

PHP



LẬP TRÌNH PHP 1

BÀI 3: HÀM TRONG PHP

PHẦN 1

Nội dung bài học

- Tổng quan về hàm trong PHP
- Khai báo hàm
- Sử dụng hàm
- Thư viện hàm



- ❑ Khởi tạo và gọi hàm
- ❑ Truyền tham số theo giá trị và tham chiếu
- ❑ Sử dụng phạm vi hoạt động của biến
- ❑ Gán giá trị mặc định cho tham số
- ❑ Sử dụng danh sách tham số có độ dài biến đổi
- ❑ Khởi tạo và sử dụng thư viện của hàm

❑ Cú pháp của một hàm

```
function function_name([$param_1, $param_2, ... , $param_n]) {  
    //viết mã cho hàm  
    [return return_value]  
}
```

- ❑ Để viết hàm có trả về dữ liệu, viết câu lệnh return cuối thân hàm. Lệnh return kết thúc việc thực hiện hàm và trả về giá trị xác định. Nếu không gán giá trị trả về thì giá trị NULL sẽ được trả về
- ❑ Để hàm không trả về dữ liệu, không viết lệnh return
- ❑ Khi thực hiện lời gọi hàm, các đối số trong danh sách đối số phải theo cùng thứ tự của các tham số trong danh sách tham số mà hàm đã xác định và phải tương thích về kiểu dữ liệu

❑ Hướng dẫn xây dựng hàm:

```
//Hàm không có tham số và trả về một giá trị
function coin_toss() {
    $coin = (mt_rand(0,1)==0)?'heads':'tails';
    return $coin;
}
```

```
//Hàm có ba tham số và trả về một giá trị
function avg_of_3($x, $y, $z) {
    $avg = ($x + $y + $z) / 3;
    return $avg;
}
```

```
//Hàm có một tham số và không trả về giá trị
function display_error($error) {
    echo '<p class="error">'.$error.'</p>';
}
```

□ Hướng dẫn gọi hàm:

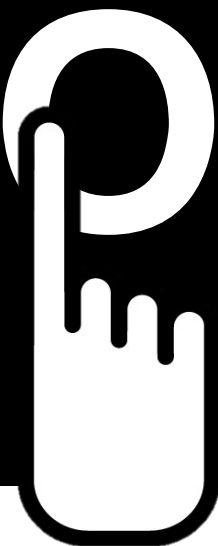
```
//Hàm trả về nhiều giá trị
$average = avg_of_3(5, 2, 8);           // $average bằng 5
echo coin_toss();                       // hiển thị heads hoặc tails

//Hàm không trả về giá trị
display_error('Value out of range.');
```



DEMO

Xây Dựng hàm nhận 3 tham số : số
thứ nhất, số thứ hai và toán tử.
Hàm trả về kết quả của biểu thức



- ❑ Mặc định, tất cả các đối số của hàm được truyền theo giá trị
- ❑ Truyền tham số theo giá trị: một bản sao của đối số sẽ được gửi tới hàm. Khi hàm thay đổi một tham số, nó chỉ thay đổi bản sao của đối số, chứ không phải đối số ban đầu
- ❑ Truyền tham số theo tham chiếu: một tham chiếu đến các tham số ban đầu sẽ được gửi tới hàm. Khi hàm thay đổi tham số, hàm thực sự thay đổi các đối số ban đầu. Cú pháp: viết ký hiệu '&' trước tham số

- Đối số được truyền theo giá trị:

```
function add_3_by_val($value) {  
    $value += 3;  
    echo 'Number: '.$value;  
}  
$number = 5;  
add_3_by_val($number);           //hiển thị 8  
echo 'Number: '.$number;         //hiển thị 5
```

- Đối số được truyền theo tham chiếu:

```
function add_3_by_ref(&$value) {  
    $value += 3;  
    echo '<p>Number: '.$value.'</p>';  
}  
$number = 5;  
add_3_by_ref($number);           //hiển thị 8  
echo '<p>Number: '.$number.'</p>'; //hiển thị 8
```

- ❑ Phạm vi của một biến xác định đoạn mã có thể truy cập biến đó

- ❑ Biến được định nghĩa bên trong hàm:
 - Có phạm vi cục bộ
 - Chỉ có hiệu lực với đoạn mã chạy bên trong hàm

- ❑ Biến được định nghĩa bên ngoài hàm:
 - Có phạm vi toàn cục
 - Chỉ có hiệu lực với đoạn mã chạy ở cấp toàn cục và không có hiệu lực trong phạm vi bất kỳ hàm nào (theo mặc định)

- ❑ Truy cập một biến toàn cục từ bên trong hàm:
sử dụng câu lệnh toàn cục để nhập một biến từ phạm vi toàn cục sang phạm vi cục bộ
- ❑ Nhận tất cả các biến được lưu trong phạm vi toàn cục: sử dụng mảng tích hợp \$GLOBALS
- ❑ Mảng \$GLOBALS là biến toàn cục tự động giống như các mảng \$_POST và \$_GET

❑ Biến có phạm vi toàn cục:

```
$a = 10;                                //$a có phạm vi toàn cục
function show_a() {                      //bên trong hàm, $a là NULL
    echo $a;
}
show_a();                                //không hiển thị gì
```

❑ Biến có phạm vi địa phương:

```
function show_d() {
    $d = 10;
    echo $d;                            //$d có phạm vi cục bộ bên trong hàm show_d
}
echo $d;                                //bên ngoài hàm, $d là NULL
```

❑ Hướng dẫn truy cập biến toàn cục từ phạm vi bên trong hàm:

```
$b = 10;                                // $b có phạm vi toàn cục
function show_b() {
    global $b;                          // $b lúc này tham chiếu tới biến toàn cục tên $b
    echo $b;
}
show_b();                               // hiển thị 10
```

❑ Cách khác để truy cập biến toàn cục từ phạm vi bên trong hàm:

```
$c = 10;                                // $c có phạm vi toàn cục
function show_c() {
    $c = $GLOBALS['c'];                 // $c lúc này tham chiếu tới biến toàn cục $c
    echo $c;
}
show_c();                               // hiển thị 10
```

□ Cú pháp:

<Tên tham số> = <giá trị mặc định>

□ Giá trị mặc định phải là giá trị hoặc mảng các giá trị vô hướng hoặc là giá trị NULL

□ Viết hàm thiết lập giá trị mặc định cho một tham số:

- Bước 1: gán giá trị NULL cho tham số đó
- Bước 2: trong phạm vi hàm, kiểm tra xem tham số này có chứa giá trị NULL không

❑ Hướng dẫn gán giá trị mặc định cho tham số:

```
//Hàm với một tham số mặc định
function get_rand_bool_text($type = 'coin') {
    $rand = mt_rand(0, 1);
    switch ($type) {
        case 'coin':
            $result = ($rand == 1)?'heads':'tails';
            break;
        case 'switch':
            $result = ($rand == 1)?'on':'off';
            break;
    }
    return $result;
}
```

```
//Hàm với một tham số bắt buộc và hai tham số mặc định
function display_error($error, $tag = 'p', $class = 'error') {
    $opentag = '<'. $tag. ' class="'. $class. '">';
    $closetag = '</'. $tag. '>';
    echo $opentag.$error.$closetag;
}
```

❑ Lời gọi hàm với một tham số mặc định:

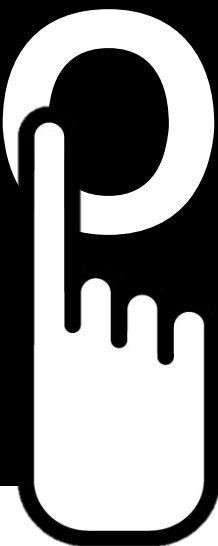
```
//Bỏ qua các tham số tùy chọn
echo get_rand_bool_text ();           //hiển thị 'heads' hoặc 'tails'
$is_leap_year = is_leap_year();       //đúng hoặc sai dựa vào ngày hiện tại

//Cung cấp các tham số tùy chọn
echo get_rand_bool_text('switch');    //hiển thị 'on' hoặc 'off'
$is_leap_year = is_leap_year (new DateTime('March 15,2015')); //sai
```




DEMO

Xây dựng hàm nhận hai tham số và
thực hiện hoán vị 2 giá trị với nhau



PHP



LẬP TRÌNH PHP 1

BÀI 3: HÀM TRONG PHP

PHẦN 2

- ❑ Danh sách tham số có chiều dài thay đổi cho phép tạo một hàm làm việc với số lượng đối số thay đổi
- ❑ Có thể yêu cầu một số lượng tối thiểu các đối số bằng cách sử dụng tham số dự trữ trong danh sách tham số

Hàm	Mô tả
<code>func_get_args()</code>	Trả về một mảng chứa các tham số được truyền vào hàm.
<code>func_num_args()</code>	Trả về số lượng tham số được truyền vào hàm.
<code>func_get_arg(\$i)</code>	Trả về tham số tại một chỉ mục cụ thể.

❑ Hướng dẫn viết hàm với danh sách tham số

```
//Hàm thêm một danh sách các số
function add() {
    $numbers = func_get_args();
    $total = 0;
    foreach($numbers as $number) {
        $total += $number;
    }
    return $total;
}
$sum = add(5, 10, 15);           //$sum bằng 30

//Hàm tính trung bình một hoặc nhiều số
function average($x) {           //$x yêu cầu ít nhất một đối số
    $count = func_num_args();
    $total = 0;
    for($i = 0; $i < $count; $i++) {
        $total += func_get_arg($i);
    }
    return $total / $count;
}
$avg = average(75, 95, 100);     //$avg bằng 90
```

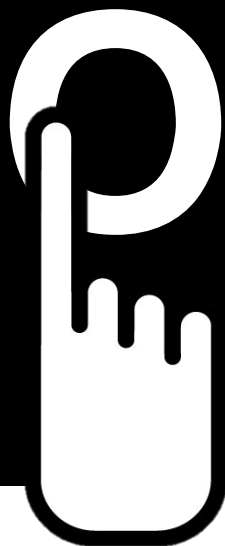
Danh sách tham số có độ dài biến đổi

```
//Sử dụng tham số bắt buộc với danh sách tham số biến đổi
function array_append(&$array, $x) {
    $values = func_get_args(); //cũng chứa $array
    array_shift($values); //loại bỏ $array từ phía trước
    foreach($values as $value) {
        $array[] = $value;
    }
}
$data = array('apples', 'oranges');
array_append($data, 'grapes', 'pears');
```



DEMO

Xây dựng hàm cộng nhưng số
lượng tham số tùy theo nhu cầu
người sử dụng. Hàm trả về tổng



- ❑ Các ứng dụng thường có rất nhiều hàm. Trong trường hợp này, việc tổ chức các hàm vào thư viện bên ngoài rất hữu ích
- ❑ Mục đích sử dụng thư viện:
 - Có thể dùng thư viện cho hơn một ứng dụng
 - Các lập trình viên có thể làm việc trên các thư viện khác nhau để giảm thời gian phát triển ứng dụng
- ❑ Hướng dẫn thiết lập đường dẫn fle chèn:
 - Lưu thư viện trong thư mục riêng để nó có thể được truy cập bởi nhiều ứng dụng
 - Thêm thư mục này vào đường dẫn fle chèn. Đường dẫn này là một danh sách các thư mục cho phép PHP tìm kiếm các fle chèn

- Các hàm làm việc với đường dẫn file chèn:

Hàm	Mô tả
<code>get_include_path()</code>	Lấy chuỗi chứa đường dẫn file chèn hiện thời.
<code>set_include_path(\$path)</code>	Thiết lập đường dẫn file chèn.

- Hướng dẫn lấy đường dẫn file chèn:

```
$include_path = get_include_path();
```

- Hướng dẫn thiết lập đường dẫn file chèn:

```
set_include_path($include_path, ';C:\xampp\htdocs\book_apps\lib');
```

- Hướng dẫn chèn một file sau khi đường dẫn file chèn được thiết lập:

```
require_once cart.php; //ví dụ file cart.php là file cần chèn
```


□ Hướng dẫn tạo và sử dụng namespace:

- Namespace có chứa một nhóm các tên không có trong phạm vi toàn cục
- Namespace cho phép tổ chức các hàm và sử dụng các tên đã được dùng trong namespace toàn cục
- Có thể hình dung namespace giống như một thư mục trên máy tính. Ví dụ, bạn có thể có hai file cùng tên report.txt miễn là chúng ở các thư mục khác nhau. Tương tự, bạn có thể có hai hàm được đặt tên là show miễn là chúng ở trong các namespace khác nhau

- Bạn có thể sử dụng namespace để chứa các hàm trong namespace thay vì sử dụng namespace toàn cục. Điều này giúp bạn tránh được việc đụng độ tên.
- Trong phạm vi một namespace, bạn có thể đưa vào các hằng số, hàm và lớp. Trong chương tiếp theo, bạn sẽ tìm hiểu thêm về lớp.
- Để thực hiện lời gọi tới một hàm trong namespace, viết tên cho namespace, dấu xỏ ngược và tên hàm.
- Để tạo ra một bí danh cho namespace, sử dụng từ khóa `using`, theo sau là tên namespace, tiếp theo là từ khóa `as`, sau đó là bí danh

- Hướng dẫn tạo namespace trong file:

```
//Sử dụng cú pháp lệnh
```

```
<?php
```

```
namespace cart;
```

```
    //các hàm trong namespace cart
```

```
?>
```

```
//Sử dụng cú pháp dấu ngoặc nhọn
```

```
<?php
```

```
namespace cart {
```

```
    //các hàm trong namespace cart
```

```
}
```

```
?>
```

- Hướng dẫn sử dụng các hàm được xác định trong namespace:

```
//Tạo file chứa namespace với một hàm
```

```
namespace murach\errors {  
function log($error) {  
    echo '<p class="error">' . $error . '</p>';  
}  
}
```

```
//Gọi tới hàm được lưu trong namespace
```

```
//nạp file lưu namespace
```

```
require_once 'errors.php';
```

```
//gọi hàm log
```

```
murach\errors\log('Invalid value');
```

```
//tạo bí danh và sử dụng nó để gọi hàm log
```

```
use murach\errors as e; //sử dụng 'e' thay cho 'murach\errors'
```

```
e\log('Invalid value');
```

Tổng kết bài học

- Tổng quan về hàm trong PHP
- Khai báo hàm
- Sử dụng hàm
- Thư viện hàm





KẾT THÚC