Basic 2

Edited by Thanh-Huy LE

Table of Contents

- Array and functions on array
- String functions
- User defined function
- PHP Forms and User Input

Array

- An array is a special variable, which can store multiple values in one single variable.
- In PHP, there are three kinds of arrays:
  - **Numeric array** - with a numeric index
  - **Associative array** - with (key-value) pair
  - **Multidimensional array** - An array containing one or more arrays

PHP Array

- Numeric array

```
$cars =
array(“Ferrari","BMW",“Ford");


or


$cars[0]=“Ferrari";
$cars[1]="BMW";
$cars[2]=“Ford";


echo $cars[0] . " and " .
$cars[1] ;


Result =??
```

- Associative Arrays

```
$ages =
array(“Ronaldo"=>27,“Messi"=>24);


or


$ages[‘Ronaldo’] = “27";
$ages[‘Messi’] = “24";


echo “Messi is " . $ages[‘Messi'] .
" years old.";


Result =??
```

Multidimensional Arrays

- element of array is also an array

```
$sport = array
(
“soccer”=>array ( “Ronaldo”, “Messi”, “Rooney” ),
“tennis” =>array (“Federer”, “Nadal” )
)
echo $sport[‘soccer’][1] . " is the best player”;
```

Result ???

Access PHP Array

- Sequential access to array elements **current** and **next**

```
$colors = array(“blue", "red", "green");
$color = current($colors);
print("$color <br />");
while ($color = next($colors))
     print ("$color <br />");
```

- Alternative: **each**, instead of **next**

```
while ($element = each($colors)) {
    print ("$element['value'] <br />");
}
```

Access PHP Array

- Using `foreach` statement
- Syntax:

```
foreach (array_name as scalar_name) { ... }
```

- Example

```
$colors = array(“blue", "red", "green");
foreach ($colors as $color) {
   print $color;
}
```

PHP Array - Useful functions

explode()

- Breaks a string into an array.
- Syntax: `explode(separator, string, limit)`

```
<?php
$str = 'one|two|three|four';

// positive limit
print_r(explode('|', $str, 2));

// negative limit (PHP 5.x)
print_r(explode('|',$str, -1));
?>
```

output

```
Array
(
 [0] => one
 [1]=> two|three|four
)
Array
(
 [0] => one
 [1] => two
 [2] => three
)
```

PHP Array - Useful functions

implode()

- returns a string from the elements of an array.
- Syntax: `implode(separator,array)`

```
<?php
$arr =
array(‘one',‘two',‘three',‘four');
echo implode(“|",$arr);
?>
```

one | two | three | four

PHP Array - Other Useful functions

- Testing whether an element exists

```
if (array_key_exists("Wed", $highs))
```

- An array can be deleted with unset

```
unset($list);
unset($list[4]);# Deletes index 4 element
```

- `sizeof(`*`an_array`*`)` returns the number of elements
- `in_array(17, $list)` returns `true` if `17` is an element of `$list`

PHP String functions

- **Finding length of a string using** `strlen()`

```
$str = ‘Monday';
$len = strlen('$str');
echo $len; // this will output 6
```

- **Return part of a string, using:**

```
substr(string string, int start [, int length])
```

```
echo substr(“Monday", 1); // => “onday”
echo substr(“Monday", 1, 3); // => “ond"
substr(“Monday", 0, 4); // => “Mond“
substr(“Monday", -1); // => “y"
substr(“Monday", -3, 2); // => “da"
```

PHP String functions

- **Remove whitespaces** and other predefined characters from both sides of a string, using:

  `trim(string)`

- Following characters are removed:

"\0" - NULL	"\t" - tab
"\n" - new line	"\x0B" - vertical tab
"\r" - carriage return	" " - ordinary **white space**

```
<?php
$str = "      i love you      ";
echo "With trim: " . trim($str); // => i love
you (no spaces)
?>
```

User Defined Function

- A function will be executed by a call it.

```
function displayName()
{
  echo ‘Christinano Ronaldo';
}
```

- Calling function

```
displayName();
```

Functions - parameters & return value

```
<?php
function add($x, $y)
{
    $total = $x + $y;
    return $total;
}
$result = add(1, 16);
echo "1 + 16 = " . $result;
?>
```

Result ???

PHP Forms and User Input

- The PHP **`$_GET`** and **`$_POST`** variables are used to retrieve information from user input (form)
- **`$_GET`** variable is used to collect values in a form with **`method="GET"`**.
- $_POST variable is used to collect values in a form with **`method="POST"`**.

PHP $_GET Variable

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

- When click the "Submit" button, the URL sent to the server could look something like this:

  …/welcome.php?fname=Ronaldo&age=27

- In "welcome.php":

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

- Result

```
Welcome Ronaldo
You are 27 years old!
```

PHP $_REQUEST Variable

- The predefined `$_REQUEST` variable contains the contents of both `$_GET`, `$_POST`, and `$_COOKIE`.
- Used to collect form data sent with both the GET and POST methods.
- Example

```
Welcome <?php echo $_REQUEST["fname"]; ?>!<br />
You are <?php echo $_REQUEST["age"]; ?> years old.
```

- Result

```
Welcome Ronaldo
You are 27 years old!
```

Demo