

Strings

Strings in PHP

- ❖ Strings in PHP can be used to store data other than textual characters. Strings in PHP can be used to store binary data of any kind.

Simple Strings

- ❖ There are many ways through which it is possible to create new strings. Placing textual characters in quotes or double quotes is one of them.

```
$text = 'ABC';
```

```
$text = "ABC";
```

- ❖ When using double quotes we can add special escape sequences directly within the string (e.g. special characters such as "Hello\n").

Embedded Variables

- ❖ We can embed variables directly within a double quoted string simply by writing their names.

```
echo "Hallo $person_name\n";
```

```
echo "temp_variable=$temp_variable";
```

The Here doc

- ❖ Creating complex strings can be done via two ways. It can be done using the double quotes – as explained in the previous slide – and it can be done using the Here doc syntax.

<<<

...

...

...

...

_____;

The identifier we choose...

The identifier we choose...

The Heredoc

```
<?php
$text_var = <<< TOKTOK
We wish you a happy birthday!
All the best!
Regards,
Friends.
TOKTOK;
echo $text_var;
?>
```

The Backslash

- ❖ When creating a string using single quotes, we can include single quotes using the backslash.

```
echo 'I love my home\'s atmosphere';
```

- ❖ When creating a string using double quotes, we don't need to use the backslash in order to include single quotes.

```
echo "I love my home's atmosphere";
```

- ❖ When creating a string using double quotes, we can include double quotes and dollar signs prefixed with backslash.

```
echo "The US \$ value is \"4.4\"";
```

The Backslash

```
<?php
echo '<BR>I love Europe\'s weather';
echo "<BR>I US \$ currency";
echo "<BR>Here is my home's cake";
?>
```


The `strlen()` Function

- ❖ Calling the `strlen()` function you can get the length of a given string.

The strlen() Function

```
<?php
$text_1 = "abcdef";
$text_2 = "abc def ";
$text_3 = "\n";
$length_1 = strlen($text_1);
$length_2 = strlen($text_2);
$length_3 = strlen($text_3);
echo "<BR>The length of \$text_1 is $length_1";
echo "<BR>The length of \$text_2 is $length_2";
echo "<BR>The length of \$text_3 is $length_3";
?>
```

The Output

The length of \$text_1 is 6
The length of \$text_2 is 8
The length of \$text_3 is 1

The `strtr()` Function

- ❖ This function returns a new string received after changing all occurrences of each one of the characters in `$from` to its equivalent character in `$to`. If the two strings have a different length, the extra characters in the longer one are ignored.

```
string strtr ( string $str , string $from , string $to )  
string strtr ( string $str , array $replace_pairs )
```

The `strtr()` Function

- ❖ The second version of this function returns a new string after repeatedly replacing each string (key) in `$replace_pairs` with its value.

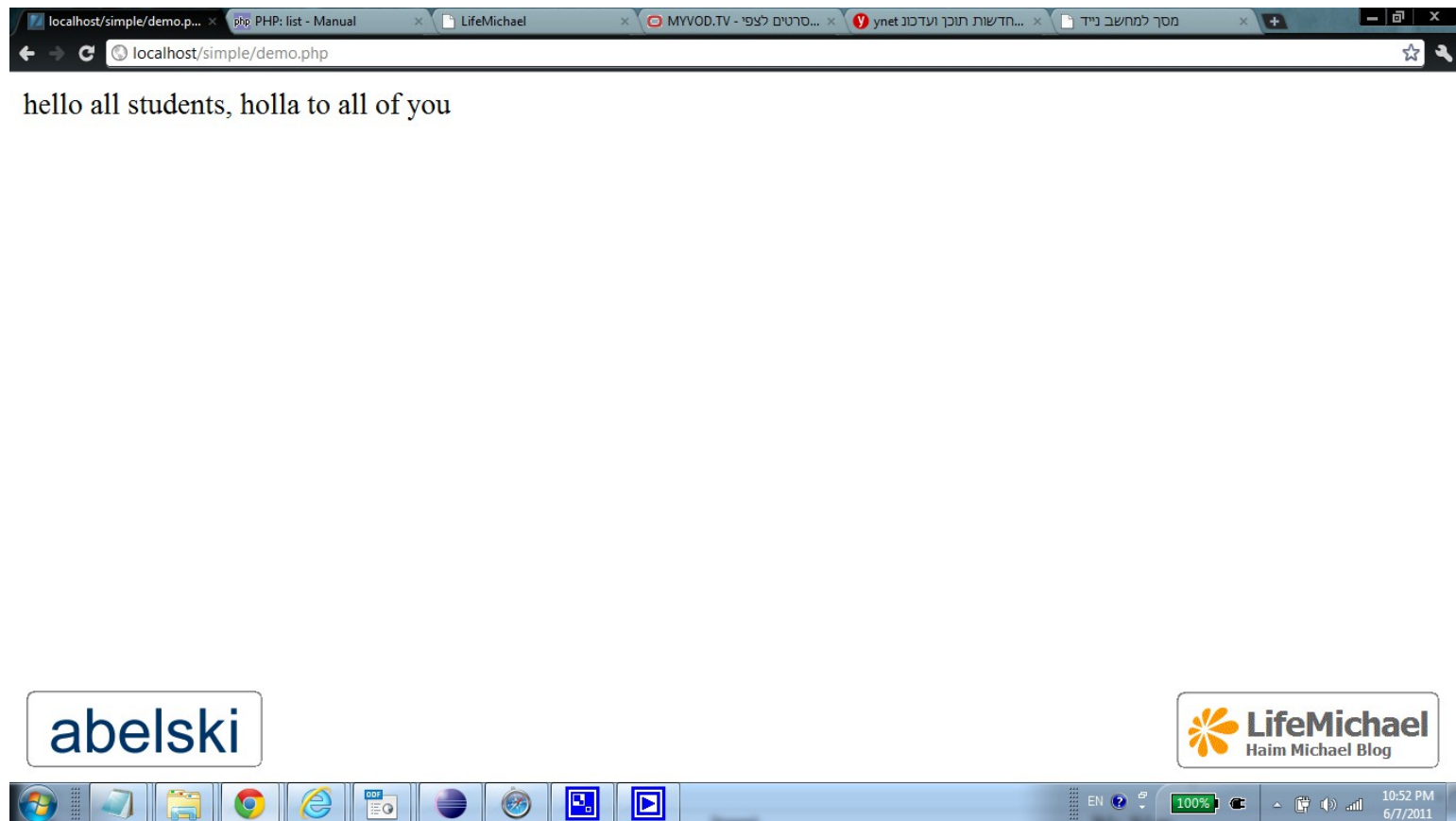
```
string strtr ( string $str , string $from , string $to )  
string strtr ( string $str , array $replace_pairs )
```

The `strtr()` Function

```
<?php
$trans = array("hello" => "holla", "hi" => "hello");
echo strtr("hi all students, hello to all of you", $trans);
?>
```



The `strstr()` Function



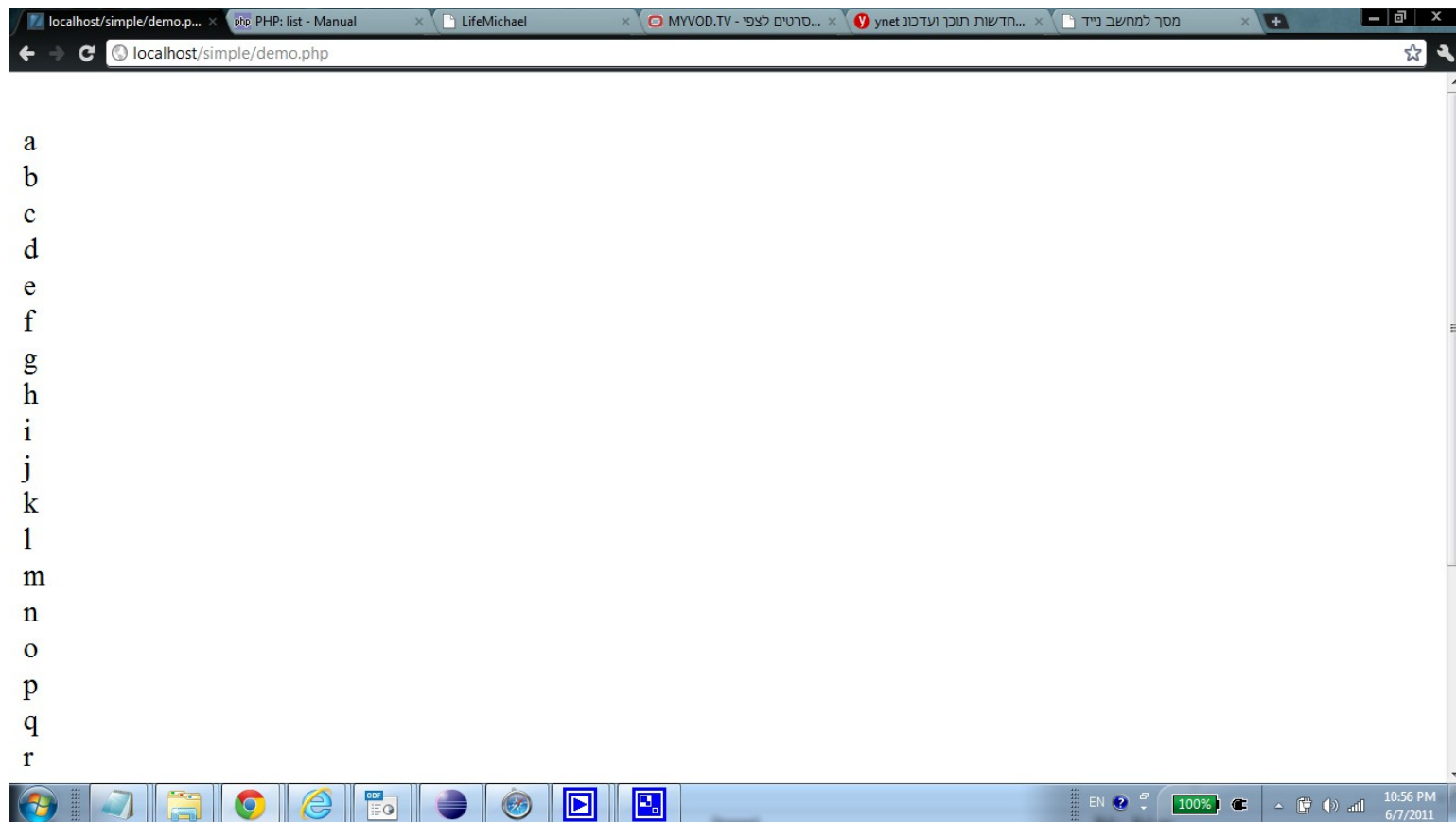
Strings as Arrays

- ❖ Each string can be treated as if it was an array.

```
<?php
$str = "abcdefghijklmnopqrstuvwxyz";
$length = strlen($str);
for($index=0; $index<$length; $index++)
{
    echo "<BR>$str[$index]";
}
?>
```



Strings as Arrays



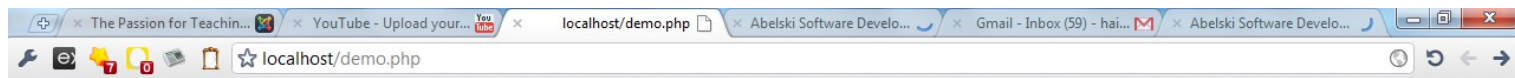
Strings Comparison

- ❖ Comparing strings using the == comparison operator is involved with a transparent conversion of textual strings to numeric values.

```
<?php
$temp_1 = '1978year';
$temp_2 = 1978;
if($temp_1==$temp_2)
    echo "'$temp_1' and $temp_2 are equal";
else
    echo "no";
?>
```



Strings Comparison



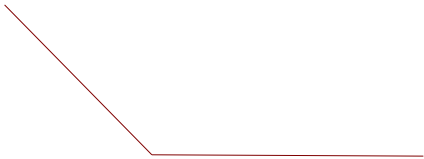
'1978year' and 1978 are equal



Strings Comparison

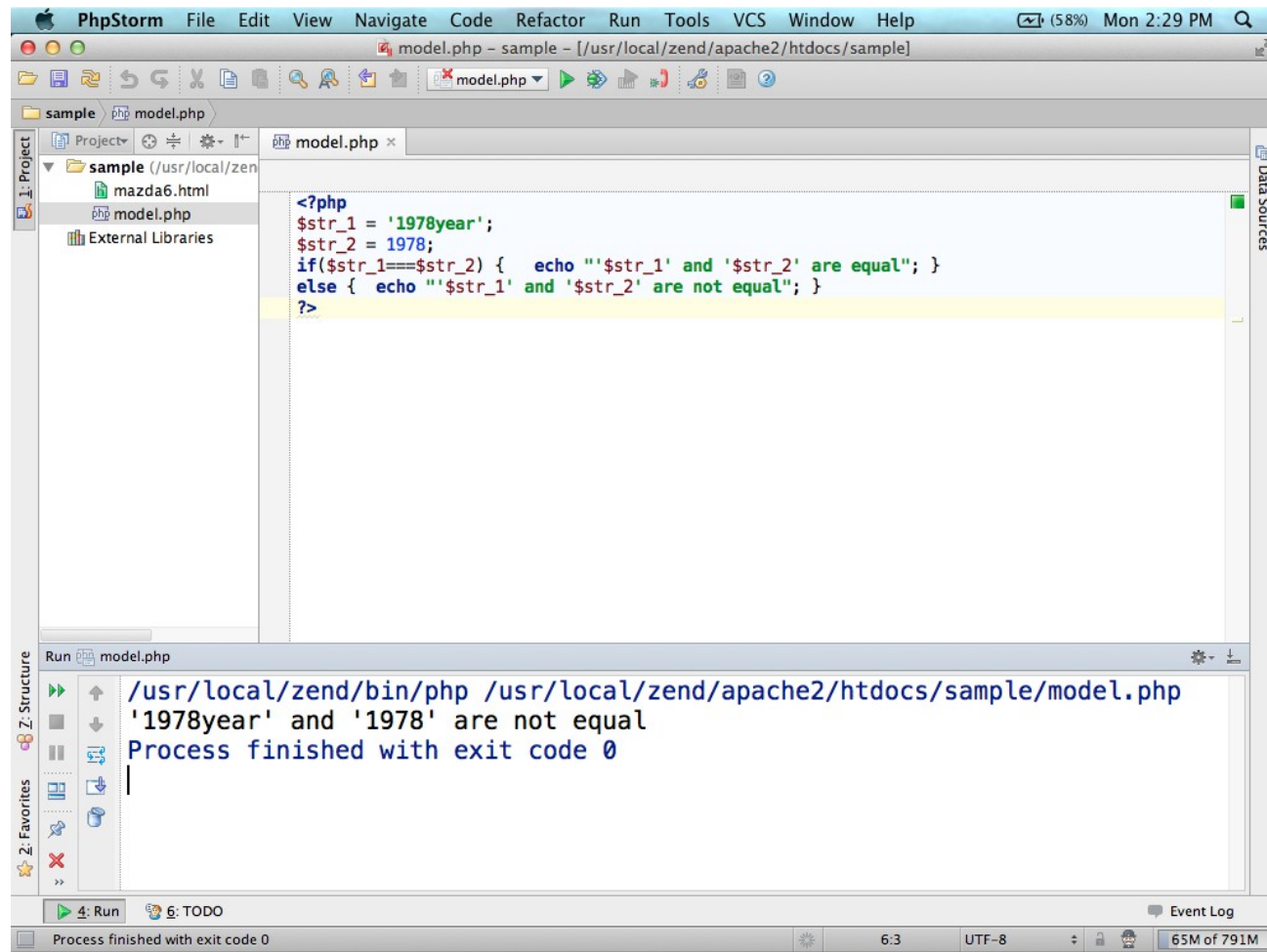
- ❖ Using the identity === operator is not involved with an automatic transparent conversion.

```
<?php
$str_1 = '1978year';
$str_2 = 1978;
if($str_1=== $str_2) { echo "'$str_1' and '$str_2' are equal"; }
else { echo "'$str_1' and '$str_2' are not equal"; }
?>
```



'1978year' and '1978' are not equal

Strings Comparison



The screenshot shows the PhpStorm IDE interface. The main editor window displays a PHP file named `model.php` with the following code:

```
<?php
$str_1 = '1978year';
$str_2 = 1978;
if($str_1== $str_2) { echo "'$str_1' and '$str_2' are equal"; }
else { echo "'$str_1' and '$str_2' are not equal"; }
?>
```

The code is highlighted in yellow. The left sidebar shows the project structure with a folder named `sample` containing `mazda6.html` and `model.php`. The bottom panel shows the Run output for `model.php`:

```
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
'1978year' and '1978' are not equal
Process finished with exit code 0
```

The status bar at the bottom indicates the process finished with exit code 0, the file is 6.3 KB, and the encoding is UTF-8.

The `strcmp()` & `strcasecmp()` Functions

- ❖ Both `strcmp()` and `strcasecmp()` receive two strings and compare them. They work the same, except that the former is case sensitive. Both functions return 0 when the two strings are equal.

```
int strcmp ( string $str1 , string $str2 )
```

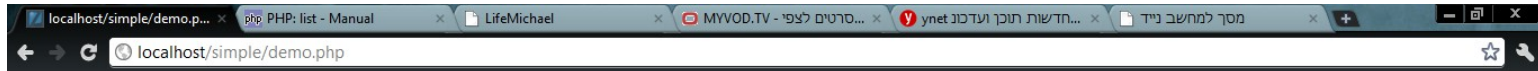
```
int strcasecmp ( string $str1 , string $str2 )
```

The strcmp () & strcasecmp () Functions

```
<?php
$str1 = "Turkey";
$str2 = "turkey";
if (strcasecmp($str1, $str2) == 0)
{
    echo '<BR>$str1 is equal to $str2 in a case insensitive comparison';
}
else
{
    echo '<BR>$str1 is not equal to $str2 in a case insensitive comparison';
}
?>
```



The strcmp() & strcasecmp() Functions



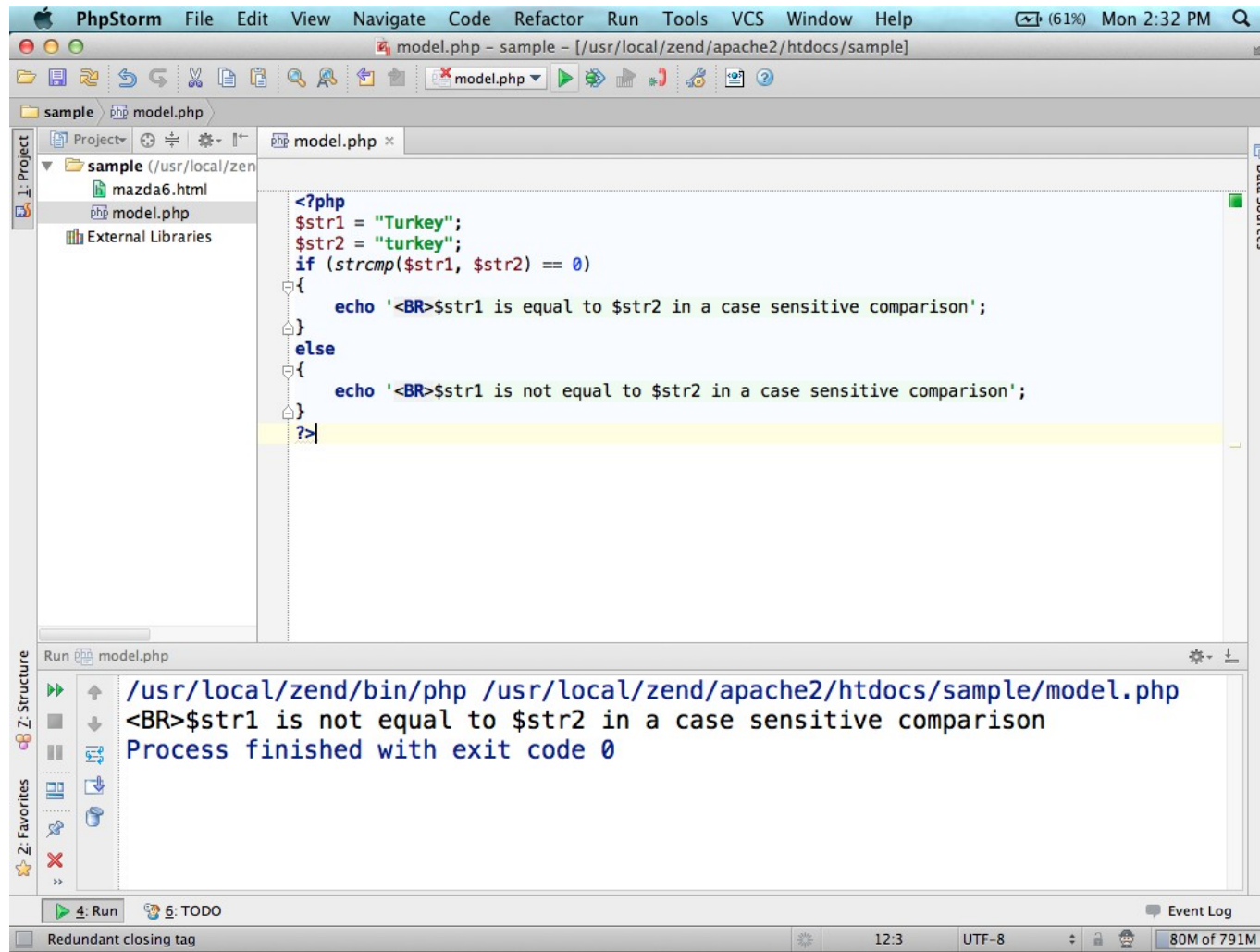
\$str1 is equal to \$str2 in a case insensitive comparison



The strcmp() & strcasecmp() Functions

```
<?php
$str1 = "Turkey";
$str2 = "turkey";
if (strcmp($str1, $str2) == 0)
{
    echo '<BR>$str1 is equal to $str2 in a case sensitive comparison';
}
else
{
    echo '<BR>$str1 is not equal to $str2 in a case sensitive comparison';
}
?>
```


The strcmp() & strcasecmp() Functions



The screenshot shows the PhpStorm IDE interface. The main editor displays a PHP file named `model.php` with the following code:

```
<?php
$str1 = "Turkey";
$str2 = "turkey";
if (strcmp($str1, $str2) == 0)
{
    echo '<BR>$str1 is equal to $str2 in a case sensitive comparison';
}
else
{
    echo '<BR>$str1 is not equal to $str2 in a case sensitive comparison';
}
?>
```

The left sidebar shows the project structure with a folder named `sample` containing `mazda6.html` and `model.php`. The bottom panel shows the execution output for `model.php`:

```
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
<BR>$str1 is not equal to $str2 in a case sensitive comparison
Process finished with exit code 0
```

The status bar at the bottom indicates a "Redundant closing tag" warning, the time is 12:3, the encoding is UTF-8, and the memory usage is 80M of 791M.

The `strncasecmp()` Function

- ❖ The `strncasecmp()` function enables to set the number of characters you want to compare.

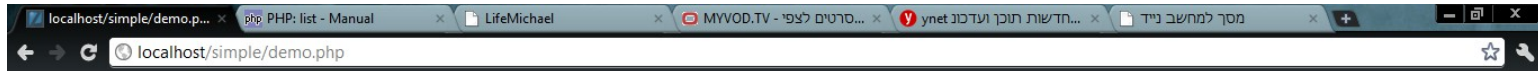
```
int strncasecmp ( string $str1 , string $str2 , int $len )
```

The strncasecmp () Function

```
<?php
$str1 = "lovsdfsdysdy Turkey";
$str2 = "Lovely turkey country!";
if (strncasecmp($str1, $str2,3) == 0)
{
    echo 'true';
}
else
{
    echo 'false';
}
?>
```



The `strncasecmp()` Function



true

abelski

 **LifeMichael**
Haim Michael Blog



The `strpos()` & `strstr()` Functions

- ❖ The `strpos()` & `strstr()` functions families provide the simplest way to search for substrings.

```
int strpos ( string $haystack ,  
             mixed $needle [, int $offset ] )
```

```
string strstr ( string $haystack ,  
               string $needle , bool $before_needle )
```

The strpos() & strstr() Functions

```
int strpos ( string $haystack ,  
            mixed $needle [, int $offset ] )
```

This function returns the index position of the first occurrence of needle within the haystack. The optional offset parameter allows specifying from which character to start the search.

The strpos() & strstr() Functions

```
string strstr ( string $haystack , string $needle ,  
               bool $before_needle )
```

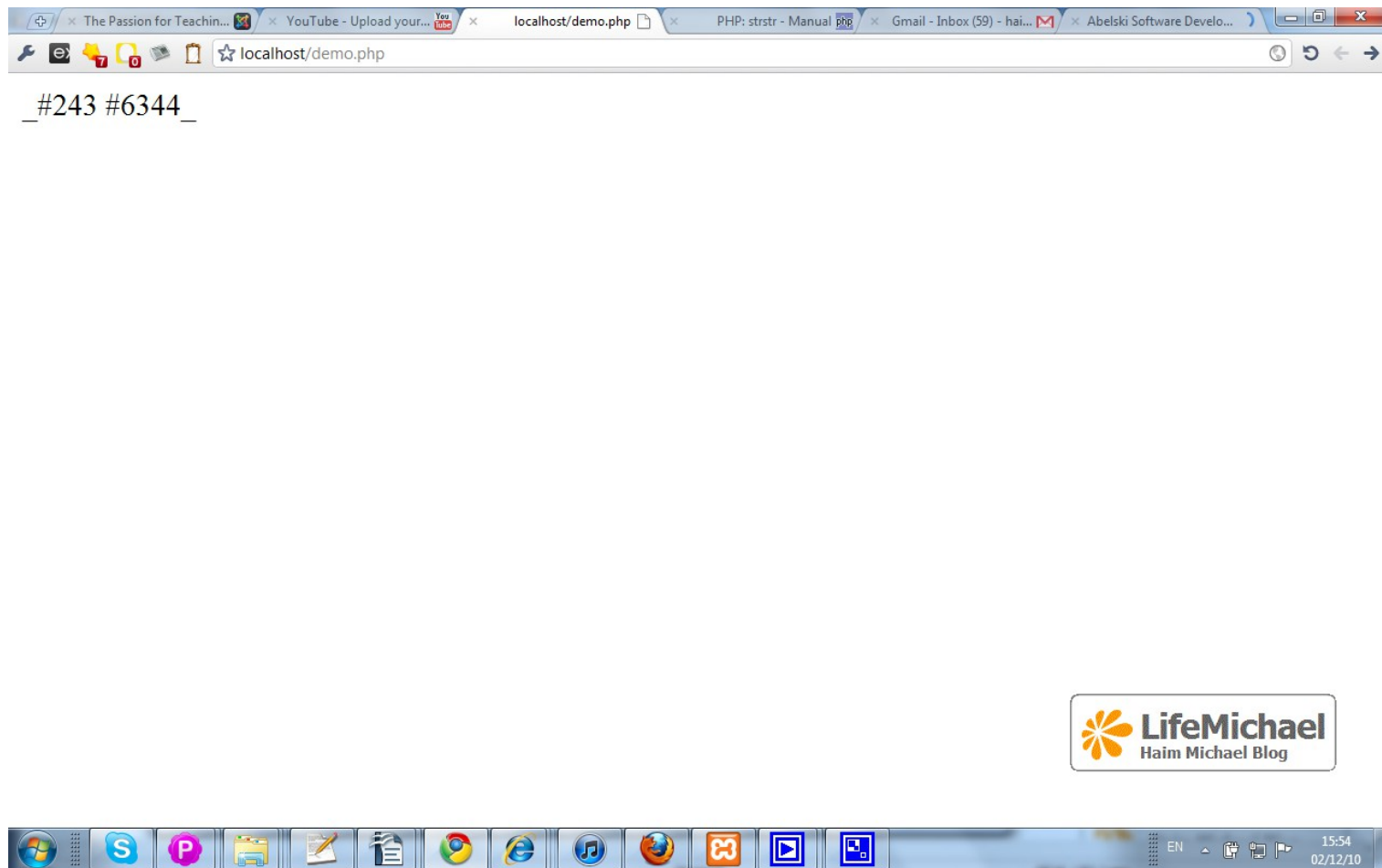
This function returns the substring that starts with the first occurrence of the needle within the haystack till the end of the haystack. If the third parameter is true then this function returns the substring from the beginning of the haystack till the needle. Default is false.

The strpos() & strstr() Functions

```
<?php  
$telephone = '972 54 5544232 #243 #6344';  
$extension = strstr($telephone, '#');  
echo "_".$extension."_";  
?>
```



The strpos() & strstr() Functions

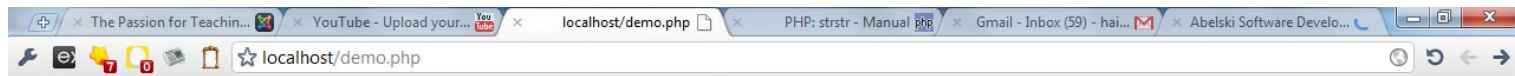


The strpos() & strstr() Functions

```
<?php
$telephone = '972 54 5544232 #243 #6344';
$extension = strpos($telephone, '#');
echo $extension;
?>
```



The strpos() & strstr() Functions



15



The `strpos()` & `strpos()` Functions

- ❖ The `strpos()` & `strpos()` do the same work
`strpos()` & `strpos()` do... with one difference: The
`strpos()` & `strpos()` are not case sensitive.

The `strrpos()` Function

- ❖ The `strrpos()` does the same work `strpos()` does...
with one difference. The `strrpos()` starts the search in a reverse order.

The `strspn()` & `strcspn()` Functions

- ❖ The `strspn()` function returns the length of the initial biggest matching mask.

```
int strspn ( string $str1 , string $str2  
            [, int $start [, int $length ]] )
```

This function returns the length of the initial segment of `str1` which consists entirely of characters in `str2` .

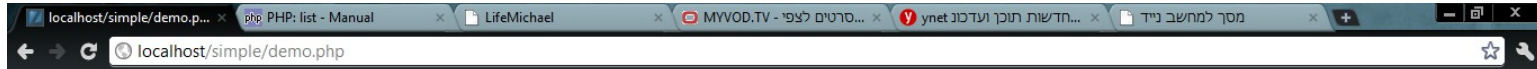
- ❖ The `strcspn()` function works the opposite. It returns the length of the initial segment of the string that doesn't contain any of the characters of `str2`.

The strstr() & strpos() Functions

```
<?php
$a = "i love java and php";
$b = "love java";
$num = strpos($a, $b);
echo $num;
?>
```



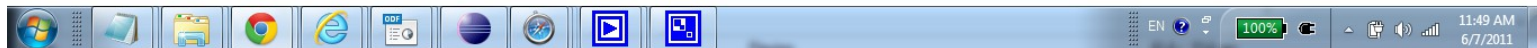
The `strspn()` & `strcspn()` Functions



0

abelski

 **LifeMichael**
Haim Michael Blog



The `str_replace()` Function

- ❖ The `str_replace()` function replaces all occurrences of the searched string with the replacement one.

```
mixed str_replace ( mixed $search ,  
                    mixed $replace ,  
                    mixed $subject [, int &$count ] )
```

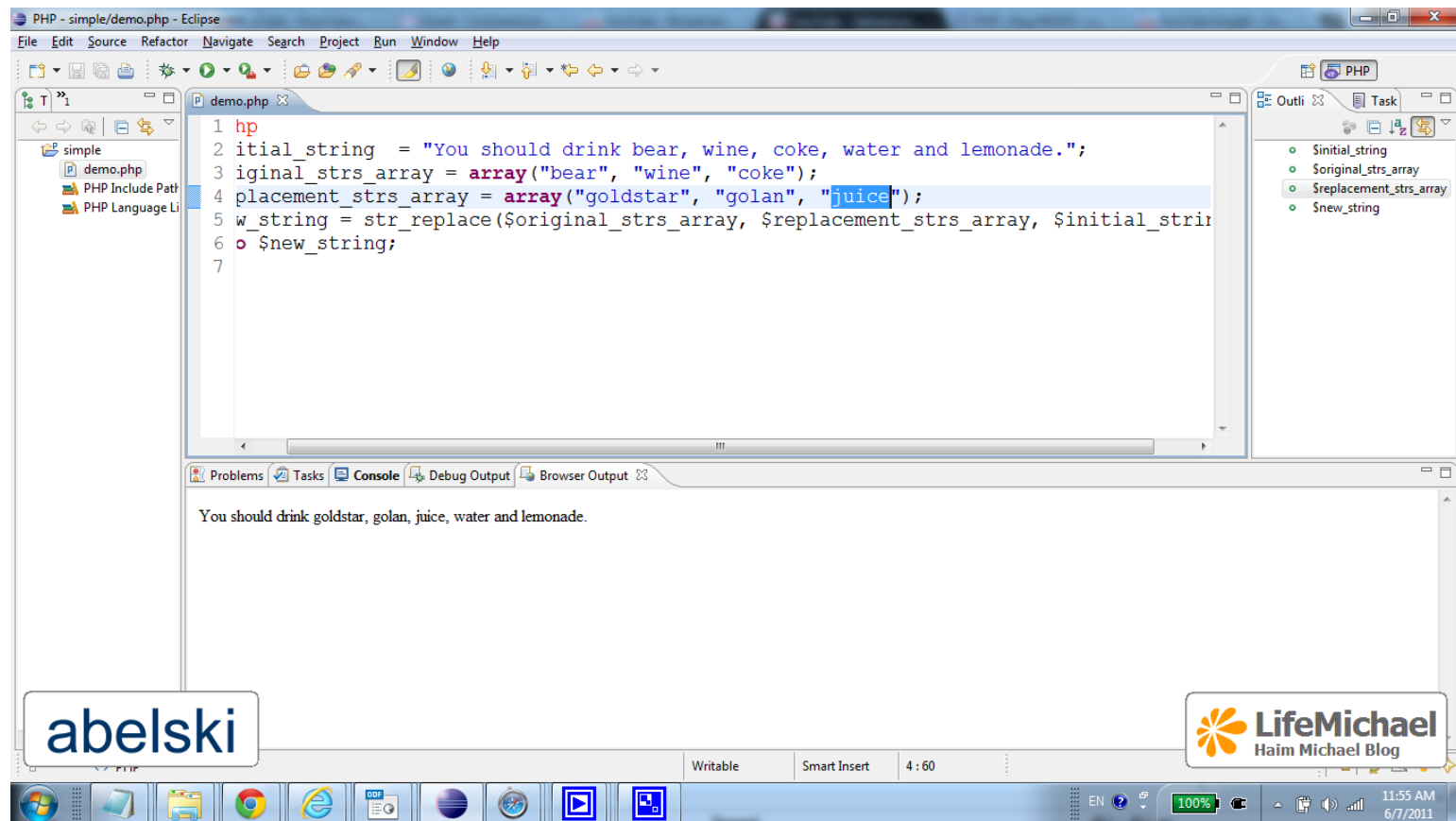
The `str_replace` function goes over the `$subject` string and replaces each `$search` occurrence with `$replace`. The `&$count` parameter is an optional one. If we pass it, the function will fill it with the number of substitutions. The `$replace` and `$search` parameters we pass can be arrays of values, which allows us to replace more than one string (process from left to right).

The str_replace() Functions

```
<?php
$initial_string = "You should drink bear, wine, coke, water and lemonade.";
$original_strs_array = array("bear", "wine", "coke");
$replacement_strs_array = array("goldstar", "golan", "juice");
$new_string = str_replace($original_strs_array, $replacement_strs_array,
    $initial_string);
echo $new_string;
?>
```



The str_replace() Functions



```
1 hp
2 $initial_string = "You should drink bear, wine, coke, water and lemonade.";
3 $original_strs_array = array("bear", "wine", "coke");
4 $replacement_strs_array = array("goldstar", "golan", "juice");
5 $new_string = str_replace($original_strs_array, $replacement_strs_array, $initial_string);
6
7
```

You should drink goldstar, golan, juice, water and lemonade.

abelski

LifeMichael
Haim Michael Blog

Writable Smart Insert 4 : 60

11:55 AM
6/7/2011

The substr_replace() Function

- ❖ The `substr_replace()` function replaces a string with another string in a character its index is passed to the function.

```
mixed substr_replace ( mixed $string , string $replacement ,  
                      int $start [, int $length ] )
```

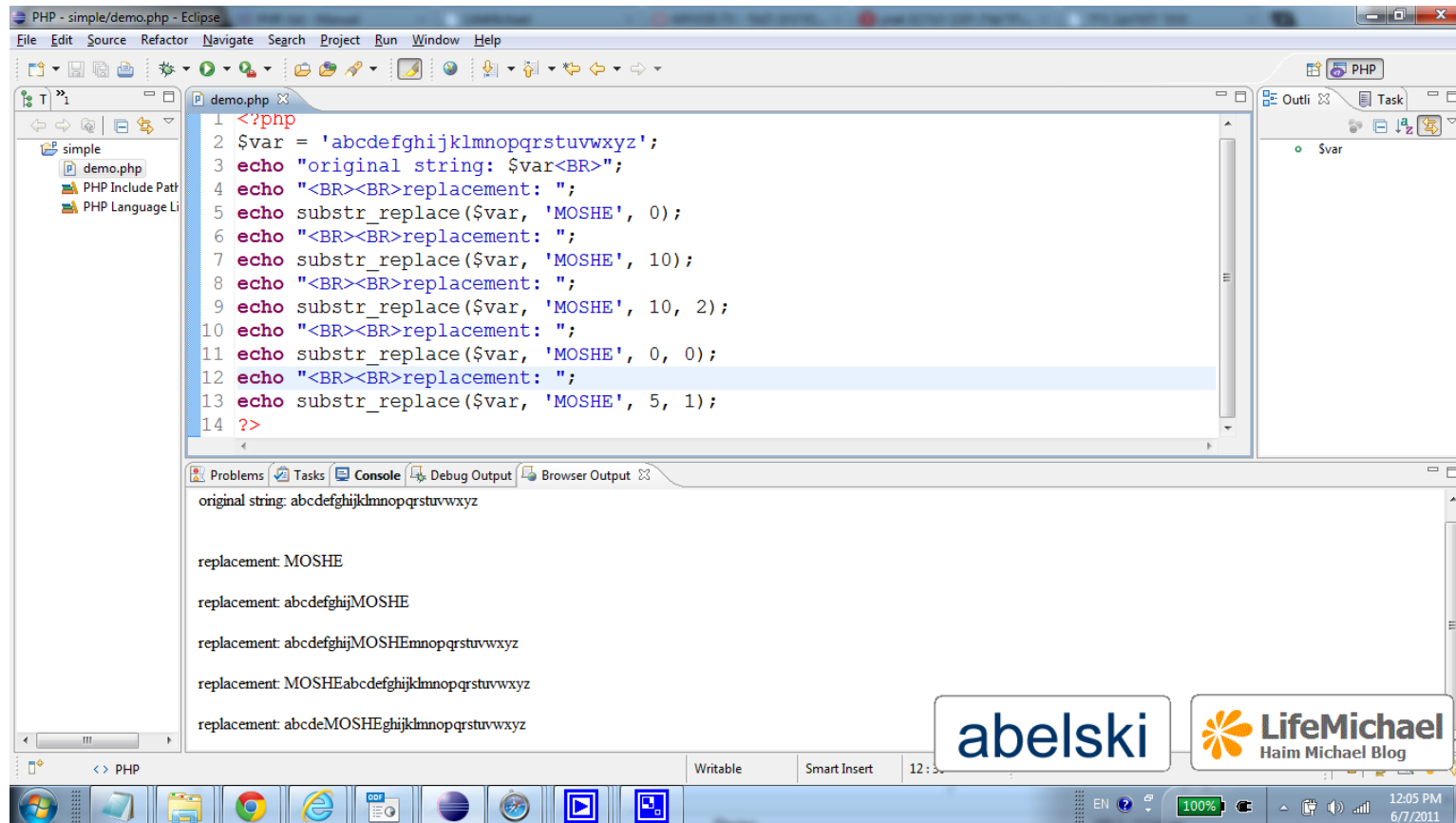
The `substr_replace` replaces portion of `$string` with `$replacement` starting in index `$start`. The optional `$length` parameter specifies the length of the string that will be replaced (out of `$string`).

The substr_replace() Function

```
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string: $var<BR>";
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 0);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 10);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 10, 2);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 0, 0);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 5, 1);
?>
```



The substr_replace() Function



```
1 <?php
2 $var = 'abcdefghijklmnopqrstuvxyz';
3 echo "original string: $var<BR>";
4 echo "<BR><BR>replacement: ";
5 echo substr_replace($var, 'MOSHE', 0);
6 echo "<BR><BR>replacement: ";
7 echo substr_replace($var, 'MOSHE', 10);
8 echo "<BR><BR>replacement: ";
9 echo substr_replace($var, 'MOSHE', 10, 2);
10 echo "<BR><BR>replacement: ";
11 echo substr_replace($var, 'MOSHE', 0, 0);
12 echo "<BR><BR>replacement: ";
13 echo substr_replace($var, 'MOSHE', 5, 1);
14 ?>
```

original string: abcdefghijklmnopqrstuvxyz

replacement: MOSHE

replacement: abcdefghijMOSHE

replacement: abcdefghijMOSHEmnopqrstuvxyz

replacement: MOSHEabcdefghijklmnopqrstuvxyz

replacement: abcdeMOSHEghijklmnopqrstuvxyz

The `substr()` Function

- ❖ The `substr()` function extracts a substring out of a given string.

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

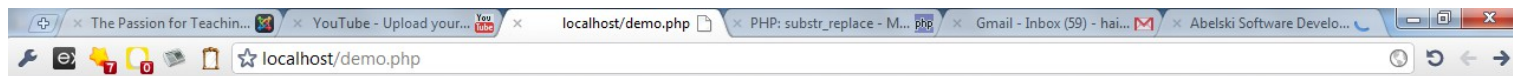
The `$string` parameter is the original string. The `$start` is the index from which the substring will start. If `$start` is negative the returned substring will start from the character its index is counted from the end.

The substr () Function

```
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string...<BR>$var<BR>";
echo "<BR><BR>substring...<BR>";
echo substr($var,0);
echo "<BR><BR>substring...<BR>";
echo substr($var,2,4);
?>
```



The substr () Function



original string...
abcdefghijklmnopqrstuvwxyz

substring...
abcdefghijklmnopqrstuvwxyz

substring...
cdef



The `setlocale()` Function

- ❖ Formatting rules are generally affected by the geographic location.
- ❖ The `setlocale()` method enables us setting a specific locale setting and indicate which functions will be affected by the change.

```
string setlocale ( int $category , array $locale )  
string setlocale ( int $category ,  
                  string $locale [, string $... ] )
```

The `setlocale()` Function

The `$category` parameter can be one of the following:

`LC_ALL` for all of the below

`LC_COLLATE` for string comparison.

`LC_CTYPE` for character classification and conversion, for example `strtoupper()`

`LC_MONETARY` for `localeconv()`

`LC_NUMERIC` for decimal separator (See also `localeconv()`)

`LC_TIME` for date and time formatting with `strftime()`

`LC_MESSAGES` for system responses (available if was compiled with `libintl`)

The `setlocale()` Function

The `$locale` parameter can be any of the valid values according to <http://www.w3.org/WAI/ER/IG/ert/iso639.htm>.

e.g.

`en_US`

`de_DE`

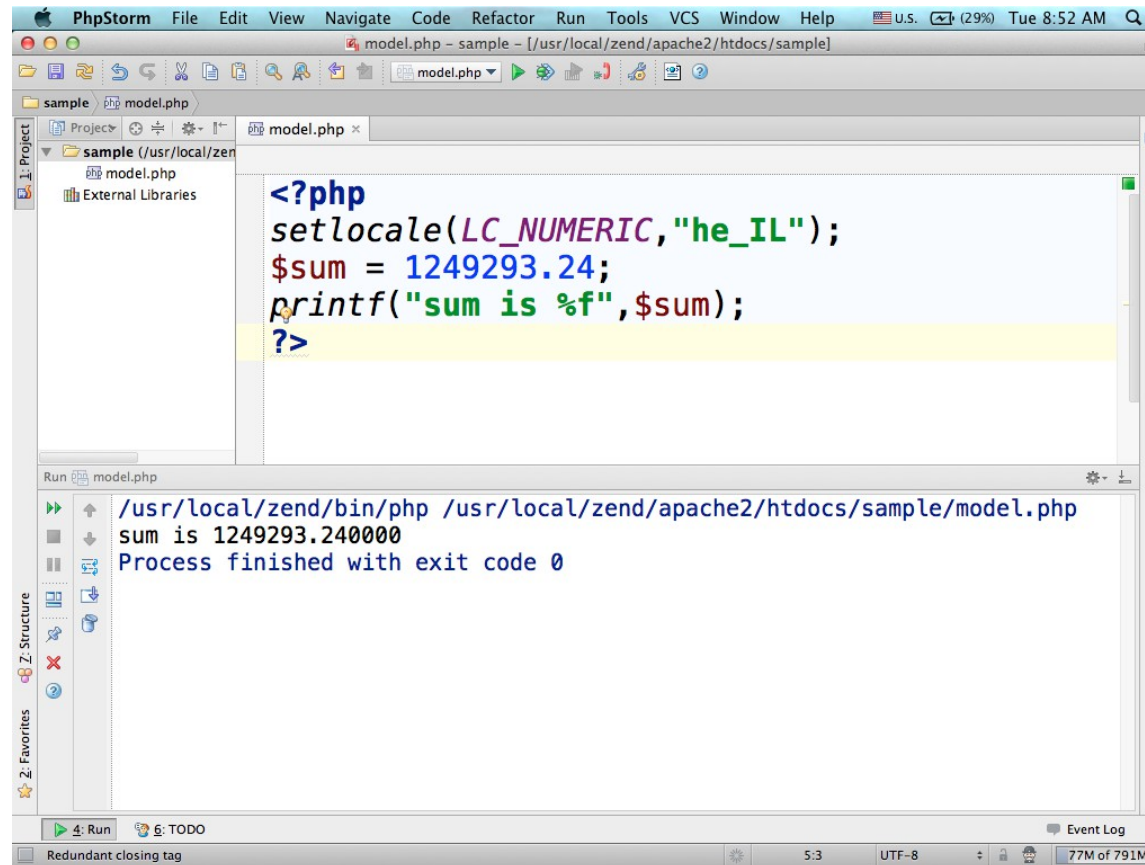
`fr_FR`

The `setlocale()` Function

```
<?php
setlocale(LC_NUMERIC, "he_IL");
$sum = 1249293.24;
printf("sum is %f", $sum);
?>
```



The `setlocale()` Function



The screenshot shows the PhpStorm IDE interface. The main editor window displays a PHP script in `model.php` with the following code:

```
<?php
setlocale(LC_NUMERIC,"he_IL");
$sum = 1249293.24;
printf("sum is %f",$sum);
?>
```

The script is executed, and the output is shown in the Run window:

```
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
sum is 1249293.240000
Process finished with exit code 0
```

The status bar at the bottom indicates a "Redundant closing tag" warning and shows the file size as 77M of 791M.

The `number_format()` Function

- ❖ The `number_format()` function is used for output a number and separated its digits into thousands, millions etc.

```
string number_format ( float $number [, int $decimals])  
string number_format ( float $number [, int $decimals  
                        [, string $dec_point ]], string $thousands_sep )
```

The `money_format()` Function

❖ The `money_format()` function is used for output a currency.

```
string money_format ( string $format , float $number )
```

This method can be called passing two parameters. The `$number` parameter is the number that should be formatted. The `$format` parameter can be composed of various special characters.

The `printf()` Function

❖ The `printf()` function allows formatting our text in various ways.

❖ The `printf()` function simply writes to the script's output.

```
int printf ( string $format [, mixed $args [, mixed $... ]] )
```

❖ The `sprintf()` function simply returns it.

```
string sprintf ( string $format [, mixed $args  
                                                    [, mixed $... ]] )
```

❖ The `fprintf()` function simply writes to a file.

```
int fprintf ( resource $handle , string $format  
                                                    [, mixed $args [, mixed $... ]] )
```

The `printf()` Function

❖ The `$format` is composed of zero (or more) of the following directives:

Ordinary Characters

They are copied directly to the result. The '%' doesn't belong to this group of directives.

Conversion Specification

Each one of the conversion specifications results in fetching its parameter. Each conversion specification consists of a percent sign (%), followed by one or more of the following elements: sign specifier, padding specifier, alignment specifier, width specifier, precision specifier and a type specifier.

The `printf()` Function

- ❖ The conversion specifications include the following possibilities:

Sign Specifier

This optional sign specifier forces the '-' / '+' sign to be used. The sign specifier forces showing the sign symbol (even when trying to force presenting '+' and the number is already positive... the '+' will be added).

```
printf("%%+d = '%+d'\n", $n);
```

Padding Specifier

This optional padding specifier tells which character to use for padding the result to the right string size. The ' should prefix that character unless it is '0' or a simple space ' '.

```
printf("[%'#12s]\n", $str);
```

The `printf()` Function

Alignment Specifier

This optional sign specifier tells whether the result should be left justified or right justified. The default is right justified. Adding '-' after the '%' will change its alignment to be left justified.

```
printf("[% -10s]\n", $str);
```

Width Specifier

This optional number says how many characters (at the minimum) the conversion result should be.

```
printf("[% '#8s]\n", $s);
```

The `printf()` Function

Precision Specifier

This optional specifier says how many decimal digits should be displayed for a floating point numbers.

```
printf("%.2f", $sum);
```

//results in two digits after the decimal point.

Type Specifier

The type specifier says what type the arguments data should be treated as.

b - treated as integer and presented as a binary number.

c - treated as integer and presented as a character.

d - treated as integer and presented as a signed decimal integer number.

e - treated as a scientific notation (e.g. 1.5e4).

The `printf()` Function

- u - treated as integer and presented as an unsigned decimal number.
- f - treated as float and presented as a floating point number (local aware).
- F- treated as float and presented as a floating point number (non local aware).
- o - treated as integer and presented as an octal number.
- s - treated as presented as a string.
- x - treated as integer and presented as hexadecimal number (lowercase).
- X - treated as integer and presented as hexadecimal number (Uppercase).

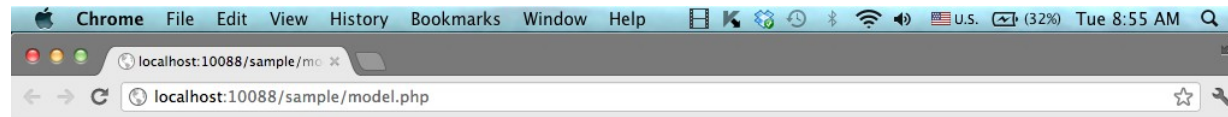
The printf() Function

```
<?php
$positive_number = 43951789;
$negative_number = -43951789;
$simple_character = 65; // ASCII 65 is 'A'
$simple_string = "mosh";
$long_string = "moshiko goes to seven grade";
printf("%%b = '%b'<BR>", $positive_number);
printf("%%c = '%c'<BR>", $simple_character);
printf("%%d = '%d'<BR>", $positive_number);
printf("%%e = '%e'<BR>", $positive_number);
printf("%%u = '%u'<BR>", $positive_number);
printf("%%u = '%u'<BR>", $negative_number);
```

The printf () Function

```
printf("%%f = '%f'<BR>", $positive_number);  
printf("%%o = '%o'<BR>", $positive_number);  
printf("%%s = '%s'<BR>", $positive_number);  
printf("%%x = '%x'<BR>", $positive_number);  
printf("%%X = '%X'<BR>", $positive_number);  
printf("_%s_<BR>",      $simple_string);  
printf("_%20s_<BR>",    $simple_string);  
printf("_%-20s_<BR>",   $simple_string);  
printf("_%020s_<BR>",   $simple_string);  
printf("_%'#20s_<BR>",  $simple_string);  
printf("%.3e<BR>", $positive_number);  
?>
```


The `printf()` Function



```
%b = '10100111101010011010101101'
%c = 'A'
%d = '43951789'
%e = '4.395179e+7'
%u = '43951789'
%u = '4251015507'
%f = '43951789.000000'
%o = '247523255'
%s = '43951789'
%x = '29ea6ad'
%X = '29EA6AD'
_mosh_
_mosh_
_mosh_
_0000000000000000mosh_
_#####mosh_
4.395e+7
```

The `sscanf()` Function

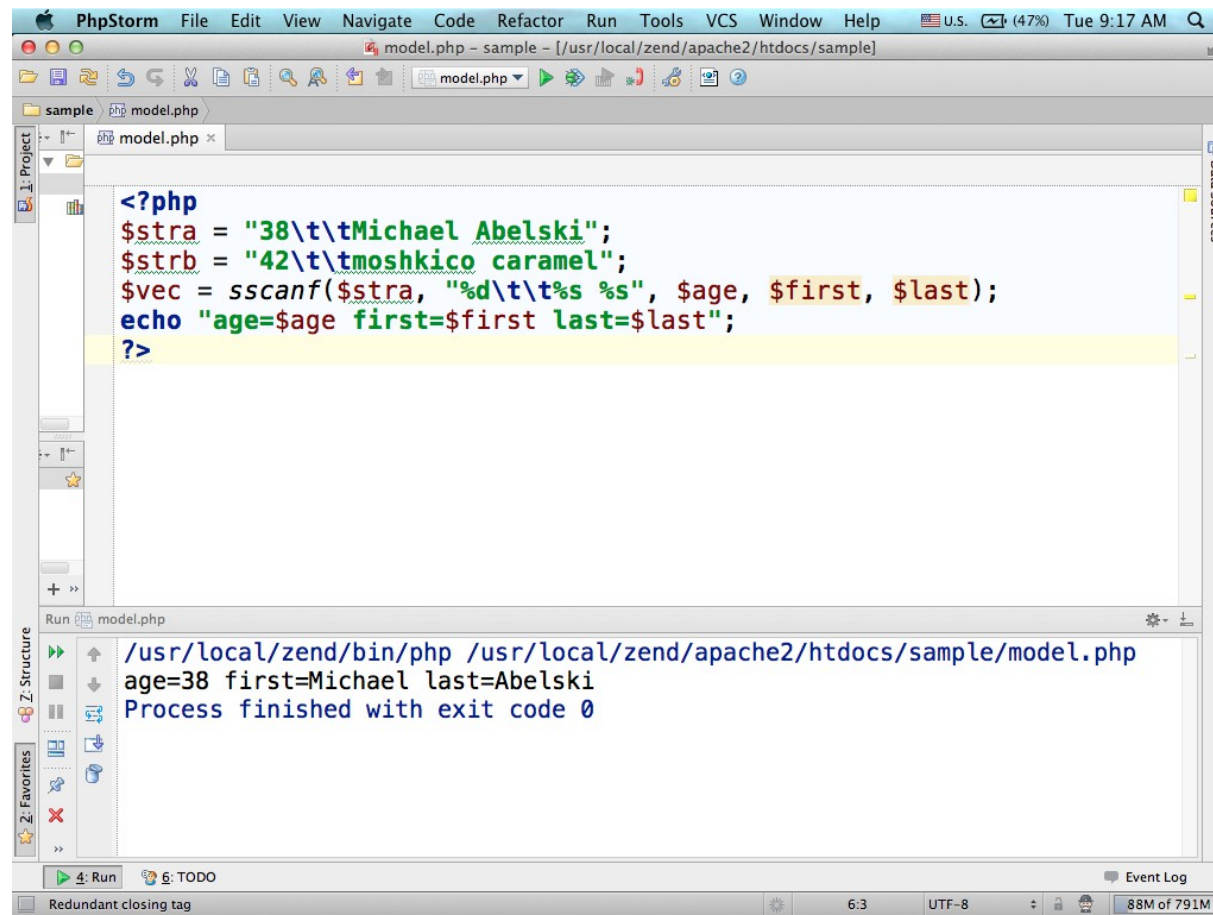
- ❖ The `sscanf()` function parse input string according to format we set. They work similarly to `printf()` functions, except that instead of allowing output the `sscanf()` function allow parsing formatted input.
- ❖ Unlike working with `printf()` function, when working with `sscanf()` function the data must exactly match the format.
- ❖ The `sscanf()` function returns an array of the parsed data.

The sscanf () Function

```
<?php
$person = "38\t\tMichael Abelski";
$vec = sscanf($person, "%d\t\t%s %s", $age, $first, $last);
echo "<person>";
echo "<age>$age</age>";
echo "<firstname>$first</firstname>";
echo "<lastname>$last</lastname>";
echo "</person>";
?>
```



The sscanf() Function



The screenshot shows the PhpStorm IDE interface. The main editor window displays a PHP script named `model.php` with the following code:

```
<?php
$str_a = "38\t\tMichael Abelski";
$str_b = "42\t\tmoshko caramel";
$vec = sscanf($str_a, "%d\t\t%s %s", $age, $first, $last);
echo "age=$age first=$first last=$last";
?>
```

The script is executed, and the output is shown in the Run window at the bottom. The output indicates that the `sscanf()` function successfully parsed the string `"38\t\tMichael Abelski"` into the variables `$age`, `$first`, and `$last`.

```
Run model.php
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
age=38 first=Michael last=Abelski
Process finished with exit code 0
```

Strings Direct Dereferencing

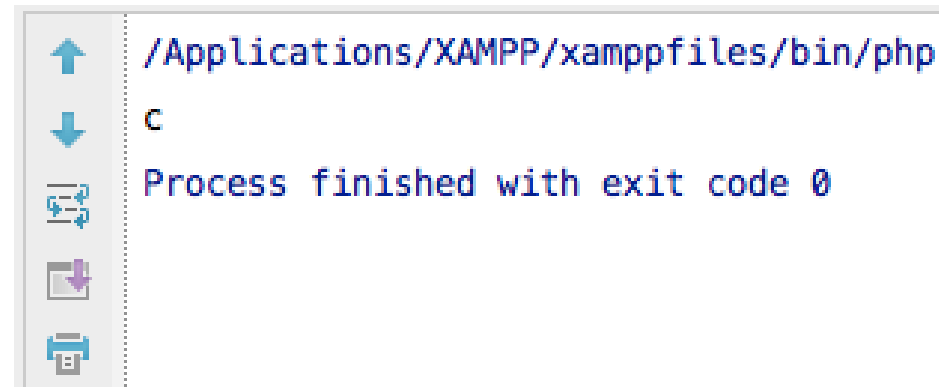
- ❖ As of PHP5.5 we can refer a specific character in our string right after the string is created in the very same statement.

Strings Direct Dereferencing

```
<?php  
echo "michael"[2];  
?>
```



Strings Direct Dereferencing



A terminal window with a light gray border. On the left is a vertical toolbar with five icons: a blue up arrow, a blue down arrow, a blue icon with two arrows forming a square, a purple icon with a square and a downward arrow, and a blue printer icon. The terminal text is as follows:

```
/Applications/XAMPP/xamppfiles/bin/php  
c  
Process finished with exit code 0
```

The Output

Strings

Strings in PHP

- ❖ Strings in PHP can be used to store data other than textual characters. Strings in PHP can be used to store binary data of any kind.

Simple Strings

- ❖ There are many ways through which it is possible to create new strings. Placing textual characters in quotes or double quotes is one of them.

```
$text = 'ABC';
```

```
$text = "ABC";
```

- ❖ When using double quotes we can add special escape sequences directly within the string (e.g. special characters such as "Hello\n").

Embedded Variables

- ❖ We can embed variables directly within a double quoted string simply by writing their names.

```
echo "Hallo $person_name\n";  
echo "temp_variable=$temp_variable";
```

The Heredoc

❖ Creating complex strings can be done via two ways. It can be done using the double quotes – as explained in the previous slide – and it can be done using the Heredoc syntax.

The diagram shows a C++ struct definition. On the left, the text is: `<<< struct {` followed by four lines of indented code: `int x;`, `int y;`, `int z;`, and `int w;`. On the right, there are two red-bordered boxes. The top box contains the text "The identifier we choose..." and has a red line pointing to the `struct` keyword. The bottom box contains the text "The identifier we choose..." and has a red line pointing to the opening curly brace `{`.

The Heredoc

```
<?php
$text_var = <<< TOKTOK
We wish you a happy birthday!
All the best!
Regards,
Friends.
TOKTOK;
echo $text_var;
?>
```

07/30/14

© Abelski eLearning

6

You can download the heredoc_sample.php code from the samples folder of this topic.

You can execute this code sample at
http://www.abelski.com/courses/php/samples/strings/heredoc_sample.php

The Backslash

- ❖ When creating a string using single quotes, we can include single quotes using the backslash.

```
echo 'I love my home\'s atmosphere';
```

- ❖ When creating a string using double quotes, we don't need to use the backslash in order to include single quotes.

```
echo "I love my home's atmosphere";
```

- ❖ When creating a string using double quotes, we can include double quotes and dollar signs prefixed with backslash.

```
echo "The US \$ value is \"4.4\"";
```

The Backslash

```
<?php
echo '<BR>I love Europe\'s weather';
echo "<BR>I US \$ currency";
echo "<BR>Here is my home's cake";
?>
```

07/30/14

© Abelski eLearning

8

You can find this code sample (backslash_sample.php) in this topic's samples folder.

You can execute this sample browsing the following URL address:
http://www.abelski.com/courses/php/samples/strings/backslash_sample.php

The `strlen()` Function

- ❖ Calling the `strlen()` function you can get the length of a given string.

The strlen() Function

```
<?php
$text_1 = "abcdef";
$text_2 = "abc def ";
$text_3 = "\n";
$length_1 = strlen($text_1);
$length_2 = strlen($text_2);
$length_3 = strlen($text_3);
echo "<BR>The length of \$text_1 is $length_1";
echo "<BR>The length of \$text_2 is $length_2";
echo "<BR>The length of \$text_3 is $length_3";
?>
```

The Output

The length of \$text_1 is 6
The length of \$text_2 is 8
The length of \$text_3 is 1

The `strtr()` Function

- ❖ This function returns a new string received after changing all occurrences of each one of the characters in `$from` to its equivalent character in `$to`. If the two strings have a different length, the extra characters in the longer one are ignored.

```
string strtr ( string $str , string $from , string $to )  
string strtr ( string $str , array $replace_pairs )
```

The `strtr()` Function

- ❖ The second version of this function returns a new string after repeatedly replacing each string (key) in `$replace_pairs` with its value.

```
string strtr ( string $str , string $from , string $to )  
string strtr ( string $str , array $replace_pairs )
```

The `strstr()` Function

```
<?php
$trans = array("hello" => "holla", "hi" => "hello");
echo strstr("hi all students, hello to all of you", $trans);
?>
```



07/30/14

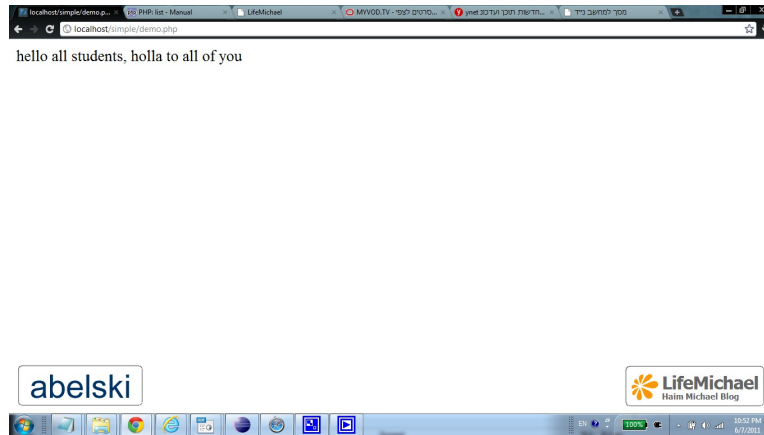
© Abelski eLearning

13

You can find this sample (`strstr_sample.php`) in the samples folder of this topic.

You can execute this sample browsing at the following URL:
http://www.abelski.com/courses/php/samples/strings/strstr_sample.php

The `strtr()` Function



07/30/14

© Abelski eLearning

14

You can find this sample (`strtr_sample.php`) in the samples folder of this topic.

You can execute this sample browsing at the following URL:
http://www.abelski.com/courses/php/samples/strings/strtr_sample.php

Strings as Arrays

❖ Each string can be treated as if it was an array.

```
<?php
$str = "abcdefghijklmnopqrstuvwxyz";
$length = strlen($str);
for($index=0; $index<$length; $index++)
{
    echo "<BR>$str[$index]";
}
?>
```



07/30/14

© Abelski eLearning

15

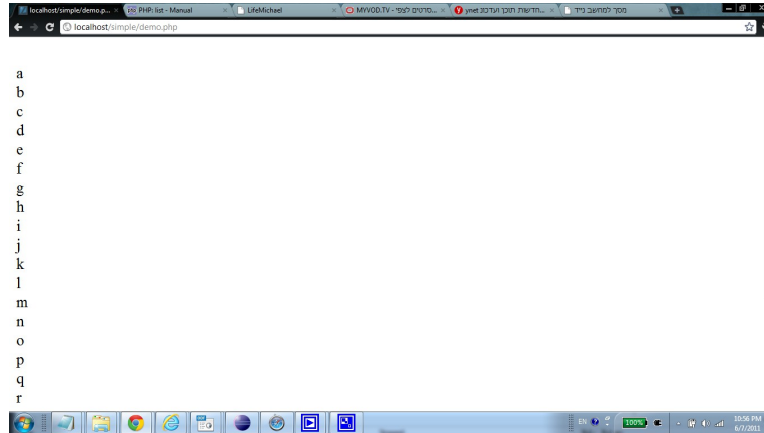
This possibility can be very useful when there is a need to go over a string char by char. The first character's index is 0. The last character's index is `strlen($str)-1`.

You can find this code sample (`string_array_sample.php`) in the samples folder of this topic.

You can execute the `string_array_sample.php` sample browsing the following URL address:

http://www.abelski.com/courses/php/samples/strings/string_array_sample.php

Strings as Arrays



07/30/14

© Abelski eLearning

16

This possibility can be very useful when there is a need to go over a string char by char. The first character's index is 0. The last character's index is `strlen($str)-1`.

You can find this code sample (`string_array_sample.php`) in the samples folder of this topic.

You can execute the `string_array_sample.php` sample browsing the following URL address:

http://www.abelski.com/courses/php/samples/strings/string_array_sample.php

Strings Comparison

- ❖ Comparing strings using the == comparison operator is involved with a transparent conversion of textual strings to numeric values.

```
<?php
$temp_1 = '1978year';
$temp_2 = 1978;
if($temp_1==$temp_2)
    echo "'$temp_1' and $temp_2 are equal";
else
    echo "no";
?>
```



07/30/14

© Abelski eLearning

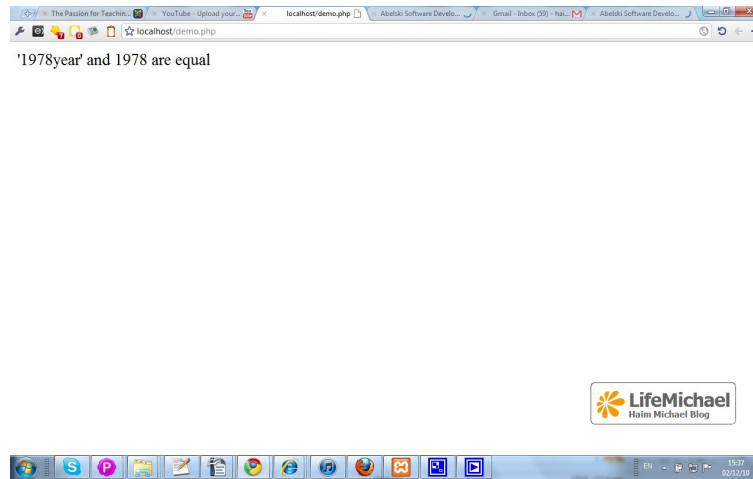
17

You can find this code sample (simple_string_comparison.php) in the samples folder of this topic.

You can execute this sample browsing the following URL address:

http://www.abelski.com/courses/php/samples/strings/simple_string_comparison.php

Strings Comparison



07/30/14

© Abelski eLearning

18

You can find this code sample (simple_string_comparison.php) in the samples folder of this topic.


You can execute this sample browsing the following URL address:

http://www.abelski.com/courses/php/samples/strings/simple_string_comparison.php

Strings Comparison

- ❖ Using the identity === operator is not involved with an automatic transparent conversion.

```
<?php
$str_1 = '1978year';
$str_2 = 1978;
if($str_1=== $str_2) { echo "'$str_1' and '$str_2' are equal"; }
else { echo "'$str_1' and '$str_2' are not equal"; }
?>
```



'1978year' and '1978' are not equal

07/30/14

© Abelski eLearning

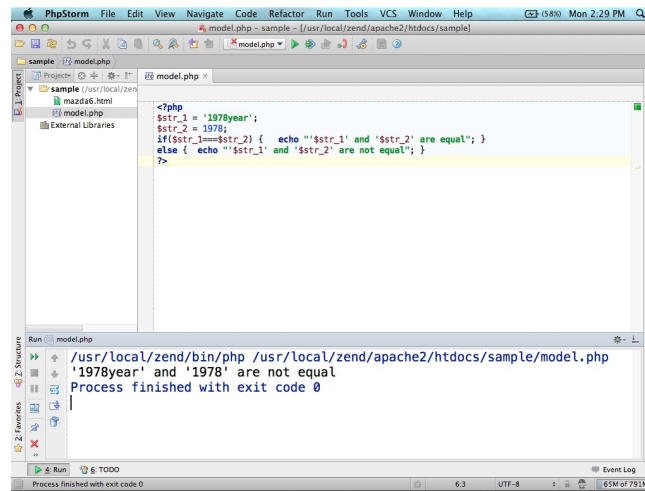
19

You can find this code sample (identity_comparison.php) in the samples folder of this topic.

You can execute this sample browsing the following URL address:

http://www.abelski.com/courses/php/samples/strings/identity_comparison.php

Strings Comparison



The screenshot shows the PhpStorm IDE interface. The main editor window displays a PHP file named `model.php` with the following code:

```
<?php
$str_1 = '1978year';
$str_2 = '1978';
if($str_1==$str_2) { echo "'$str_1' and '$str_2' are equal"; }
else { echo "'$str_1' and '$str_2' are not equal"; }
?>
```

The `Run` tab at the bottom shows the execution output:

```
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
'1978year' and '1978' are not equal
Process finished with exit code 0
```

07/30/14

© Abelski eLearning

20

You can find this code sample (`identity_comparison.php`) in the samples folder of this topic.

You can execute this sample browsing the following URL address:

http://www.abelski.com/courses/php/samples/strings/identity_comparison.php

The `strcmp()` & `strcasecmp()` Functions

- ❖ Both `strcmp()` and `strcasecmp()` receive two strings and compare them. They work the same, except that the former is case sensitive. Both functions return 0 when the two strings are equal.

```
int strcmp ( string $str1 , string $str2 )  
int strcasecmp ( string $str1 , string $str2 )
```

The strcmp() & strcasecmp() Functions

```
<?php
$str1 = "Turkey";
$str2 = "turkey";
if (strcasecmp($str1, $str2) == 0)
{
    echo '<BR>$str1 is equal to $str2 in a case insensitive comparison';
}
else
{
    echo '<BR>$str1 is not equal to $str2 in a case insensitive comparison';
}
?>
```



07/30/14

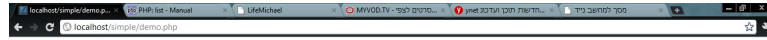
© Abelski eLearning

22

You can find this code sample (strcmp_sample.php) in this topic's samples folder.

You can execute this code sample browsing the following URL address:
http://www.abelski.com/courses/php/samples/strings/strcmp_sample.php

The `strcmp()` & `strcasecmp()` Functions



\$str1 is equal to \$str2 in a case insensitive comparison



07/30/14

© Abelski eLearning

23

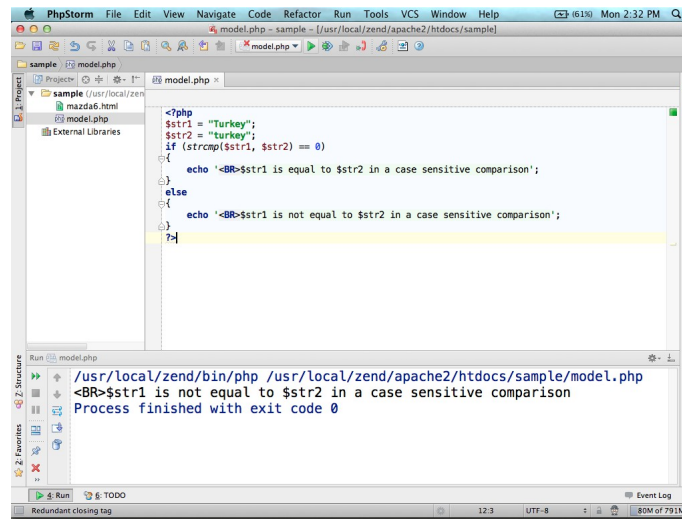
You can find this code sample (`strcmp_sample.php`) in this topic's samples folder.

You can execute this code sample browsing the following URL address:
http://www.abelski.com/courses/php/samples/strings/strcmp_sample.php

The strcmp() & strcasecmp() Functions

```
<?php
$str1 = "Turkey";
$str2 = "turkey";
if (strcmp($str1, $str2) == 0)
{
    echo '<BR>$str1 is equal to $str2 in a case sensitive comparison';
}
else
{
    echo '<BR>$str1 is not equal to $str2 in a case sensitive comparison';
}
?>
```

The strcmp() & strcasecmp() Functions



The screenshot shows the PhpStorm IDE interface. The main editor window displays a PHP file named `model.php` with the following code:

```
<?php
$str1 = "Turkey";
$str2 = "turkey";
if (strcmp($str1, $str2) == 0)
{
    echo "<BR>$str1 is equal to $str2 in a case sensitive comparison";
}
else
{
    echo "<BR>$str1 is not equal to $str2 in a case sensitive comparison";
}
?>
```

The `else` block is highlighted in yellow. Below the editor, the `Run` window shows the execution output:

```
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
<BR>$str1 is not equal to $str2 in a case sensitive comparison
Process finished with exit code 0
```

The status bar at the bottom indicates a "Redundant closing tag" warning.

07/30/14

© Abelski eLearning

25

The `strncasecmp()` Function

- ❖ The `strncasecmp()` function enables to set the number of characters you want to compare.

```
int strncasecmp ( string $str1 , string $str2 , int $len )
```

The strncasecmp () Function

```
<?php
$str1 = "lovsdfsdfsdy Turkey";
$str2 = "Lovely turkey country!";
if (strncasecmp($str1, $str2,3) == 0)
{
    echo 'true';
}
else
{
    echo 'false';
}
?>
```



07/30/14

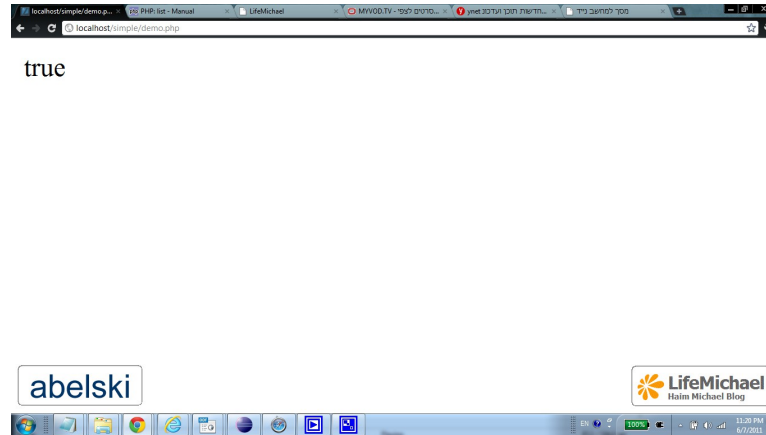
© Abelski eLearning

27

You can find this code sample (strncasecmp_sample.php) in the samples folder of this topic.

You can execute this code sample browsing the following URL:
http://www.abelski.com/courses/php/samples/strings/strncasecmp_sample.php

The `strncasecmp()` Function



07/30/14

© Abelski eLearning

28

You can find this code sample (`strncasecmp_sample.php`) in the samples folder of this topic.

You can execute this code sample browsing the following URL:
http://www.abelski.com/courses/php/samples/strings/strncasecmp_sample.php

The `strpos()` & `strstr()` Functions

- ❖ The `strpos()` & `strstr()` functions families provide the simplest way to search for substrings.

```
int strpos ( string $haystack ,  
            mixed $needle [, int $offset ] )  
  
string strstr ( string $haystack ,  
               string $needle , bool $before_needle )
```

The `strpos()` & `strstr()` Functions

```
int strpos ( string $haystack ,  
            mixed $needle [, int $offset ] )
```

This function returns the index position of the first occurrence of needle within the haystack. The optional offset parameter allows specifying from which character to start the search.

The strpos() & strstr() Functions

```
string strstr ( string $haystack , string $needle ,  
               bool $before_needle )
```

This function returns the substring that starts with the first occurrence of the needle within the haystack till the end of the haystack. If the third parameter is true then this function returns the substring from the beginning of the haystack till the needle. Default is false.

The strpos() & strstr() Functions

```
<?php
$telephone = '972 54 5544232 #243 #6344';
$extension = strstr($telephone, '#');
echo "_".$extension."_";
?>
```



07/30/14

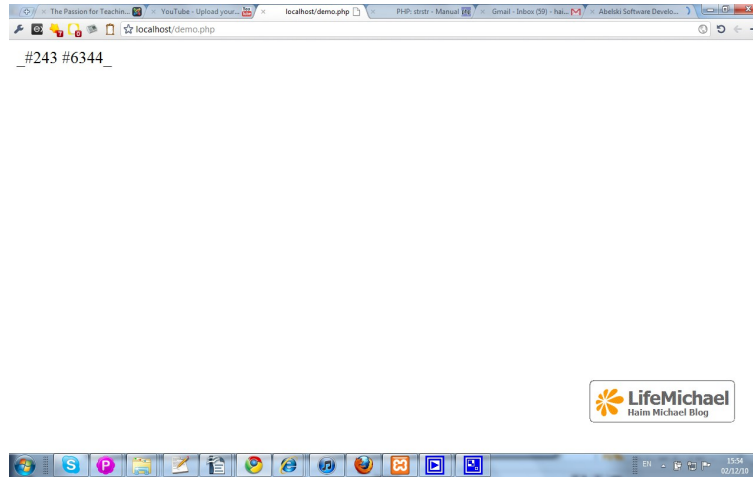
© Abelski eLearning

32

You can find this code sample (strstr_sample.php) in the samples folder of this topic.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/strstr_sample.php

The strpos() & strstr() Functions



07/30/14

© Abelski eLearning

33

You can find this code sample (strstr_sample.php) in the samples folder of this topic.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/strstr_sample.php

The strpos() & strstr() Functions

```
<?php
$telephone = '972 54 5544232 #243 #6344';
$extension = strpos($telephone, '#');
echo $extension;
?>
```



07/30/14

© Abelski eLearning

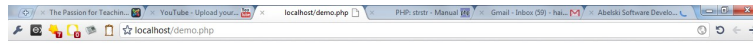
34

You can find this code sample (strpos_sample.php) in the samples folder of this topic.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/strpos_sample.php

Note that if the needle is in the beginning of the string (starts the string) then strpos() will return the value zero. For that reason, using the strpos() returned value within a conditional statement must be done by comparing the returned value to 'true' or 'false', otherwise, a returned value of zero, which PHP automatically translates into 'false' might lead to wrong results.

The strpos() & strstr() Functions



15



07/30/14

© Abelski eLearning

35

You can find this code sample (strpos_sample.php) in the samples folder of this topic.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/strpos_sample.php

Note that if the needle is in the beginning of the string (starts the string) then strpos() will return the value zero. For that reason, using the strpos() returned value within a conditional statement must be done by comparing the returned value to 'true' or 'false', otherwise, a returned value of zero, which PHP automatically translates into 'false' might lead to wrong results.

The `strpos()` & `strpos()` Functions

- ❖ The `strpos()` & `strpos()` do the same work
`strpos()` & `strpos()` do... with one difference: The
`strpos()` & `strpos()` are not case sensitive.

The `strrpos()` Function

- ❖ The `strrpos()` does the same work `strpos()` does... with one difference. The `strrpos()` starts the search in a reverse order.

The `strspn()` & `strcspn()` Functions

- ❖ The `strspn()` function returns the length of the initial biggest matching mask.

```
int strspn ( string $str1 , string $str2  
            [, int $start [, int $length ]] )
```

This function returns the length of the initial segment of `str1` which consists entirely of characters in `str2` .

- ❖ The `strcspn()` function works the opposite. It returns the length of the initial segment of the string that doesn't contain any of the characters of `str2`.

07/30/14

© Abelski eLearning

38

Both functions have optional parameters to specify where to start the search and the length of the string (from `str1`) that should be examined.

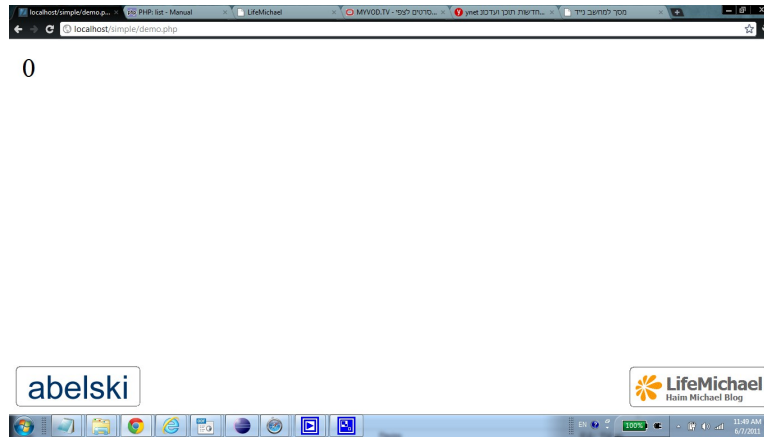
The strpos() & strrpos() Functions

```
<?php
$a = "i love java and php";
$b = "lo v i e javaa";
$num = strpos($a, $b);
echo $num;
?>
```



Both functions have optional parameters to specify where to start the search and the length of the string (from str1) that should be examined.

The `strspn()` & `strcspn()` Functions



07/30/14

© Abelski eLearning

40

Both functions have optional parameters to specify where to start the search and the length of the string (from `str1`) that should be examined.

The `str_replace()` Function

- ❖ The `str_replace()` function replaces all occurrences of the searched string with the replacement one.

```
mixed str_replace ( mixed $search ,  
                    mixed $replace ,  
                    mixed $subject [, int &$count ] )
```

The `str_replace` function goes over the `$subject` string and replaces each `$search` occurrence with `$replace`. The `&$count` parameter is an optional one. If we pass it, the function will fill it with the number of substitutions. The `$replace` and `$search` parameters we pass can be arrays of values, which allows us to replace more than one string (process from left to right).

The `str_replace()` Functions

```
<?php
$initial_string = "You should drink bear, wine, coke, water and lemonade.";
$original_strs_array = array("bear", "wine", "coke");
$replacement_strs_array = array("goldstar", "golan", "juice");
$new_string = str_replace($original_strs_array, $replacement_strs_array,
    $initial_string);
echo $new_string;
?>
```



07/30/14

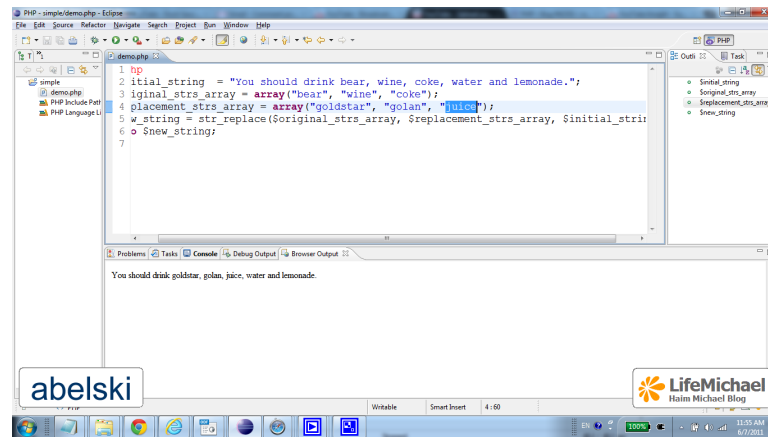
© Abelski eLearning

42

You can find this code sample (`str_replace_sample.php`) in the samples folder of this topic.

You can execute this code sample browsing at
http://www.abelski.com/courses/php/samples/strings/str_replace_sample.php

The `str_replace()` Functions



07/30/14

© Abelski eLearning

43

You can find this code sample (`str_replace_sample.php`) in the samples folder of this topic.

You can execute this code sample browsing at
http://www.abelski.com/courses/php/samples/strings/str_replace_sample.php

The substr_replace() Function

- ❖ The `substr_replace()` function replaces a string with another string in a character its index is passed to the function.

```
mixed substr_replace ( mixed $string , string $replacement ,  
                      int $start [, int $length ] )
```

The `substr_replace` replaces portion of `$string` with `$replacement` starting in index `$start`. The optional `$length` parameter specifies the length of the string that will be replaced (out of `$string`).

The substr_replace() Function

```
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string: $var<BR>";
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 0);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 10);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 10, 2);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 0, 0);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 5, 1);
?>
```



07/30/14

© Abelski eLearning

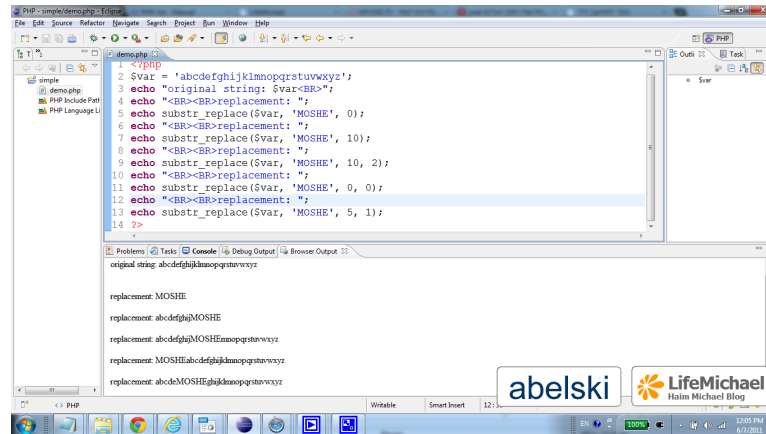
45

You can find the substr_sample.php code sample in the samples folder of this topic.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/substr_sample.php

The optional length parameter represents the length of the portion of the string that will be replaced. When the length value is negative it represents the number of characters from the end of the string at which to stop replacing. Its default value is strlen(string). If length is zero then this function will have the effect of inserting replacement into string at the given start offset.

The substr_replace() Function



The screenshot shows a PHP IDE with a file named 'demo.php'. The code in the file is as follows:

```
1 <?PHP
2 $var = 'abcdefghijklmnopqrstuvwxyz';
3 echo "original string: $var<br>";
4 echo "<br><br>replacement: ";
5 echo substr_replace($var, 'MOSHE', 0);
6 echo "<br><br>replacement: ";
7 echo substr_replace($var, 'MOSHE', 10);
8 echo "<br><br>replacement: ";
9 echo substr_replace($var, 'MOSHE', 10, 2);
10 echo "<br><br>replacement: ";
11 echo substr_replace($var, 'MOSHE', 0, 0);
12 echo "<br><br>replacement: ";
13 echo substr_replace($var, 'MOSHE', 5, 1);
14 ?>
```

The IDE's output window shows the following results:

```
original string: abcdefghijklmnopqrstuvwxyz
replacement: MOSHE
replacement: abcdefghMOSHE
replacement: abcdefghMOSHEmnopqrstuvwxyz
replacement: MOSHEabdefghijklmnopqrstuvwxyz
replacement: abcdeMOSHEghijklmnopqrstuvwxyz
```

At the bottom of the IDE, there are logos for 'abelski' and 'LifeMichael'.

07/30/14

© Abelski eLearning

46

You can find the substr_sample.php code sample in the samples folder of this topic.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/substr_sample.php

The optional length parameter represents the length of the portion of the string that will be replaced. When the length value is negative it represents the number of characters from the end of the string at which to stop replacing. Its default value is strlen(string). If length is zero then this function will have the effect of inserting replacement into string at the given start offset.

The `substr()` Function

- ❖ The `substr()` function extracts a substring out of a given string.

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

The `$string` parameter is the original string. The `$start` is the index from which the substring will start. If `$start` is negative the returned substring will start from the character its index is counted from the end.

The substr () Function

```
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string...<BR>$var<BR>";
echo "<BR><BR>substring...<BR>";
echo substr($var,0);
echo "<BR><BR>substring...<BR>";
echo substr($var,2,4);
?>
```



07/30/14

© Abelski eLearning

48

The substr.php code sample can be found in the samples folder of this topic.

You can execute this code sample browsing at <http://www.abelski.com/courses/php/samples/strings/substr.php>

The `substr()` Function



07/30/14

© Abelski eLearning

49

The `substr.php` code sample can be found in the samples folder of this topic.

You can execute this code sample browsing at <http://www.abelski.com/courses/php/samples/strings/substr.php>

The `setlocale()` Function

- ❖ Formatting rules are generally affected by the geographic location.
- ❖ The `setlocale()` method enables us setting a specific locale setting and indicate which functions will be affected by the change.

```
string setlocale ( int $category , array $locale )  
string setlocale ( int $category ,  
                  string $locale [, string $... ] )
```

The `setlocale()` Function

The `$category` parameter can be one of the following:

`LC_ALL` for all of the below

`LC_COLLATE` for string comparison.

`LC_CTYPE` for character classification and conversion, for example `strtoupper()`

`LC_MONETARY` for `localeconv()`

`LC_NUMERIC` for decimal separator (See also `localeconv()`)

`LC_TIME` for date and time formatting with `strftime()`

`LC_MESSAGES` for system responses (available if was compiled with `libintl`)

The `setlocale()` Function

The `$locale` parameter can be any of the valid values according to <http://www.w3.org/WAI/ER/IG/ert/iso639.htm>.

e.g.

`en_US`

`de_DE`

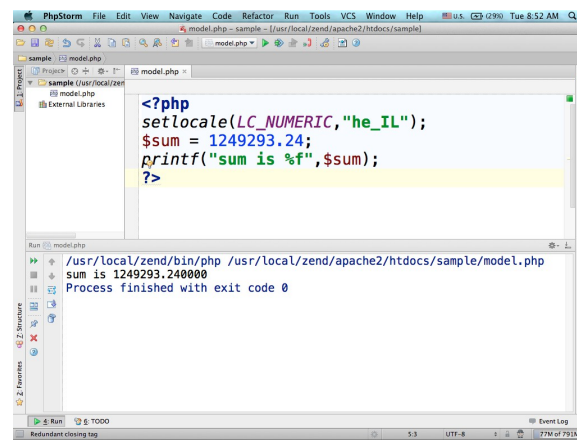
`fr_FR`

The `setlocale()` Function

```
<?php
setlocale(LC_NUMERIC, "he_IL");
$sum = 1249293.24;
printf("sum is %f", $sum);
?>
```



The `setlocale()` Function



The screenshot shows the PHPStorm IDE interface. The main editor window displays a PHP script named `model.php` with the following code:

```
<?php
setlocale(LC_NUMERIC, "he_IL");
$sum = 1249293.24;
printf("sum is %f", $sum);
?>
```

The `printf` function call is highlighted in yellow. Below the editor, the Run window shows the execution command and output:

```
/usr/local/zend/bin/php /usr/local/zend/apache2/htdocs/sample/model.php
sum is 1249293.240000
Process finished with exit code 0
```

The status bar at the bottom indicates the file is encoded in UTF-8 and the current time is 7:14 PM on 7/30/14.

The `number_format()` Function

- ❖ The `number_format()` function is used for output a number and separated its digits into thousands,millions etc.

```
string number_format ( float $number [, int $decimals])  
string number_format ( float $number [, int $decimals  
                        [, string $dec_point ]], string $thousands_sep )
```

The `money_format()` Function

❖ The `money_format()` function is used for output a currency.

```
string money_format ( string $format , float $number )
```

This method can be called passing two parameters. The `$number` parameter is the number that should be formatted. The `$format` parameter can be composed of various special characters.

The `printf()` Function

❖ The `printf()` function allows formatting our text in various ways.

❖ The `printf()` function simply writes to the script's output.

```
int printf ( string $format [, mixed $args [, mixed $... ]] )
```

❖ The `sprintf()` function simply returns it.

```
string sprintf ( string $format [, mixed $args  
[ , mixed $... ]] )
```

❖ The `fprintf()` function simply writes to a file.

```
int fprintf ( resource $handle , string $format  
[ , mixed $args [, mixed $... ]] )
```


The `printf()` Function

❖ The `$format` is composed of zero (or more) of the following directives:

Ordinary Characters

They are copied directly to the result. The '%' doesn't belong to this group of directives.

Conversion Specification

Each one of the conversion specifications results in fetching its parameter. Each conversion specification consists of a percent sign (%), followed by one or more of the following elements: sign specifier, padding specifier, alignment specifier, width specifier, precision specifier and a type specifier.

The `printf()` Function

- ❖ The conversion specifications include the following possibilities:

Sign Specifier

This optional sign specifier forces the '-' / '+' sign to be used. The sign specifier forces showing the sign symbol (even when trying to force presenting '+' and the number is already positive... the '+' will be added).

```
printf("%%+d = %+d\n", $n);
```

Padding Specifier

This optional padding specifier tells which character to use for padding the result to the right string size. The ' should prefix that character unless it is '0' or a simple space ' '.

```
printf("[%'#12s]\n", $str);
```

The conversion sign follows '%'.

The `printf()` Function

Alignment Specifier

This optional sign specifier tells whether the result should be left justified or right justified. The default is right justified. Adding '-' after the '%' will change its alignment to be left justified.

```
printf("[% -10s]\n", $str);
```

Width Specifier

This optional number says how many characters (at the minimum) the conversion result should be.

```
printf("[% '#8s]\n", $s);
```

The `printf()` Function

Precision Specifier

This optional specifier says how many decimal digits should be displayed for a floating point numbers.

```
printf("%.2f", $sum);  
//results in two digits after the decimal point.
```

Type Specifier

The type specifier says what type the arguments data should be treated as.

- b - treated as integer and presented as a binary number.
- c - treated as integer and presented as a character.
- d - treated as integer and presented as a signed decimal integer number.
- e - treated as a scientific notation (e.g. 1.5e4).

The `printf()` Function

u - treated as integer and presented as an unsigned decimal number.

f - treated as float and presented as a floating point number (local aware).

F - treated as float and presented as a floating point number (non local aware).

o - treated as integer and presented as an octal number.

s - treated as presented as a string.

x - treated as integer and presented as hexadecimal number (lowercase).

X - treated as integer and presented as hexadecimal number (Uppercase).

The `printf()` Function

```
<?php
$positive_number = 43951789;
$negative_number = -43951789;
$simple_character = 65; // ASCII 65 is 'A'
$simple_string = "mosh";
$long_string = "moshiko goes to seven grade";
printf("%b = '%b'<BR>", $positive_number);
printf("%c = '%c'<BR>", $simple_character);
printf("%d = '%d'<BR>", $positive_number);
printf("%e = '%e'<BR>", $positive_number);
printf("%u = '%u'<BR>", $positive_number);
printf("%u = '%u'<BR>", $negative_number);
```

The `printf()` Function

```
printf("%%f = '%f'<BR>", $positive_number);  
printf("%%o = '%o'<BR>", $positive_number);  
printf("%%s = '%s'<BR>", $positive_number);  
printf("%%x = '%x'<BR>", $positive_number);  
printf("%%X = '%X'<BR>", $positive_number);  
printf("_%s_<BR>", $simple_string);  
printf("_%20s_<BR>", $simple_string);  
printf("_%-20s_<BR>", $simple_string);  
printf("_%020s_<BR>", $simple_string);  
printf("_%#20s_<BR>", $simple_string);  
printf("%.3e<BR>", $positive_number);  
?>
```

07/30/14

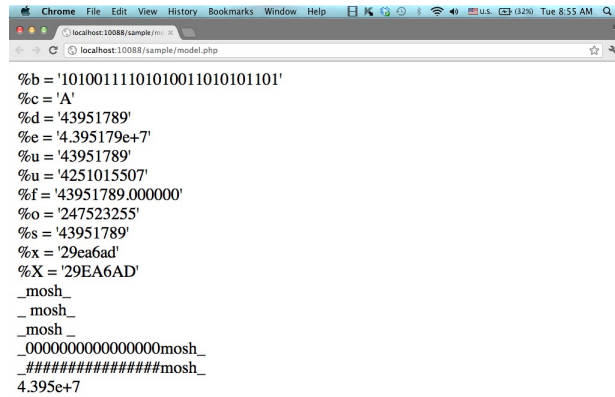
© Abelski eLearning

64

You can find this code sample in the samples folder of this topic. The filename is `printf_sample.php`.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/printf_sample.php

The `printf()` Function



A screenshot of a Google Chrome browser window. The address bar shows 'localhost:10088/sample/model.php'. The page content displays the output of a `printf` statement, showing various data types: binary, character, decimal, exponential, unsigned decimal, floating-point, octal, string, hexadecimal, and a custom format with underscores. The output is as follows:

```
%b = '10100111101010011010101101'  
%c = 'A'  
%d = '43951789'  
%e = '4.395179e+7'  
%u = '43951789'  
%u = '4251015507'  
%f = '43951789.000000'  
%o = '247523255'  
%s = '43951789'  
%x = '29ea6ad'  
%X = '29EA6AD'  
_mosh_  
_mosh_  
_mosh_  
_0000000000000000mosh_  
_#####mosh_  
4.395e+7
```

07/30/14

© Abelski eLearning

65

You can find this code sample in the samples folder of this topic. The filename is `printf_sample.php`.

You can execute this code sample browsing at http://www.abelski.com/courses/php/samples/strings/printf_sample.php

The `sscanf()` Function

- ❖ The `sscanf()` function parse input string according to format we set. They work similarly to `printf()` functions, except that instead of allowing output the `sscanf()` function allow parsing formatted input.
- ❖ Unlike working with `printf()` function, when working with `sscanf()` function the data must exactly match the format.
- ❖ The `sscanf()` function returns an array of the parsed data.

The sscanf () Function

```
<?php
$person = "38\t\tMichael Abelski";
$vec = sscanf($person, "%d\t\t%s %s", $age, $first, $last);
echo "<person>";
echo "<age>$age</age>";
echo "<firstname>$first</firstname>";
echo "<lastname>$last</lastname>";
echo "</person>";
?>
```



07/30/14

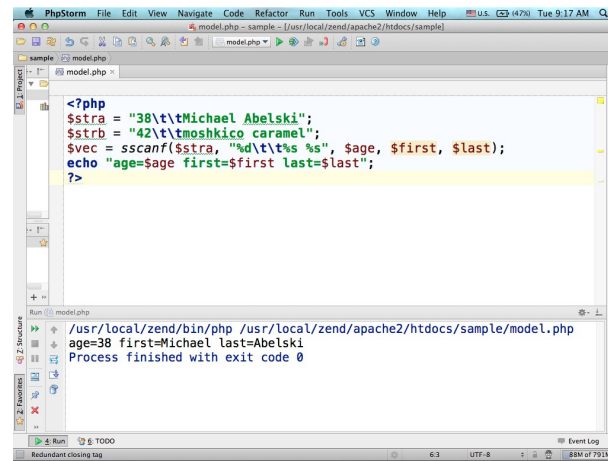
© Abelski eLearning

67

You can find the `sscanf_sample.php` file in the samples folder of this topic.

You can execute this sample browsing at
http://www.abelski.com/courses/php/samples/strings/sscanf_sample.php

The sscanf () Function



The screenshot shows the PhpStorm IDE interface. The main editor window displays a PHP script in a file named 'model.php'. The script uses the `sscanf` function to parse a string. Below the editor, the 'Run' window shows the output of the script, which is 'age=38 first=Michael last=Abelski'.

```
<?php
$str = "38\t\tMichael Abelski";
$strb = "42\t\tmoshkico caramel";
$vec = sscanf($str, "%d\t\t%s %s", $age, $first, $last);
echo "age=$age first=$first last=$last";
?>
```

Run /usr/local/Zend/bin/php /usr/local/Zend/apache2/htdocs/sample/model.php
age=38 first=Michael last=Abelski
Process finished with exit code 0

07/30/14

© Abelski eLearning

68

You can find the `sscanf_sample.php` file in the `samples` folder of this topic.

You can execute this sample browsing at http://www.abelski.com/courses/php/samples/strings/sscanf_sample.php

Strings Direct Dereferencing

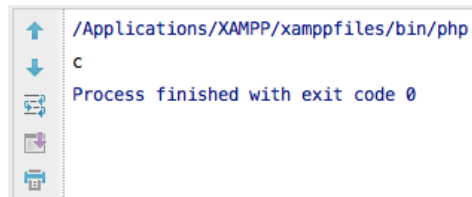
- ❖ As of PHP5.5 we can refer a specific character in our string right after the string is created in the very same statement.

Strings Direct Dereferencing

```
<?php  
echo "michael"[2];  
?>
```



Strings Direct Dereferencing



```
/Applications/XAMPP/xamppfiles/bin/php  
c  
Process finished with exit code 0
```

A screenshot of a terminal window. The title bar is not visible. The window has a light gray background. On the left side, there is a vertical toolbar with five icons: a blue up arrow, a blue down arrow, a green icon with a magnifying glass, a purple icon with a plus sign, and a blue icon with a minus sign. The main area of the terminal is white and contains the following text: the first line is the path `/Applications/XAMPP/xamppfiles/bin/php` in blue; the second line is the character `c` in blue; and the third line is the text `Process finished with exit code 0` in blue.

The Output