KHÁI NIỆM

1. PHP LÀ GÌ?

- PHP viết tắt của **Hypertext Preprocessor.**
- PHP được sử dụng rộng rãi, là một ngôn ngữ kịch bản (scripting language) mã nguồn mở.
- Các tập lệnh PHP được thực thi trên máy chủ.

2. FILE PHP

- Tệp **PHP** có thể chứa **văn bản, HTML, CSS, JavaScript** và **mã PHP**.
- Mã **PHP** được thực thi trên máy chủ, và kết quả trả về trình duyệt dưới dạng **HTML thuần**.
- Tệp **PHP** có phần mở rộng là **".php"**.

3. ỨNG DỤNG CỦA PHP

- **PHP** có thể tạo nội dung trang web động.
- **PHP** có thể tạo, mở, đọc, ghi, xóa và đóng tệp trên máy chủ.
- **PHP** có thể thu thập dữ liệu từ biểu mẫu.
- **PHP** có thể gửi và nhận cookie.
- **PHP** có thể thêm, xóa, sửa đổi dữ liệu trong cơ sở dữ liệu.
- **PHP** có thể được sử dụng để kiểm soát quyền truy cập của người dùng.
- **PHP** có thể mã hóa dữ liệu.

Với **PHP**, bạn không chỉ giới hạn ở việc xuất ra **HTML**. Bạn có thể tạo **hình ảnh, tệp PDF** hoặc bất kỳ dạng văn bản nào khác, chẳng hạn như **XHTML** và **XML**.

4. TẠI SAO NÊN CHỌN PHP

- **PHP** có thể chạy trên nhiều nền tảng khác nhau (**Windows, Linux, Unix, Mac OS X**, v.v.).
- **PHP** tương thích với hầu hết các máy chủ web hiện nay (**Apache, IIS**, v.v.).
- **PHP** hỗ trợ nhiều hệ quản trị cơ sở dữ liệu khác nhau.
- **PHP** hoàn toàn miễn phí. Bạn có thể tải về từ trang chính thức: www.php.net.

- **PHP** dễ học và chạy hiệu quả trên máy chủ.

5. CÚ PHÁP PHP

Có thể đặt tập lệnh PHP ở bất kỳ vị trí nào trong tài liệu.

Một tập lệnh PHP bắt đầu <?php và kết thúc bằng ?>:

```php
<?php
// PHP code goes here
?>
```

Phần mở rộng tệp mặc định cho tệp PHP là " .php".

Một tệp PHP thường chứa các thẻ HTML và một số mã lệnh PHP.

6. KHAI BÁO BIẾN PHP

Trong PHP, một biến bắt đầu bằng dấu $, theo sau là tên của biến:

```php
$x = 5;
$y = "John";
```

Lưu ý: Không giống như các ngôn ngữ lập trình khác, PHP không có lệnh để khai báo biến. Biến được tạo ra ngay khi bạn gán giá trị cho nó lần đầu tiên.

Quy tắc cho biến PHP:

- Một biến bắt đầu bằng $dấu, theo sau là tên của biến
- Tên biến phải bắt đầu bằng một chữ cái hoặc ký tự gạch dưới
- Tên biến không thể bắt đầu bằng số
- Tên biến chỉ có thể chứa các ký tự chữ và số và dấu gạch dưới (Az, 0-9 và _)
- Tên biến phân biệt chữ hoa chữ thường ($age và $AGE là hai biến khác nhau)

1) Phạm vi biến PHP

Trong PHP, biến có thể được khai báo ở bất cứ đâu trong tập lệnh.

Phạm vi của một biến là phần của tập lệnh mà biến đó có thể được tham chiếu/sử dụng.

PHP có ba phạm vi biến khác nhau:

- Local
- global
- static (tĩnh)

Một biến được khai báo bên ngoài một hàm có **PHẠM VI GLOBAL** và chỉ có thể được truy cập bên ngoài một hàm:

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

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

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

Output:

Variable x inside function is:

Variable x outside function is: 5

Từ khóa **global** được sử dụng để truy cập biến toàn cục từ bên trong một hàm.

Để thực hiện việc này, hãy sử dụng từ khóa **global** trước các biến (bên trong hàm):

```php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
```

PHP cũng lưu trữ tất cả các biến toàn cầu trong một mảng có tên

$Globals [index]. index giữ tên của biến. Mảng này cũng có thể truy cập từ bên trong các hàm và có thể được sử dụng để cập nhật các biến toàn cầu trực tiếp.

```php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
```

Một biến được khai báo trong một hàm có **PHẠM VI LOCAL** và chỉ có thể được truy cập trong hàm đó:

Biến có phạm vi cục bộ:

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

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

Bạn có thể có các biến cục bộ có cùng tên trong các hàm khác nhau, vì biến cục bộ chỉ được nhận dạng bởi hàm mà chúng được khai báo.

Thông thường, khi một hàm hoàn thành/thực thi, tất cả các biến của nó sẽ bị xóa. Tuy nhiên, đôi khi chúng ta muốn một biến cục bộ KHÔNG bị xóa. Chúng ta cần nó cho một công việc khác.

Để thực hiện việc này, hãy sử dụng **từ khóa Static** khi bạn khai báo biến lần đầu tiên:

```php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
myTest();
myTest();
```

output:

0

1

2

Sau đó, mỗi lần hàm được gọi, biến đó vẫn sẽ giữ nguyên thông tin chứa trong lần cuối cùng hàm được gọi.

Lưu ý: Biến vẫn là biến cục bộ của hàm.

2) kiểu dữ liệu biến PHP

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

cách lấy kiểu dữ liệu: sử dụng hàm **var_dump()**

```php
$x = 5;
var_dump($x);
```

sự khác biệt giữa dấu nháy đơn và dấu ngoặc kép:

Chuỗi ký tự được trích dẫn kép thực hiện các thao tác cho các ký tự đặc biệt:

```php
$x = "John";
echo "Hello $x";
```

output: Hello John

Chuỗi được trích dẫn đơn không thực hiện các hành động như vậy, nó trả về chuỗi như cách nó được viết, với tên biến:

Chuỗi ký tự được trích dẫn đơn trả về chuỗi nguyên bản:

```php
$x = "John";
echo 'Hello $x';
```

output: Hello $x

cách lấy độ dài của một chuỗi: sử dụng Hàm PHP **strlen()**

```php
echo strlen("Hello world!");
```

cách đếm số từ trong một chuỗi: sử dụng Hàm PHP **str_word_count()**

```php
echo str_word_count("Hello world!");
```

Hàm PHP **strpos()** tìm kiếm một đoạn văn bản cụ thể trong một chuỗi. Nếu tìm thấy kết quả khớp, hàm sẽ trả về vị trí ký tự của kết quả khớp đầu tiên. Nếu không tìm thấy kết quả khớp, hàm sẽ trả về FALSE.

ví dụ: tìm kiếm “World” trong chuỗi “Hello World!”

```php
echo strpos("Hello world!", "world");
```

output: 6

một số hàm liên quan tới chỉnh sửa chuỗi:
https://www.w3schools.com/php/php_string_modify.asp

cách nối chuỗi A và chuỗi B: sử dụng toán tử **.**

```php
$x = "Hello";
$y = "World";
$z = $x . $y;
echo $z;
```

hoặc sử dụng dấu ngoặc kép, cũng đưa ra kết quả tương tự:

```php
$x = "Hello";
$y = "World";
$z = "$x $y";
echo $z;
```

cách cắt chuỗi: sử dụng **substr()**, trả về một lượng các ký tự bằng cách nêu vị trí và số lượng ký tự trả về từ vị trí đó, ví dụ:

```php
$x = "Hello World!";
echo substr($x, 6, 5);
```

output: World!

nếu không ghi số lượng ký tự trả về, hàm sẽ lấy ký tự từ vị trí bắt đầu cho đến ký tự cuối cùng của chuỗi.

```php
$x = "Hello World!";
echo substr($x, 6);
```

output: World!

cắt chuỗi từ phía cuối chuỗi: sử dụng chỉ số âm. ký tự đầu tiên có chỉ số 0, và ký tự cuối cùng có chỉ số -1.

Lấy 3 ký tự, bắt đầu từ chữ "o" trong “Hello World!” (chỉ mục -5):

```php
$x = "Hello World!";
echo substr($x, -5, 3);
```

output: orl

Chiều dài âm
Sử dụng độ dài âm để chỉ định số ký tự cần bỏ qua, bắt đầu từ cuối chuỗi:
Từ chuỗi "Xin chào, bạn khỏe không?", lấy các ký tự bắt đầu từ chỉ mục 5 và tiếp tục cho đến khi bạn đạt đến ký tự thứ 3 tính từ cuối (chỉ mục -3). output kết thúc bằng "ow are y":

```php
$x = "Hi, how are you?";
echo substr($x, 5, -3);
```

chuyển đổi kiểu dữ liệu:

- **(string)** - Chuyển đổi sang kiểu dữ liệu String
- **(int)** - Chuyển đổi sang kiểu dữ liệu Integer
- **(float)** - Chuyển đổi sang kiểu dữ liệu Float
- **(bool)** - Chuyển đổi sang kiểu dữ liệu Boolean
- **(array)** - Chuyển đổi sang kiểu dữ liệu Mảng
- **(object)** - Chuyển đổi sang kiểu dữ liệu Object
- **(unset)** - Chuyển đổi sang kiểu dữ liệu NULL

ví dụ:

```php
$a = 5;       // Integer
$b = 5.34;    // Float
$c = "hello"; // String
$d = true;    // Boolean
$e = NULL;    // NULL

$a = (string) $a;
$b = (string) $b;
$c = (string) $c;
$d = (string) $d;
$e = (string) $e;

//To verify the type of any object in PHP, use the var_dump() function:
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
var_dump($e);
```

7. HẰNG SỐ PHP

- Hằng số giống như biến, ngoại trừ việc một khi đã được định nghĩa thì chúng không thể thay đổi hoặc hủy định nghĩa.
- Hằng số là một định danh (tên) cho một giá trị đơn giản. Giá trị không thể thay đổi trong quá trình thực hiện tập lệnh.
- Tên hằng hợp lệ bắt đầu bằng một chữ cái hoặc dấu gạch dưới (không có dấu $ trước tên hằng).

Lưu ý: Không giống như biến, hằng số tự động mang tính toàn cục trong toàn bộ tập lệnh.

Để tạo hằng số, hãy sử dụng hàm **define()**

cú pháp:

```php
define(name, value);
```

name : Chỉ định tên của hằng số
value : Chỉ định giá trị của hằng số

hoặc chúng ta có thể sử dụng hàm **const** để tạo hằng số:

```php
const MYCAR = "Volvo";
echo MYCAR;
```

const so với **define()**

- **const** không thể được tạo bên trong phạm vi khối khác, như bên trong một hàm hoặc bên trong một câu lệnh **if**.
- **define** có thể được tạo bên trong phạm vi khối khác.

tạo một mảng hằng số bằng define():

```php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
```

Các hằng số tự động mang tính toàn cục và có thể được sử dụng trong toàn bộ tập lệnh. Ví dụ này sử dụng hằng số bên trong hàm, ngay cả khi nó được định nghĩa bên ngoài hàm:

```php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
  echo GREETING;
}

myTest();
```

hằng số ma thuật:

PHP có chín hằng số được xác định trước, có thể thay đổi giá trị tùy thuộc vào nơi chúng được sử dụng, còn được gọi là "hằng số ma thuật".

Các hằng số ma thuật này được viết bằng dấu gạch dưới kép ở đầu và cuối, ngoại trừ hằng số **ClassName::class**.

Constant	Description	
__CLASS__	If used inside a class, the class name is returned.	Try it »
__DIR__	The directory of the file.	Try it »
__FILE__	The file name including the full path.	Try it »
__FUNCTION__	If inside a function, the function name is returned.	Try it »
__LINE__	The current line number.	Try it »
__METHOD__	If used inside a function that belongs to a class, both class and function name is returned.	Try it »
__NAMESPACE__	If used inside a namespace, the name of the namespace is returned.	Try it »
__TRAIT__	If used inside a trait, the trait name is returned.	Try it »
ClassName::class	Returns the name of the specified class and the name of the namespace, if any.	Try it »

ví dụ __CLASS__

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

<?php
class Fruits {
  public function myValue(){
    return __CLASS__;
  }
}
$kiwi = new Fruits();
echo $kiwi->myValue();
?>

</body>
</html>
```

output: Fruits

link: https://www.w3schools.com/php/php_magic_constants.asp

8. IN RA MÀN HÌNH

Với PHP, có hai cách cơ bản để lấy kết quả: **echo** và **print**.

- sự giống nhau: đều sử dụng để in output ra màn hình
- sự khác biệt:
  + **echo:** không có giá trị trả về và có thể lấy về tham số, nhanh hơn **print** một chút.

+ **print:** có giá trị trả về là 1 nên có thể được sử dụng trong biểu thức, chỉ có thể lấy một đối số.

1) Echo

Câu lệnh này **echo** có thể được sử dụng có hoặc không có dấu ngoặc đơn: echo hoặc **echo()**.

```
echo "Hello";
//same as:
echo("Hello");
```

Ví dụ sau đây cho thấy cách xuất văn bản và biến bằng **echo** câu lệnh:

```
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo "<h2>$txt1</h2>";
echo "<p>Study PHP at $txt2</p>";
```

Chuỗi được bao quanh bởi dấu ngoặc kép, nhưng có sự khác biệt giữa dấu ngoặc đơn và dấu ngoặc kép trong PHP.

Khi sử dụng dấu ngoặc kép, các biến có thể được chèn vào chuỗi như trong ví dụ trên.

Khi sử dụng dấu ngoặc đơn, các biến phải được chèn bằng toán tử "**.**", như sau:

```
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo '<h2>' . $txt1 . '</h2>';
echo '<p>Study PHP at ' . $txt2 . '</p>';
```

2) print

Câu lệnh **print** có thể được sử dụng có hoặc không có dấu ngoặc đơn: **print** hoặc **print()**.

```
print "Hello";
//same as:
print("Hello");
```

Ví dụ sau đây cho thấy cách xuất văn bản bằng **print** lệnh (lưu ý rằng văn bản có thể chứa mã đánh dấu HTML):

```
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
```

9. TOÁN TỬ PHP

Toán tử được sử dụng để thực hiện các thao tác trên biến và giá trị.

PHP chia các toán tử thành các nhóm sau:

- Toán tử số học
- Toán tử gán
- Toán tử so sánh
- Toán tử tăng/giảm
- Toán tử logic
- Toán tử chuỗi
- Toán tử mảng
- Toán tử gán có điều kiện

Operator	Name	Example	Result	Try it
+	Addition	$x + $y	Sum of $x and $y	Try it »
-	Subtraction	$x - $y	Difference of $x and $y	Try it »
*	Multiplication	$x * $y	Product of $x and $y	Try it »
/	Division	$x / $y	Quotient of $x and $y	Try it »
%	Modulus	$x % $y	Remainder of $x divided by $y	Try it »
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power	Try it »

link: https://www.w3schools.com/php/php_operators.asp

toán tử so sánh

Operator	Name	Example	Result	Try it
==	Equal	$x == $y	Returns true if $x is equal to $y	Try it »
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	Try it »
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	Try it »
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y	Try it »
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type	Try it »
>	Greater than	$x > $y	Returns true if $x is greater than $y	Try it »
<	Less than	$x < $y	Returns true if $x is less than $y	Try it »
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y	Try it »
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y	Try it »
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.	Try it »

ví dụ:

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

<?php
$x = 5;
$y = 10;

echo ($x <=> $y); // returns -1 because $x is less than $y
echo "<br>";

$x = 10;
$y = 10;

echo ($x <=> $y); // returns 0 because values are equal
echo "<br>";

$x = 15;
$y = 10;

echo ($x <=> $y); // returns +1 because $x is greater than $y
?>

</body>
</html>
```

toán tử tăng/ giảm

Operator	Same as...	Description	Try it
++$x	Pre-increment	Increments $x by one, then returns $x	Try it »
$x++	Post-increment	Returns $x, then increments $x by one	Try it »
--$x	Pre-decrement	Decrements $x by one, then returns $x	Try it »
$x--	Post-decrement	Returns $x, then decrements $x by one	Try it »

toán tử logic

Operator	Name	Example	Result	Try it
and	And	$x and $y	True if both $x and $y are true	Try it »
or	Or	$x or $y	True if either $x or $y is true	Try it »
xor	Xor	$x xor $y	True if either $x or $y is true, but not both	Try it »
&&	And	$x && $y	True if both $x and $y are true	Try it »
||	Or	$x || $y	True if either $x or $y is true	Try it »
!	Not	!$x	True if $x is not true	Try it »

toán tử chuỗi

Operator	Name	Example	Result	Try it
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2	Try it »
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1	Try it »

ví dụ:

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

<?php
$txt1 = "Hello";
$txt2 = " world!";
$txt1 .= $txt2;
echo $txt1;
?>

</body>
</html>
```

toán tử mảng:

Operator	Name	Example	Result	Try it
+	Union	$x + $y	Union of $x and $y	Try it »
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs	Try it »
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types	Try it »
!=	Inequality	$x != $y	Returns true if $x is not equal to $y	Try it »
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y	Try it »
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y	Try it »

ví dụ:

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

<?php
$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");

var_dump($x == $y);
?>

</body>
</html>
```

output: bool(false)

toán tử gán có điều kiện

Các toán tử gán có điều kiện của PHP được sử dụng để đặt giá trị tùy thuộc vào các điều kiện:

Operator	Name	Example	Result	Try it
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE	Try it »
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7	Try it »

10. CÂU ĐIỀU KIỆN

Trong PHP chúng ta có các câu lệnh điều kiện sau:

- **if** - thực thi một số mã nếu một điều kiện là đúng
- **if...else** - thực thi một số mã nếu điều kiện là đúng và một mã khác nếu điều kiện đó là sai
- **if...else if...else** - thực thi các mã khác nhau cho hơn hai điều kiện
- **switch** - chọn một trong nhiều khối mã để thực thi

cách sử dụng tương tự C++

11. VÒNG LẶP

Trong PHP, chúng ta có các kiểu vòng lặp sau:

- **while** - lặp qua một khối mã miễn là điều kiện được chỉ định là đúng
- **do...while** - lặp qua một khối mã một lần, sau đó lặp lại vòng lặp miễn là điều kiện được chỉ định là đúng
- **for** - lặp qua một khối mã với số lần xác định
- **foreach** - lặp qua một khối mã cho mỗi phần tử trong một mảng

…

foreach:

Công dụng phổ biến nhất của vòng lặp foreach là lặp qua các phần tử của một mảng.

ví dụ:

```
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {
  echo "$x <br>";
}
```

Với mỗi lần lặp, giá trị của phần tử mảng hiện tại được gán cho biến $x. Lặp lại tiếp tục cho đến khi đạt đến phần tử mảng cuối cùng.

khóa và giá trị:

Mảng ở trên là mảng có chỉ mục , trong đó phần tử đầu tiên có khóa là 0, phần tử thứ hai có khóa là 1, v.v.

Mảng kết hợp thì khác, mảng kết hợp sử dụng các khóa được đặt tên mà bạn gán cho chúng và khi lặp qua các mảng kết hợp, bạn có thể muốn giữ nguyên cả khóa và giá trị.

Điều này có thể thực hiện được bằng cách chỉ định cả khóa và giá trị trong **foreach** định nghĩa, như sau:

Ví dụ in cả khóa và giá trị từ **$members** mảng:

```php
$members = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach ($members as $x => $y) {
  echo "$x : $y <br>";
}
```

output:

Peter : 35

Ben : 37

Joe : 43

Vòng lặp foreach cũng có thể được sử dụng để lặp qua các thuộc tính của một đối tượng:

Ví dụ in ra tên thuộc tính và giá trị của đối tượng **$myCar**:

```php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
}

$myCar = new Car("red", "Volvo");

foreach ($myCar as $x => $y) {
  echo "$x: $y <br>";
}
```

output:

color: red

model: Volvo

Foreach Byref

Khi lặp qua các mục mảng, bất kỳ thay đổi nào được thực hiện đối với mục mảng sẽ, theo mặc định, KHÔNG ảnh hưởng đến mảng ban đầu:

```php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {
  if ($x == "blue") $x = "pink";
}

var_dump($colors);
```

output:

```
array(4) {
  [0]=>
  string(3) "red"
  [1]=>
  string(5) "green"
  [2]=>
  string(4) "blue"
  [3]=>
  string(6) "yellow"
}
```

NHƯNG, bằng cách sử dụng ký tự **&** trong khai báo **foreach**, mục mảng được gán theo **tham chiếu** , dẫn đến bất kỳ thay đổi nào được thực hiện đối với mục mảng cũng sẽ được thực hiện đối với mảng ban đầu:

Ví dụ: Bằng cách gán các mục mảng theo tham chiếu , những thay đổi sẽ ảnh hưởng đến mảng ban đầu:

```php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as &$x) {
  if ($x == "blue") $x = "pink";
}

var_dump($colors);
```

output:

```
array(4) {
  [0]=>
  string(3) "red"
  [1]=>
  string(5) "green"
  [2]=>
  string(4) "pink"
  [3]=>
  string(6) "yellow"
}
```

12. CÁC HÀM PHP

1. Các hàm tích hợp sẵn

PHP có hơn 1000 hàm tích hợp có thể được gọi trực tiếp từ bên trong một tập lệnh để thực hiện một tác vụ cụ thể.

Vui lòng tham khảo tài liệu tham khảo PHP của chúng tôi để có cái nhìn tổng quan đầy đủ về các hàm tích hợp sẵn của PHP.

2. Hàm do người dùng tự định nghĩa

tạo một hàm: (chức năng gọi hàm cũng giống c++)

Khai báo hàm do người dùng định nghĩa bắt đầu bằng từ khóa **function**, theo sau là tên hàm:

```php
function myMessage() {
  echo "Hello world!";
}
```

Lưu ý: Tên hàm phải bắt đầu bằng một chữ cái hoặc dấu gạch dưới. Tên hàm KHÔNG phân biệt chữ hoa chữ thường.

…

Số lượng đối số thay đổi

Bằng cách sử dụng ...toán tử trước tham số hàm, hàm chấp nhận một số lượng đối số không xác định. Điều này cũng được gọi là hàm variadic.

Đối số của hàm variadic trở thành một mảng.

Ví dụ: Một hàm không biết sẽ nhận được bao nhiêu tham số:

```php
function sumMyNumbers(...$x) {
  $n = 0;
  $len = count($x);
  for($i = 0; $i < $len; $i++) {
    $n += $x[$i];
  }
  return $n;
}

$a = sumMyNumbers(5, 2, 6, 2, 7, 7);
echo $a;
```

Bạn chỉ có thể có một đối số có độ dài thay đổi và đó phải là tham số cuối cùng.

Ví dụ: Đối số variadic phải là đối số cuối cùng:

```php
function myFamily($lastname, ...$firstname) {
  txt = "";
  $len = count($firstname);
  for($i = 0; $i < $len; $i++) {
    $txt = $txt."Hi, $firstname[$i] $lastname.<br>";
  }
  return $txt;
}

$a = myFamily("Doe", "Jane", "John", "Joey");
echo $a;
```

Việc sử dụng toán tử "..." ở đối số đầu tiên trong hai đối số sẽ gây ra lỗi:

```php
function myFamily(...$firstname, $lastname) {
  $txt = "";
  $len = count($firstname);
  for($i = 0; $i < $len; $i++) {
    $txt = $txt."Hi, $firstname[$i] $lastname.<br>";
  }
  return $txt;
}

$a = myFamily("Doe", "Jane", "John", "Joey");
echo $a;
```

3. Khai báo kiểu dữ liệu cho một hàm

thông thường, và cả những ví dụ trên, ta không cần khai báo kiểu dữ liệu cho các hàm, vì PHP sẽ tự động liên kết một kiểu dữ liệu với biến, tùy thuộc vào giá trị của nó. Vì các kiểu dữ liệu không được thiết lập theo nghĩa nghiêm ngặt, bạn có thể làm những việc như thêm một chuỗi vào một số nguyên mà không gây ra lỗi.

Trong PHP 7, khai báo kiểu đã được thêm vào. Điều này cung cấp cho chúng ta tùy chọn để chỉ định kiểu dữ liệu mong muốn khi khai báo một hàm và bằng cách thêm **strict** khai báo, nó sẽ đưa ra "**Lỗi nghiêm trọng**" nếu kiểu dữ liệu không khớp.

ví dụ nếu không sử dụng **strict**:

```php
<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

Để chỉ định, **strict** chúng ta cần đặt **declare(strict_types=1);**. Điều này phải nằm ở dòng đầu tiên của tệp PHP.

Trong ví dụ sau, chúng ta thử gửi cả số và chuỗi đến hàm, nhưng ở đây chúng ta đã thêm **strict** khai báo:

```php
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

Tuyên bố strict này buộc mọi thứ phải được sử dụng theo cách dự định.

4. Khai báo kiểu trả về của PHP

PHP 7 cũng hỗ trợ Khai báo Kiểu cho **return** câu lệnh. Giống như khai báo kiểu cho đối số hàm, bằng cách bật yêu cầu nghiêm ngặt, nó sẽ đưa ra "Lỗi nghiêm trọng" khi không khớp kiểu.

Để khai báo kiểu cho hàm trả về, hãy thêm dấu hai chấm (**:**) và kiểu dữ liệu trả về ngay trước {dấu ngoặc nhọn mở ( ) khi khai báo hàm.

Trong ví dụ sau, chúng tôi chỉ định kiểu trả về cho hàm:

```php
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
  return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

Bạn có thể chỉ định kiểu trả về khác với kiểu đối số, nhưng hãy đảm bảo kiểu trả về là đúng:

```php
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
  return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
```

13. MẢNG TRONG PHP

Trong PHP, có ba loại mảng:

- Mảng có chỉ mục - Mảng có chỉ mục số
- Mảng kết hợp - Mảng có khóa được đặt tên
- Mảng đa chiều - Mảng chứa một hoặc nhiều mảng

1. Mảng mục

- các phần tử trong một mảng có thể có các kiểu dữ liệu khác nhau. ví dụ:

```php
$myArr = array("Volvo", 15, ["apples", "bananas"], myFunction);
```

- có rất nhiều hàm hỗ trợ cho mảng trong PHP, ví dụ hàm Count() đếm từng phần tử có trong mảng.

```php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
```

- truy cập mảng bằng chỉ số:

```php
$cars = array("Volvo", "BMW", "Toyota");
echo $cars[0];
```

- thay đổi giá trị của một phần tử:

```php
$cars = array("Volvo", "BMW", "Toyota");
$cars[1] = "Ford";
var_dump($cars);
```

- sử dụng vòng lặp:

```php
$cars = array("Volvo", "BMW", "Toyota");
foreach ($cars as $x) {
  echo "$x <br>";
}
```

- thông thường, các chỉ mục sẽ được sắp xếp theo thứ tự tăng dần một đơn vị, nhưng đối với những mảng có chỉ mục hỗn loạn, chỉ số của một mục mới sẽ là chỉ mục cao nhất hiện tại + 1 đơn vị.

ví dụ, mảng có chỉ số 5, 7, 10, khi thêm một phần tử A, p.tử này sẽ có chỉ mục 11.

…

https://www.w3schools.com/php/php_arrays_update.asp

14. BIẾN TOÀN CỤC PHP - SUPERGLOBALS

Một số biến được xác định trước trong PHP là "siêu toàn cục", nghĩa là chúng luôn có thể truy cập được, bất kể phạm vi - và bạn có thể truy cập chúng từ bất kỳ hàm, lớp hoặc tệp nào mà không cần phải làm bất cứ điều gì đặc biệt.

Các biến siêu toàn cục của PHP là:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

1. $GLOBALS

Tham chiếu đến biến toàn cục **$x** bên trong một hàm:

```php
$x = 75;

function myfunction() {
  echo $GLOBALS['x'];
}

myfunction()
```

Bạn cũng có thể tham chiếu đến các biến toàn cục bên trong hàm bằng cách định nghĩa chúng là biến toàn cục với từ khóa **global.**

Định nghĩa $x là toàn cục bên trong một hàm:

```php
$x = 75;

function myfunction() {
  global $x;
  echo $x;
}

myfunction()
```

2. PHP $_SERVER

$_SERVER là biến toàn cục siêu PHP chứa thông tin về tiêu đề, đường dẫn và vị trí tập lệnh.

Ví dụ dưới đây cho thấy cách sử dụng một số phần tử trong **$_SERVER**:

```php
echo $_SERVER['PHP_SELF'];
echo $_SERVER['SERVER_NAME'];
echo $_SERVER['HTTP_HOST'];
echo $_SERVER['HTTP_REFERER'];
echo $_SERVER['HTTP_USER_AGENT'];
echo $_SERVER['SCRIPT_NAME'];
```

Bảng sau đây liệt kê những thành phần quan trọng nhất có thể đưa vào bên trong **$_SERVER**:

Thành phần/Mã	Mô tả
`$_SERVER['PHP_SELF']`	Trả về tên tệp của tập lệnh đang thực thi hiện tại
`$_SERVER['GATEWAY_INTERFACE']`	Trả về phiên bản của Common Gateway Interface (CGI) mà máy chủ đang sử dụng
`$_SERVER['SERVER_ADDR']`	Trả về địa chỉ IP của máy chủ lưu trữ
`$_SERVER['SERVER_NAME']`	Trả về tên của máy chủ lưu trữ (chẳng hạn như www.w3schools.com)
`$_SERVER['SERVER_SOFTWARE']`	Trả về chuỗi nhận diện của máy chủ (chẳng hạn như Apache/2.2.24)
`$_SERVER['SERVER_PROTOCOL']`	Trả về tên và phiên bản của giao thức thông tin (chẳng hạn như HTTP/1.1)
`$_SERVER['REQUEST_METHOD']`	Trả về phương thức yêu cầu được sử dụng để truy cập trang (chẳng hạn như POST)
`$_SERVER['REQUEST_TIME']`	Trả về dấu thời gian khi bắt đầu yêu cầu (chẳng hạn như 1377687496)
`$_SERVER['QUERY_STRING']`	Trả về chuỗi truy vấn nếu trang được truy cập thông qua một chuỗi truy vấn
`$_SERVER['HTTP_ACCEPT']`	Trả về tiêu đề Accept từ yêu cầu hiện tại
`$_SERVER['HTTP_ACCEPT_CHARSET']`	Trả về tiêu đề Accept_Charset từ yêu cầu hiện tại (chẳng hạn như utf-8, ISO-8859-1)
`$_SERVER['HTTP_HOST']`	Trả về tiêu đề Host từ yêu cầu hiện tại
`$_SERVER['HTTP_REFERER']`	Trả về URL đầy đủ của trang hiện tại (không đáng tin cậy vì không phải tất cả trình duyệt đều hỗ trợ)
`$_SERVER['HTTPS']`	Kiểm tra xem tập lệnh có được truy vấn thông qua giao thức HTTP bảo mật không
`$_SERVER['REMOTE_ADDR']`	Trả về địa chỉ IP của người dùng đang truy cập trang hiện tại
`$_SERVER['REMOTE_HOST']`	Trả về tên máy chủ từ nơi người dùng đang truy cập trang
`$_SERVER['REMOTE_PORT']`	Trả về cổng đang được sử dụng trên máy của người dùng để giao tiếp với máy chủ web
`$_SERVER['SCRIPT_FILENAME']`	Trả về đường dẫn tuyệt đối của tập lệnh đang thực thi
`$_SERVER['SERVER_ADMIN']`	Trả về giá trị của chỉ thị `SERVER_ADMIN` trong tệp cấu hình máy chủ web (nếu tập lệnh chạy trên máy chủ ảo, giá trị này sẽ là giá trị được xác định cho máy chủ ảo đó, chẳng hạn như someone@w3schools.com)
`$_SERVER['SERVER_PORT']`	Trả về cổng trên máy chủ đang được sử dụng bởi máy chủ web để giao tiếp (chẳng hạn như 80)
`$_SERVER['SERVER_SIGNATURE']`	Trả về phiên bản máy chủ và tên máy chủ ảo được thêm vào các trang do máy chủ tạo
`$_SERVER['PATH_TRANSLATED']`	Trả về đường dẫn hệ thống tệp đến tập lệnh hiện tại
`$_SERVER['SCRIPT_NAME']`	Trả về đường dẫn của tập lệnh hiện tại
`$_SERVER['SCRIPT_URI']`	Trả về URI của trang hiện tại

3. PHP $_REQUEST

$_REQUEST là một biến siêu toàn cục (super global) trong PHP, chứa dữ liệu từ các biến **$_GET**, **$_POST**, và **$_COOKIE**. Nó giúp truy xuất dữ liệu được gửi từ form hoặc từ URL một cách linh hoạt.

Sử dụng $_REQUEST với $_POST

Dữ liệu **POST** thường được gửi từ một biểu mẫu HTML (**<form>**).

Ví dụ:

```
<form method="post" action="demo_request.php">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
```

Xử lý dữ liệu trong PHP (demo_request.php):

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

Trong trường hợp này, khi người dùng nhập tên và nhấn "**Submit**", dữ liệu sẽ được gửi đến file demo_request.php và hiển thị tên nhập vào.

Ví dụ kết hợp xử lý bảo mật:

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = htmlspecialchars($_REQUEST['fname']);
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
```

Kiểm tra xem phương thức gửi dữ liệu có phải là **POST** không.

Sử dụng **htmlspecialchars()** để bảo vệ khỏi tấn công XSS.

Kiểm tra nếu ô nhập trống thì thông báo lỗi.

Sử dụng $_REQUEST với $_GET

Ngoài việc nhận dữ liệu từ form, **$_REQUEST** cũng có thể lấy dữ liệu từ query string (chuỗi truy vấn trên URL).

Ví dụ: Gửi dữ liệu qua URL (GET request)

```
<a href="demo_phpfile.php?subject=PHP&web=W3schools.com">Test $REQUEST</a>
```

Xử lý trong file demo_phpfile.php

```
<?php
echo "Study " . $_REQUEST['subject'] . " at " . $_REQUEST['web'];
?>
```

Khi người dùng nhấp vào liên kết, dữ liệu subject=PHP và web=W3schools.com sẽ được gửi đến demo_phpfile.php, và kết quả hiển thị sẽ là:

```
Study PHP at W3schools.com
```

4. PHP - $_POST

Khái niệm

$_POST là một biến siêu toàn cục trong PHP, được sử dụng để thu thập dữ liệu từ các biểu mẫu HTML (**<form>**) khi phương thức gửi là **POST**. Dữ liệu được gửi qua POST không hiển thị trong URL và có thể bao gồm lượng thông tin lớn hơn so với phương thức **GET**.

Sử dụng $_POST để thu thập dữ liệu từ biểu mẫu

Khi người dùng gửi biểu mẫu bằng phương thức **POST**, dữ liệu sẽ được truyền đến máy chủ và có thể được truy cập thông qua biến **$_POST**.

Ví dụ về biểu mẫu HTML:

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

Xử lý dữ liệu trong tệp welcome.php:

```
<?php
$name = $_POST['fname'];
echo "Welcome " . $name;
?>
```

Trong ví dụ này, khi người dùng nhập tên và nhấn "**Submit**", dữ liệu sẽ được gửi đến tệp welcome.php, nơi tên được nhập sẽ được hiển thị với thông điệp "Welcome".

Lưu ý về bảo mật

Mặc dù dữ liệu gửi qua POST không hiển thị trong URL, nhưng vẫn cần thực hiện các biện pháp bảo mật để đảm bảo an toàn cho ứng dụng:

- Kiểm tra và xác thực dữ liệu đầu vào: Đảm bảo rằng dữ liệu người dùng nhập vào phù hợp với định dạng mong muốn và không chứa các ký tự hoặc chuỗi độc hại.

- Sử dụng hàm **htmlspecialchars()**: Chuyển đổi các ký tự đặc biệt trong dữ liệu đầu vào thành các thực thể HTML để ngăn chặn tấn công XSS (Cross-Site Scripting).

```
<?php
$name = htmlspecialchars($_POST['fname']);
echo "Welcome " . $name;
?>
```

Bằng cách sử dụng **htmlspecialchars()**, các ký tự đặc biệt trong tên người dùng nhập sẽ được chuyển đổi, giúp ngăn chặn các lỗ hổng bảo mật tiềm ẩn.

Ví dụ về xử lý dữ liệu an toàn:

4. PHP $_GET

Khái niệm

Biểu thức chính quy (Regular Expressions) là một chuỗi ký tự đặc biệt giúp bạn tìm kiếm hoặc khớp các mẫu trong văn bản. Trong PHP, có hai hàm chính để làm việc với biểu thức chính quy:

- **preg_match()**: Kiểm tra xem một chuỗi có khớp với mẫu biểu thức chính quy hay không.

- **preg_match_all()**: Tìm tất cả các khớp trong một chuỗi với mẫu biểu thức chính quy.

Sử dụng preg_match()

Hàm **preg_match()** kiểm tra xem một chuỗi có khớp với mẫu biểu thức chính quy hay không và trả về số lần khớp (0 hoặc 1).

Cú pháp:

```
preg_match(pattern, input, matches, flags, offset)
```

pattern: Biểu thức chính quy cần tìm.

input: Chuỗi cần kiểm tra.

matches: Mảng lưu trữ kết quả khớp (tùy chọn).

flags: Cờ tùy chọn (tùy chọn).

offset: Vị trí bắt đầu tìm kiếm (tùy chọn).

ví dụ:

```php
<?php
$str = "Visit W3Schools";
$pattern = "/w3schools/i";
echo preg_match($pattern, $str);
?>
```

kết quả: 1 (khớp, được tìm thấy)

Sử dụng preg_match_all()

Hàm **preg_match_all()** tìm tất cả các khớp trong một chuỗi với mẫu biểu thức chính quy và trả về số lượng khớp.

Cú pháp:

```php
preg_match_all(pattern, input, matches, flags, offset)
```

ví dụ:

```php
<?php
$str = "The rain in SPAIN falls mainly on the plains.";
$pattern = "/ain/i";
echo preg_match_all($pattern, $str, $matches);
?>
```

Kết quả: 4 (có 4 khớp với mẫu "ain" không phân biệt chữ hoa/thường).

Sử dụng preg_replace()

Hàm **preg_replace()** tìm kiếm các mẫu trong chuỗi và thay thế chúng bằng một chuỗi khác.

Cú pháp:

```php
preg_replace(pattern, replacement, input, limit, count)
```

replacement: Chuỗi thay thế.

limit: Giới hạn số lần thay thế (tùy chọn).

count: Biến lưu trữ số lần thay thế (tùy chọn).

ví dụ:

```php
<?php
$str = "Visit Microsoft!";
$pattern = "/microsoft/i";
$replacement = "W3Schools";
echo preg_replace($pattern, $replacement, $str);
?>
```

Kết quả: Visit W3Schools!

Lưu ý: Khi sử dụng biểu thức chính quy trong PHP, các mẫu thường được đặt giữa dấu gạch chéo (/). Tham số **i** sau dấu gạch chéo kết thúc mẫu cho biết tìm kiếm không phân biệt chữ hoa/thường.