S@ifur's Solutions

69/B Monwara Plaza (Level-5)
East Panthapath
Dhaka-1205
Cell: 01713 432003

Lecture Sheet on "PHP+MySQL"
Day-10: (PHP+MySQL)

Objectives
- ❖ Understanding of some useful functions
- ❖ Implementation of these examples

Topics
- ❖ Trim()
- ❖ uniqid()
- ❖ substr()
- ❖ rand()
- ❖ md5()
- ❖ foreach()
- ❖ list()

trim()

trim — Strip whitespace (or other characters) from the beginning and end of a string

Description:

string **trim** (string $str [, string $charlist])

Parameters

str : The **string** that will be trimmed.

Charlist: Optionally, the stripped characters can also be specified using the *charlist* parameter. Simply list all characters that you want to be stripped. With .. you can specify a range of characters.

Return Values: The trimmed string.

*Note.: **ltrim()** and **rtrim()** functions are used as their name suggest.*

uniqid()

uniqid — Generate a unique ID

Description

string **uniqid** ([string $prefix [, bool $more_entropy]])

Gets a prefixed unique identifier based on the current time in microseconds.

Parameters

prefix : Can be useful, for instance, if you generate identifiers simultaneously on several hosts that might happen to generate the identifier at the same microsecond. With an empty *prefix* , the returned string will be 13 characters long. If *more_entropy* is **TRUE**, it will be 23 characters.

more_entropy **:** If set to **TRUE**, **uniqid()** will add additional entropy (using the combined linear congruential generator) at the end of the return value, which should make the results more unique.

Return Values

Returns the unique identifier, as a string.

Examples

If you need a unique identifier or token and you intend to give out that token to the user via the network (i.e. session cookies), it is recommended that you use something along these lines: This will create a 32 character identifier (a 128 bit hex number) that is extremely difficult to predict.

Example #1 uniqid() Example

```
<?php
//no prefix
//works only in PHP5 and later versions
$token = md5(uniqid());

//better, difficult to guess
$better_token = md5(uniqid(rand(), true));
?>
```

substr()

substr — Return part of a string

Description

string **substr** (string $string , int $start [, int $length])
Returns the portion of *string* specified by the *start* and *length* parameters.

Parameters

string : The input string.
Start : If *start* is non-negative, the returned string will start at the *start* 'th position in ***string*** , counting from zero. For instance, in the string '*abcdef*', the character at position *0* is '*a*', the character at position *2* is '*c*', and so forth. If *start* is negative, the returned string will start at the *start* 'th character from the end of *string* .
Example #1 Using a negative *start*

```
    <?php
    $rest = substr("abcdef", -1);    // returns "f"
    $rest = substr("abcdef", -2);    // returns "ef"
    $rest = substr("abcdef", -3, 1); // returns "d"
    ?>
```

length
If *length* is given and is positive, the string returned will contain at most *length* characters beginning from *start* (depending on the length of *string*). If *string* is less than or equal to *start* characters long, **FALSE** will be returned. If *length* is given and is negative, then that many characters will be omitted from the end of *string* (after the start position has been calculated when a *start* is negative). If *start* denotes a position beyond this truncation, an empty string will be returned.
Example #2 Using a negative *length*

```
<?php
$rest = substr("abcdef", 0, -1);  // returns "abcde"
$rest = substr("abcdef", 2, -1);  // returns "cde"
$rest = substr("abcdef", 4, -4);  // returns ""
$rest = substr("abcdef", -3, -1); // returns "de"
?>
```

Return Values

Returns the extracted part of string.

rand()

rand — Generate a random integer

Description

int **rand** (void)
int **rand** (int $min , int $max)

If called without the optional *min* , *max* arguments **rand()** returns a pseudo-random integer between 0 and getrandmax(). If you want a random number between 5 and 15 (inclusive), for example, use *rand(5, 15)*.

Note: On some platforms (such as Windows), getrandmax() is only 32768. If you require a range larger than 32768, specifying *min* and *max* will allow you to create a range larger than this, or consider using mt_rand() instead.

Parameters

min : The lowest value to return (default: 0)
max : The highest value to return (default: getrandmax())

Return Values

A pseudo random value between *min* (or 0) and *max* (or getrandmax(), inclusive).

Example:

```
<?php
echo rand() . "\n";
echo rand() . "\n";

echo rand(5, 15);
?>
```

The above example will output something similar to:

```
7771
22264
11
```

md5()

md5 — Calculate the md5 hash of a string

Description

string **md5** (string $str [, bool $raw_output])

Calculates the MD5 hash of *str* using the » RSA Data Security, Inc. MD5 Message-Digest Algorithm, and returns that hash.

Parameters

str : The string.
raw_output : If the optional *raw_output* is set to **TRUE**, then the md5 digest is instead returned in raw binary format with a length of 16. Defaults to **FALSE**.

Return Values

Returns the hash as a 32-character hexadecimal number.

Examples

```
<?php
$str = 'apple';

if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') {
   echo "Would you like a green or red apple?";
   exit;
}
?>
```

foreach

PHP 4 introduced a *foreach* construct, much like Perl and some other languages. This simply gives an easy way to iterate over arrays. *foreach* works only on arrays, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two syntaxes; the second is a minor but useful extension of the first:

```
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement
```

The first form loops over the array given by *array_expression*. On each loop, the value of the current element is assigned to *$value* and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).
The second form does the same thing, except that the current element's key will be assigned to the variable *$key* on each loop.

As of PHP 5, it is possible to iterate objects too.

Note: When *foreach* first starts executing, the internal array pointer is automatically reset to the first element of the array. This means that you do not need to call reset() before a *foreach* loop.

> **Note**: Unless the array is referenced, *foreach* operates on a copy of the specified array and not the array itself. *foreach* has some side effects on the array pointer. Don't rely on the array pointer during or after the foreach

without resetting it.

As of PHP 5, you can easily modify array's elements by preceding *$value* with &. This will assign reference instead of copying the value.

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
   $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
unset($value); // break the reference with the last element
?>
```

list

list — Assign variables as if they were an array

Description

void **list** (mixed $varname [, mixed $...])

Like array(), this is not really a function, but a language construct. **list()** is used to assign a list of variables in one operation.

Parameters

varname

A variable.

Return Values

No value is returned.

Examples

```
<?php

$info = array('coffee', 'brown', 'caffeine');

// Listing all the variables
list($drink, $color, $power) = $info;
echo "$drink is $color and $power makes it special.\n";

// Listing some of them
list($drink, , $power) = $info;
echo "$drink has $power.\n";

// Or let's skip to only the third one
list( , , $power) = $info;
echo "I need $power!\n";
```

```
// list() doesn't work with strings
list($bar) = "abcde";
var_dump($bar); // NULL
?>
```

Example #2 An example use of list()

```
<table>
 <tr>
  <th>Employee name</th>
  <th>Salary</th>
 </tr>

<?php

$result = mysql_query("SELECT id, name, salary FROM employees", $conn);
while (list($id, $name, $salary) = mysql_fetch_row($result)) {
    echo " <tr>\n" .
          "  <td><a href=\"info.php?id=$id\">$name</a></td>\n" .
          "  <td>$salary</td>\n" .
          " </tr>\n";
}

?>
</table>
```