Bài 5

PHP cơ bản

Viện CNTT & TT

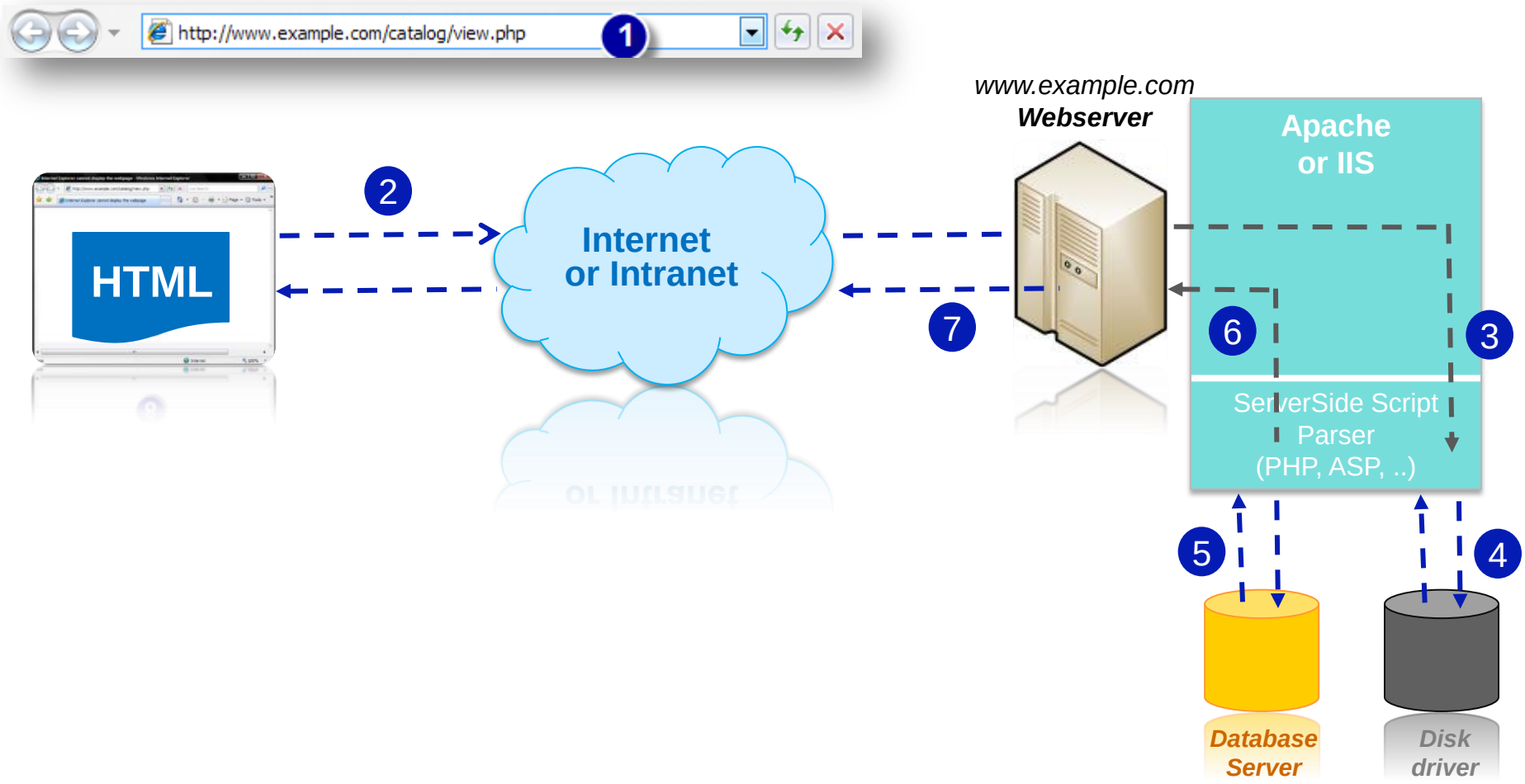
1. Giới thiệu về PHP

- PHP là gì?
  - PHP = **P**HP **H**ypertext **P**reprocessor, tên gốc là Personal Home Pages.
  - PHP là ngôn ngữ viết web động.
  - Bộ biên dịch PHP là phần mềm mã nguồn mở.
  - Là ngôn ngữ server-side script, tương tự như ASP, JSP, … thực thi ở phía WebServer
  - Thường kết nối với hệ quản trị CSDL MySQL

Giới thiệu về PHP – Lịch sử phát triển

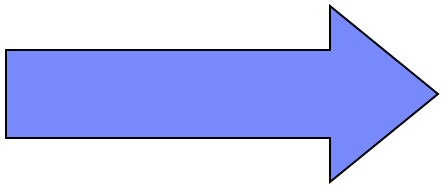
- **PHP :** Rasmus Lerdorf in 1994 (được phát triển để phát sinh các form đăng nhập sử dụng giao thức HTTP của Unix)
- **PHP 2 (1995) :** Chuyển sang ngôn ngữ script xử lý trên server. Hỗ trợ CSDL, Upload File, khai báo biến, mảng, hàm đệ quy, câu điều kiện, biểu thức, …
- **PHP 3 (1998) :** Hỗ trợ ODBC, 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 5.4.0 *(www.php.net)*

Cơ chế hoạt động của WebServer



Cài đặt

- **Để thiết kế trang web sử dụng PHP & MySQL, cần cài đặt:**
  - Máy chủ web Apache
  - PHP
  - Hệ quản trị cơ sở dữ liệu MySQL



XAMPP

Một số khái niệm

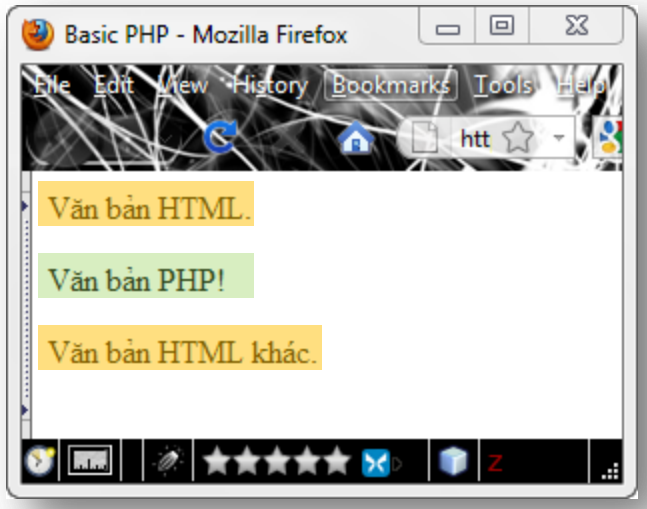
- PHP nhúng vào HTML
  - Có thể nhúng mã PHP vào mọi vị trí trong tài liệu HTML.
  - Chèn mã PHP vào file HTML: Có 3 dạng chính

```
<?php echo("Hello World!"); ?>

<? echo("Hello World!"); ?>

<script language="php">
        echo("Hello World!");
</script>
```

  - Phần mở rộng của tập tin chứa mã PHP thường là **.php:** index.php, giohang.php, ...

Ví dụ 1

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5          <title>Basic PHP</title>
6      </head>
7      <body>
8          <p>Văn bản HTML.</p>
9          <?php
10             echo '<p>Văn bản PHP!</p>';
11         ?>
12         <p>Văn bản HTML khác.</p>
13     </body>
14 </html>
```



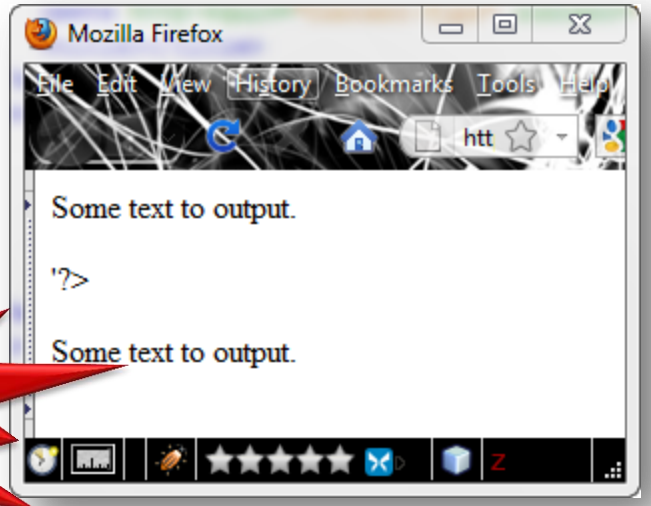
Ví dụ 2

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5          <title></title>
6      </head>
7      <body>
8          <?php
9              echo '<p>Khối dữ liệu PHP 1.</p>';
10         ?>
11         <p>Dữ liệu HTML, <?php echo 'Dữ liệu PHP 2'; ?></p>
12
13         <?php echo '<b>'; ?>
14             Một ví dụ kết hợp HTML và PHP.
15         <?php echo '</b>'; ?>
16     </body>
17 </html>
```

http://localhost/01BasicPhp/01Quyuoc_VD3.php

Ví dụ 3

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5          <title></title>
6      </head>
7      <body>
8          <?='<p>Some text to output.</p>'?>
9
10         <?php
11             echo '<p>Some text to output.</p>';
12         ?>
13     </body>
14 </html>
```



Không nên sử dụng cú pháp PHP viết tắt

Chỉnh sửa php.ini

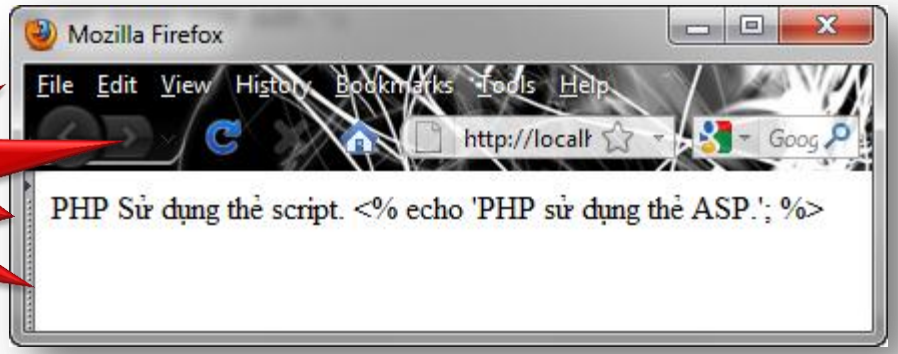
Tham số	Ý nghĩa
short_open_tag = Off	Cho phép sử dụng <? ?>
asp_tags = Off	Cho phép sử dụng <% %>
post_max_size = 8M	Kích thước tối đa của dữ liệu gửi lên server
file_uploads = On	Cho phép upload file
upload_max_filesize = 2M	Kích thước tối đa của mỗi file upload

Ví dụ 4

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5          <title></title>
6      </head>
7      <body>
           <script language="php">
9              echo 'PHP Sử dụng thẻ script. ';
10         </script>
11
12         <%
13             echo 'PHP sử dụng thẻ ASP.';
14         %>
15     </body>
16 </html>
```

Cẩn thận với Javascript

Cú pháp ASP không hỗ trợ PHP6



Một số khái niệm

- Đặc điểm của PHP
  - Có khả năng đối tượng
  - Thông dịch
  - Phân biệt chữ hoa và chữ thường
  - Lệnh kết thúc bằng dấu chấm phẩy “ ; ”
  - PHP là một ngôn ngữ kịch bản ràng buộc lỏng:
    - Không cần khai báo trước, việc khai báo sẽ được tự động thực hiện khi sử dụng.
    - Không cần định kiểu. Kiểu giá trị sẽ được xác định phù hợp với dữ liệu đầu vào

Một số khái niệm

- Tại sao cần dùng PHP?
  - PHP dễ học, dễ viết.
  - Có khả năng truy xuất hầu hết CSDL có sẵn.
  - Thể hiện được tính bền vững, chặn chẽ, phát triển không giới hạn.
  - PHP miễn phí, mã nguồn mở.

Viết ghi chú trong PHP

Để ghi chú trong PHP có 3 dạng sau:

Dạng 1: # đây là ghi chú.

Dạng này chỉ áp dụng ghi đó chỉ nằm trên một dòng văn bản

Dạng 2: // đây là ghi chú.

Dạng này cũng chỉ áp dụng ghi đó chỉ nằm trên một dòng văn bản

Dạng 3: /* đây là một ghi chú dài

Áp dụng cho nhiều hàng */

1. Biến

- Khái niệm
- Khai báo và gán giá trị cho biến
- Phạm vi hoạt động của biến

Khai báo và gán giá trị cho biến

- Khai báo biến
  - Cú pháp: $tên_biến
  - Ví dụ: $tong
- Quy tắc đặt tên cho biến
  - Tên biến phải bắt đầu bằng ký tự $, theo sau là 1 ký tự hoặc dấu _, tiếp đó là ký tự, ký số hoặc dấu _
  - Nên khởi tạo giá trị ban đầu cho biến
  - Tên biến không trùng với tên hàm
  - Biến không nên bắt đầu bằng ký số

Khai báo và gán giá trị cho biến

- Lưu ý
  - Tên biến có phân biệt chữ HOA – chữ thường
  - Ví dụ: biến $a và biến $A là hai biến khác nhau

Khai báo biến – Ví dụ

- Ví dụ :

STT	Tên biến
1	`$size`
2	`$0Zero`
3	`$my_drink_size`
4	`$_drinks`
5	`$Size`
6	`$drink4you`
7	`$$2hot4u`
8	`$drink-Size`
9	`Size`

Khai báo và gán giá trị cho biến

- Gán giá trị cho biến
  - Gán giá trị trực tiếp
    - Cú pháp: $tên_biến = <giá_trị>;
    - Ví dụ:

```
<?php
        $so_luong = 100;
?>
```

Phạm vi hoạt động của biến

- Biến cục bộ
- Biến toàn cục
- Biến static

Phạm vi hoạt động của biến

- Biến cục bộ
  - Biến được khai báo trong hàm => biến cục bộ
  - Khi ra khỏi hàm => biến cục bộ và giá trị của nó sẽ bị hủy bỏ
  - Lưu ý: khi có cùng tên thì biến bên trong hàm và biến bên ngoài hàm là hai biến hoàn toàn khác nhau

Phạm vi hoạt động của biến

- Biến cục bộ
  - Ví dụ:

```
<?php
function Test()
{
    $a=5;
    echo $a; // phạm vi cục bộ
}
Test(); → 5
echo $a; → khong co
?>
```

Phạm vi hoạt động của biến

- Biến toàn cục
  - Có thể truy xuất bất cứ nơi nào trong trang
  - Khi muốn sử dụng và cập nhật biến toàn cục trong hàm thì phải dùng từ khóa global phía trước biến hoặc dùng $_GLOBALS[“tên_biến”]

Phạm vi hoạt động của biến

- Biến toàn cục
  - Ví dụ: dùng từ khóa global

```
<?php
$a = 1;
$b = 2;
function Sum()
{
        global $a, $b;
        $b = $a + $b;
}
Sum();
echo $b; → 3
?>
```

Phạm vi hoạt động của biến

- Biến toàn cục
  - Ví dụ: dùng biến $_GLOBALS

```
<?php
$a = 1;
$b = 2;
function Sum()
{
    $_GLOBALS['b'] = $_GLOBALS['a'] + $_GLOBALS['b'];
}
Sum();
echo $b; → 3
?>
```

Phạm vi hoạt động của biến

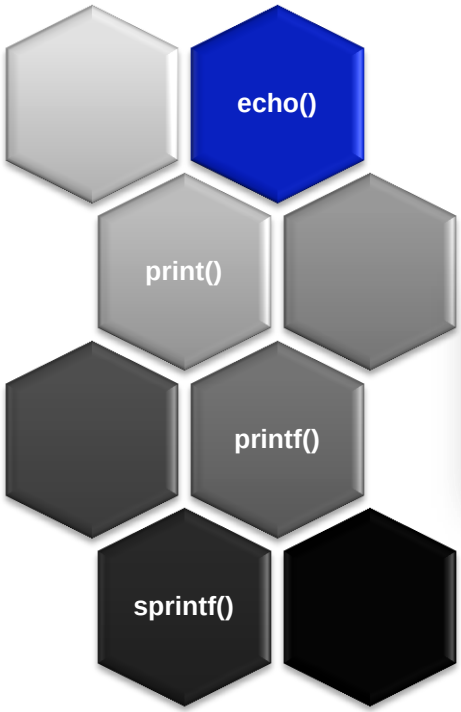
- Biến static
  - Không mất đi giá trị khi ra khỏi hàm
  - Sẽ giữ nguyên giá trị trước đó khi hàm được gọi một lần nữa
  - Phía trước tên biến static phải có từ khóa static

Phạm vi hoạt động của biến

- Biến static
  - Ví dụ:

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
Test(); → 0
Test(); → 1
Test(); → 2
?>
```

Xuất dữ liệu ra trình duyệt

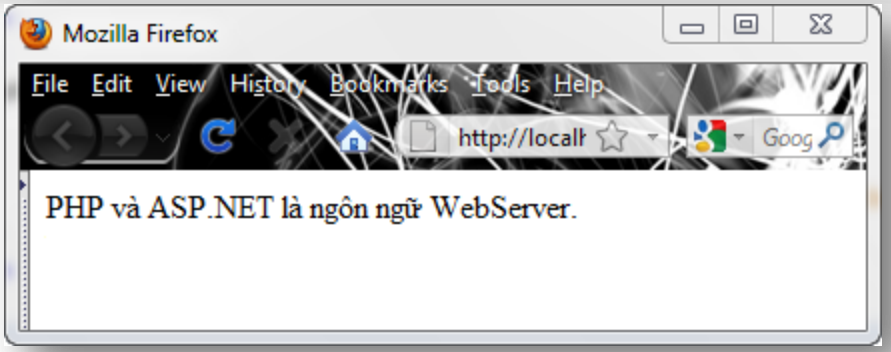


- **Cú pháp:**

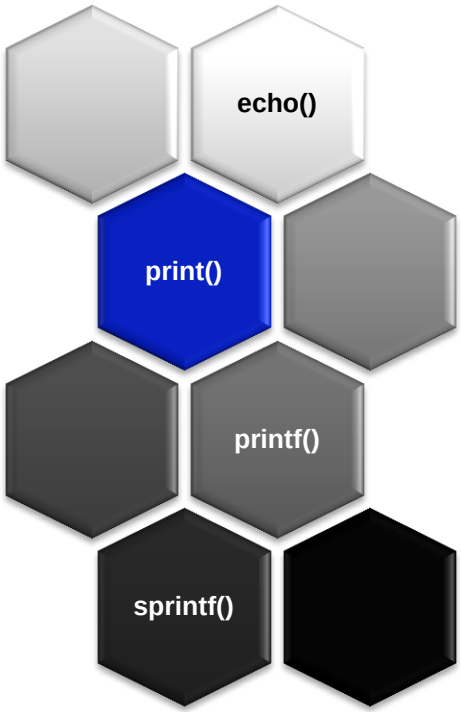
```
void echo(tham số chuỗi [, tham số chuỗi [, tham số chuỗi]])
```

- **Ví dụ:**

```
<?php
    $NgonNgu1 = "PHP";
    $NgonNgu2 = "ASP.NET";
    echo $NgonNgu1, " và ", $NgonNgu2, " là ngôn ngữ WebServer.";
?>
```



Xuất dữ liệu ra trình duyệt

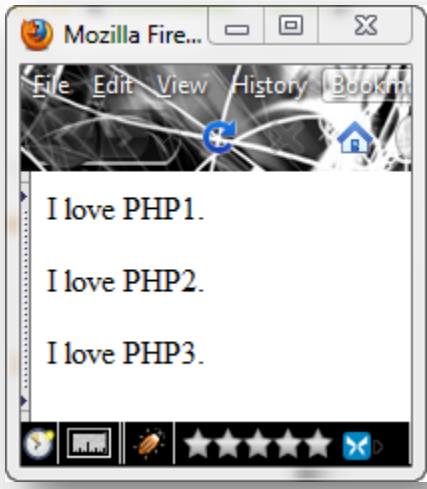


- **Cú pháp:**

```
int print(tham số)
```

- **Ví dụ:**

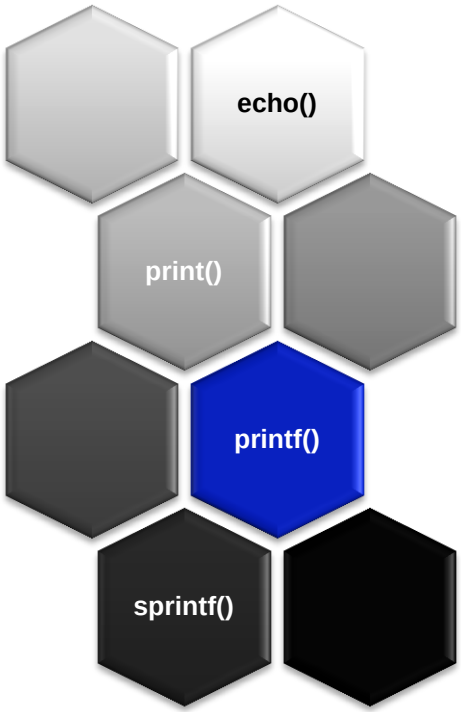
```
<?php
    print("<p>I love PHP1.</p>");
?>

<?php
    $WLang = "<p>I love PHP2.</p>";
    print $WLang;
?>

<?php
    print "<p>I love
                    PHP3.</p>";
?>
```



Xuất dữ liệu ra trình duyệt

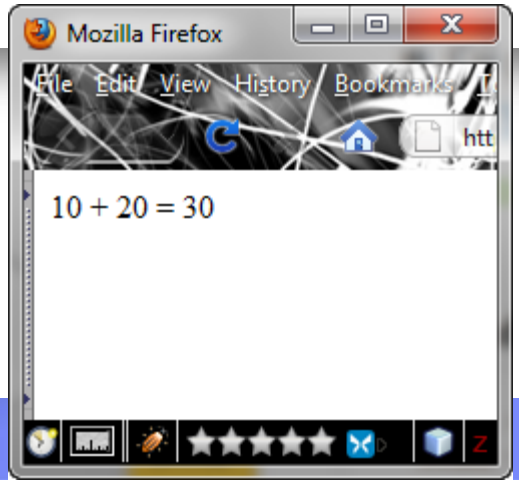
- **Cú pháp:**

```
boolean printf(string
format [, mixed args])
```

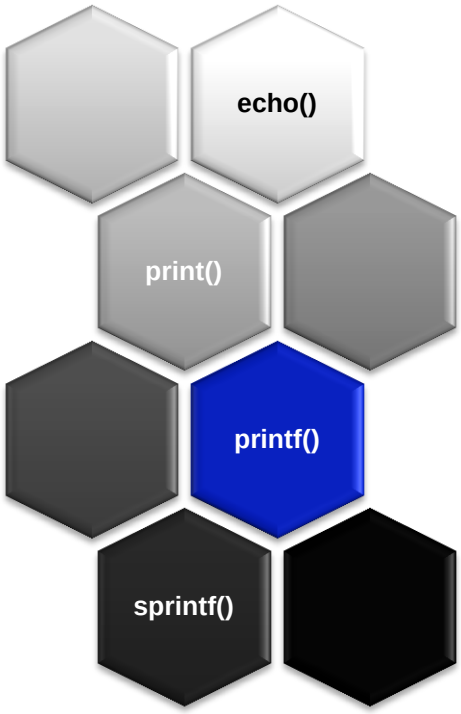


- **Ví dụ:**

```
<?php
    $num1 = 10;
    $num2 = 20;
    printf("%d + %d = %d", $num1, $num2, $num1+$num2);
?>
```



Xuất dữ liệu ra trình duyệt

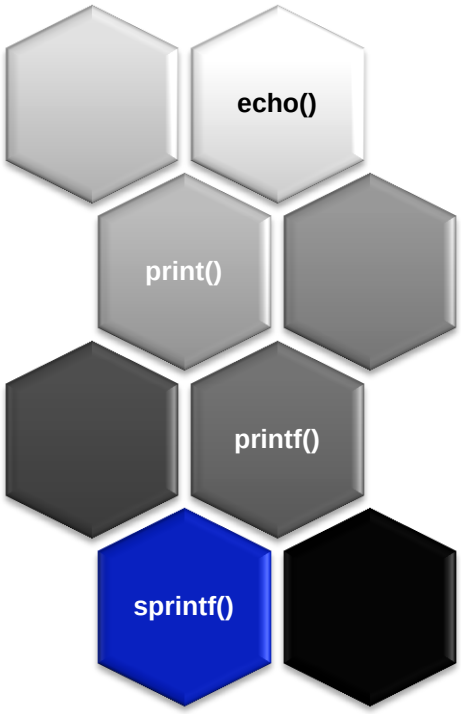


Ký hiệu	Kiểu tham số	Thể hiện ra
%b	Số nguyên	Số nhị phân
%c	Số nguyên	1 ký tự ACSII
%d	Số nguyên	Số nguyên
%f	Số thực	Số thực
%o	Số nguyên	Số bát phân
%s	Chuỗi	Chuỗi
%u	Số nguyên	Số nguyên không dấu
%x	Số nguyên	Số thập lục phân thường
%X	Số nguyên	Số thập lục phân hoa

```
printf("$%.2f", 43.2);
```

```
// $43.20
```

Xuất dữ liệu ra trình duyệt



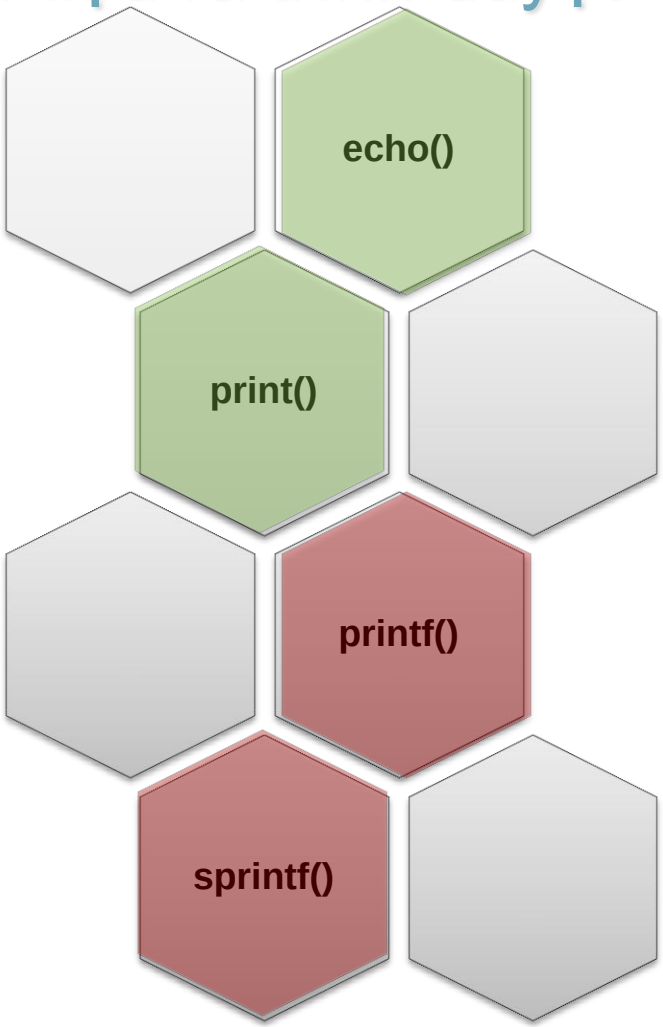
- **Cú pháp:**

```
string sprint(string format
[, mixed args])
```

- **Ví dụ:**

```
$cost = sprintf("$%.2f", 43.2);

// $cost = $43.20
```

Xuất dữ liệu ra trình duyệt



Xuất chuỗi "tĩnh"

Xuất chuỗi "động"

2. Hằng

- Khai báo
- Sử dụng

Khai báo

- Khái niệm: Hằng là một giá trị không thể chỉnh sửa được trong quá trình thực hiện chương trình.
- Quy tắc đặt tên hằng cũng giống như quy tắc đặt tên biến.
- Dùng hàm define() để định nghĩa. Một khi hằng được định nghĩa, nó không bị thay đổi.
- Chỉ có các kiểu dữ liệu boolean, integer, float, string mới có thể chứa các hằng.

Khai báo

- Cú pháp: define(“TÊN_HẰNG”, giá_trị);
  - Ví dụ: khai báo hằng số PI có giá trị là 3.14

```
<?php
        define(“PI”, 3.14);
?>
```

Sử dụng

- Dùng tên hằng mỗi khi muốn sử dụng
  - Ví dụ: tính diện tích và chu vi hình tròn với bán kính r=10

```
<?php
        define(“PI”, 3.14);
        $r=10;
        $s= PI * pow($r ,2); → $s = 314
        $p = 2 * PI * $r; → $p = 62.8
?>
```

3. Kiểu dữ liệu

- Kiểu dữ liệu – mô tả
- Chuyển đổi kiểu dữ liệu

Kiểu dữ liêu trong PHP

- PHP hỗ trợ 4 kiểu dữ liệu
  - Kiểu số.
  - Kiểu chuỗi
  - Kiểu lôgic
  - Kiểu mảng và kiểu đối tượng

Kiểu dữ liệu – mô tả

- Boolean: chỉ có một trong hai giá trị là TRUE và FALSE
  - Ví dụ:

```
<?php
   $ket_qua = True;
?>
```

Kiểu số

- Số nguyên từ -231 đến 231-1
  - Hệ thập phân: VD: $a = 16;
  - Hệ 16 (hexa): VD: $a=0x10;
  - Hệ 8 (bát phân): VD: 020;
- Số thực (thập phân): từ 1.7E-308 đến 1.7E+308
  - Biểu diễn: $a = 0.017
  - Dạng khoa học: $a = 17.0E-03

Kiểu dữ liệu – mô tả

- Integer: Kiểu số nguyên.
- Giá trị có thể là số trong hệ thập phân, thập lục phân và bát phân.
  - Ví dụ:

```
<?php
   $a = 1234; // hệ thập phân
   $b = -123; // số âm hệ thập phân
   $c = 0123; // hệ bát phân (bắt đầu bằng 0 theo sau là các ký số)
   $d = 0x1A; // hệ thập lục phân (bắt đầu bằng 0x theo sau là các ký số)
?>
```

Kiểu dữ liệu – mô tả

- Float / double: Kiểu dữ liệu số thực.
  - Ví dụ:

```
<?php
   $a = 1.234;
   $b = 1.2e3; → 1.2 * 1000 = 1200
?>
```

Kiểu dữ liệu – mô tả

- String: Kiểu dữ liệu chuỗi, ký tự. Trong đó, mỗi ký tự chiếm 1 byte.
- Mỗi chuỗi có thể chứa một hay nhiều ký tự thuộc 256 ký tự khác nhau.
- Chuỗi không có giới hạn về kích thước.
  - Ví dụ:

```
<?php
   $chuoi = 'Chúc mừng năm mới';
   echo "$chuoi 2008";  → "Chúc mừng năm mới 2008"
?>
```

Kiểu chuỗi

- Giới hạn bởi nháy đơn (‘) hoặc kép (“)
- Chuỗi đặt trong nháy kép bị thay thế và xử lý ký tự thoát. Trong nháy đơn thì không.
- PHP **không** hỗ trợ Unicode, để làm việc với Unicode bạn phải sử dụng UTF8 với các hàm **utf8_encode**() – **utf8_decode**().
- Ví dụ:

```
$a = “Hello”;
$b = “$a world”; //tương đương $b=“Hello world”
$c = ‘$a world’; //$c=‘$a world’ (không thay đổi)
```

Kiểu dữ liệu – mô tả

- Array: Kiểu dữ liệu mảng các phần tử
  - Ví dụ:

```
<?php
   // tạo và in một mảng
   $array = array(1, 2, 3, 4, 5);
   print_r($array);
?>
```

Kiểu dữ liệu – mô tả

- Array:
  - $n = 100;

    $arr1 = array($n);
  - $arr2 = array(giá trị 1, giá trị 2,..., giá trị n);
  - $arr3 = array(); //Mảng động
  - $arr4 = array(hoten => "HTùng", //Mảng kết hợp

        quequan => "LX",

        tuoi => 24,

        IQ => "Rất cao");

Kiểu dữ liệu – mô tả

- Object: Kiểu dữ liệu là đối tượng của một lớp
  - Ví dụ:

```
<?php
class a
{
   function chao_a()
   {
      echo "Xin chào a";
   }
}

$b = new a();
$b->chao_a(); // Kết quả: “Xin chào a”
?>
```

Chuyển đổi kiểu dữ liệu

- Trong quá trình tính toán, kiểu dữ liệu cũ của biến có thể không còn phù hợp nữa (kết quả tính toán vượt khỏi phạm vi của kiểu dữ liệu cũ) => chuyển đổi kiểu dữ liệu
- Cách thực hiện: ghi tên kiểu dữ liệu mà biến muốn chuyển đổi vào phía trước biến
  - Ví dụ:

```
<?php
        $don_gia = 5000;
        $so_luong = 100;
        $thanh_tien = (double)($so_luong*$don_gia);
?>
```

Chuyển kiểu dữ liệu

- Cú pháp
  - 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	

4. Các toán tử

- Toán tử số học
- Toán tử nối chuỗi
- Toán tử gán kết hợp
- Toán tử so sánh
- Toán tử luận lý

Các toán tử

Các phép toán cơ bản:

- Phép gán: $biến = biểu_thức;
- Các phép toán số học: +, -, *, /, %
- Các phép toán so sánh: ==, ===, !=, >=, <=, >, <
- Các phép toán kết hợp: ++, --, +=, -=, *=, /=
- Các phép toán logic: !, &&, ||
- Toán tử tam phân: (điều_kiện **?** giá_trị_1 **:** giá_trị_2)
- Phép ghép chuỗi: **.** (dấu chấm)
- Toán tử error: @, ngăn không cho thông báo lỗi

Các phép toán trong PHP

- Phép toán toán học

Phép toán	Ví dụ	Kết quả
+	$y =2; $x=$y+2;	4
-	$y =2; $x=5-$y;	3
*	$y =2; $x=$y*5;	10
/	$y =5; $x=$y/2;	2.5
%	$y =5%3; $x =16%8;	2 0
++	$y =5; $x++;	x=6
--	$y =5; $x--;	x=4

Các phép toán trong PHP

- Phép gán

Phép toán	Ví dụ	Tương đương
=	$x=$y;	$x=$y;
+=	$x+=$y;	$x=$x+$y;
-=	$x-=$y;	$x=$x-$y;
*=	$x*=$y;	$x=$x*$y;
/=	$x/=$y;	$x=$x/$y;
.=	$x.=$y;	$x=$x.$y;
%=	$x%=$y;	$x=$x%$y;

Các phép toán trong PHP

- Phép so sánh

Phép toán	Ví dụ
==	5==8 kết quả false
!=	5!=8 kết quả true
<>	5<>8 kết quả true
>	5>8 kết quả false
<	5<8 kết quả true
>=	5>=8 kết quả false
<=	5<=8 kết quả true

Các phép toán trong PHP

- Phép toán logic

Phép toán	Ví dụ
&&	$x=6; $y=3; ($x < 10 && $y > 1) kết quả true
||	$x=6; $y=3; ($x==5 || $y==5) kết quả false
!	$x=6; $y=3; !($x==$y) kết quả true

Toán tử nối chuỗi

- Sử dụng toán tử "." để nối các chuỗi lại với nhau
- Ví dụ:

```
<?php
        $chuoi_1 = "vietnam";
        $chuoi_2 = "airline.com.vn";
        $chuoi = $chuoi_1 . $chuoi_2;
        → "vietnamairline.com.vn"
?>
```

- Kết quả khi sử dụng toán tử nối chuỗi có kiểu là kiểu chuỗi

Toán tử so sánh

- Toán tử ==
  - Thực hiện phép so sánh bằng
  - Ví dụ:

```
<?php
        $a = $_POST[“a”];
        $b = $_POST[“b”];
        $kq = “”;
        if ($a==$b)
                $kq = “a bằng b”;
?>
```

Toán tử so sánh

- Toán tử ===
  - Thực hiện phép so sánh bằng với các đối tượng có cùng kiểu dữ liệu
  - Ví dụ:

```
<?php
        $a = $_POST[“a”];
        $b = $_POST[“b”];
        $kq = “”;
        if ($a===$b)
                $kq = “ a bằng b và có cùng kiểu dữ liệu”;
?>
```

5. Tham chiếu

- Khái niệm
- Cách thức làm việc của tham chiếu

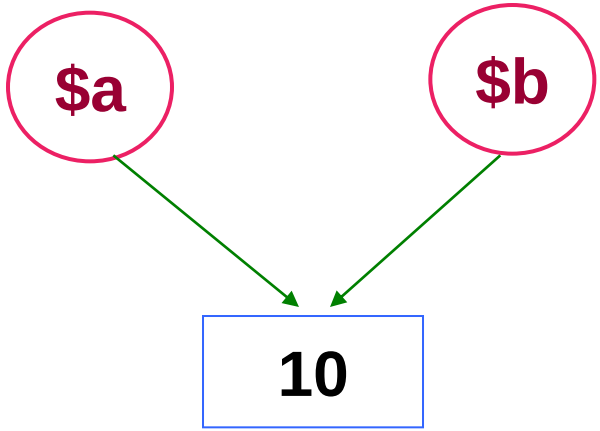
Khái niệm

- Trong PHP tham chiếu có nghĩa là lấy cùng một giá trị bằng nhiều tên biến khác nhau.
- Ký hiệu tham chiếu: **&**

Cách thức làm việc của tham chiếu

- Tham chiếu trong PHP cho phép tạo ra hai hay nhiều biến có cùng một nội dung.
  - Ví dụ:

```
<?php
        $a = 10;
        $b = &$a;
        echo $a; → 10
        echo $b; → 10
?>
```



6. Các hàm kiểm tra giá trị của biến

- Kiểm tra tồn tại isset()
- Kiểm tra giá trị rỗng empty()
- Kiểm tra trị kiểu số is_numeric()
- Kiểm tra kiểu dữ liệu của biến
- Xác định kiểu của biến gettype()

Kiểm tra tồn tại: isset()

- Ý nghĩa: dùng để kiểm tra biến có giá trị hay không
- Cú pháp: isset(<tên biến 1>, <tên biến 2>, …)
- Kết quả trả về:
  - TRUE: nếu tất cả các biến đều có giá trị
  - FALSE: nếu một biến bất kỳ không có giá trị

Kiểm tra tồn tại: isset()

- Ví dụ:

```
<?php
        if(isset($_POST[”ten_dang_nhap”]))
                echo ”Xin chào ”.$_POST[”ten_dang_nhap”];
        else
                echo ”Vui lòng nhập tên đăng nhập”;
?>
```

Kiểm tra giá trị rỗng: empty()

- Ý nghĩa: dùng để kiểm tra biến có giá trị rỗng hay không
- Cú pháp: empty(<tên biến>)
  - Kết quả trả về:
    - TRUE: nếu biến có giá trị rỗng
    - FALSE: nếu một biến có giá trị khác rỗng
  - Các giá trị được xem là rỗng:
    - “” (chuỗi rỗng), NULL
    - 0 (khi kiểu là integer), FALSE, array()
    - var $var (biến trong lớp được khai báo nhưng không có giá trị)

Kiểm tra giá trị rỗng: empty()

- Ví dụ:

```
<?php
if(empty($_POST[“ten_dang_nhap”]))
        {
                echo “Vui lòng nhập tên đăng nhập”;
                exit;
        }
else
        echo ”Xin chào ”.$_POST[”ten_dang_nhap”];
?>
```

Kiểm tra trị kiểu số: is_numeric()

- Ý nghĩa: dùng để kiểm tra biến có giá trị kiểu số hay không
- Cú pháp: is_numeric(<tên biến>)
  - Kết quả trả về:
    - TRUE: nếu biến có giá trị kiểu số
    - FALSE: nếu biến có giá trị không phải kiểu số

Kiểm tra trị kiểu số: is_numeric()

- Ví dụ:

```
<?php
        if(is_numeric($_POST[“so_luong”]))
        {
                $so_luong = $_POST[“so_luong”];
                $thanh_tien = $so_luong * $don_gia;
        }
        else
                echo “Số lượng phải là kiểu số!”;
?>
```

Kiểm tra kiểu dữ liệu của biến

- is_int() / is_long()
- is_string()
- is_double()
- is_array()
- is_bool()
- is_file()
- is_float()
- is_null()
- is_object()
  - Ý nghĩa: kiểm tra giá trị của biến có phải là kiểu integer - long – string – double hay không
  - Cú pháp chung: tên_hàm(<tên_biến>)

Kiểm tra kiểu dữ liệu của biến

- Ví dụ:

```
<?php
        $a = “15”;
        is_int($a); → 0 (FALSE)
        $b = 15;
        is_int($b); → 1 (TRUE)
        $a = “hello”;
        is_string($a); → 1 (TRUE)
        $b = 12.5;
        is_string($b); → 0 (FALSE)
        $x = 4.2135;
        is_double($x); → 1 (TRUE)
?>
```

Xác định kiểu của biến: gettype()

- Ý nghĩa; kiểm tra biến hoặc giá trị có kiểu dữ liệu nào: integer, string, double, array, object, class, …
- Cú pháp: gettype(<tên biến> hoặc <giá trị>)
- Kết quả trả về: kiểu của giá trị hay kiểu của biến

Kiểm tra kiểu dữ liệu của biến

- Ví dụ:

```
<?php
        $n = “Đây là chuỗi”;
        $a = 123;
        $b = 123.456;
        $mang = array(1,2,3);
        echo gettype($n); → string
        echo gettype($a); → integer
        echo gettype($b); → double
        echo gettype($mang); → array
?>
```

7. Cấu trúc điều khiển

- Cấu trúc rẽ nhánh
- Cấu trúc chọn lựa
- Cấu trúc lặp
- Sử dụng break và continue trong cấu trúc lặp

Cấu trúc rẽ nhánh if

- Dạng 1: if
  - Cú pháp:

```
if (điều kiện)
        {
                khối lệnh
        }
```

  - Ý nghĩa:

    Nếu điều kiện đúng thì

    thực hiện khối lệnh

  - Điều kiện là một biểu thức logic trả về đúng (TRUE) hoặc sai (FALSE)

Cấu trúc rẽ nhánh if

- Dạng 1: if
  - Ví dụ: Tìm số lớn

```
<?php
        $x = 10;
        $y = 5;
        if($x>=$y)
                $so_lon = $x;
        if($x<$y)
                $so_lon = $y;
?>
```

Cấu trúc rẽ nhánh if

- Dạng 2: if … else
  - Cú pháp:

```
if(điều kiện)
        {
                khối lệnh 1
        }
else
        {
                khối lệnh 2
        }
```

Cấu trúc rẽ nhánh if

- Dạng 2: if … else
  - Ý nghĩa:

    Nếu điều kiện đúng thì

    thực hiện khối lệnh 1

    Ngược lại thì

    thực hiện khối lệnh 2
  - Điều kiện là một biểu thức logic trả về đúng (TRUE) hoặc sai (FALSE)
  - Cấu trúc if có thể lồng nhau

Cấu trúc rẽ nhánh if

- Dạng 2: if … else
  - Ví dụ: Tìm số lớn

```
<?php
        $x = $_POST[“x”];
        $y = $_POST[“y”];
        if($x>$y)
                $so_lon = $x;
        else
                $so_lon = $y;
?>
```

Toán tử ?:

- Cú pháp:

  (điều kiện)?<kết quả khi điều kiện đúng>:<kết quả khi điều kiện sai>

  – Ý nghĩa: dùng để thay thế cho cấu trúc điều khiển if…else với một câu lệnh bên trong

  – Có thể lồng nhiều toán tử ?: với nhau

Toán tử ?:

- Ví dụ: Tìm số lớn nhất trong hai số

```
<?php
        $a = $_POST[“a”];
        $b=$_POST[“b”];
        $c = ($a>$b)?$a:$b;
        // Kết quả cũng tương tự như khi sử dụng if …else
?>
```

Toán tử ?:

- Ví dụ: Tìm số lớn nhất trong ba số

```
<?php
        $a=$_POST[“a”];
        $b=$_POST[“b”];
        $c=$_POST[“c”];
        $so_lon_nhat=($a>$b)?(($a>$c)?$a:$c):$b;
?>
```

Cấu trúc rẽ nhánh if

- **Dạng 3: if … elseif … else**
  - Cú pháp:

```
if(điều kiện 1)
        {
                khối lệnh 1
        }
elseif(điều kiện 2)
        {
                khối lệnh 2
        }
…
else
        {
                khối lệnh khi không thỏa các điều kiện trên
        }
```

Cấu trúc rẽ nhánh if

- Dạng 3: if … elseif … else
  - Ý nghĩa:

    Nếu điều kiện 1 đúng thì

    thực hiện khối lệnh 1

    Ngược nếu điều kiện 2 đúng thì

    thực hiện khối lệnh 2

    …

    Ngược lại thì

    thực hiện khối lệnh khi không thỏa các điều kiện trên
  - Điều kiện là một biểu thức logic trả về đúng (TRUE) hoặc sai (FALSE)

Cấu trúc rẽ nhánh if

- Dạng 1: if … elseif … else
  - Ví dụ: Tính thành tiền cho một hàng hóa: khi số lượng nhỏ hơn 10 đơn vị thì không giảm giá, từ 10->20 đơn vị thì giảm 5%, trên 20 đơn vị thì giảm 10%

```
<?php
        $so_luong = $_POST[“so_luong”];
        $don_gia = $_POST[“don_gia”];
        if ($so_luong <10)
                $thanh_tien = $so_luong * $don_gia;
        elseif ($so_luong >= 10 and $so_luong <=20)
                $thanh_tien = ($so_luong * $don_gia) * 0.95;
        else
                $thanh_tien = ($so_luong * $don_gia) * 0.9;
?>
```

Cấu trúc chọn lựa switch

- Cấu trúc switch cũng tương tự như cấu trúc if trong trường hợp có nhiều điều kiện
- Những trường hợp khác nhau có những cách xử lý khác nhau => dùng switch

Cấu trúc chọn lựa switch

- Dạng 1: mỗi trường hợp một cách xử lý khác nhau
  - Cú pháp

```
switch(biến điều kiện)
{
        case giá trị 1:
                khối lệnh 1
                break;
        case giá trị 2:
                khối lệnh 2
                break;
        ...
        [default: khối lệnh khi không thỏa tất cả các case trên]
```

Cấu trúc chọn lựa switch

- Dạng 1: mỗi trường hợp một cách xử lý khác nhau
  - Ý nghĩa

Xem xét biến điều kiện

trường hợp biến điều kiện = giá trị 1

thì thực hiện khối lệnh 1

trường hợp biến điều kiện = giá trị 2

thì thực hiện khối lệnh 2

…

[mặc định

thì thực hiện khối lệnh khi không thỏa tất cả các trường hợp trên]

Cấu trúc chọn lựa switch

- Dạng 1: mỗi trường hợp một cách xử lý khác nhau
  - Ví dụ: Đổi một số bất kỳ từ 0-9 thành chữ

```
<?php
        $so = $_POST[“so”];
        switch($so)
        {       case 1:
                        $chu = “Một”;
                        break;
                case 2:
                        $chu = “Hai”;
                        break;
                ...
                default:
                        $chu = “Đây không phải là số”;
        }
?>
```

Cấu trúc chọn lựa switch

- Dạng 2: mỗi nhóm các trường hợp có cùng một cách xử lý
  - Cú pháp

```
switch(biến điều kiện)
{
   case giá trị 1:
   case giá trị 2:
   ...
   khối lệnh
   ...
   [default: khối lệnh khi không thỏa tất cả các case trên]
}
```

Cấu trúc chọn lựa switch

- Dạng 2: mỗi nhóm các t/hợp có cùng một cách xử lý
  - Ý nghĩa

Xem xét biến điều kiện

trường hợp biến điều kiện = giá trị 1

hoặc biến điều kiện = giá trị 2

hoặc …

thì thực hiện khối lệnh

…

[mặc định

thì thực hiện khối lệnh khi không thỏa tất cả các trường hợp trên]

Cấu trúc chọn lựa switch

- Dạng 2: mỗi nhóm các t/hợp có cùng một cách xử lý
  - Ví dụ: In câu chúc theo thứ trong tuần

```
<?php
        switch($thu)
        {       case “Thứ hai”:
                case “Thứ ba”:
                case “Thứ tư”:
                case “Thứ năm”:
                case “Thứ sáu”:
                        echo “Chúc một ngày làm việc tốt!”;
                        break;
                case “Thứ bảy”:
                case “Chủ nhật”:
                        echo “Cuối tuần vui vẻ”;
                        break;
        }
?>
```

Cấu trúc chọn lựa switch

- So sánh với cách viết dùng if
  - Ví dụ: In câu chúc theo thứ trong tuần

```
<?php
        if ($thu==“Thứ hai” || $thu==“Thứ ba” || $thu==“Thứ tư” ||
                $thu==“Thứ năm” || $thu==“Thứ sáu”)
                        echo “Chúc một ngày làm việc tốt!”;
        elseif ($thu==“Thứ bảy” || $thu==“Chủ nhật”)
                        echo “Cuối tuần vui vẻ”;
?>
```

Cấu trúc lặp

- Cấu trúc lặp cho phép thực hiện nhiều lần một khối lệnh của chương trình khi thỏa điều kiện
- Gồm có các cấu trúc: for, foreach, while, do…while

Cấu trúc lặp for

- Công dụng:
  - for được sử dụng khi chúng ta biết trước số lần cần lặp, biến đếm chạy trong khoảng giới hạn của vòng lặp, và giá trị lặp.
- Cú pháp:

```
for($biến_đếm = giá trị khởi đầu của vòng lặp for; điều kiện giới hạn của vòng lặp for; giá trị lặp của vòng lặp for)
{
        khối lệnh
}
```

Cấu trúc lặp for

- Ý nghĩa:
  - Khối lệnh chỉ được thực hiện khi biến đếm vẫn còn nằm trong khoảng giới hạn của vòng lặp.
  - Vòng lặp sẽ kết thúc khi biến đếm vượt qua khoảng giới hạn của vòng lặp
  - Cần chỉ định giá trị lặp của biến đếm. Sau mỗi lần lặp biến đếm sẽ tăng lên (hoặc giảm đi)

Cấu trúc lặp for

- Ví dụ 1: tính tổng các số từ 1 đến 10

```
<?php
        $tong = 0;
        for($i=1; $i<=10;$i++)
        {
                $tong = $tong + $i;
        }
        echo $tong; → 55
?>
```

Cấu trúc lặp for

- Ví dụ 2: tính tổng các số lẻ từ 1 đến 10

```php
<?php
        $tong = 0;
        for($i=1; $i<=10;$i=$i+2)
        {
                $tong = $tong + $i;
        }
        echo $tong;
?>
```

Hoặc

```php
<?php
        $tong = 0;
        for($i=1; $i<=10;$i++)
        {
                if($i%2!=0)
                        $tong = $tong + $i;
        }
        echo $tong;
?>
```

Cấu trúc lặp foreach

- Công dụng:
  - foreach thường được dùng để duyệt tập hợp (mảng).
- Cú pháp duyệt giá trị các phần tử trong mảng:

```
foreach ($ten_mang as $gia_tri)
{
        khối lệnh
}
```

- Cú pháp duyệt cả khóa và giá trị các phần tử trong mảng:

```
foreach ($ten_mang as $tu_khoa => $gia_tri)
{
        khối lệnh
}
```

Cấu trúc lặp foreach

- Ý nghĩa:
  - Cấu trúc này sẽ duyệt từ phần tử đầu tiên đến phần tử cuối cùng của tập hợp (mảng) để thực hiện các xử lý tương ứng

Cấu trúc lặp foreach

- Ví dụ 1: duyệt và in nội dung của các phần tử trong mảng

```
<?php
        $mang_ten = array(“mai”, “lan”, “cúc”, “trúc”);
        foreach($mang_ten as $ten)
        {
                echo $ten . “  ”;
        }
        → mai lan cúc trúc
?>
```

Cấu trúc lặp foreach

- Ví dụ 2: duyệt và in nội dung của khóa và giá trị của các phần tử trong mảng

```
<?php
        $a = array("one" => 1, "two" => 2, "three" => 3, "four" => 4);
        foreach ($a as $tu_khoa => $gia_tri)
                {
                        echo "a[$tu_khoa] => $gia_tri ";
                }
?>
→ a[one] => 1 a[two] => 2 a[three] => 3 a[four] => 4
```

Cấu trúc lặp while

- Công dụng
  - Thực hiện lặp đi lặp lại một công việc nào đó khi thỏa điều kiện.
  - while được sử dụng khi không xác định được số lần lặp (số lần lặp phụ thuộc vào điều kiện tại thời điểm thực thi)
- Cú pháp

```
while(điều kiện)
{
        khối lệnh
}
```

Cấu trúc lặp while

- Ý nghĩa
  - Điều kiện là một biểu thức logic (có kết quả TRUE hoặc FALSE)
  - Sau mỗi lần thực hiện khối lệnh trong while, điều kiện sẽ được kiểm tra lại.

    Nếu giá trị True thì thực hiện lại vòng lặp

    Nếu giá trị False thì chấm dứt vòng lặp
  - Cấu trúc này kiểm tra điều kiện trước khi thực hiện các lệnh nên sẽ không thực hiện lần nào nếu ngay lần đầu tiên điều kiện có giá trị False.

Cấu trúc lặp while

- Ví dụ 1: tính tổng các số nguyên dương <10

```
<?php
        $i = 1;
        $tong = 0;
        while($i<10)
        {
                $tong = $tong +$i;
                $i= $i+1;
        }
        echo $tong; → 45
?>
```

Cấu trúc lặp while

- Ví dụ 2: tìm số nguyên lớn nhất trong khoảng từ 1-100 chia hết cho n (0<n<=100)

```
<?php
        $n= $_POST[“n”];
        $i=100;
        while($i>=1)
        {
                if($i%$n==0)
                {
                        $lon_nhat = $i;
                        break;
                }
                $i--;
        }
?>
```

Cải tiến → `while($i>=$n)`

Cấu trúc lặp do … while

- Công dụng
  - Thực hiện lặp đi lặp lại một công việc nào đó khi thỏa điều kiện.
  - do… while: việc kiểm tra điều kiện sẽ được thực hiện sau khi thực hiện khối lệnh do { …}
- Cú pháp

```
do
{
        khối lệnh
}
while(điều kiện);
```

Cấu trúc lặp do … while

- Ý nghĩa
  - Điều kiện là một biểu thức logic (có kết quả TRUE hoặc FALSE)
  - Sau mỗi lần thực hiện khối lệnh trong do … while, điều kiện sẽ được kiểm tra lại.

    Nếu giá trị True thì thực hiện lại vòng lặp

    Nếu giá trị False thì chấm dứt vòng lặp
  - Do điều kiện được kiểm tra sau khi thực hiện khối lệnh nên nếu ngay lần đầu điều kiện có giá trị FALSE thì khối lệnh cũng được thực hiện một lần.

Cấu trúc lặp do … while

- Ví dụ 1: tính tổng các số nguyên dương <10

```
<?php
        $i = 1;
        $tong = 0;
        do
                {
                        $tong = $tong +$i;
                        $i= $i+1;
                }
        while($i<10);
        echo $tong; → 45
?>
```

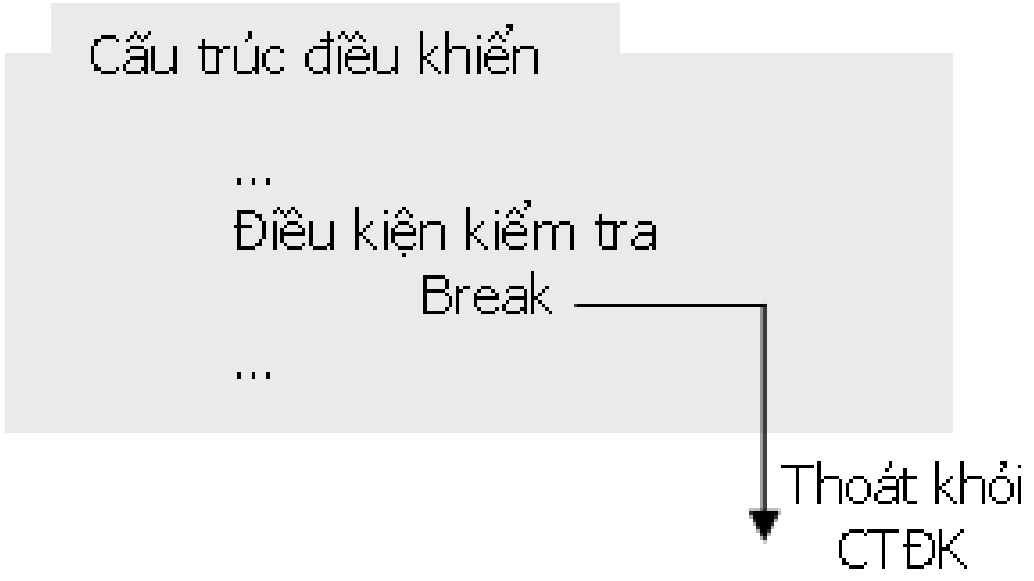
Cấu trúc lặp do … while

- Ví dụ 2:

```
<?php
        $so = 1;
        do
        {
                echo "Số = $so<br />"; → 1
                $so++;
         }
        while (($so > 2) && ($so < 4));
?>
```

Sử dụng break và continue trong cấu trúc lặp

- break
  - Công dụng: thoát khỏi cấu trúc điều khiển dựa trên kết quả của biểu thức luận lý kèm theo (điều kiện kiểm tra)



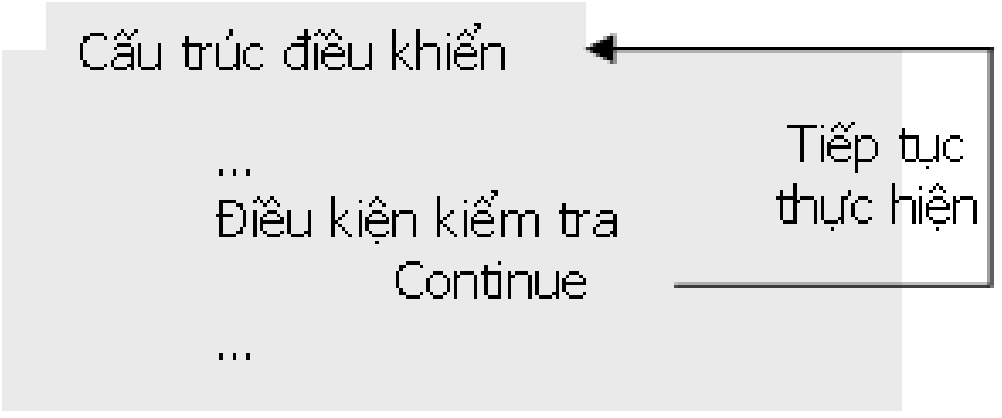
Sử dụng break và continue trong cấu trúc lặp

- break
  - Ví dụ: Kiểm tra số nguyên tố

```
<?php
        $so = $_POST[“so”];
        $kq = true;
        for($i=2; $i<$so; $i++)
        {       if($so%$i==0)
                        {
                                $kq = false;
                                break;
                        }
        }
?>
```

Sử dụng break và continue trong cấu trúc lặp

- continue
  - Công dụng: Khi gặp continue, các lệnh bên dưới continue tạm thời không thực hiện tiếp, khi đó con trỏ sẽ nhảy về đầu vòng lặp để kiểm tra giá trị của biểu thức điều kiện còn đúng hay không.
  - continue thường đi kèm với một biểu thức luận lý.



Sử dụng break và continue trong cấu trúc lặp

- continue
  - Ví dụ: Tính tổng các số lẻ từ đến 10

```
<?php
        $tong =0;
        for($i=1;$i<=10;$i++)
        {       if($i%2==0)
                        continue;
                $tong = $tong + $i;
        }
        echo $tong; → 25
?>
```

9. Hàm

- Hàm thư viện
- Hàm do người dùng tự định nghĩa

Hàm thư viện

- Là các hàm được PHP xây dựng sẵn, chỉ cần gọi khi sử dụng
- Có rất nhiều nhóm hàm trong thư viện: nhóm hàm chuỗi, toán học, thời gian, lịch, mảng, thư mục, tập tin, mail, CSDL, …
- Cách dùng: viết đúng tên hàm và truyền vào các giá trị cần thiết.

Hàm thư viện

- Ví dụ: dùng hàm round() để làm tròn một số, hàm date() để lấy ngày hiện tại có định dạng ngày tháng năm

```
<?php
        $so = 121.1542;
        round($so,2); → 121.15
        round($so,-1); → 120
        date("d/m/Y"); → 25/02/2008
?>
```

Hàm do người dùng tự định nghĩa

- Cách khai báo:

```
function Tên_hàm(Danh_sách_các_tham_số)
{
        khối lệnh bên trong hàm
        return giá_trị;
}
```

Hàm do người dùng tự định nghĩa

- Trong đó
  - Tên hàm: được sử dụng khi gọi hàm, tên hàm nên có ý nghĩa gợi nhớ
  - Danh sách các tham số: dùng để truyền dữ liệu bên ngoài vào, hàm có thể có hoặc không có tham số
  - Giá trị: là kết quả trả về của hàm. Hàm có thể có hoặc không có giá trị trả về

Hàm do người dùng tự định nghĩa

- Ví dụ: xây dựng hàm xuất câu chào

```
<?php
        function xuat_cau_chao()
        {
                echo “Chào mừng các bạn đến với PHP”;
        }
?>
```

Hàm do người dùng tự định nghĩa

- Sử dụng:

Tên_hàm(Danh_sách_các_giá_trị)

  - Tên hàm gọi sử dụng phải giống tên hàm đã xây dựng
  - Danh sách các giá trị: cung cấp các thông tin cho tham số của hàm. Với:
    - Số lượng các giá trị bằng số lượng các tham số của hàm
    - Thứ tự tương ứng theo thứ tự các tham số
    - Kiểu dữ liệu của giá trị phải tương ứng với kiểu của tham số
    - Nếu hàm không có giá trị truyền vào thì phía sau tên hàm cũng phải có cặp ()

Hàm do người dùng tự định nghĩa

- Ví dụ: gọi hàm xuất câu chào

```
<?php
        xuat_cau_chao();
?>
```

Hàm do người dùng tự định nghĩa

- Ví dụ: xây dựng hàm tính tổng hai số a và b với hai tham số là a, b. Hàm có kết quả trả về là tổng của a và b.

```
<?php
        function tinh_tong($a,$b)
        {
                $tong = 0;
                $tong = $a+$b;
                return $tong;
        }
?>
```

Hàm do người dùng tự định nghĩa

- Ví dụ: gọi hàm tính tổng

```
<?php
        $so_a = $_POST[“a”];
        $so_b = $_POST[“b”];
        $tong_a_b = tinh_tong($so_a,$so_b);
        echo “Tổng của hai số a và b là: ” . $tong_a_b;
?>
```

Hàm do người dùng tự định nghĩa

- Phân loại tham số của hàm: có hai loại
  - Tham trị: truyền tham số theo giá trị
  - Tham biến: truyền tham số theo địa chỉ

Hàm do người dùng tự định nghĩa

- Tham trị
  - Tham số truyền giá trị từ bên ngoài vào cho hàm.
  - Khi thay đổi giá trị của tham trị bên trong hàm thì giá trị của nó ở ngoài hàm vẫn không bị thay đổi.

Hàm do người dùng tự định nghĩa

- Tham trị
  - Ví dụ:

```
<?php
        function them_vao_chuoi($chuoi)
        {
                $chuoi .= "và chuỗi sau khi thêm.";
                return $chuoi;
        }
$chuoi_goc= "Đây là chuỗi gốc, ";
echo them_vao_chuoi($chuoi_goc); → "Đây là chuỗi gốc, và chuỗi sau khi thêm."
echo $chuoi_goc ; → "Đây là chuỗi gốc,  "
?>
```

Hàm do người dùng tự định nghĩa

- Tham biến:
  - Tham số truyền giá trị từ bên ngoài cho hàm và trả giá trị ở trong hàm ra bên ngoài.
  - Khi thay đổi giá trị của tham biến bên trong hàm thì giá trị của nó ở ngoài hàm cũng sẽ thay đổi sau khi chúng ta gọi hàm đã xây dựng.
  - Đối với tham biến chúng ta sẽ sử dụng cú pháp với ký tự & ở phía trước.

Hàm do người dùng tự định nghĩa

- Tham biến:
  - Cú pháp:

```
function Tên_hàm(   Tên_tham_biến, ...)
{
                    ...
                    Tên_tham_biến = giá trị;
                    return ...;
}
```

Hàm do người dùng tự định nghĩa

- Tham biến:
  - Ví dụ:

```
<?php
        function them_vao_chuoi(&$chuoi)
        {
                $chuoi .= "và chuỗi sau khi thêm.";
                return $chuoi;
        }
$chuoi_goc= "Đây là chuỗi gốc, ";
echo them_vao_chuoi($chuoi_goc); → "Đây là chuỗi gốc, và chuỗi sau khi thêm."
echo $chuoi_goc; →"Đây là chuỗi gốc, và chuỗi sau khi thêm."
?>
```

Hàm do người dùng tự định nghĩa

- Tham số tùy chọn
  - Là những tham số có thể được truyền giá trị hoặc không
  - Cho phép tạo sẵn các giá trị mặc định cho tham số
  - Những tham số này chỉ xuất hiện ở cuối danh sách các tham số

Hàm do người dùng tự định nghĩa

- Tham số tùy chọn:
  - Cú pháp:

```
function Tên_hàm(Danh sách các tham trị, tham biến, $tham_số_tùy_chọn=giá_trị)
{
        khối lệnh bên trong hàm
        return giá_trị;
}
```

Hàm do người dùng tự định nghĩa

- Tham số tùy chọn:
  - Ví dụ:

```
<?php
        function so_thich($nuoc_uong = "cà phê")
        {
                return "Tôi thích uống $nuoc_uong.";
        }
echo so_thich(); →“Tôi thích uống cà phê.”
echo so_thich("nước ép trái cây"); → “Tôi thích uống nước ép trái cây.”
?>
```

8. Xử lý lỗi

- Phân loại lỗi
- Dò và sửa lỗi thủ công
- Dùng try…catch để dò và sửa lỗi

Phân loại lỗi

- Lỗi cú pháp (syntax error)
  - Xuất hiện khi ta viết code.
  - Được thông báo khi ta thực thi trang.
  - Nguyên nhân: viết sai hoặc thiếu cú pháp.
- Lỗi thực thi
  - Xảy ra khi ta thực thi trang.
  - Khó xác định hơn lỗi cú pháp.
  - Nguyên nhân: Mở một tập tin đang tồn tại, chia cho 0, truy xuất bảng không tồn tại trong CSDL, …

Phân loại lỗi

- Lỗi luận lý
  - Xảy ra khi ta thực thi trang
  - Được thể hiện dưới những hình thức hoặc những kết quả không mong đợi.
  - Nguyên nhân: sai lầm trong thuật giải.

Dò và sửa lỗi thủ công

- Khi có lỗi phát sinh thì trang thực thi sẽ tự động thông báo lỗi
- Cách sửa lỗi:
  - Xem thông báo lỗi và dòng xảy ra lỗi
  - Mở code và sửa lỗi tại dòng đó (hoặc dòng lân cận)

Dùng try…catch để dò và sửa lỗi

- Try … catch: cho phép thử thực hiện một khối lệnh xem có bị lỗi hay không, nếu có sẽ bẫy và xử lý lỗi.
- Cấu trúc try … catch có hai khối:
  - Khối try: các câu lệnh có khả năng gây ra lỗi
  - Khối catch: các câu lệnh để bẫy và xử lý lỗi phát sinh trên khối try.
- Một lỗi xảy ra khi thực thi trang gọi là một Exception.
- Nếu dòng nào trong khối lệnh có khả năng tạo ra lỗi thì gọi trả về lỗi đó.

Dùng try…catch để dò và sửa lỗi

- Trong PHP lỗi sẽ không tự động trả về => gọi lỗi đó bằng cú pháp sau:

  throw new Exception(“Câu thông báo lỗi”, code);

- Với câu thông báo lỗi và mã lỗi (code) đều là các tham số tùy chọn

Dùng try…catch để dò và sửa lỗi

- Cú pháp

```
try
  {
      - khối lệnh có khả năng phát sinh lỗi
      - các lỗi
      throw new Exception("Câu thông báo lỗi", code);
  }
catch (Exception $e)
  {
      echo "<p>Lỗi: " . $e->getMessage() . "</p>"; //xuất lỗi
  }
```

Dùng try…catch để dò và sửa lỗi

- Lưu ý:
  - Một khối try có thể dùng một hay nhiều khối catch
  - Mỗi khối catch hiển thị một loại lỗi khác nhau

10. Các hàm toán học

- Giá trị tuyệt đối
- Pi
- Lũy thừa
- Phát sinh ngẫu nhiên
- Làm tròn
- Tính căn

Giá trị tuyệt đối abs()

- Kết quả trả về là giá trị tuyệt đối của một số
- Cú pháp:

abs(số)

Giá trị tuyệt đối abs()

- Ví dụ

```
<?php
echo abs(-4.2); → 4.2 (kiểu float/double)
echo abs(-5); → 5 (kiểu integer)
echo abs(4); → 4 (kiểu integer)
?>
```

Pi

- pi()
  - Kết quả trả về là hằng số PI trong toán học
  - Cú pháp:

    pi()
  - Ví dụ:

```
<?php
echo pi(); → 3.1415926535898
?>
```

Lũy thừa

- pow()
  - Kết quả trả về là phép tính lũy thừa của một số theo một số mũ chỉ định
  - Cú pháp:

    pow(số, lũy_thừa)

Lũy thừa

- Ví dụ

```
<?php
echo pow(2,5); → 32
?>
```

Phát sinh ngẫu nhiên

- rand()
  - Kết quả trả về là một giá trị số nguyên ngẫu nhiên bất kỳ (nằm trong khoảng từ 0 đến 32768)
  - Khi hàm rand() có hai tham số là giá trị nhỏ nhất (min) và giá trị lớn nhất (max) thì giá trị trả về sẽ là một giá trị số nằm trong khoảng từ giá trị nhỏ nhất đến giá trị lớn nhất.
  - Cú pháp:

    rand ([min, max])

Phát sinh ngẫu nhiên

- Ví dụ

```
<?php
echo rand(); → 10325
echo rand(); → 7774
echo rand(10, 100); → 57
?>
```

Làm tròn

- round()
  - Kết quả trả về là một số đã được làm tròn.
  - Nếu chỉ có một tham số là số được truyền vào thì kết quả trả về là một số nguyên.
  - Nếu có cả hai tham số là số và vị trí làm tròn thì kết quả trả về là một số được làm tròn dựa trên vị trí làm tròn.
- Cú pháp:

  round (số [, vị trí làm tròn])
  - Vị trí làm tròn là một số nguyên âm hoặc nguyên dương dùng để chỉ định vị trí muốn làm tròn, được tính từ vị trí dấu chấm thập phân

Làm tròn

- Ví dụ

```
<?php
echo round(3.4); → 3
echo round(3.5); → 4
echo round(577.545, 2); → 577.55
echo round(577.545,-1); → 580
?>
```

Tính căn

- sqrt()
  - Tính căn bậc hai của một số dương bất kỳ
  - Kết quả trả về là căn bậc hai của số
  - Cú pháp:

    sqrt(số)

Tính căn

- Ví dụ

```
<?php
echo sqrt(9); → 3
?>
```

11. Các hàm chuỗi

- Định dạng chuỗi
- So sánh chuỗi
- Tìm kiếm và thay thế
- Kết hợp và tách chuỗi

Định dạng chuỗi - Bỏ ký tự thừa

- ltrim()
  - Loại bỏ các ký tự thừa bên trái chuỗi.
  - Kết quả trả về là một chuỗi không có ký tự thừa bên trái chuỗi.
  - Cú pháp:

    ltrim(str [, charlist])

    - Tham số bắt buộc là chuỗi str
    - Tham số tùy chọn là danh sách các ký tự muốn loại bỏ bên trái chuỗi.

Định dạng chuỗi - Bỏ ký tự thừa

- Ví dụ

```
<?php
        $text = "\t\tXin chào các bạn :)    ...";
        echo ltrim($text);
        → "Xin chào các bạn :) ..."
?>
```

Định dạng chuỗi - Bỏ ký tự thừa

- rtrim(), chop()
  - Loại bỏ các ký tự thừa bên phải chuỗi.
  - Kết quả trả về là một chuỗi không có ký tự thừa bên phải chuỗi.
  - Cú pháp:

    rtrim(str [, charlist])

    chop(str [, charlist])

Định dạng chuỗi - Bỏ ký tự thừa

- Ví dụ

```
<?php
        $text = “\t\tXin chào các bạn :)    ...    ”;
        echo rtrim($text);
        → “\t\tXin chào các bạn :) ...”
        echo chop($text, “\t.”);
        → “Xin chào các bạn :)”
?>
```

Định dạng chuỗi - Bỏ ký tự thừa

- trim()
  - Loại bỏ các ký tự thừa bên phải và bên trái chuỗi.
  - Kết quả trả về là một chuỗi không có ký tự thừa bên phải và bên trái chuỗi.
  - Cú pháp:

    trim(str [, charlist])

Định dạng chuỗi - Bỏ ký tự thừa

- Ví dụ

```
<?php
        $text = “\t\tXin chào các bạn :)    ...    ”;
        echo trim($text);
        → “Xin chào các bạn :) ...”
        echo trim($text, “\t.”);
        → “Xin chào các bạn :)”
?>
```

Định dạng chuỗi - Bỏ ký tự thừa

- Lưu ý: nếu không có tham số thứ hai thì hàm ltrim(), rtrim(), chop() sẽ tự động loại bỏ các ký tự sau:
  - " " (ASCII 32 (0x20)), ký tự khoảng trắng.
  - "\t" (ASCII 9 (0x09)), ký tự tab.
  - "\n" (ASCII 10 (0x0A)), ký tự về đầu dòng mới.
  - "\r" (ASCII 13 (0x0D)), ký tự xuống dòng.
  - "\0" (ASCII 0 (0x00)), byte NULL.
  - "\x0B" (ASCII 11 (0x0B)), tab dọc.
- Ngoài ra, có thể liệt kê các ký tự muốn loại bỏ ở charlist

Định dạng chuỗi – Xuống dòng

- nl2br()
  - Khi trình bày dữ liệu trên trang Web, cần phải định dạng lại những ký tự “\n” trong cơ sở dữ liệu thành thẻ “
” để trình bày xuống hàng trên trình duyệt Web
  - Cú pháp:

    nl2br(str)

Định dạng chuỗi – Xuống dòng

- Ví dụ

```
<?php
        echo nl2br("Xin chào các bạn \n");
        → "Xin chào các bạn <br>"
?>
```

Định dạng chuỗi – Chuyển đổi kiểu chữ

- strtoupper(): chuyển đổi chữ thành chữ HOA
- strtolower(): chuyển đổi chữ thành chữ thường
- ucfirst(): viết hoa ký tự đầu tiên của chuỗi
- ucwords(): viết hoa ký tự đầu tiên của mỗi từ
- Cú pháp

```
strtoupper(str)
strtolower(str)
ucfirst(str)
ucwords(str)
```

Định dạng chuỗi – Chuyển đổi kiểu chữ

- Ví dụ

```
<?php
        echo strtoupper(“happy new year”);
        → “HAPPY NEW YEAR”
        echo strtolower(“HAPPY NEW YEAR”);
        → “happy new year”
        echo ucfirst(“happy new year”);
        → “Happy new year”
        echo ucwords(“happy new year”);
        → “Happy New Year”
?>
```

Định dạng chuỗi – Chiều dài chuỗi

- strlen()
  - Kết quả trả về là chiều dài của một chuỗi
  - Cú pháp

    strlen(str)
  - Ví dụ:

```
<?php
        echo strlen(“abcdef”); → 6
?>
```

Định dạng chuỗi – Đếm ký tự

- **count_chars()**
  - Kết quả trả về là một mảng với các phần tử có khóa là mã ASCII và giá trị là số lần xuất hiện của các ký tự trong chuỗi.
  - Thứ tự của các khóa được sắp tăng dần theo bảng mã ASCII
  - Cú pháp

    count_chars(str [, mode])

Định dạng chuỗi – Đếm ký tự

- Mode: là kiểu đếm (tham số tùy chọn)
- Mode = 0 (hoặc không sử dụng mode): liệt kê tất cả các ký tự ASCII từ 0 đến 255 và số lần xuất hiện.
- Mode = 1 : chỉ liệt kê danh sách có số lần xuất hiện lớn hơn 0
- Mode = 2: chỉ liệt kê danh sách có số lần xuất hiện bằng 0
- Mode = 3: chuỗi mới mà mỗi ký tự trong chuỗi gốc chỉ xuất hiện ở chuỗi mới 1 lần theo thứ tự tăng dần của bảng mã ASCII.
- Mode = 4: trả về một chuỗi mới với các ký tự theo thứ tự tăng dần theo bảng mã ASCII, mà các ký tự này không xuất hiện ở chuỗi gốc.

Định dạng chuỗi – Đếm ký tự

- Ví dụ: mode = 1

```
<?php
        $str = "Hello World!";
        print_r(count_chars($str,1));
        →
        Array
        (
        [32] => 1 [33] => 1 [72] => 1
        [87] => 1 [100] => 1 [101] => 1
        [108] => 3 [111] => 2 [114] => 1
        )
?>
```

Định dạng chuỗi – Đếm ký tự

- Ví dụ: mode = 3

```
<?php
        $str = "Hello World!";
        echo count_chars($str,3);
        → !HWdelor
?>
```

So sánh chuỗi – Có phân biệt HOA - thường

- strcmp()
  - So sánh hai chuỗi có phân biệt chữ HOA, chữ thường.
  - Kết quả trả về:
    - =0: nếu hai chuỗi bằng nhau
    - <0: nếu chuỗi str1 nhỏ hơn chuỗi str2
    - >0: nếu chuỗi str1 lớn hơn chuỗi str2
- Cú pháp

  strcmp(str1, str2)
- Lưu ý: Việc so sánh chuỗi dựa trên bảng mã ASCII của các ký tự

So sánh chuỗi – Có phân biệt HOA - thường

- Ví dụ:

```
<?php
        echo strcmp("hello", "Hello");
        → 1 (chuỗi thứ nhất lớn hơn chuỗi thứ hai) vì
        trong bảng mã ASCII ký tự 'h' có giá trị là 101
        lớn hơn ký tự 'H' có giá trị là 72, các ký tự còn
        lại trong chuỗi đều giống nhau
        echo strcmp("Hôm nay", "Thứ tư"); → -1
?>
```

So sánh chuỗi – Không phân biệt HOA - thường

- strcasecmp(), strnatcmp()
  - So sánh hai chuỗi không phân biệt chữ HOA, chữ thường.
  - Kết quả trả về:
    - =0: nếu hai chuỗi bằng nhau
    - <0: nếu chuỗi str1 nhỏ hơn chuỗi str2
    - >0: nếu chuỗi str1 lớn hơn chuỗi str2
- Cú pháp

strcasecmp(str1, str2)

strnatcmp(str1, str2)

So sánh chuỗi – Không phân biệt HOA - thường

– Ví dụ:

```
<?php
        echo strnatcmp(“hello”, “Hello”); → 0
?>
```

Tìm kiếm và thay thế - Tìm một chuỗi trong chuỗi

- strstr(), strchr()
  - Tìm một chuỗi trong chuỗi.
  - Kết quả trả về là một chuỗi con của chuỗi str1 được lấy từ vị trí xuất hiện đầu tiên của chuỗi str2 đến hết chuỗi str1 nếu tìm thấy chuỗi str2 trong chuỗi str1, nếu không tìm thấy chuỗi str2 trong chuỗi str1 thì trả về FALSE.
- Cú pháp

strstr(str1, str2)

strchr(str1, str2)

Tìm kiếm và thay thế - Tìm một chuỗi trong chuỗi

- Ví dụ:

```
<?php
        $email = “phuong@yahoo.com”;
        $domain = strstr($email, “@”);
        echo $domain; →“@yahoo.com”
?>
```

Tìm kiếm và thay thế - Tìm vị trí của chuỗi con

- strpos()
  - Kết quả trả về là vị trí tìm thấy của chuỗi str2 trong chuỗi str1, nếu trong chuỗi str1 có nhiều chuỗi str2 thì kết quả trả về sẽ là vị trí đầu tiên mà chuỗi str2 xuất hiện trong chuỗi str1. Nếu không tìm thấy chuỗi str2 trong chuỗi str1 thì kết quả trả về là FALSE
- Cú pháp

  strpos(str1, str2)
- Lưu ý: Để kiểm tra kết quả thì phải dùng toán tử === thay cho toán tử ==

Tìm kiếm và thay thế - Tìm vị trí của chuỗi con

- Ví dụ:

```
<?php
        $str1 = “abc def a”;
        $str2 = “g”;
        $vi_tri = strpos($str1, $str2);
        if($vi_tri === FALSE)
                echo “Không tìm thấy”;
        else
                echo “Vị trí tìm thấy : “. $vi_tri;
?>
```

Tìm kiếm và thay thế

- str_replace()
  - Tìm kiếm chuỗi str1 trong chuỗi str, nếu tìm thấy thì thay thế chuỗi str1 bằng chuỗi str2 trong chuỗi str.
- Cú pháp

  str_replace(str1, str2, str)

Tìm kiếm và thay thế

- Ví dụ

```
<?php
        echo str_replace(“em”, “bé”, “Hôm qua em đến trường”);
        → “Hôm qua bé đến trường”
?>
```

Kết hợp và tách chuỗi – Tách chuỗi

- explode()
  - Tách chuỗi str thành nhiều chuỗi con bằng cách chỉ định chuỗi tách separator và gán từng chuỗi con này vào các phần tử của mảng
- Cú pháp

  explode(separator, str)

Kết hợp và tách chuỗi – Tách chuỗi

- Ví dụ

```
<?php
        $chuoi = “happy new year”;
        $mang_chuoi = explode(“ ”, $chuoi);
        echo $mang_chuoi[0]; → “happy”
        echo $mang_chuoi[1]; → “new”
        echo $mang_chuoi[2]; → “year”
?>
```

Kết hợp và tách chuỗi – Kết hợp chuỗi

- implode()
  - Kết hợp các phần tử của mảng thành một chuỗi và các phần tử khi ráp thành chuỗi sẽ cách nhau bằng chỉ định cách separator
- Cú pháp

  implode(separator, array)

Kết hợp và tách chuỗi – Kết hợp chuỗi

- Ví dụ

```
<?php
        $mang_chuoi = array(“happy”, “new”, “year”);
        $chuoi = implode(“ ”, $mang_chuoi);
        echo $chuoi; → “happy new year”
?>
```

Đổi ký tự

- chr()
  - Hàm chr() có tham số truyền vào là một số trong bảng mã ASCII và có kết quả trả về là một ký tự tương ứng với số đó.
  - Bảng mã ASCII quy định mỗi ký tự sẽ có một số tương ứng. Phạm vi các số từ 0 đến 255.
- Cú pháp

  chr(số)

Đổi ký tự

- Ví dụ

```
<?php
        echo chr(52); → 4
        echo chr(052); → *
        echo chr(0x52); → R
?>
```

12. Các hàm thời gian

- Kiểm tra ngày hợp lệ
- Định dạng ngày
- Lấy các giá trị của ngày hiện tại
- Lấy thời gian hiện tại
- Chuyển đổi ngày
- Chuyển chuỗi thời gian thành giá trị thời gian
- Đổi thời gian sang đơn vị giây
- Định dạng thời gian thành số nguyên

Kiểm tra ngày hợp lệ

- checkdate()
  - Kiểm tra ngày với tháng, ngày, năm được truyền vào, nếu ngày hợp lệ thì kết quả trả về là TRUE, ngược lại trả về FALSE
- Cú pháp:

  checkdate(tháng, ngày, năm)

Kiểm tra ngày hợp lệ

- Ví dụ:

```
<?php
        echo checkdate(11,30,2007); → TRUE
?>
```

Định dạng ngày

- date()
  - Kết quả trả về là ngày hiện tại sau khi đã được định dạng
- Cú pháp:

  date(chuỗi định dạng)

Định dạng ngày

- Ví dụ:

```
<?php
        echo date(“d/m/Y h:m:s”);
        → 28-11-2007 11:11:07
        echo date(“D-M-Y”);
        → Thu–Dec-2007
?>
```

Lấy các giá trị của ngày hiện tại

- getdate()
  - Kết quả trả về là một mảng gồm 10 phần tử (có khóa dạng chuỗi) chứa các thông tin cần thiết của ngày hiện tại.

[seconds] – giây, [minutes] - phút, [hours] – giờ

[mday] – ngày trong tháng, [wday] – ngày trong tuần

[mon] – tháng (dạng số), [year] – năm, [yday] – ngày trong năm

[weekday] – thứ trong tuần, [month] – tháng (dạng chữ)

Lấy các giá trị của ngày hiện tại

- Cú pháp:

  getdate()

- Ví dụ:

```
<?php
        $today = getdate();
        print_r($today);
?>
        → Array
        (       [seconds] => 18 [minutes] => 18 [hours] => 13
                [mday] => 28 [wday] => 3 [mon] => 11 [year] => 2007
                [yday] => 331 [weekday] => Wednesday
                [month] => November
        )
```

Lấy các giá trị của ngày hiện tại

- localtime()
  - Kết quả trả về là một mảng (có khóa dạng số) trong đó mỗi phần tử của mảng là một thành phần của ngày hiện tại (giờ, phút, giây, ngày, …). Thứ tự các phần tử như sau:
    - [0] - giây, [1] – phút, [2] - giờ, [3] - ngày trong tháng
    - [4] - tháng trong năm (với 0 là tháng 1 – January)
    - [5] - Năm bắt đầu từ năm 1900, [6] - Ngày trong tuần
    - [7] - ngày trong năm, [8] - 0 – nếu là ngày và 1 – nếu là đêm
  - Lưu ý: tháng bắt đầu từ 0 (tháng 1) đến 11 (tháng 12) và ngày trong tuần bắt đầu từ 0 (Chủ nhật) đến 6 (Thứ bảy)

Lấy các giá trị của ngày hiện tại

- Cú pháp:

  localtime()

  – Ví dụ:

```
<?php
        $today = localtime();
        print_r($today);
?>
        → Array
        (
                [0] => 13 [1] => 26  [2] => 13
                [3] => 28 [4] => 10  [5] => 107
                [6] => 3  [7] => 331 [8] => 0
        )
```

Lấy thời gian hiện tại

- time()
  - Kết quả trả về là thời gian hiện tại được đo lường theo giá trị giây (tính từ 1/1/1970 00:00:00 GMT)
- Cú pháp:

  time()

Lấy thời gian hiện tại

- Ví dụ

```
<?php
        $tuan_sau = time() + (7 * 24 * 60 * 60);
        // 7 days; 24 hours; 60 mins; 60secs
        echo ”Hiện tại:”. date(”Y-m-d”) ."<br>";
        → Hiện tại: 2007-11-28
        echo ”Tuần sau: ”. date(”Y-m-d”, $tuan_sau) ."<br>";
        → Tuần sau: 2007-12-05
?>
```

Chuyển đổi ngày

- cal_to_ jd()
  - Chuyển đổi một ngày sang giá trị đếm số ngày Julian (sử dụng khá nhiều khi làm việc với các định dạng lịch khác nhau).
  - Chuẩn của nó dựa trên việc đếm ngày Julian (đếm ngày theo Julian là đếm số ngày từ ngày bắt đầu January 1, 4713 B.C đến ngày được đưa vào để đếm).
  - Sử dụng để đếm số ngày Julian dựa trên ngày của một hệ thống lịch được chọn.
  - Số ngày = ngày được truyền vào trong hàm cal_to_jd - January 1, 4713 B.C (January 1, 4713 B.C là ngày bắt đầu)

Chuyển đổi ngày

- Cú pháp:

  cal_to_jd(loại lịch ,tháng, ngày, năm)

  Loại lịch có thể là:

  - CAL_GREGORIAN (lịch châu Âu – thường dùng)
  - CAL_JULIAN (lịch Julian)
  - CAL_JEWISH (lịch Do Thái)
  - CAL_FRENCH (lịch Pháp)

Chuyển đổi ngày

- Ví dụ

```
<?php
        $jd = cal_to_jd(CAL_GREGORIAN, 12, 4, 2007);
        echo "<br> jd: ". $jd; → 2454439

?>
```

Chuyển chuỗi thời gian thành giá trị thời gian

- strtotime()
  - Kết quả trả về là tổng số giây của thời gian nếu chuỗi thời gian đúng, ngược lại sẽ trả về FALSE
  - Tổng số giây = thời gian được truyền vào trong hàm strtotime() - January 1 1970 00:00:00 (vì thời gian bắt đầu được tính từ: January 1 1970 00:00:00)
- Cú pháp:

  strtotime(chuỗi thời gian)

Chuyển chuỗi thời gian thành giá trị thời gian

- Ví dụ

```
<?php
        echo strtotime("now");
        → 1.204.795.981
        // Với now = 06-03-2008 16:33:01
?>
```

Đổi thời gian sang đơn vị giây

- mktime()
  - Kết quả trả về là tổng số giây của thời gian nếu các tham số thời gian được truyền vào phù hợp, ngược lại sẽ trả về FALSE
  - Tổng số giây = thời gian được truyền vào trong hàm mktime() - January 1 1970 00:00:00 (vì thời gian bắt đầu là: January 1 1970 00:00:00)
- Cú pháp:

  mktime ([hour [, minute [, second [, month [, day [, year [, is_dst]]]]]]])

Đổi thời gian sang đơn vị giây

- Ví dụ

```
<?php
        echo mktime(0, 0, 0, 1, 1, 1998);
        → 883.587.600
        echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1998));
        → Jan-01-1998
        echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 98));
        → Jan-01-1998
?>
```

Định dạng thời gian thành số nguyên

- idate()
  - Kết quả trả về là thời gian hiện tại có kiểu số nguyên căn cứ vào ký tự định dạng
- Cú pháp:

  idate(ký tự định dạng)

  Các ký tự định dạng:
  - d - ngày trong tháng, h - giờ (Định dạng 12 giờ), H - giờ (Định dạng 24 giờ), i – phút, L - trả về 1 nếu là năm nhuận, ngược lại trả về 0, m – tháng, s – giây, w - thứ trong tuần (chủ nhật (Sunday) =0), y - năm (1 hoặc 2 ký số), Y - năm (4 ký số), z - ngày trong năm, …

Định dạng thời gian thành số nguyên

- Ví dụ

```
<?php
        // in năm đầy đủ (4 ký tự)
        echo idate('Y');
        → 2008
        echo idate('L');
        → 1 (vì năm 2008 là năm nhuận)
?>
```