```
function create_table($data){
  echo '<table border="1">';
  reset($data); // Remember this is used to point to the beginning
  $value = current($data);
  while ($value){
    echo "<tr><td>$value</td>     \n";
    $value = next($data);
  }
  echo '</table>';
}
```

PHP

Hypertext Preprocessor

06 PHP III

PHP III

- Functions
- Internal or Built-in Functions
- User-defined Functions
- Testing Variable Functions
- Workshop

Functions

- Using Functions in PHP
  - A function is a self-contained module of code that prescribes a calling interface, performs some task, and optionally returns a result.
    - To separate code that performs a single, well-defined task.
    - To makes the code easier to read and allows us to reuse the code each time we need to do the same task.
  - Calling Functions
    - The following line is the simplest possible call to a function:
      - function_name();
        - » This calls a function named function_name that does not require parameters.
        - » This line of code ignores any value that might be returned by this function.
        - » For example: phpinfo();
    - Most functions do require one or more parameters –information given to a function.
      - function_name(7.993);   function_name($variable);

Functions

Using Functions in PHP (cont.)

- You can see how many parameters a function takes, what each represents, and what data type each needs to be from the function 's *prototype* .
- This is the prototype for the function fopen():
  - **int fopen(string *filename*, string *mode*, [int *use_include_path*]);**
    - The prototype tells us a number of things, and it is important that you know how to correctly interpret these specifications.
    - In this case, the word int before the function name tells us that this function will return an integer.
    - The function parameters are inside the parentheses.
    - In the case of fopen(), three parameters are shown in the prototype.
      - » The parameter filename and mode are strings and the last parameter is an integer.
      - » The square brackets around use_include_path indicate that this parameter is optional.
      - » We can provide values for optional parameters or we can choose to ignore them, and the default value will be used.

```
$name = 'myfile.txt';
$openmode = 'r';
$fp = fopen($name, $openmode)
```

Functions

- Using Functions in PHP (cont.)
  - Case and Function Names
    - Calls to functions are not case sensitive, so calling function_name(), Function_Name(), or FUNCTION_NAME() are all valid and will all have the same result.
    - It is important to note that function names behave differently to variable names.
      - Variable names are case sensitive, so $Name and $name are two separate variables
      - Function names, Name() and name() are the same function.

Functions

- Why Should You Define Your Own Functions?
  - Declaring a function allows you to use your own code in the same way as the built-in functions.
  - You can call and reuse the same function many times throughout your script.
    - Save time to write the code
    - Easy to debug
  - Basic function structure
    - A function declaration creates or declares a new function.
    - The declaration begins with the keyword **function** provides the **function name**, the **parameters required**, and contains the code that will be executed each time this function is called.

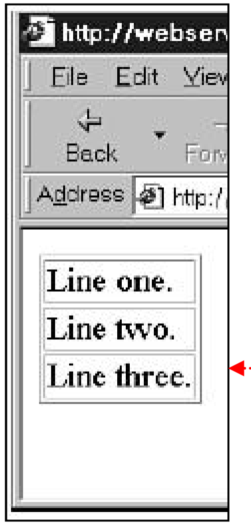
```
function my_function(){
   echo 'My function was called';
}
```

Functions

Parameters

- A parameter allows you to pass data into a function.
- Most functions require one or more parameters.
  - For example: If we call our create_table() function as follows

```
$my_array = array('Line one.','Line two.','Line three.');
create_table($my_array);
```



Line one.
Line two.
Line three.

```
function create_table($data){
  echo '<table border="1">';
  reset($data); // Remember this is used to point to the beginning
  $value = current($data);
  while ($value){
    echo "<tr><td>$value</td></tr>\n";
    $value = next($data);
  }
  echo '</table>';
}
```

Functions

Scope

- A variable 's scope controls where that variable is visible and useable.
  - Variables declared **inside a function** are in scope from the statement in which they are declared to the closing brace at the end of the function. This is called *function scope* .These variables are called *local variables* .
  - Variables declared **outside of functions** are in scope from the statement in which they are declared to the end of the file, but *not inside functions* .This is called *global scope* .These variables are called *global variables* .
  - Using require() and include() statements does not affect scope.
    - If the statement is used within a function, **function scope applies.**
    - If it is not inside a function, **global scope applies.**
  - The keyword **global** can be used to manually specify that a variable defined or used within a function will have global scope.
  - Variables can be manually deleted by calling unset*(variable_name).*
    - A variable is no longer in scope if it has been unset.

Functions

Scope (cont.)

- For example:

```
function fn(){
  $var = 'contents';
}
echo $var;
```

```
...
function fn(){
  echo 'inside the function, $var = '.$var.'<br />';
  $var = 'contents2';
  echo 'inside the function, $var = '.$var.'<br />';
}
$var = 'contents 1';
fn();
echo 'outside the function, $var = '.$var.'<br />';
...
```

```
inside the function, $var =
inside the function, $var = contents 2
outside the function, $var = contents 1
```

Functions

- Code Blocks

```
for($i=0; $i<3; $i++ )
  echo 'Line 1<br />';
  echo 'Line 2<br />';
```

```
Line 1
Line 1
Line 1
Line 2
```

```
for($i=0; $i<3; $i++ ){
  echo 'Line 1<br />';
  echo 'Line 2<br />';
}
```

```
Line 1
Line 2
Line 1
Line 2
Line 1
Line 2
```

Internal or Built-in Functions

- Date and Time Functions
- Array Functions
- Character Type Functions
- Directory Functions
- String Functions
- Mathematical Functions
- Mail Functions
- Filesystem Functions

Internal or Built-in Functions

- Date and Time Functions

```
<?php
  echo "Today is ".date("l \\t\h\e jS \o\\f F, Y.")."<br>";
  echo "Now, it is ".date("H:i:s A").".<br>";
  echo "December 5, 2007 is on ".date("l", mktime(0, 0, 0, 12, 5, 2007)).".<br>";
?>
```

```
Today is Saturday the 5th of May, 2007.
Now, it is 18:27:36 PM.
December 5, 2007 is on Wednesday.
```

NOTE: \t is tab and \f is formfeed, so that we need to escape them.

Internal or Built-in Functions

Array Functions

```
<?php
  $a = array(1, 3, 5, 7, 9);
  echo 'Sum of all values in $a = '.array_sum($a).'<br>';
  echo 'Array $a in original order: ';
  foreach($a as $val){
          echo "$val ";
  }
  $a = array_reverse($a);
  echo '<br>Array $a in reversed order: ';
  foreach($a as $val){
          echo "$val ";
  }
  echo '<br>Display shuffle numbers from 1 to 10: ';
  $numbers = range(1,10);
  shuffle($numbers);
  foreach ($num as $number) {
    echo "$num ";
  }
?>
```

```
Sum of all values in $a = 25
Array $a in original order: 1 3 5 7 9
Array $a in reversed order: 9 7 5 3 1
Display shuffle numbers from 1 to 10: 7 3 2 8 9 6 4 5 1 10
```

Internal or Built-in Functions

Character Type Functions

```
<?php
  $str = array("SUT Korat 2007", "Yahoo1999DotCom");
  foreach ($str as $s){
           echo "The string \"$s\" ";
    if(ctype_alnum($s)){
       echo "consists of all letters or digits.<br>";
    }else{
       echo "does not consist of all letters or digits.<br>";
    }
  }
?>
```

```
The string "SUT Korat 2007" does not consist of all letters or digits.
The string "Yahoo1999DotCom" consists of all letters or digits.
```

ctype_alnum -- Check for alphanumeric character(s), either a letter or a digit

Internal or Built-in Functions

- Directory Functions

```
<?php
  echo getcwd().'<br>';
  $dir = 'D:\AppServ\www\php_book';
  $files = scandir($dir);
  foreach($files as $value){
    echo "$value<br>";
  }
  echo getcwd().'<br>';
  chdir('..\cp4');
  echo getcwd().'<br>';
?>
```

```
D:\AppServ\www\php_book\cp5
.
..
cp3
cp4
cp5
D:\AppServ\www\php_book\cp5
D:\AppServ\www\php_book\cp4
```

Internal or Built-in Functions

- String Functions

```
<?php
  $data2 = "This is a sample of using string functions.";
  echo "$data2<br>";
  $data2_out1 = ucwords($data2);
  echo "$data2_out1<br>";
  $data2_out2 = str_replace("a sample of","the",$data2);
  echo "$data2_out2<br>";
  $ans = strcmp("somsak", "sombat");
  echo "The comparison between \"somsak\" and \"sombat\" is $ans.<br>";
?>
```

```
This is a sample of using string functions.
This Is A Sample Of Using String Functions.
This is the using string functions.
The comparison between "somsak" and "sombat" is 1.
```

Internal or Built-in Functions

- Mathematical Functions

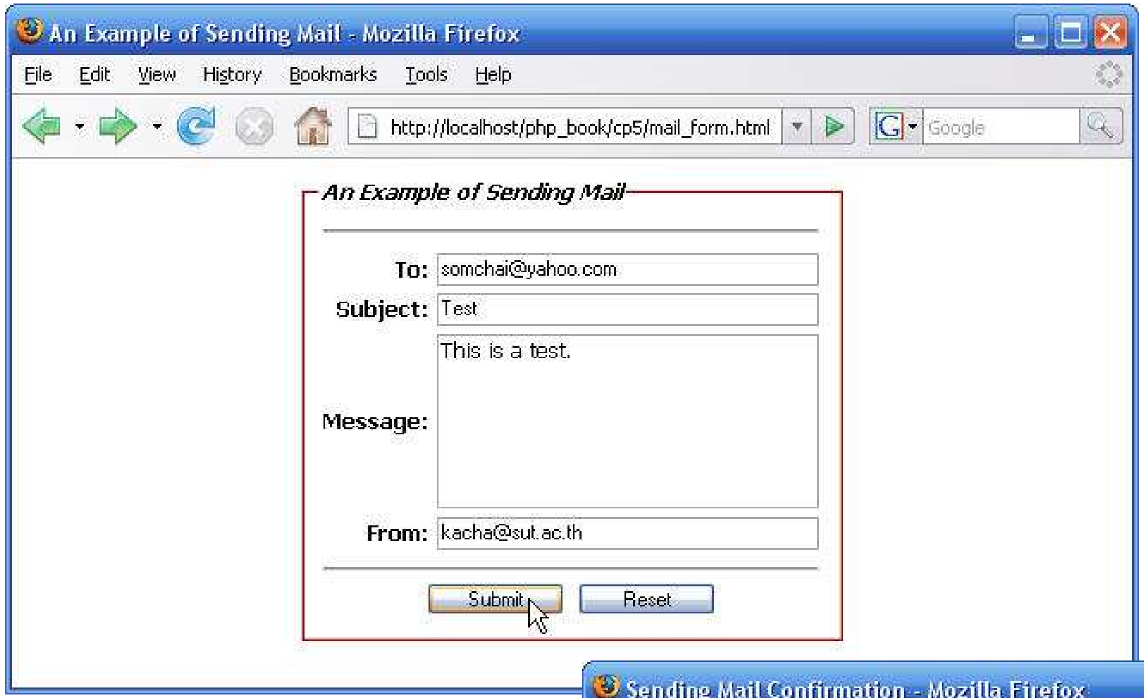
```
<?php
  echo 'abs(-25) = '.abs(-25).'<br>';
  echo 'max(8,9,3,4,1,2,7) = '.max(8,9,3,4,1,2,7).'<br>';
  echo 'min(8,9,3,4,1,2,7) = '.min(8,9,3,4,1,2,7).'<br>';
  echo 'decoct(9) = '.decoct(9).'<br>';
  echo 'decbin(9) = '.decbin(9).'<br>';
  echo 'bindec(1001) = '.bindec(1001).'<br>';
  echo 'ceil(3.25) = '.ceil(3.25). '<br>';
  echo 'floor(5.95) = '.floor(5.95).'<br>';
  echo 'sqrt(9) = '.sqrt(9).'<br>';
  echo 'deg2rad(45) = '.deg2rad(45).'<br>';
  echo 'pow(2,4) = '.pow(2,4).'<br>';
  echo 'rand(1, 30) = '.rand(1,30).'<br>';
?>
```

```
abs(-25) = 25
max(8,9,3,4,1,2,7) = 9
min(8,9,3,4,1,2,7) = 1
decoct(9) = 11
decbin(9) = 1001
bindec(1001) = 9
ceil(3.25) = 4
floor(5.95) = 5
sqrt(9) = 3
deg2rad(45) = 0.785398163397
pow(2,4) = 16
rand(1, 30) = 5
```

Internal or Built-in Functions

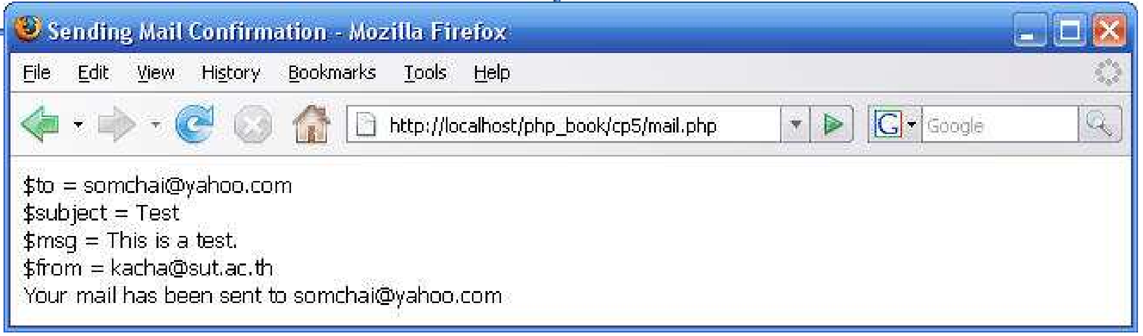
Mail Functions

```
bool mail ( string to, string subject, string message [, string
            additional_headers [, string additional_parameters]] )
```


An Example of Sending Mail - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/php_book/cp5/mail_form.html
Google
Submit
Reset

An Example of Sending Mail

To:	somchai@yahoo.com
Subject:	Test
Message:	This is a test.
From:	kacha@sut.ac.th


Sending Mail Confirmation - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/php_book/cp5/mail.php
Google

```
$to = somchai@yahoo.com
$subject = Test
$msg = This is a test.
$from = kacha@sut.ac.th
Your mail has been sent to somchai@yahoo.com
```

Internal or Built-in Functions

Filesystem Functions

```
<?php
  $dir = "D:";
  echo $dir.'<br>';
  $c_file = 0;
  $c_dir = 0;
  echo "Drive $dir has total space = ".disk_total_space($dir)." Bytes<br>";
  echo "Drive $dir has space available = ".disk_free_space($dir)." Bytes<br>";
  echo "The following information are files and directory in drive $dir<br>";
  if($dh = opendir($dir)){
     while(($name = readdir($dh)) !== false){
       $type = filetype($dir.$name);
       echo "&nbsp;&nbsp;&nbsp;&nbsp;NAME: $name --- TYPE: ".$type."<br>";
       if($type=="dir")
         $c_dir++;
       else
         $c_file++;
     }
     closedir($dh);
  }
  echo "Total File = $c_file<br>Total Directory = $c_dir<br>";
?>
```

Internal or Built-in Functions

- Filesystem Functions

```
D:
Drive D: has total space = 50009669632 Bytes
Drive D: has space available = 46083821568 Bytes
The following information are files and directory in drive D:
    NAME: AppServ --- TYPE: dir
    NAME: background2.jpeg --- TYPE: file
    NAME: bg1.psd --- TYPE: file
    NAME: Documents and Settings --- TYPE: dir
    NAME: fixedtableheads.css --- TYPE: file
    NAME: gd --- TYPE: dir
    NAME: MSOCache --- TYPE: dir
    NAME: Patch --- TYPE: dir
    NAME: patipan2 --- TYPE: dir
    NAME: patipan2.tar.gz --- TYPE: file
    NAME: RECYCLER --- TYPE: dir
    NAME: scroll_column --- TYPE: dir
    NAME: sutmots_April_11_06.06 --- TYPE: dir
    NAME: System Volume Information --- TYPE: dir
    NAME: Winning Eleven 8I --- TYPE: dir
Total File = 4
Total Directory = 11
```

User-defined Functions

Function Arguments

- *Pass by value*

```
function function_name(argument){
        statement;
}
function_name(variable);
```

```
<?php
  function summation($num1,$num2){
    $sum = $num1 + $num2;
    return $sum;
  }
  $n1 = 5;
  $n2 = 20;
  echo "$n1 + $n2 = ".summation($n1,$n2);
?>
```

```
5 + 20 = 25
```

User-defined Functions

Function Arguments

- *Pass by reference*

```
function function_name(argument){
         statement;
}
function_name(&variable);
```

```
<?php
  function new_value(&$num){
    $num = $num * $num;
  }
  $n1 = 20;
  echo 'Before calling the function: $n1 = '.$n1.'<br>';
  new_value($n1);
  echo 'After calling the function: $n1 = '.$n1.'<br>';
?>
```

```
...
...  ($num)
...
...
...
...
new_value(&$n1);
...
...
```

```
Before calling the function: $n1 = 20
After calling the function: $n1 = 400
```

User-defined Functions

Function Arguments

- *Default argument values*

```
function function_name(argument){
        statement;
}
function_name(); // function_name(variable);
```

```
<?php
  function food($type = "steak"){
    return "I like to eat $type.<br>";
  }
  echo food();
  $str = "noodle";
  echo food($str);
  echo food();
?>
```

```
I like to eat steak.
I like to eat noodle.
I like to eat steak.
```

User-defined Functions

- Variable Functions
  - if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it.

```
<?php
  function abc(){
    echo "This is in function abc()<br>";
  }
  function message($year = "2004"){
    echo "Function message(): $year<br>";
  }
  $f = "abc";
  $f();
  $f = "message";
  $f();
  $f(2007);
?>
```

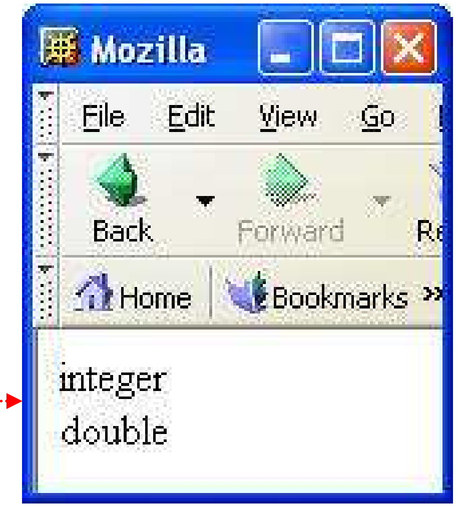
```
This is in function abc()
Function message(): 2004
Function message(): 2007
```

Testing Variable Functions

Testing and Setting Variable Types

- gettype()
  - Prototype: **string gettype(mixed var);**
  - we pass it a variable. It will determine the type and return a string containing the type name, or "unknown type" if it is not one of the standard types; that is, integer, double, string, array, or object.
- settype()
  - Prototype: **bool settype (mixed var, string type);**
  - we pass it a variable that we would like to change the type of, and a string containing the new type for that variable from the previous list.

```
$a = 56;
echo gettype($a).'<br />';
settype($a, 'double');
echo gettype($a).'<br />';
```



```
integer
double
```

Testing Variable Functions

- Testing and Setting Variable Types (cont.)
  - PHP also provides some specific type testing functions. Each of these takes a variable as argument and returns either true or false. The functions are
    - is_array()
    - is_double(), is_float(), is_real() (All the same function)
    - is_long(), is_int(), is_integer() (All the same function)
    - is_string()
    - is_object()

Testing Variable Functions

Testing Variable Status

- PHP has several ways to test the status of a variable.
  - The first of these is **isset()**, which has the following prototype:

    bool isset(mixed var);

    - This function takes a variable name as argument and returns true if it exists and false otherwise.
  - You can wipe a variable out of existence by using its companion construct, **unset()**, which has the following prototype:

    void unset(mixed var);

    - This gets rid of the variable it is passed and returns true.
  - Finally there is **empty()**. This checks to see if a variable exists and has a non-empty, non-zero value. It has the following prototype:

    boolean empty(mixed var);

The following things are considered to be empty
"" (an empty string)
0 (0 as an integer)
0.0 (0 as a float)
"0" (0 as a string)
NULL
FALSE
array() (an empty array)
var $var; (a variable declared, but without a value in a class)

String Manipulations

- Formatting Strings
- Joining and Splitting Strings w/ String Functions
- Comparing Strings
- Matching and Replacing Substring w/ String Functions

String Manipulations

- Formatting Strings
  - Trimming strings: chop(), ltrim(), and trim()
    - The trim() function strips whitespace from the start and end of a string, and returns the resulting string.
    - The characters it strips by default are newlines and carriage returns (\n and \r , horizontal and vertical tabs (\t and \v), end of string characters (\0) , and spaces.
    - The ltrim() function removes whitespace from the start (or left) only.
    - The chop() function removes whitespace from the end (or right) only.
  - Formatting strings for presentation
    - The nl2br() function takes a string as parameter and replaces all the newlines in it with the XHTML <br /> tag (or the HTML <br> tag in versions prior to 4.0.5).

String Manipulations

Formatting Strings (cont.)

Formatting a String for Printing

- PHP also supports a **print()** construct, which does the same thing as **echo**.
- You can apply some more sophisticated formatting using the functions printf() and sprintf().
- The **printf()** prints a formatted string to the browser.
- The **sprintf()** returns a formatted string.
- For example:

```
echo "Total amount of order is $total.";
```

```
printf ("Total amount of order is %s.", $total);
```

- The %s in the format string is called a conversion specification.
- The advantage of printf() is that we can use a more useful conversion specification to specify that $total is actually a floating point number, and that it should have two decimal places after the decimal point, as follows:
  - » printf ("Total amount of order is %.2f", $total);

String Manipulations

Formatting Strings (cont.)

Conversion Specification Type Codes

Type	Meaning
b	Interpret as an integer and print as a binary number.
c	Interpret as an integer and print as a character.
d	Interpret as an integer and print as a decimal number.
f	Interpret as a double and print as a floating point number.
o	Interpret as an integer and print as an octal number.
s	Interpret as a string and print as a string.
x	Interpret as an integer and print as a hexadecimal number with lowercase letters for the digits a-f.
X	Interpret as an integer and print as a hexadecimal number with uppercase letters for the digits A-F.

Changing the Case of String

Function	Description	Use	Value
		$subject	Feedback from web site
strtoupper()	Turns string to uppercase	strtoupper($subject)	FEEDBACK FROM WEB SITE
strtolower()	Turns string to lowercase	strtolower($subject)	feedback from web site
ucfirst()	Capitalizes first character of string if it ’s alphabetic	ucfirst($subject)	Feedback from web site
ucwords()	Capitalizes first character of each word in the string that begins with an alphabetic character	ucwords($subject)	Feedback From Web Site

String Manipulations

- Joining and Splitting Strings w/ String Functions
  - Using **explode(), implode(), and join()**
    - Prototype of function explode()
      - array **explode**(string *separator* , string *input* [, int *limit*]);
      - This function takes a string *input* and splits it into pieces on a specified *separator* string. The pieces are returned in an array.
    - Prototype of implode()
      - string **implode** (string glue, array pieces)
      - This function returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.
    - Function join()
      - This function is an alias of **implode()**.

```
$text = "This is a book.";
$part = explode(" ", $text);
echo $part[0];  //  This
echo $part[1];  //  is

$a  = array('lastname', 'email', 'phone');
$comma = implode(",", $a);
echo $comma; // lastname,email,phone
```

String Manipulations

- Joining and Splitting Strings w/ String Functions (cont.)
  - Using strtok()
    - The function **strtok()** gets pieces (called tokens) from a string one at a time.
    - Function **strtok()** is a useful alternative to using explode() for processing words from a string one at a time.
    - The prototype for strtok()
      - string **strtok**(string *input* , string *separator*);
  - Using substr()
    - The substr() function enables you to access a substring between given start and end points of a string.
    - The substr() function has the following prototype:
      - string **substr**(string *string* , int *start* [, int *length*]);

String Manipulations

- Comparing Strings
  - String Ordering: strcmp(), strcasecmp(), and strnatcmp()
    - The prototype for strcmp() is
      - int **strcmp**(string *str1* , string *str2*);
        - » If they are equal, it will return 0.
        - » If str1 comes after (or is greater than) str2 in lexicographic order, strcmp() will return a number greater than zero.
        - » If str1 is less than str2 strcmp() will return a number less than zero.
        - » This function is case sensitive.
    - The function **strcasecmp()** is identical except that it is not case sensitive.
    - The function **strnatcmp()** compares strings according to a "natural ordering," which is more the way human would do it.
      - For example, strcmp() would order the string "2" as greater than the string "12" because it is lexicographically greater, but strnatcmp() would do it the other way round.
  - Testing String Length with strlen()
    - The function strlen() checks the length of the string and returns its length.
      - For example strlen('hello') returns 5

String Manipulations

Matching and Replacing Substring w/ String Functions

- Finding Strings in Strings: strstr(), strchr(), strrchr(), and stristr()
  - The function **strstr()** is used to find a string or character match within a longer string. In PHP, function **strchr()** is exactly the same as **strstr()**.
  - Prototype of **strstr()**:
    - string **strstr**(string *haystack* , string *needle*);
      - » You pass the function a *haystack* to be searched and a *needle* to be found.
      - » If an exact match of the *needle* is found, the function returns the *haystack* from the *needle* onward, otherwise it returns false.
      - » If the *needle* occurs more than once, the returned string will start from the first occurrence of *needle* .

```
<?php
   $email  = 'user@example.com';
   $domain = strstr($email, '@');
   echo $domain;
?>
```

```
@example.com
```

  - The function **stristr()**, which is nearly identical, but is not case sensitive.
  - The function **strrchr()**, which is again nearly identical ,but will return the *haystack* from the last occurrence of the *needle* onward

String Manipulations

- Matching and Replacing Substring w/ String Functions
  - Finding the Position of a Substring: strpos() and strrpos()
    - The functions **strpos()** and **strrpos()** operate in a similar fashion to **strstr()**, except, instead of returning a substring, they return the numerical position of a *needle* within a *haystack* .
    - The **strpos()** function has the following prototype:
      - int **strpos**(string *haystack* , string *needle* , int [*offset*]);
      - For example, the following code will echo the value 4 to the browser:
        - » $test = 'Hello world';
        - » echo strpos($test, 'o');
      - In this case, we have only passed in a single character as the needle, but it can be a string of any length.
      - The optional offset parameter is used to specify a point within the *haystack* to start searching.
      - For example, the following code will echo the value 7 to the browser:
        - » echo strpos($test, 'o', 5);
    - The **strrpos()** function is almost identical, but will return the position of the last occurrence of the *needle* in the *haystack* .
      - it only works with a single character *needle*

String Manipulations

- Matching and Replacing Substring w/ String Functions
  - Replacing Substrings: str_replace()
  - The function str_replace() has the following prototype:
    - mixed **str_replace**(mixed search, mixed replace, mixed subject [, int &count]) This function will replace all the instances of *search* in *subject* with *replace* and return the new version of the *subject*.

```
<?php
  $str = "I hate PHP script.";
  $new_str = str_replace("hate","LOVE",$str);
  echo $new_str;
?>
```

```
I LOVE PHP script.
```

```
<?php
$vowels = array("a", "e", "i", "o", "u");
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");
  echo $onlyconstants;
?>
```

```
Hll Wrld f PHP
```

String Manipulations

Matching and Replacing Substring w/ String Functions

Replacing Substrings: substr_replace()

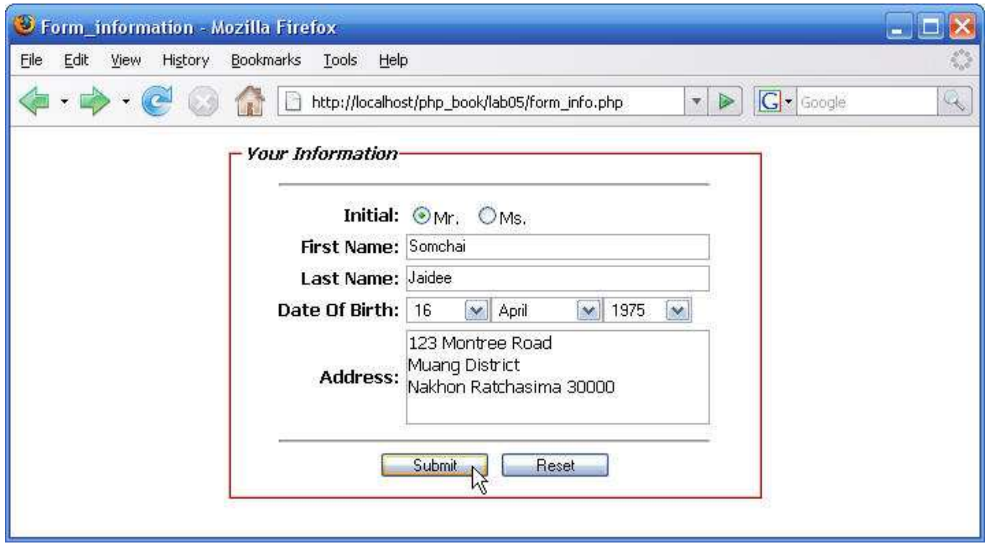
- The function **substr_replace()** is used to find and replace a particular substring of string based on its position. It has the following prototype:
  - string **substr_replace**(string *string* , string *replacement* , int *start* , int *[length]*);
  - This function will replace part of the string *string* with the string *replacement*
    - » if *start* is positive, the replacing will begin at the *start*th offset into *string*.
    - » If *start* is negative, the replacing will begin at the *start*th character from the end of *string*.
    - » If *length* is given and is positive, it represents the length of the portion of *string* which is to be replaced.
    - » If it is negative, it represents the number of characters from the end of *string* at which to stop replacing.
    - » If it is not given, then it will default to strlen (*string*)

```
<?php
  $var = 'ABCDEFGH:/MNOPQR/';
  echo "Original: $var"."<br>";
  echo substr_replace($var, 'xyz', 3)."<br>";
  echo substr_replace($var, 'xyz', -5)."<br>";
  echo substr_replace($var, 'xyz', 0,2)."<br>";
  echo substr_replace($var, 'xyz', 3,-2)."<br>";
?>
```

```
Original: ABCDEFGH:/MNOPQR/
ABCxyz
ABCDEFGH:/MNxyz
xyzCDEFGH:/MNOPQR/
ABCxyzR/
```

Workshop

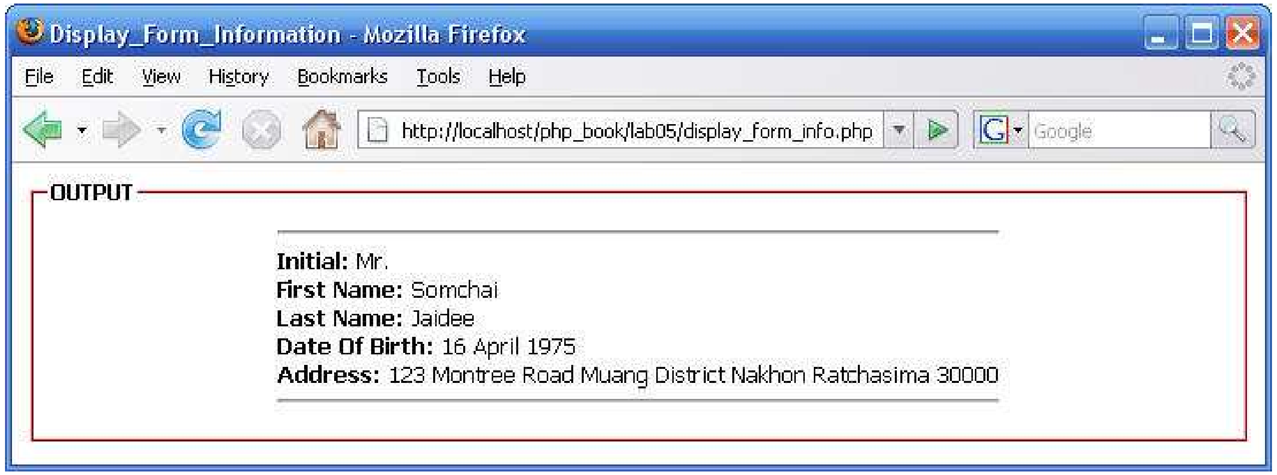
- PHP III: Writing Form and Display Information
  - Page 147 (1-3)


Form_information - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
Submit
Reset

http://localhost/php_book/lab05/form_info.php

Your Information

Initial:	[✓ Mr.] Ms.
First Name:	Somchai
Last Name:	Jaidee
Date Of Birth:	16 April 1975
Address:	123 Montree Road Muang District Nakhon Ratchasima 30000


Display_Form_Information - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google

http://localhost/php_book/lab05/display_form_info.php

OUTPUT

```
Initial: Mr.
First Name: Somchai
Last Name: Jaidee
Date Of Birth: 16 April 1975
Address: 123 Montree Road Muang District Nakhon Ratchasima 30000
```