dấu "$"
$123a="PHP" //Có lỗi vì phần tên bắt đầu của biến là dạng số.
?>
```

```php
<?
define ("C", "COMPANY");
define ("YELLOW", "#ffff00");
echo "Gia tri cua C la". C;
?>
```

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

Kiểu dữ liệu

- boolean (bool)
- integer (int)
- double (float, real)
- string
- array
- object

1 Biến trong PHP có thể lưu **bất kỳ kiểu dữ liệu** nào.

Sự chuyển đổi kiểu dữ liệu

Có hai hình thức ép kiểu chính

- *Ép kiểu ngầm định*

  Xảy ra tự động khi thực hiện các toán tử đòi hỏi hai biểu thức cùng kiểu

- *Ép kiểu chỉ định*

  Chỉ định một kiểu dữ liệu cụ thể đặt trong cặp () trước biểu thức cần ép kiểu

Một số hàm liên quan đến ép kiểu

- bool is_*type* ($tên_biến hay biểu thức):

  is_integer, is_float, is_numeric, is_string, is_bool, is_array, is_double, is_real, is_int, is_object

  - *Kiểm tra dữ liệu của một biến, kết quả trả về true hoặc false*

- string gettype($tên_biến hay biểu thức)
  - *Trả về loại kiểu dữ liệu như: integer, double, long …*

- int settype($tên_biến, “kiểu_dữ_liệu”)
  - *Gán kiểu dữ liệu cho tên biến*

Chuyển kiểu dữ liệu

Chuyển kiểu dữ liệu

- Cách 1 **(automatic)**

```
$var = "100" + 15;
$var = "100" + 15.0;
$var = 39 . " Steps";
```

- Cách 2: **(datatype) $var**
- Cách 3: **settype($var, "datatype")**

$var	(int)$var	(bool)$var	(string)$var
null	0	false	""
true	1		"1"
false	0		""
"6 feet"	6	true	
"foo"	0	true	

Chuyển kiểu dữ liệu

- Kiểm tra kiểu dữ liệu

gettype **is_string** **isset**
is_integer **is_array** **unset**
is_double **is_object** **empty**

Ví dụ:

```
$var = "test";
if (isset($var))
  echo "Variable is Set";
if (empty($var))
  echo "Variable is Empty";
```

Kiểu số - int, float

- Một số hàm xử lý số
  - **abs** **pow** **decbin** **srand(seed)**
  - **ceil** **sqrt** **bindec** **rand**
  - **Floor** **log** **dechex** **rand(min, max)**
  - **round** **log10** **hexdec** **…**

- **Ví dụ**

```
// Generate a seed
$seed = (float) microtime(  ) * 100000000;

// Seed the pseudo-random number generator
srand($seed);

// Generate some random numbers
print rand();      // between 0 and getmaxrand(  )
print rand(1, 6);  // between 1 and 6 (inclusive)
```

Kiểu chuỗi - string

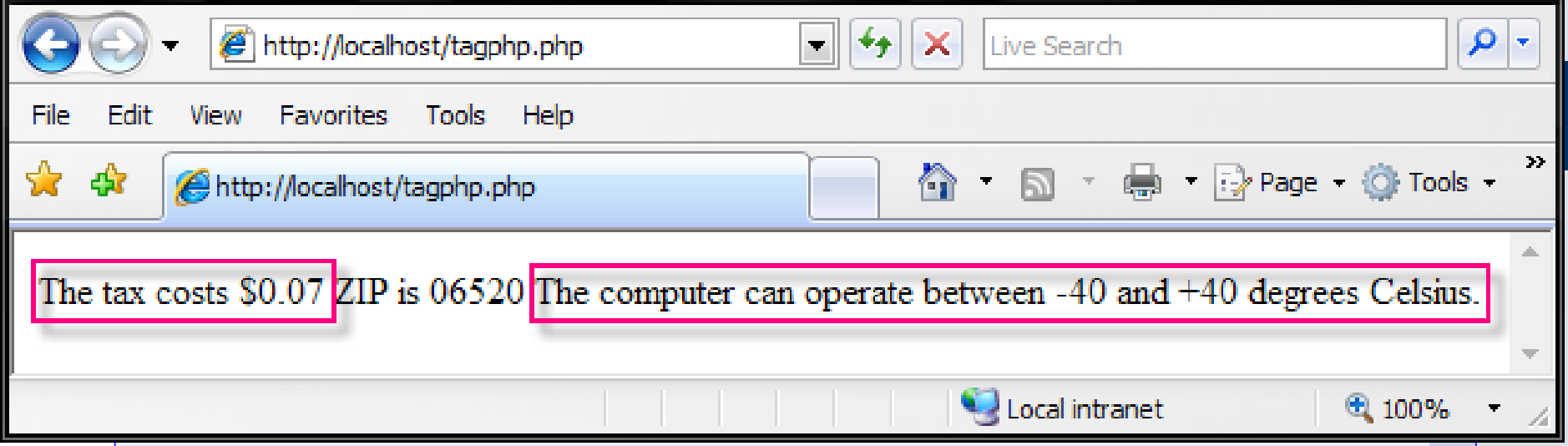
- Toán tử nối chuỗi : dấu chấm .

```
$s = “Hello” . “ World”;      // $s = “Hello World”
```

- Phân biệt dấu nháy đơn và nháy kép

```
$user = “Bill”;
print ‘Hi $user’;             // Hi $user
print “Hi $user”;             // Hi Bill
print ‘Hi’ . $user;           // ????
print ‘Hi’ . ‘$user’;         // ????
```

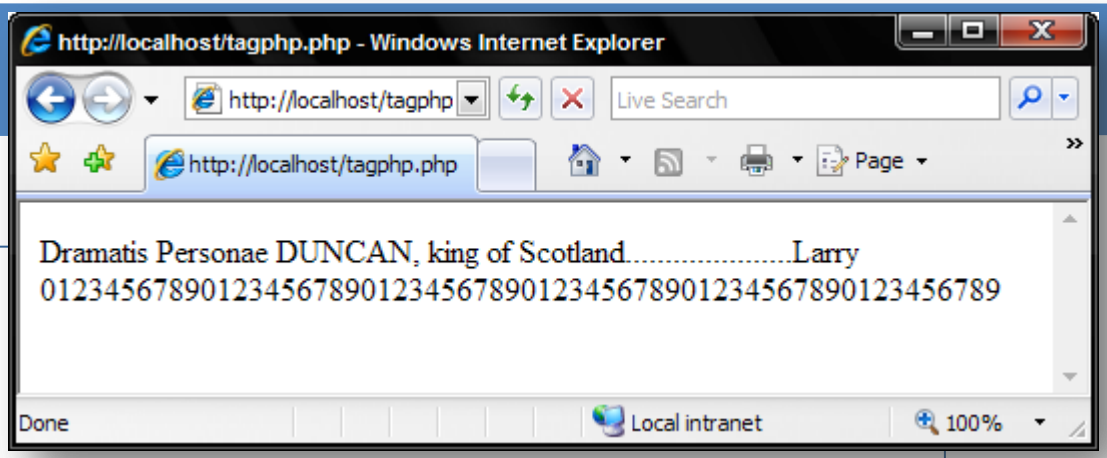
- Một số hàm xử lý chuỗi
  - **printf** **trim** **strtolower**
  - **str_pad** **str_replace** **strtoupper**
  - **strlen** **substr** **strcasecmp**
  - **…**

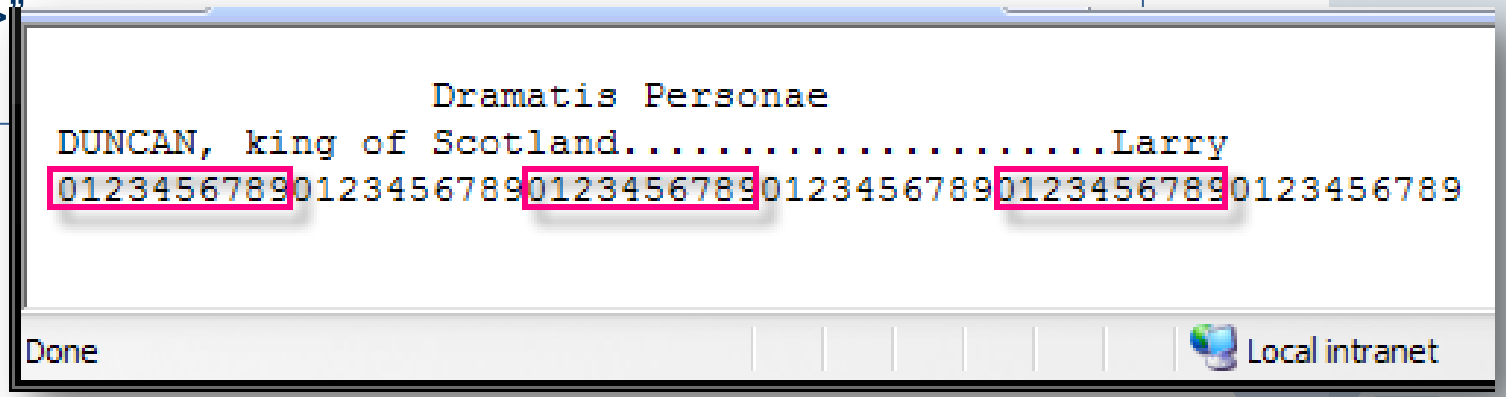



```
$zip = '6520';
printf("ZIP is %05d”, $zip);

$min = -40; $max = 40;
printf("The computer can operate between %+d and %+d
degrees Celsius.", $min, $max);
?>
```

Ví dụ

```
<?
echo “<pre>”

// Print a heading
echo str_pad(“Dramatis Personae", 50, " ", STR_PAD_BOTH)
     . “\n";

// Print an index line
echo     str_pad(“DUNCAN, king of Scotland”, 30, ".")
         . str_pad(“Larry”, 20, ".", STR_PAD_LEFT)
         . “\n";
echo “</pre>”
?>
```

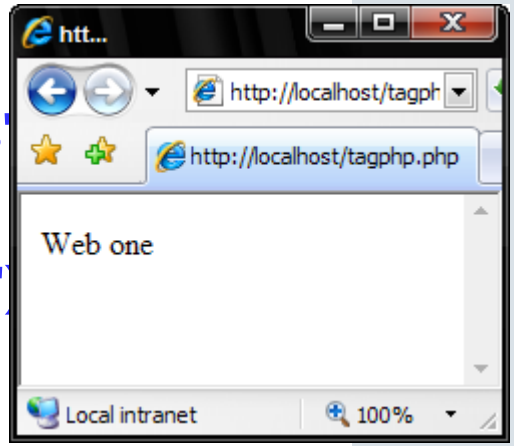




Mảng - array

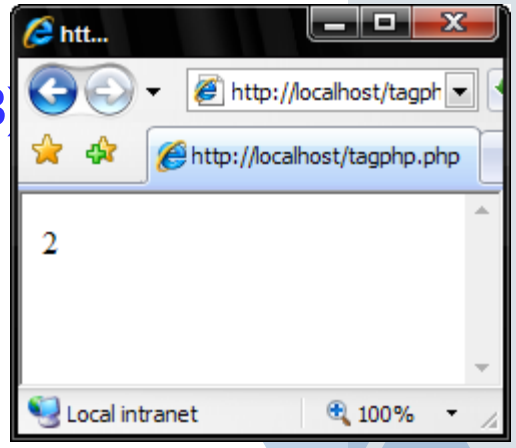
Numbered array

```
$words = array("Web", "Database", "Applications"
echo $words[0];
$numbers = array(1=>"one", "two", "three", "four"
echo $numbers[1];
```



Associated array

```
$array = array("first"=>1, "second"=>2, "third"=>3
echo $array["second"];
```



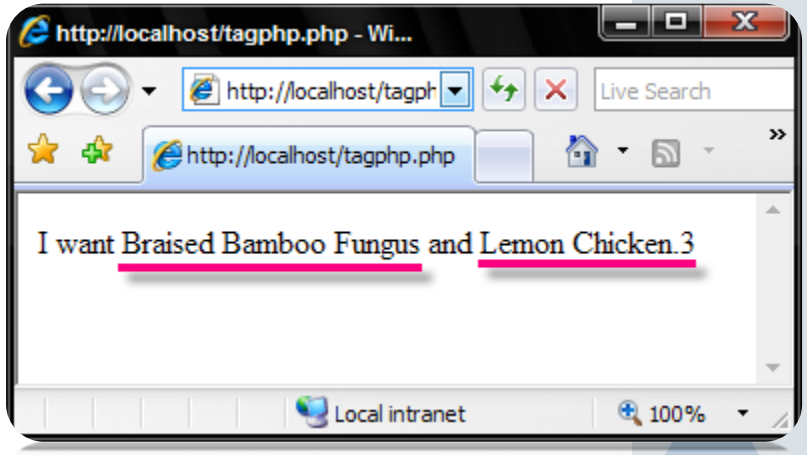
Mảng - array

Một số hàm xử lý trên mảng

- count is_array sort asort ksort usort
- min array_reverse rsort arsort krsprt uasort
- max uksort

Ví dụ:

```
$dinner = array( 'Sweet Corn and Asparagus',
                 'Lemon Chicken',
                 'Braised Bamboo Fungus');
sort($dinner);
print "I want $dinner[0] and $dinner[1].";
$dishes = count($dinner);
print $dishes;
```



Mảng - array

Một số hàm liên quan đến mảng

- reset(array)
- array_push(array, elements) : Thêm elements vào cuối mảng
- array_pop(array) : Lấy phần tử cuối ra khỏi mảng
- array_unshift(array, elements) : Thêm elements vào đầu mảng
- array_shift(array) : Lấy phần tử đầu ra khỏi mảng
- array_merge(array, array) : kết 2 mảng lại và trả ra mảng mới
- shuffle(array) : Sort random mảng
- sort(array, flag) : flag = {sort_regular, sort_numeric, sort_string, sort_locale_string}

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

Toán tử: gán và số học

Gán	Số học	Kết hợp
=	+	+=
	-	-=
	*	*=
	/	/=
	%	%=

Toán tử: so sánh

Ký hiệu	Ý nghĩa
==	Bằng giá trị
===	Bằng giá trị và cùng kiểu
!=	Khác giá trị
<>	Khác giá trị
!==	Khác giá trị hoặc khác kiểu
<	Nhỏ hơn
>	Lớn hơn
<=	Nhỏ hơn hoặc bằng
>=	Lớn hơn hoặc bằng

Toán tử: logic

Ký hiệu	Ý nghĩa
and	And
&&	And
or	Or
||	Or
xor	Xor
!	Not

Toán tử: bitwise

Ký hiệu	Ý nghĩa
&	And
|	Or
^	Xor
~	Not
<<	Dịch trái
>>	Dịch phải

Toán tử: tăng giảm 1

Ký hiệu	Ý nghĩa
++	Tăng 1
--	Giảm 1

Thứ tự ưu tiên phép toán

Qui tắc liên kết	Toán tử	Ghi chú
	new	Tạo 1 đối tượng từ 1 class, toán tử này chỉ áp dụng trên 1 toán hạng nên không có qui tắc liên kết
Bên phải trước	[	Toán tử truy cập 1 phần tử trong màng
	++ --	Tăng/Giảm 1 đơn vị, toán tử này chỉ áp dụng trên 1 toán hạng nên không có qui tắc liên kết
	! ~ - (int) (float) (string) (array) (object) @	Các toán tử này chỉ áp dụng trên 1 toán hạng nên không có qui tắc liên kết
Bên trái trước	* / %	
Bên trái trước	+ - .	
Bên trái trước	<< >>	
	== != === !===	Toán tử so sánh, chỉ áp dụng trên 2 toán hạng nên không có qui tắc liên kết
Bên trái trước	&	
Bên trái trước	^	
Bên trái trước	|	
Bên trái trước	&&	
Bên trái trước	||	
Bên trái trước	? :	
Bên phải trước	= += -= *= /= .= %= &= |= ^= <<= >>=	
Bên trái trước	and	
Bên trái trước	xor	
Bên trái trước	or	
Bên trái trước	,	

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

Cấu trúc điều khiển

- Điều kiện if
- Điều khiển switch
- Vòng lặp for
- Vòng lặp while
- Vòng lặp do.. While
- Vòng lặp foreach
- Từ khóa break, continue
- Câu lệnh *Return*
- Câu lệnh *Include*

Điều kiện if

```
if (condition)
{
    statement[s] if true
}
else (condition)
{
    statement[s] if false
}
```

Ví dụ:

```
$x = 5;
if ($x < 4)
  echo "$x is less than 4";
else
  print '$x isn't less than 4';
```

$x isn't less than 4

Điều khiển switch

```
switch (expression)
{
  case label :
       statementlist
       break;
  case label :
       statementlist
       break;
      ...
  default :
       statementlist
}
```

Ví dụ:

```
$menu = 3;
switch ($menu){
  case 1:
    echo "You picked one";
    break;
  case 2:
    echo "You picked two";
    break;
  case 3:
    echo "You picked three";
  case 4:
    echo "You picked four";
    break;
  default:
    echo "You picked another
option";
}
```

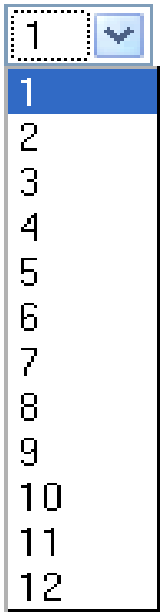
You picked three You picked four

Vòng lặp for

```
for ([initial expression]; [condition]; [update expression])
{
        statement[s] inside loop
}
```

- **Ví dụ:**

```
print "<select>";
for ($i = 1; $i <= 12; $i++) {
   print
   "<option>$i</option>";
}
print "</select>";
```



Vòng lặp while, do...while

```
while (expression)
{
   statements
}
```

```
do
{
      statements
}while (expression);
```

Ví dụ:

```
$i = 1; $j = 9;
while ($i <= 10) {
   $temp = $i * $j;
   print "$j * $i =
$temp<br>";
   $i++;
}
```

```
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

Vòng lặp foreach

```
foreach (array as variable)
{
        statements
}
```

breakfast	Walnut Bun
lunch	Cashew Nuts and White Mushrooms
dinner	Eggplant with Chili Sauce

Ví dụ:

```
$meal = array('breakfast' => 'Walnut Bun',
          'lunch' => 'Cashew Nuts and White
Mushrooms',
          'dinner' => 'Eggplant with Chili Sauce');

print "<table border='1'>\n";
foreach ($meal as $key => $value) {
    print
"<tr><td>$key</td><td>$value</td></tr>\n";
}
print '</table>';
```

Các lệnh ngắt lặp

- *Break*

  Dừng và thoát ra khỏi vòng lặp *for, foreach, while, do-while* và *switch*

- *Continue*

  Dừng thực hiện lần lặp hiện hành để chuyển sang lần lặp tiếo theo

Câu lệnh *Return*

- Trong một hàm, câu lệnh ***Return*** kết thúc việc thực thi hàm và trả về kết quả. Nó cũng kết thúc thực hiện script.

```
<?php
   function test() {
     return;
   }

   echo gettype(test()) . "\n";
   echo (test()?'true':'false') . "\n";
   echo (!test()?'true':'false') . "\n";
   echo (test() === false?'true':'false') . "\n";
?>
```

Câu lệnh *Include*

Chèn code của một file khác vào trang PHP hiện tại.

File *vars.php*

```
<?php

$color = 'green';
$fruit = 'apple';

?>
```

File test.php

```
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

Hàm - function

```
function functionName ([parameter1]...[,parameterN])
{
        statement[s] ;
}
```

```
function functionName ([parameter1]...[,parameterN])
{
        statement[s] ;
        return ….. ;
}
```

Hàm – Phạm vi biến

```
<?php
function doublevalue($var=5)
{
    global $temp;
    $temp = $var * 2;
}

$temp = 5;
doublevalue();
echo "\$temp is: $temp";
?>
```

$temp is:

$temp is: 10

Hàm – Phạm vi biến giá trị mặc định

```
<?php
    function makecoffee($type = "cappuccino")
    {
        return "Making a cup of $type.\n";
    }

    echo makecoffee();
    echo makecoffee(null);
    echo makecoffee("espresso");
?>
```

Kết quả:

```
Making a cup of cappuccino.
Making a cup of .
Making a cup of espresso.
```

Hàm – Tham trị vs Tham biến

```
<?php
function doublevalue(&$var)
{
    $var = $var * 2;
}

$variable = 5;
doublevalue($variable);
echo "\$variable is: $variable";
?>
```

$variable is: 5

$variable is: 10

Hàm có số lượng tham số không xác định

- Khai báo danh sách tham số rỗng ()
- Sử dụng các hàm sau để lấy danh sách các tham số:
  - func_num_args(): số lượng tham số khi hàm được gọi
  - func_get_arg(i): giá trị các tham số thứ i được truyền (bắt đầu từ 0)
  - func_get_args(): danh sách tất cả các tham số

Ví dụ

```
<?php
   function foo()
   {
       $numargs = func_num_args();
       echo "Number of arguments: $numargs\n";
   }

   foo(1, 2, 3);
?>
```

Biến tĩnh

Thêm từ khóa ***static*** khi khai báo biến

Được khởi tạo (và gán giá trị) một lần đầu tiên duy nhất trong suốt quá trình thực thi của script

Sử dụng biến toàn cục

Khai báo lại biến toàn cục với từ khóa global (bên trong hàm) để có thể sử dụng được biến toàn cục này bên trong hàm

Sử dụng các hàm sau để lấy danh sách các tham số:

func_num_args(): số lượng tham số khi hàm được gọi

func_get_arg(i): giá trị các tham số thứ i được truyền (bắt đầu từ 0)

func_get_args(): danh sách tất cả các tham số

Ví dụ

```
Function Test ()
    {
       static $a = 0;
       echo $a;
       $a++;
    }
```

- Với khai báo như trên, $a sẽ không mất đi giá trị sau khi gọi hàm Test() mà $a sẽ được tăng lên 1 sau mỗi lần gọi hàm Test().

Hàm

- ## Phạm vi

  Có giá trị sử dụng trong toàn script, ngay cả trước và sau khi định nghĩa

- ## Lồng hàm

  Cho phép định nghĩa lồng hàm, thậm chí lồng bên trong một cấu trúc điều khiển (if, switch, while/do, while…)

  Loại hàm này có phạm vi trong toàn script và không thể định nghĩa lại

Hàm

Hàm biến

Khi một biến kiểu chuỗi được khai báo và gán giá trị trùng khớp với tên một hàm được định nghĩa thì tên biến đó có thể được dùng như một cách gọi hàm khác với cách gọi hàm bình thường bằng tên hàm.

Một số hàm không thể dùng như hàm biến

- echo
- print
- var_dump
- print_r
- isset
- unset
- is_null
- is_type

Ví dụ

```php
<?php
function foo() {
    echo "In foo()<br />\n";
}

function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br />\n";
}

function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();        // This calls foo()

$func = 'bar';
$func('test');  // This calls bar()

$func = 'echoit';
$func('test');  // This calls echoit()
?>
```

Ví dụ hàm *echo*

*void **echo** (string $arg1 [, string $...])*

```php
<?php
// You can use variables inside of an echo statement
$foo = "foobar";
$bar = "barbaz";
echo "foo is $foo"; // foo is foobar
// You can also use arrays
$baz = array("value" => "foo");
echo "this is {$baz['value']} !"; // this is foo !
// Using single quotes will print the variable name, not the value
echo 'foo is $foo'; // foo is $foo
// Some people prefer passing multiple parameters to echo over concatenation.
echo 'This ', 'string ', 'was ', 'made ', 'with multiple parameters.', chr(10);
echo 'This ' . 'string ' . 'was ' . 'made ' . 'with concatenation.' . "\n";
echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon. no extra whitespace!
END;
// Because echo does not behave like a function, the following code is invalid.
//($some_var) ? echo 'true' : echo 'false';
// However, the following examples will work:
($some_var) ? print 'true' : print 'false'; // print is also a construct, but
                                            // it behaves like a function, so
                                            // it may be used in this context.
echo $some_var ? 'true': 'false'; // changing the statement around
?>
```

Ví dụ hàm *print*

```php
<?php
print("Hello World");
print "print('abc'); also works without parentheses.";
print "This spans
multiple lines.
The newlines will be
output as well";
print "This spans\nmultiple lines. The newlines will be\noutput as well.";
print "escaping characters is done \"Like this\".";
// You can use variables inside of a print statement
$foo = "foobar";
$bar = "barbaz";
print "foo is $foo"; // foo is foobar
// You can also use arrays
$bar = array("value" => "foo");
print "this is {$bar['value']} !"; // this is foo !
// Using single quotes will print the variable name, not the value
print 'foo is $foo'; // foo is $foo
// If you are not using any other characters, you can just print variables
print $foo;              // foobar
print <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;
?>
```

Ví dụ hàm *var_dump*

- Hiển thị kiểu dữ liệu và giá trị của nó

```php
<?php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

The above example will output:

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```

Ví dụ hàm *print_r*

- Hiển thị thông tin về một biến
- Cú pháp

  mixed **print_r** (mixed $expression [, bool $return = false])

  - Mixed : chỉ định như một biến chấp nhận nhiều kiểu dữ liệu
  - Nếu bạn muốn giữ lại kết quả đầu ra thì trả về cho một biến, tham số $return đặt là *True*.

Ví dụ

```
<pre>
<?php
$a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y', 'z'));
print_r ($a);
?>
</pre>
```

The above example will output:

```
<pre>
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

Hàm *isset* và *unset*

```php
<?php
$var = '';
// This will evaluate to TRUE so the text will be printed.
if (isset($var)) {
    echo "This var is set so I will print.";
}
// In the next examples we'll use var_dump to output
// the return value of isset().
$a = "test";
$b = "anothertest";
var_dump(isset($a));      // TRUE
var_dump(isset($a, $b)); // TRUE

unset ($a);
var_dump(isset($a));      // FALSE
var_dump(isset($a, $b)); // FALSE

$foo = NULL;
var_dump(isset($foo));    // FALSE
?>
```

Hàm *is_null*

- Trả về *True* nếu biến là Null, *False* trong các trường hợp khác.

```php
<?php

error_reporting(E_ALL);

$foo = NULL;
var_dump(is_null($inexistent), is_null($foo));

?>
```

Sử dụng lại các Hàm

- Sử dụng hai hàm ***require()*** và ***include()*** để chèn các tệp *PHP, text, HTML và cả class PHP*

Hàm – include & require

```
// functions.inc
<?php
function bold($string)
{
  echo "<b>" . $string .
"</b>\n";
}
?>

require "functions.inc";
```

```
// index.php
<html>
<head>
  <title>Simple Function
Call</title>
</head>
<body bgcolor="#ffffff">
<?
include "functions.inc";

bold("this is bold");

$myString = "this is bold";
bold($myString);
?>
</body></html>
```

Sự khác nhau giữa hàm ***require()*** và ***include()***

- Dùng Require:
  - Thông báo lỗi *"fatal"* và dừng thực thi script.
- Dùng Include:
  - Thông báo lỗi và tiếp tục thực thi.

Mảng

- Tạo mảng và gán giá trị

  $tên_biến = array([khóa =>]giá trị, [khóa =>]giá trị,…)

  Trường hợp không định nghĩa các khóa thì mảng sẽ được gán khóa mặc định theo kiểu số nguyên tăng dần bắt đầu từ 0

- Ví dụ
  - $a=array("Kenny","Maria","Julia","Kenvin");
  - $a= array (name => "Kenny", job => "Teacher", age=>"45", email => "webmaster@vietchuyen.com.vn")

- Tạo mảng từ một mảng có sẵn

  $tên_mảng_mới = $tên_mảng_cũ

Mảng

- Thêm một phần tử vào mảng

  $tên_mảng[khóa] = giá trị

  - Phần tử luôn được thêm vào cuối mảng
  - Nếu khóa đã tồn tại thì không có phần tử nào được thêm
  - Trường hợp không chỉ định khóa thì khóa sẽ được chọn bằng khóa có giá trị số nguyên lớn nhất cộng 1
- Xóa một phần tử khỏi mảng

  unset($tên_mảng[khóa])
- Đếm số phần tử của mảng

  count($tên_mảng)

Ví dụ

```php
<?php
$arr = array("foo" => "bar", 12 => true);

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

```php
<?php
$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["somearray"][6];    // 5
echo $arr["somearray"][13];   // 9
echo $arr["somearray"]["a"];  // 42
?>
```

Mảng

- Truy xuất

  $mảng[khóa]

  Khi dùng khóa chuỗi bên trong một chuỗi, không được dùng cặp ‘’ hoặc “”, nếu không thì phải đặt truy xuất bên trong cặp {}

  vd:

```
// sai lỗi cú pháp
echo "My PC has a $computer['processor'] processor<br/>\n";
echo "My PC has a $computer[""processor""] processor<br/>\n";

// đúng cú pháp nhưng không nên dùng
echo "My PC has a $computer[processor] processor<br/>\n";

// cách dùng tốt nhất
echo "My PC has a {$computer['processor']} processor<br/>\n";
```

Ví dụ

```php
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // This is the same as $arr[13] = 56;
                // at this point of the script

$arr["x"] = 42; // This adds a new element to
                // the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr);    // This deletes the whole array
?>
```

- Duyệt mảng với vòng lặp foreach

```
foreach (mảng as [khóa =>] giá trị)
    Khối lệnh;
```

Ví dụ

```php
<?php
// Create a simple array.
$array = array(1, 2, 3, 4, 5);
print_r($array);

// Now delete every item, but leave the array itself intact:
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array);

// Append an item (note that the new key is 5, instead of 0).
$array[] = 6;
print_r($array);

// Re-index:
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

Duyệt mảng với vòng lặp for

Tạo mảng khóa số nguyên trung gian

$mảng_khóa = array_keys($mảng)

Truy xuất thông qua mảng khóa

$mảng[$mảng_khóa[i]]

Duyệt mảng với con trỏ mảng

Con trỏ mảng trỏ vào phần tử đầu tiên khi mảng được tạo ra

Các hàm di chuyển con trỏ mảng

- reset($mảng)
- end ($mảng)
- current($mảng) / pos($mảng)
- each($mảng)/next($mảng): di chuyển con trỏ tới vị trí tiếp theo.
- prev($mảng)

Mảng đa chiều

Khai báo

```
$mảng_đa_chiều = array(khóa_1 => array(khóa_2 => …),
                       khóa_1 => array(khóa_2 => …),
                       …
                       khóa_1 => array(khóa_2 => …))
```

Truy xuất

```
$mảng_đa_chiều[khóa_1][khóa_2][…][khóa_n]
```

Ví dụ

```php
<?php
$fruits = array ( "fruits"  => array ( "a" => "orange",
                                       "b" => "banana",
                                       "c" => "apple"
                                     ),
                  "numbers" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6
                                     ),
                  "holes"   => array (      "first",
                                       5 => "second",
                                            "third"
                                     )
                );

// Some examples to address values in the array above
echo $fruits["holes"][5];    // prints "second"
echo $fruits["fruits"]["a"]; // prints "orange"
unset($fruits["holes"][0]);  // remove "first"

// Create a new multi-dimensional array
$juices["apple"]["green"] = "good";
?>
```

Các hàm xử lý mảng

Sắp xếp

Theo giá trị

- sort($mảng) / asort($mảng) // tăng dần
- rsort($mảng) / arsort($mảng) // giảm dần
- natsort($mảng) / natcasesort($mảng) // tăng dần, dùng cho chuỗi
- usort($mảng, ”hàm_so_sánh”) // tự định nghĩa thứ tự
- uasort($mảng, ”hàm_so_sánh”) // tự định nghĩa thứ tự

Theo khóa

- ksort($mảng) // tăng dần
- krsort($mảng) // giảm dần
- uksort($mảng, ”hàm_so_sánh”) // tự định nghĩa thứ tự

Các hàm xử lý mảng

- Nối ghép hai mảng

  array_merge($mảng1, $mảng2)

  array_combine($mảng1, $mảng2)

  array_intersect($mảng1, $mảng2)

- Tìm kiếm

  array_search($giá_trị, $mảng)

Chuỗi (***String***)

- Một chuỗi là một dãy các ký tự
- Một ký tự giống như một byte
- PHP không giới hạn kích thước kiểu chuỗi, nó chỉ phụ thuộc vào bộ nhớ mà PHP đang chạy.
- Có 4 cách để biểu diễn một chuỗi
  - Dùng dấu ngoặc đơn (‘’)
  - Dùng dấu ngoặc kép (“”)
  - Dùng câu lệnh *“heredoc”*
  - Dùng câu lệnh *“nowdoc”* (PHP 5.3.0)

Dùng dấu ngoặc đơn (‘’)

- Đây là cách đơn giản nhất để thể hiện chuỗi, đặt chuỗi cần hiện trong dấu ngoặc đơn
- Để hiển thị dấu nháy đơn trong chuỗi in ra thì dùng ký tự / (backsplash) trước ký tự ‘
- Chú ý:
  - Các biến trong dấu ngoặc đơn và các ký tự đặc biệt cho xuống dòng sẽ không được PHP biên dịch trong dấu ngoặc đơn.

Ví dụ

```php
<?php
$expand = "variable";
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: You deleted C:\*.*?
echo 'You deleted C:\\*.*?';

// Outputs: You deleted C:\*.*?
echo 'You deleted C:\*.*?';

// 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';
?>
```

Dùng dấu ngoặc kép (“”)

- PHP sẽ biên dịch các biến và các ký tự đặc biệt ở trong dấu ngoặc kép.

Sequence	Meaning
\n	linefeed (LF or 0x0A (10) in ASCII)
\r	carriage return (CR or 0x0D (13) in ASCII)
\t	horizontal tab (HT or 0x09 (9) in ASCII)
\v	vertical tab (VT or 0x0B (11) in ASCII) (since PHP 5.2.5)
\f	form feed (FF or 0x0C (12) in ASCII) (since PHP 5.2.5)
\\	backslash
\$	dollar sign
\"	double-quote
\[0-7]{1,3}	the sequence of characters matching the regular expression is a character in octal notation
\x[0-9A-Fa-f]{1,2}	the sequence of characters matching the regular expression is a character in hexadecimal notation

Ví dụ

```php
<?php
$ex= "variable";
echo "This is a simple string";

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

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

- Kết quả:

```
This is a simple stringThis will not expand:
 a newlineVariables do not
```

Dùng câu lệnh "*heredoc*"

- Là cách thứ 3 phân định một chuỗi.
- Câu lệnh:
  - <<< từ_khóa_nhận_diện

    Các dòng văn bản ở đây

    từ_khóa_nhận_diện;
- Ví dụ

```
<?php
$str = <<<EOD
        Example of string
        spanning multiple lines
        using heredoc syntax.
EOD;
echo $str;
?>
```

Dùng câu lệnh *"nowdoc"*

- Cú pháp giống *"heredoc", nhưng ký tự nhận diện được thêm ''*
- Chức năng: giống như dùng dấu ngoặc đơn ('')
- Ví dụ:

```
<?php
$str = <<<'EOD'
Example of string
spanning multiple lines
using nowdoc syntax.
EOD;
echo $str;
?>
```

Các hàm xử lý chuỗi

- Các xử lý cơ bản
  - trim($chuỗi, [’ký tự muốn cắt’])
  - ltrim($chuỗi, [’ký tự muốn cắt’])
  - rtrim($chuỗi, [’ký tự muốn cắt’])
  - strlen($chuỗi)
  - substr($chuỗi, $vị trí, $chiều_dài)
  - strtoupper ($chuỗi)
  - strtolower ($chuỗi)
  - iconv(mã nguồn, mã đích, $chuỗi)

Ví dụ hàm trim()

```php
<?php

$text   = "\t\tThese are a few words :) ...  ";
$binary = "\x09Example string\x0A";
$hello  = "Hello World";
var_dump($text, $binary, $hello);

print "\n";

$trimmed = trim($text);
var_dump($trimmed);

$trimmed = trim($text, " \t.");
var_dump($trimmed);

$trimmed = trim($hello, "Hdle");
var_dump($trimmed);

// trim the ASCII control characters at the beginning and end of $binary
// (from 0 to 31 inclusive)
$clean = trim($binary, "\x00..\x1F");
var_dump($clean);

?>
```

Ví dụ hàm iconv() -> convert chuôi theo loại mã nào đó

```php
<?php
$text = "This is the Euro symbol '€'.";

echo 'Original : ', $text, PHP_EOL;
echo 'TRANSLIT : ', iconv("UTF-8", "ISO-8859-1//TRANSLIT", $text), PHP_EOL;
echo 'IGNORE   : ', iconv("UTF-8", "ISO-8859-1//IGNORE", $text), PHP_EOL;
echo 'Plain    : ', iconv("UTF-8", "ISO-8859-1", $text), PHP_EOL;

?>
```

The above example will output something similar to:

```
Original : This is the Euro symbol '€'.
TRANSLIT : This is the Euro symbol 'EUR'.
IGNORE   : This is the Euro symbol ''.
Plain    :
Notice: iconv(): Detected an illegal character in input string in .\iconv-example.php on line 7
This is the Euro symbol '
```

Các hàm xử lý chuỗi

Tìm kiếm

- strpos($chuỗi, $chuỗi_con, [$vị_trí_bắt đầu])
- strrpos ($chuỗi, $chuỗi_con, [$vị_trí_bắt đầu])

So sánh

- strcmp($chuỗi_1, $chuỗi_2)
- strncmp($chuỗi_1, $chuỗi_2, $chiều_dài)
- strcasecmp($chuỗi_1, $chuỗi_2)
- strncasecmp($chuỗi_1, $chuỗi_2, $chiều_dài)
- strnatcmp($chuỗi_1, $chuỗi_2)
- strnatcasecmp($chuỗi_1, $chuỗi_2)

Ví dụ sử dụng hàm strpos()

```php
<?php
$mystring = 'abc';
$findme   = 'a';
$pos = strpos($mystring, $findme);

// Note our use of ===.  Simply == would not work as expected
// because the position of 'a' was the 0th (first) character.
if ($pos === false) {
    echo "The string '$findme' was not found in the string '$mystring'";
} else {
    echo "The string '$findme' was found in the string '$mystring'";
    echo " and exists at position $pos";
}
?>
```

PHP và Unicode

Thiết lập môi trường trong php.ini

Thiết lập	Giá trị
extension	php_mbstring.dll
extension_dir	“đường dẫn đến thư mực extension của php"
mbstring.language	Neutral
mbstring.internal_encoding	UTF-8
mbstring.encoding_translation	
mbstring.http_input	UTF-8
mbstring.http_output	UTF-8
mbstring.substitute_character	?
mbstring.func_overload	7

Cú pháp & Quy ước trong PHP

- Quy ước
- Khai báo biến
- Kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Hàm
- Lớp đối tượng

Class và object

Khai báo lớp

class ***tên_lớp***
{
    các thuộc tính và phương thức
}

Tạo và hủy một đối tượng

$tên_biến = new ***tên_lớp***();

Đối tượng sẽ tự động bị hủy khi không còn tham chiếu nào đến nó

$tên_biến = NULL;

Member và *method*

- Các từ khóa khai báo

  public: có thể sử dụng bên ngoài lớp

  private: chỉ sử dụng cục bộ bên trong lớp

  protected: sử dụng được bởi các lớp kế thừa

- Một số quy tắc chung
  - Không thể khai báo hai method trùng tên
  - Method phải được khai báo ngay bên trong khai báo lớp
  - Dùng biến giả **$this** để truy xuất các member và method trong lớp
  - Dùng toán tử -> để truy xuất đến member và method

Ví dụ

```
<?php
    class Cart {
      var $items;  // Items in our shopping cart

      // Add $num articles of $artnr to the cart

      function add_item ($artnr, $num) {
          $this->items[$artnr] += $num;
      }

      // Take $num articles of $artnr out of the cart
    function remove_item ($artnr, $num) {
          if ($this->items[$artnr] > $num) {
              $this->items[$artnr] -= $num;
              return true;
          } else {
              return false;
          }
      }
    }
?>
```

- Lớp Cart ở đây là một kiểu dữ liệu, vì vậy bạn có thể tạo một biến có kiểu này với toán tử new

```
$cart = new Cart;
$cart->add_item("10", 1);
```

Constructor và destructor

- Khai báo

```
public function __construct(danh sách tham số)
{
    khởi tạo giá trị các member;
}
```

constructor được tự động thực hiện khi đối tượng được tạo

```
public function __destruct()
{
    dọn dẹp;
}
```

destructor được tự động thực hiện khi đối tượng bị hủy

Constant

- Khai báo

```
const TÊN_HẰNG = giá trị;
```

- Truy xuất

```
tên_lớp::TÊN_HẰNG      // ngoài lớp
seft::TÊN_HẰNG         // trong lớp
```

Static member

- Khai báo

  … static $thuộc_tính

- Truy xuất

  tên_lớp::$thuộc_tính // ngoài lớp

  seft::$thuộc_tính // trong lớp

Static method

- Khai báo

  … static function phương_thức(…)

- Truy xuất

  tên_lớp::phương_thức(…) // ngoài lớp

  seft:: phương_thức(…) // trong lớp

Thừa kế

Khai báo lớp con

```
class lớp_con extends lớp_cha
{
    các thuộc tính và phương thức
}
```

Tất cả các member và method được khai báo public hay protected trong lớp cha được thừa kế và có thể sử dụng trong lớp con

Phương thức nạp chồng

- Gọi một method lớp cha

parent::phương_thức(…)

Bằng cách định nghĩa lại một phương thức đã có ở lớp cha, tất cả các lời gọi đến phương thức này mà không chỉ định rõ như trên sẽ được hiểu là gọi phương thức có cùng tên của lớp con

Khai báo lớp trừu tượng

```
abstract class lớp_trừu_tượng
{
    // các thuộc tính
    abstract public function
    phương_thức_trừu_tượng(…);
    …
    // các phương thức khác
}
```

Không thể tạo đối tượng trực tiếp từ lớp trừu tượng

Lớp con bắt buộc phải định nghĩa các phương thức trừu tượng của lớp cha

Ngăn kế thừa và nạp chồng

- Lớp không thể kế thừa

```
final class không_thể_kế_thừa { … }
```

- Phương thức không thể nạp chồng

```
final public function không_thể_nạp_chồng(…) { … }
```

Interface

Khai báo Interface

```
interface giao_diện
{
    public function phương_thức();
    …
}
```

Khai báo lớp theo mẫu Interface

```
abstract class tên_lớp implements giao_diện
{
    …
}
```

Các lớp sử dụng Interface hay kế thừa từ một lớp sử dụng Interface bắt buộc phải định nghĩa tất cả các phương thức trong Interface đó

Một số lưu ý

Phép gán đối tượng

```
$a = new lớp();
$b = $a;          // $a và $b cùng trỏ đến một thực thể của lớp
```

Nhân bản đối tượng

```
$b = clone $a    // $b được tạo mới và sao chép giá trị từ $a
```

phương thức __clone():

sau khi sao chép toàn bộ các giá trị từ $a vào $b, phương thức này sẽ được từ động gọi nếu được định nghĩa trong lớp của $a và $b

Hỏi và giải đáp