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 and 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"`.

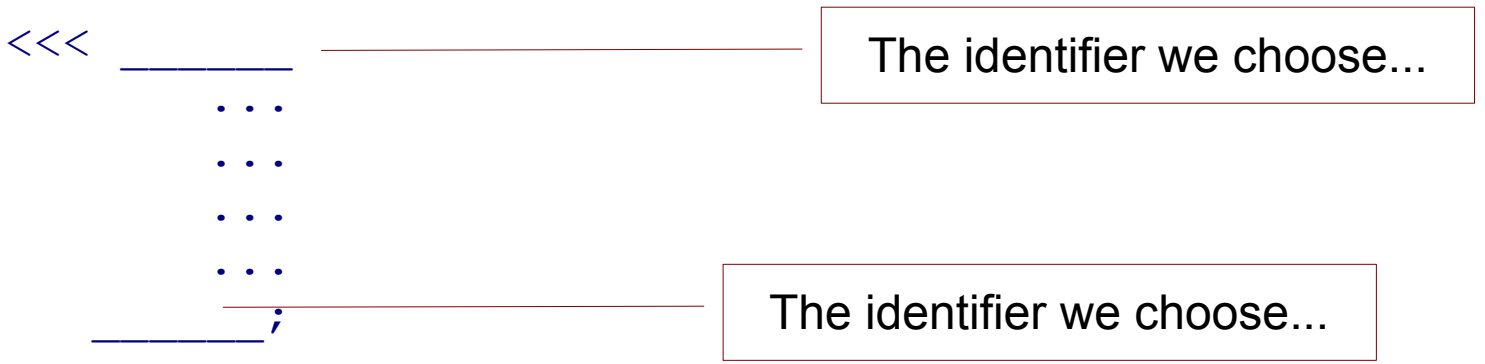
Embedded Variables

- 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 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

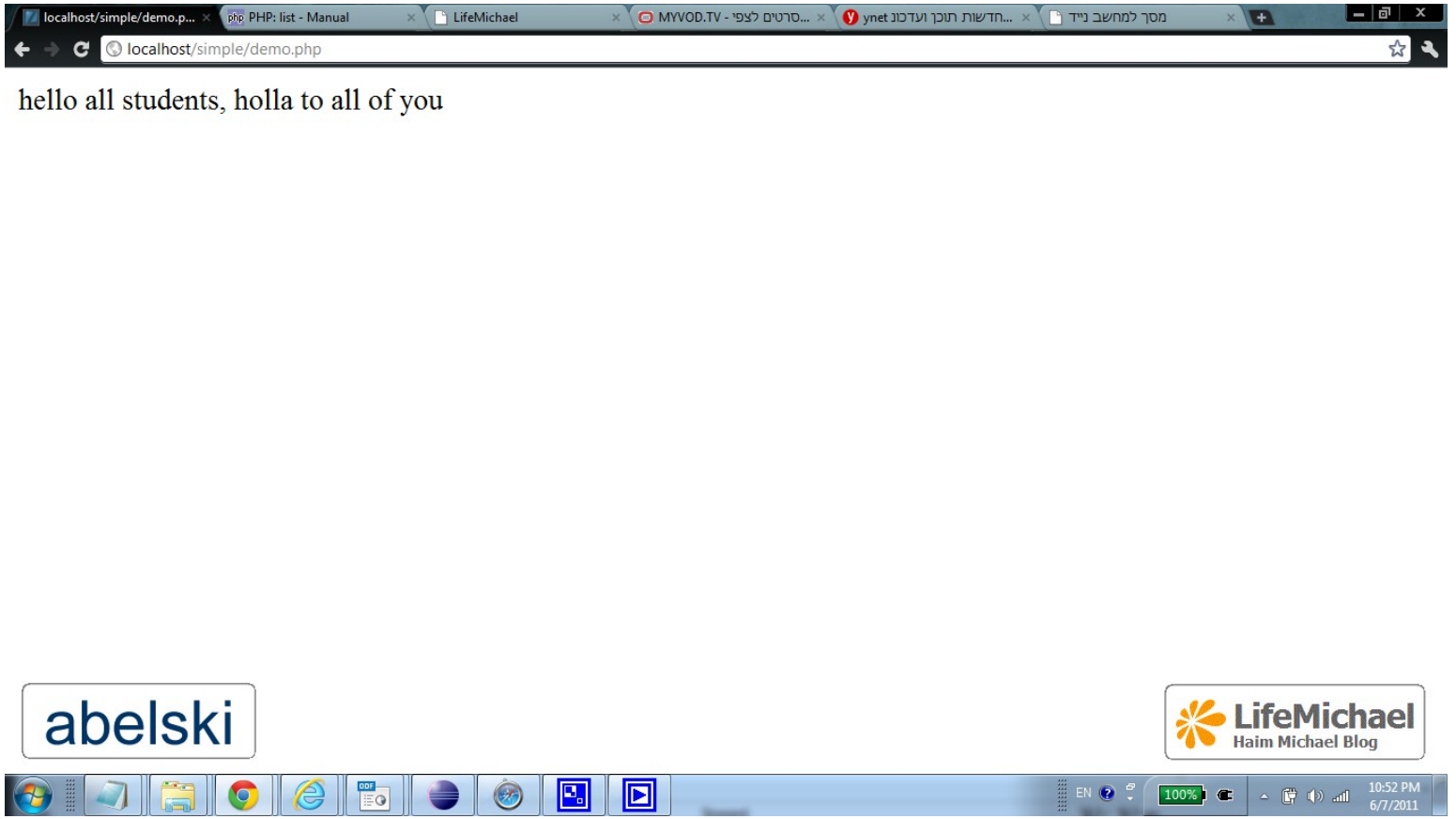
- The second version of this function returns a new string after repeatedly replacing each string (key) in `$replace_pairs` with its value.

```
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 `strtr()` Function

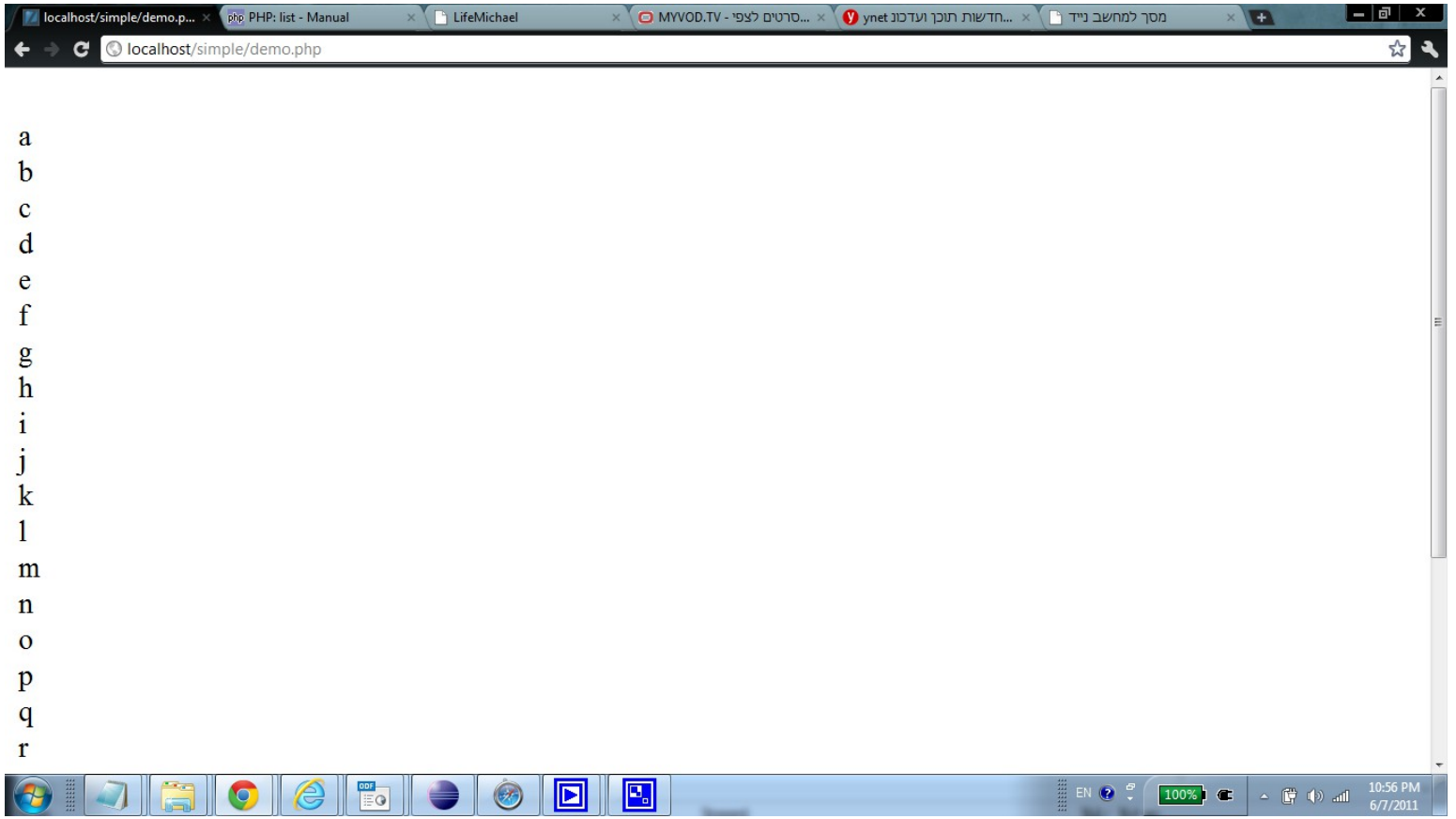


Strings as Arrays

- 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

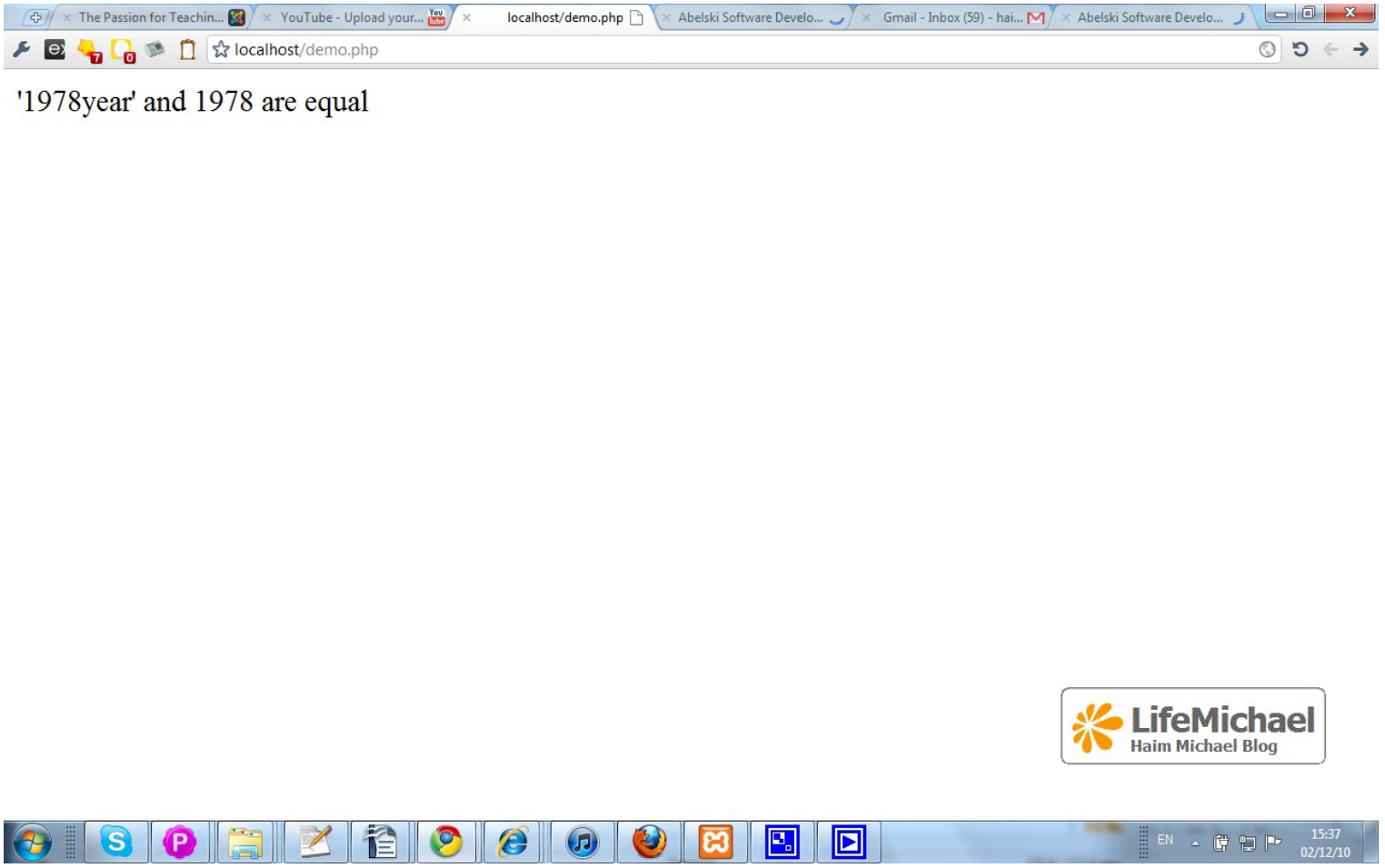


Strings Comparison

- 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



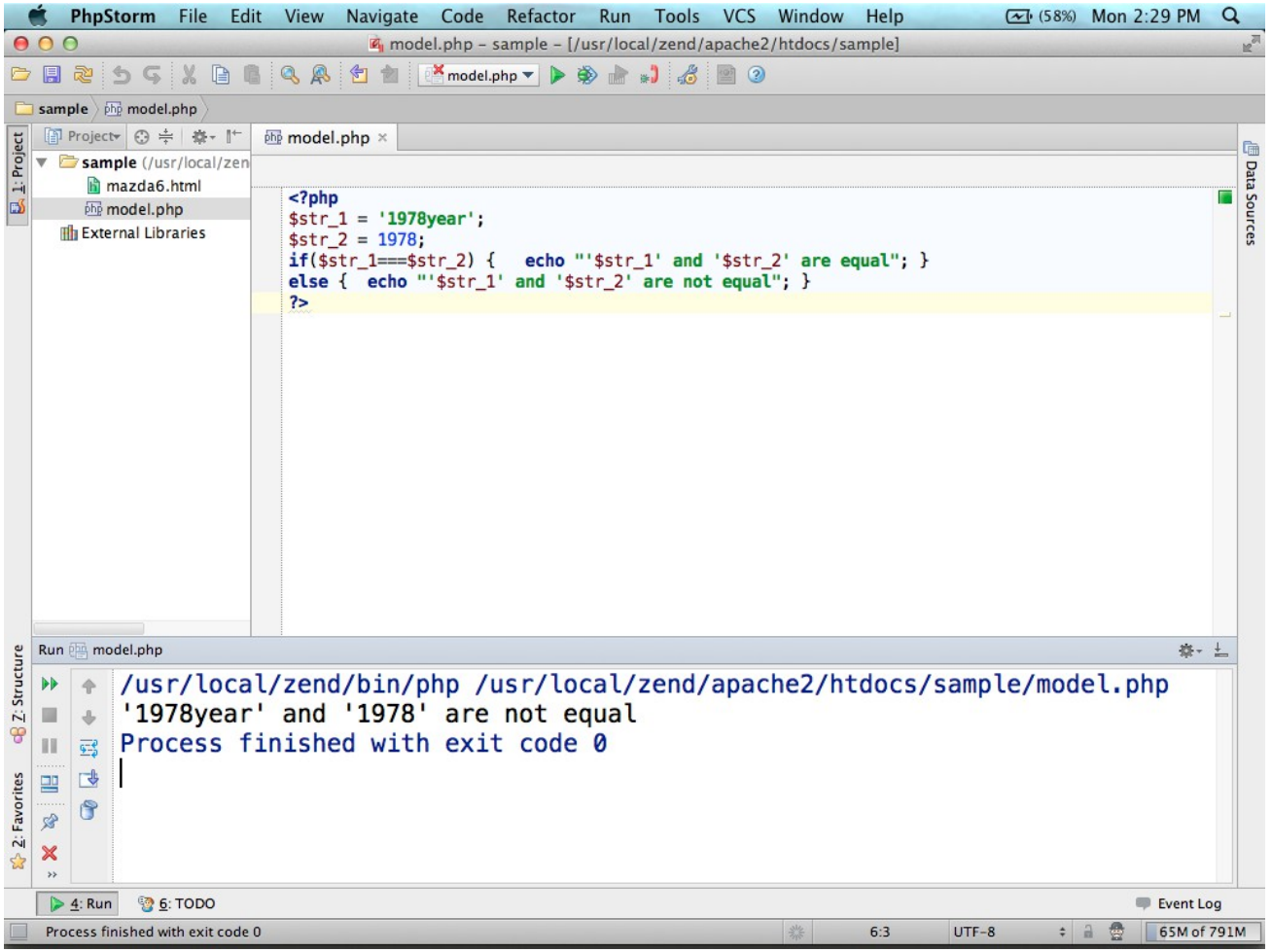
Strings Comparison

- 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 `strcmp()` & `strcasecmp()` Functions

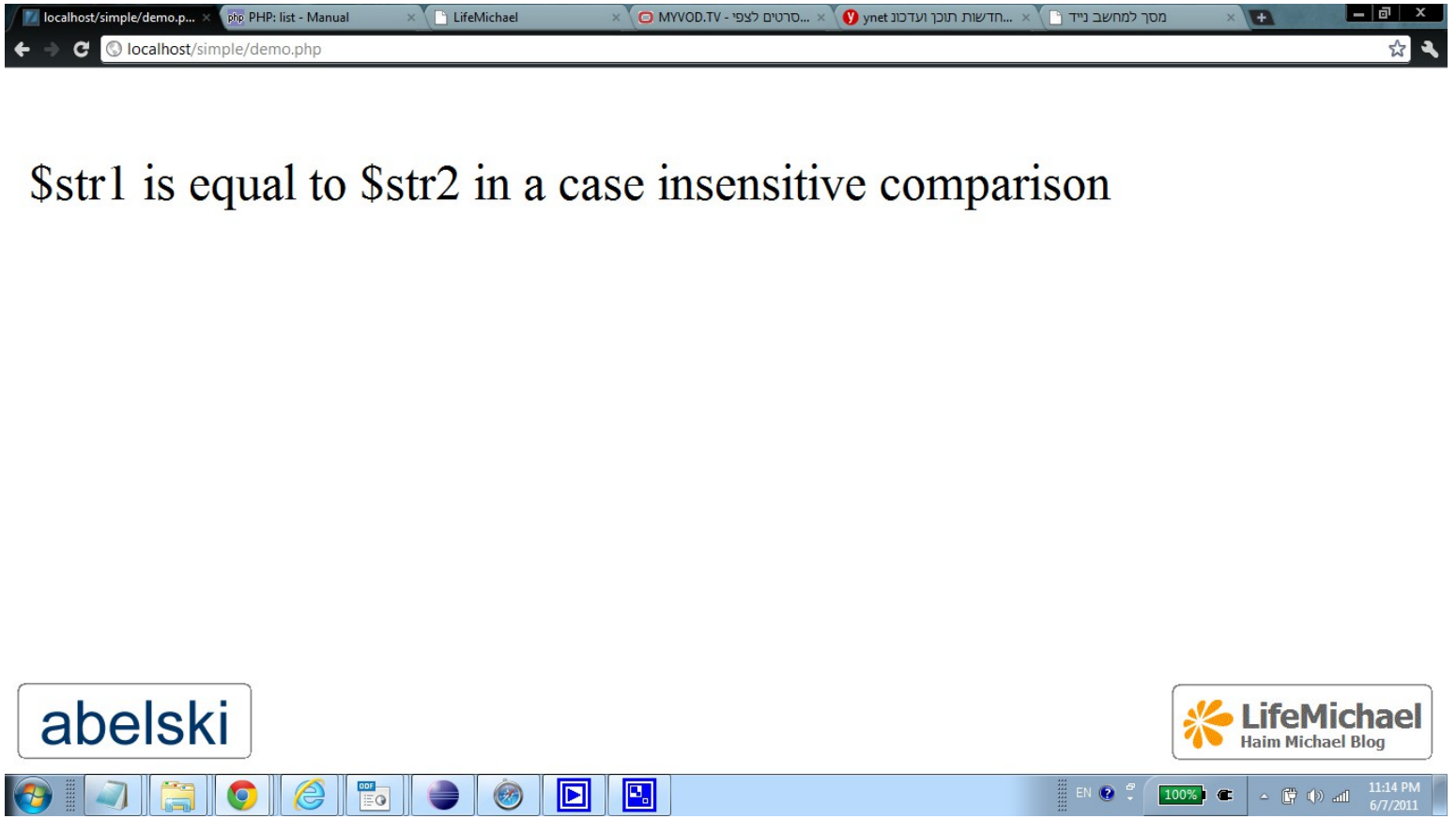
- Both `strcmp()` and `strcasecmp()` receive two strings and compare them. They work the same, except that the former is case sensitive. Both functions return 0 when the two strings are equal.

```
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';
}
?>
```

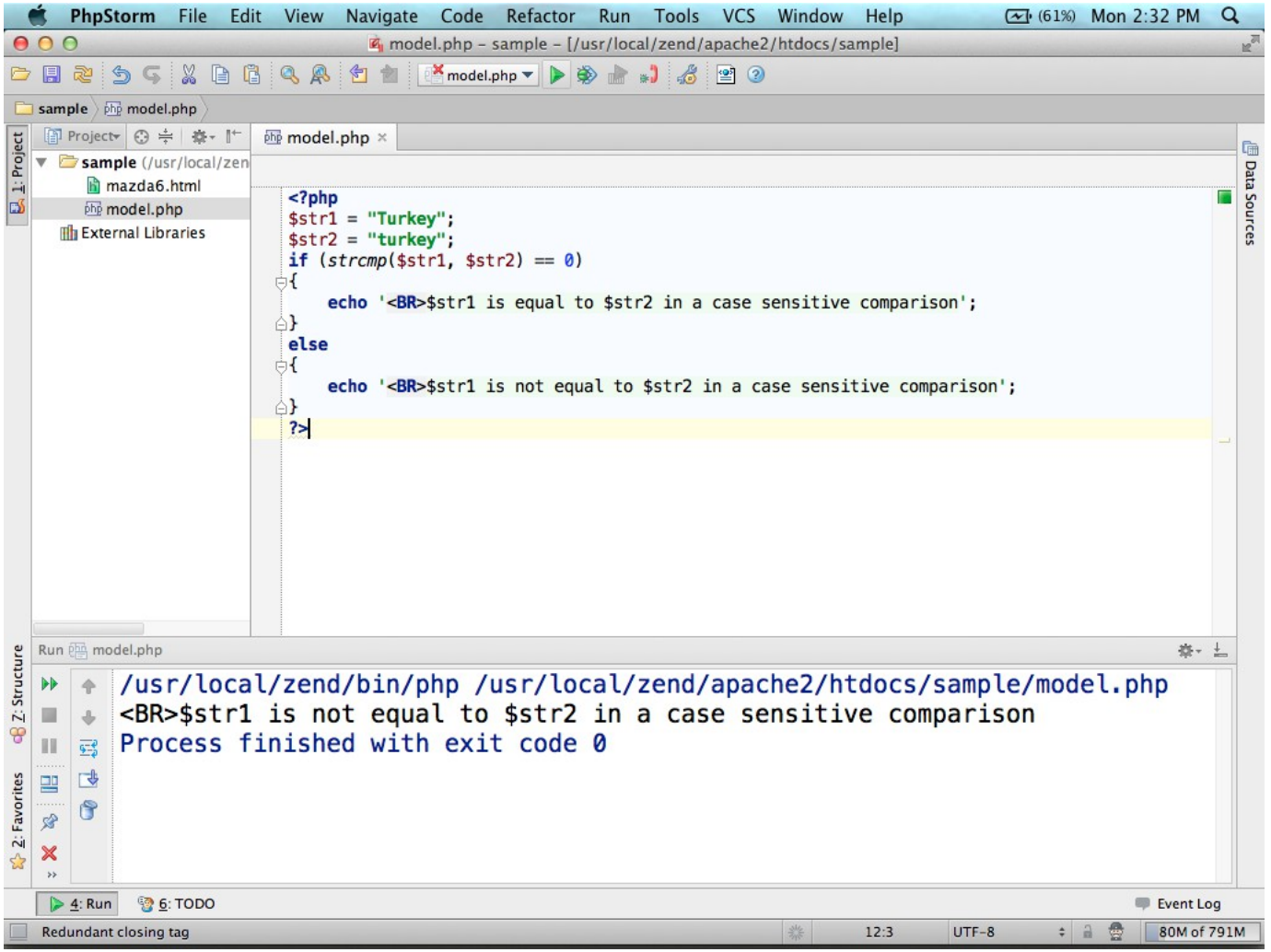
The `strcmp()` & `strcasecmp()` Functions



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 `strncasecmp()` Function

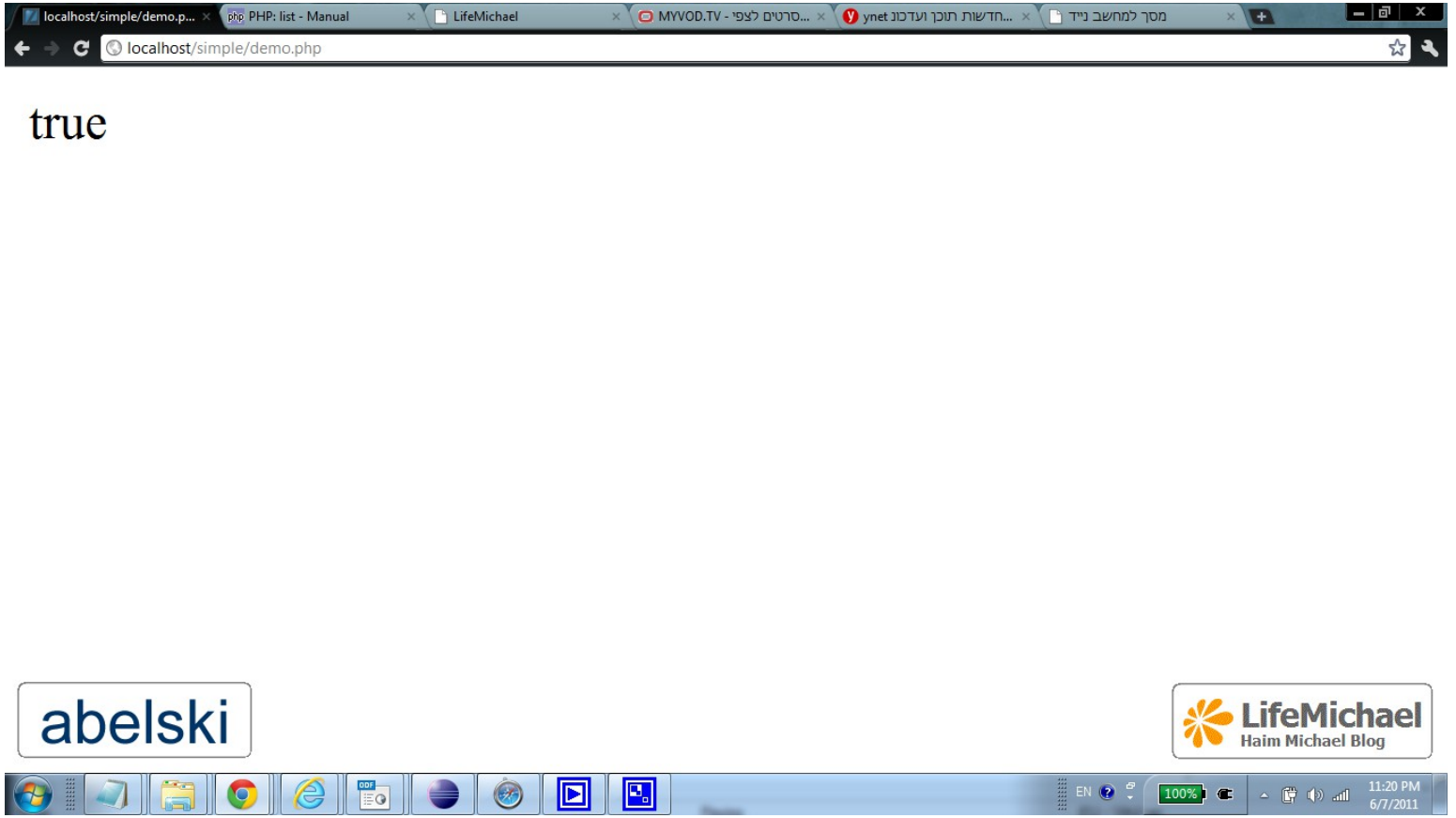
- The `strncasecmp()` function enables to set the number of characters you want to compare.

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

The `strncasecmp()` Function

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

The `strncasecmp()` Function



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

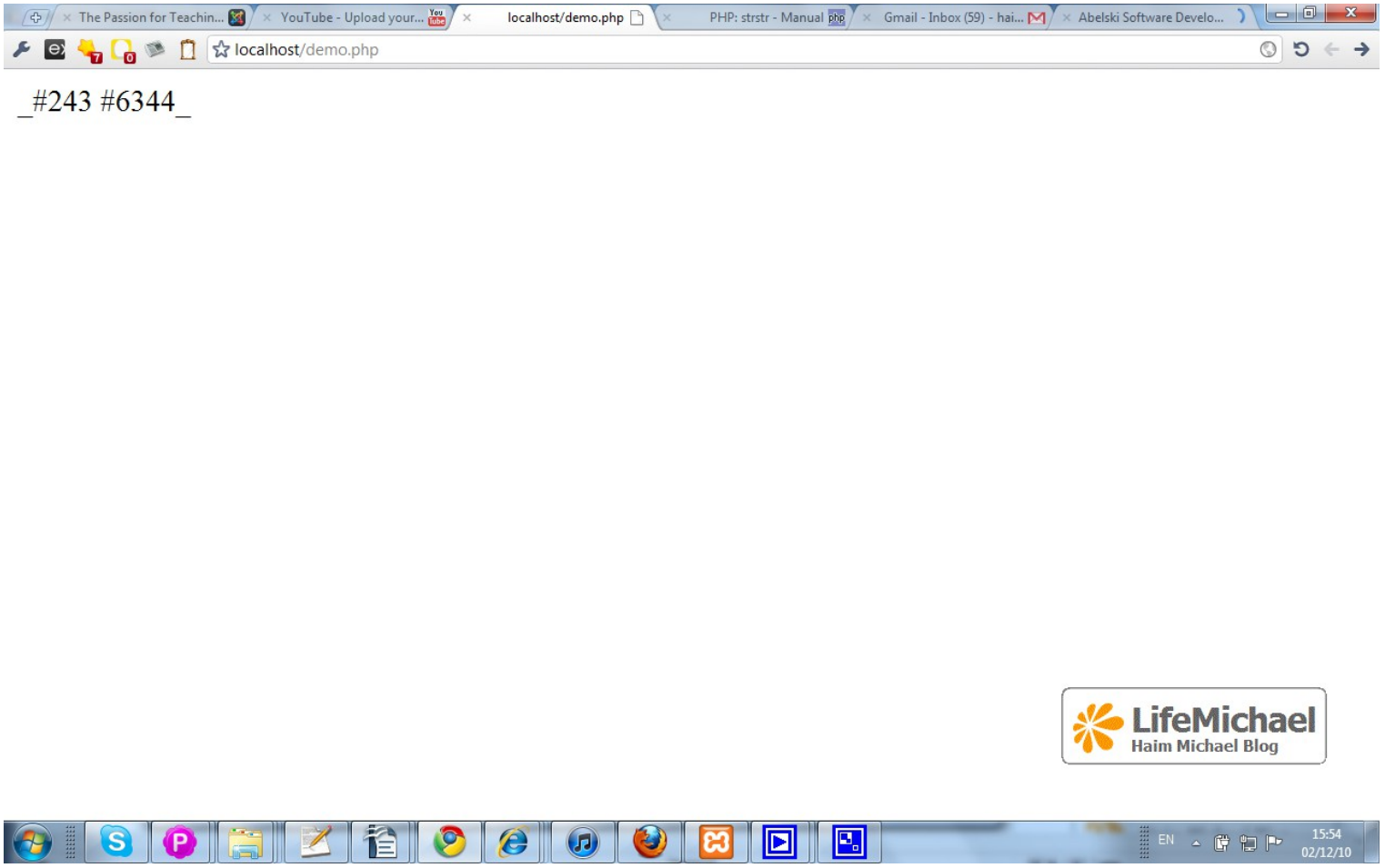
```
string strstr ( string $haystack , string $needle ,
    bool $before_needle )
```

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."_";
?>
```

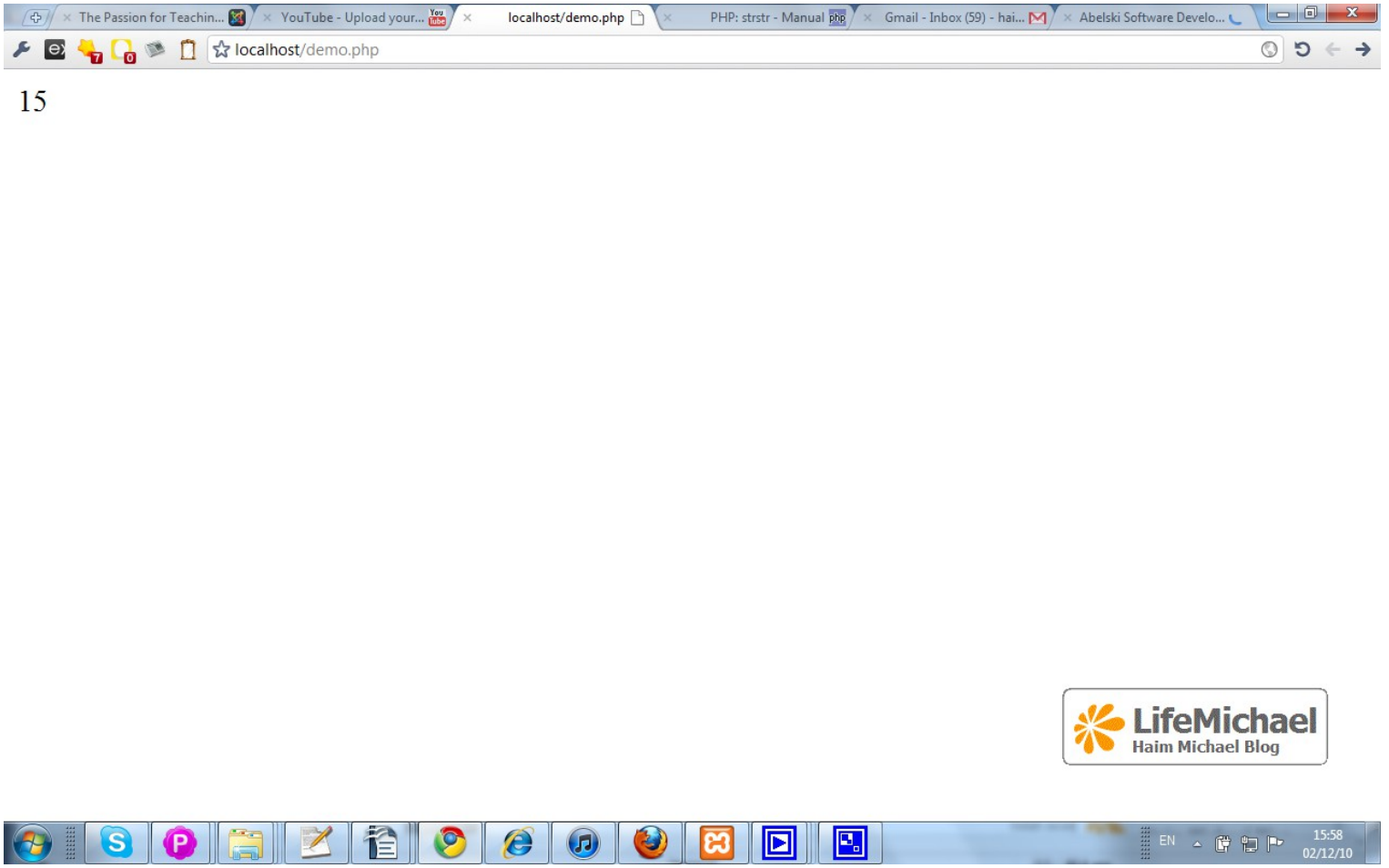
The `strpos()` & `strstr()` Functions



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

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

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



The `stripos()` & `stristr()` Functions

- The `stripos()` & `stristr()` do the same work `strpos()` & `strstr()` do... with one difference: The `stripos()` & `stristr()` 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 ]] )
```

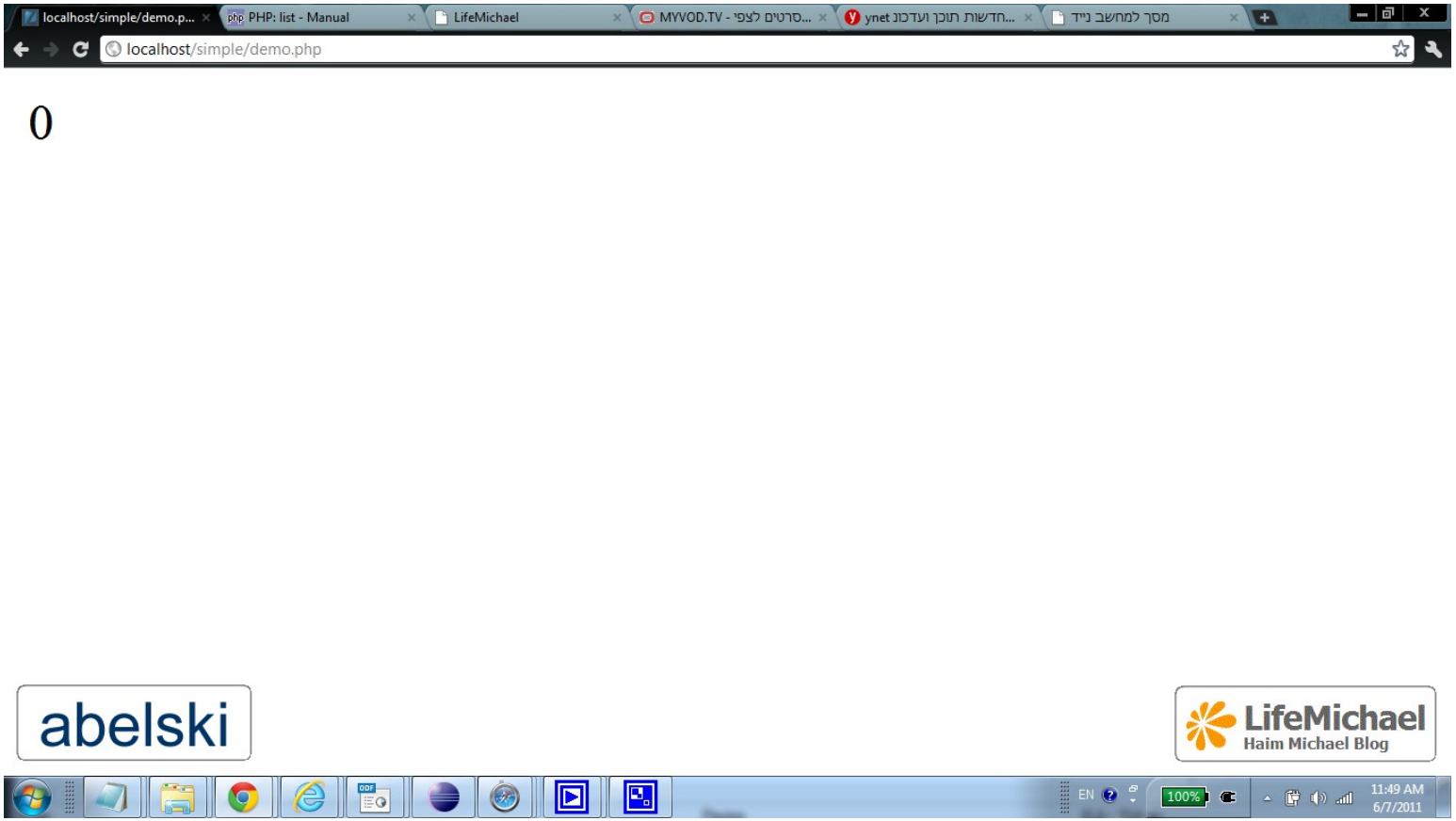
This function returns the length of the initial segment of str1 which consists entirely of characters in str2 .

- 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 `strspn()` & `strcspn()` Functions

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

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



The `str_replace()` Function

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

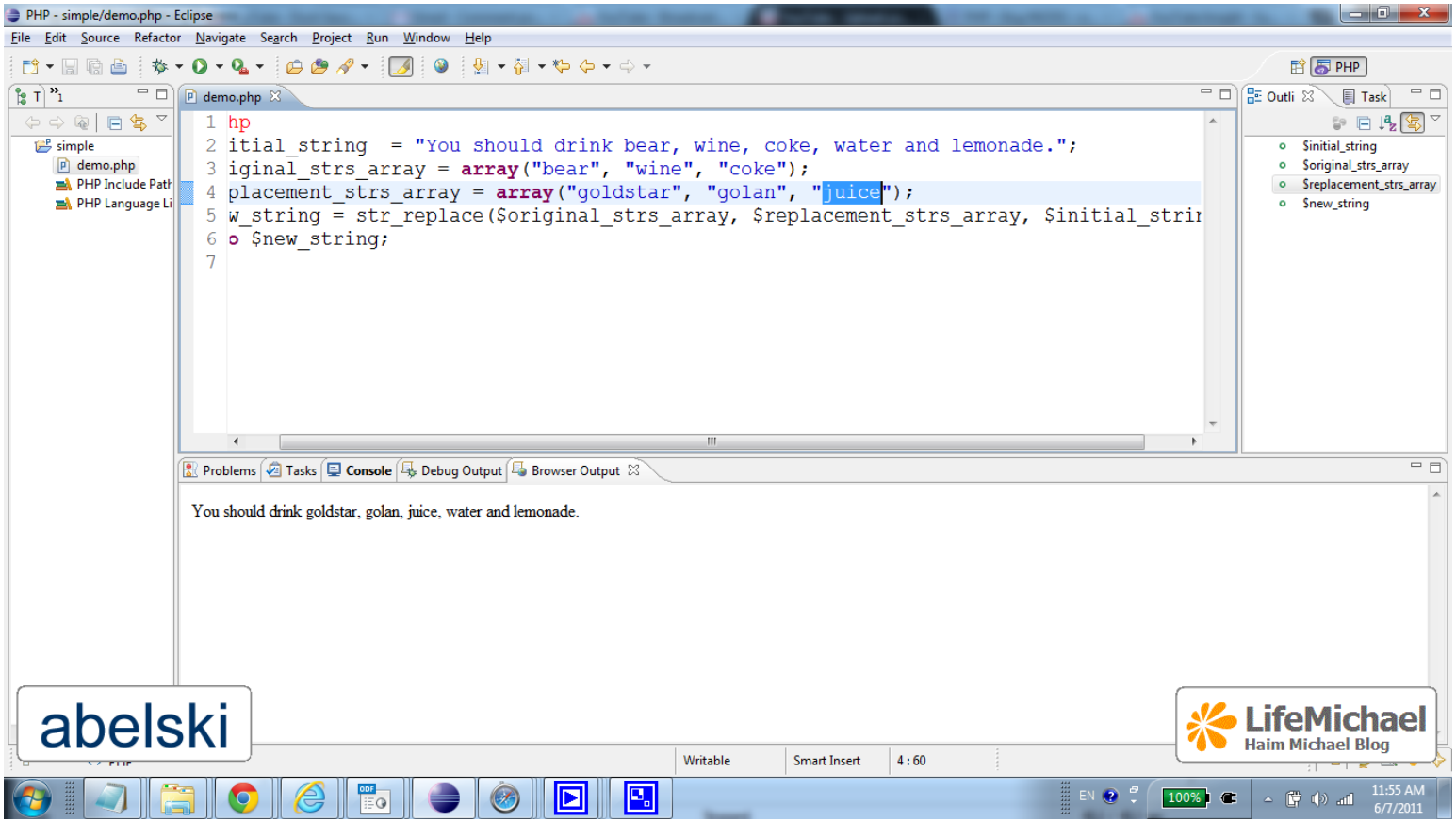
```
mixed str_replace ( mixed $search ,
                    mixed $replace ,
                    mixed $subject [, int &$count ] )
```

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
<?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



The `substr_replace()` Function

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

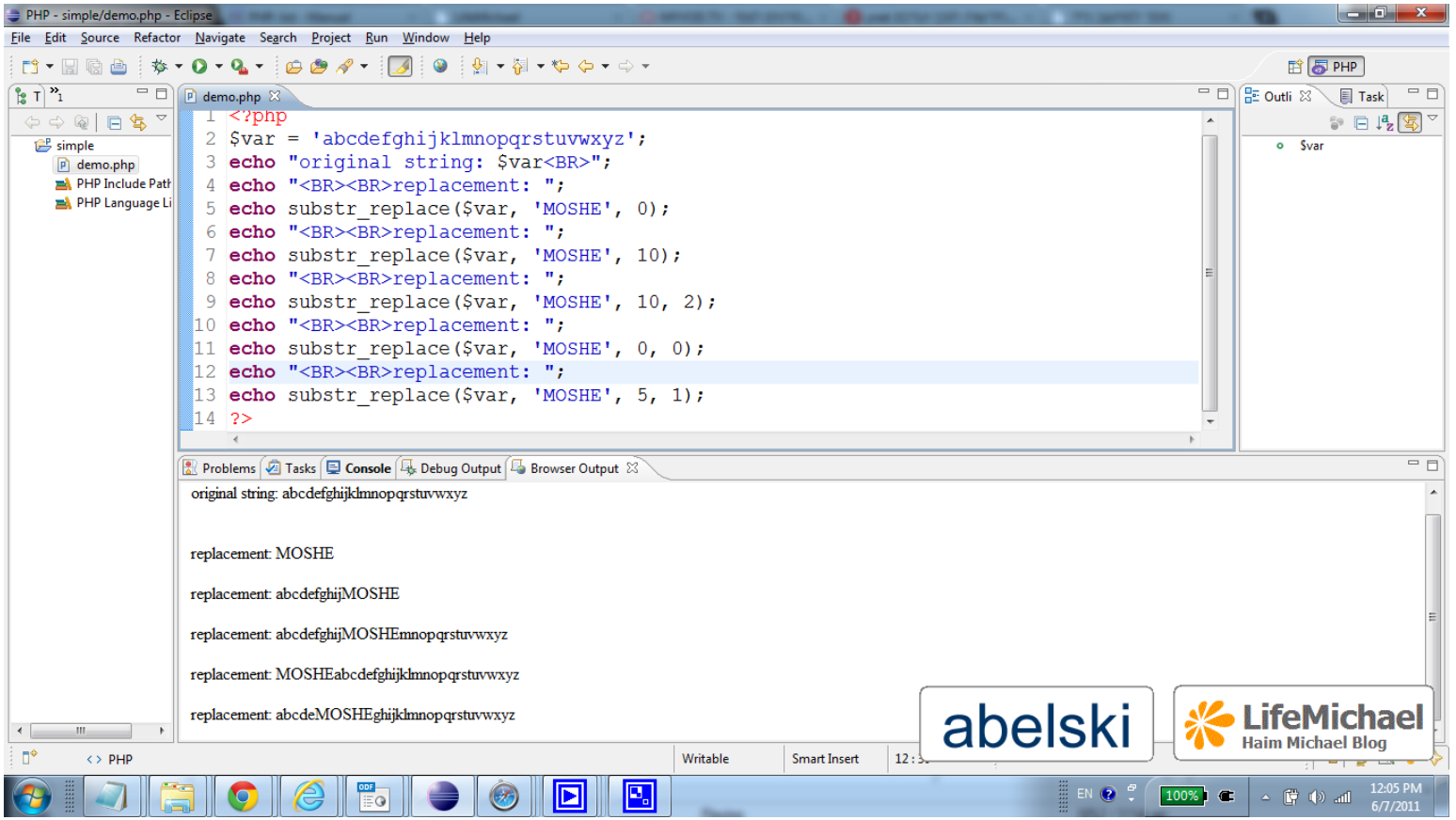
```
mixed substr_replace ( mixed $string , string $replacement ,
                                    int $start [, int $length ] )
```

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
<?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



The `substr()` Function

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

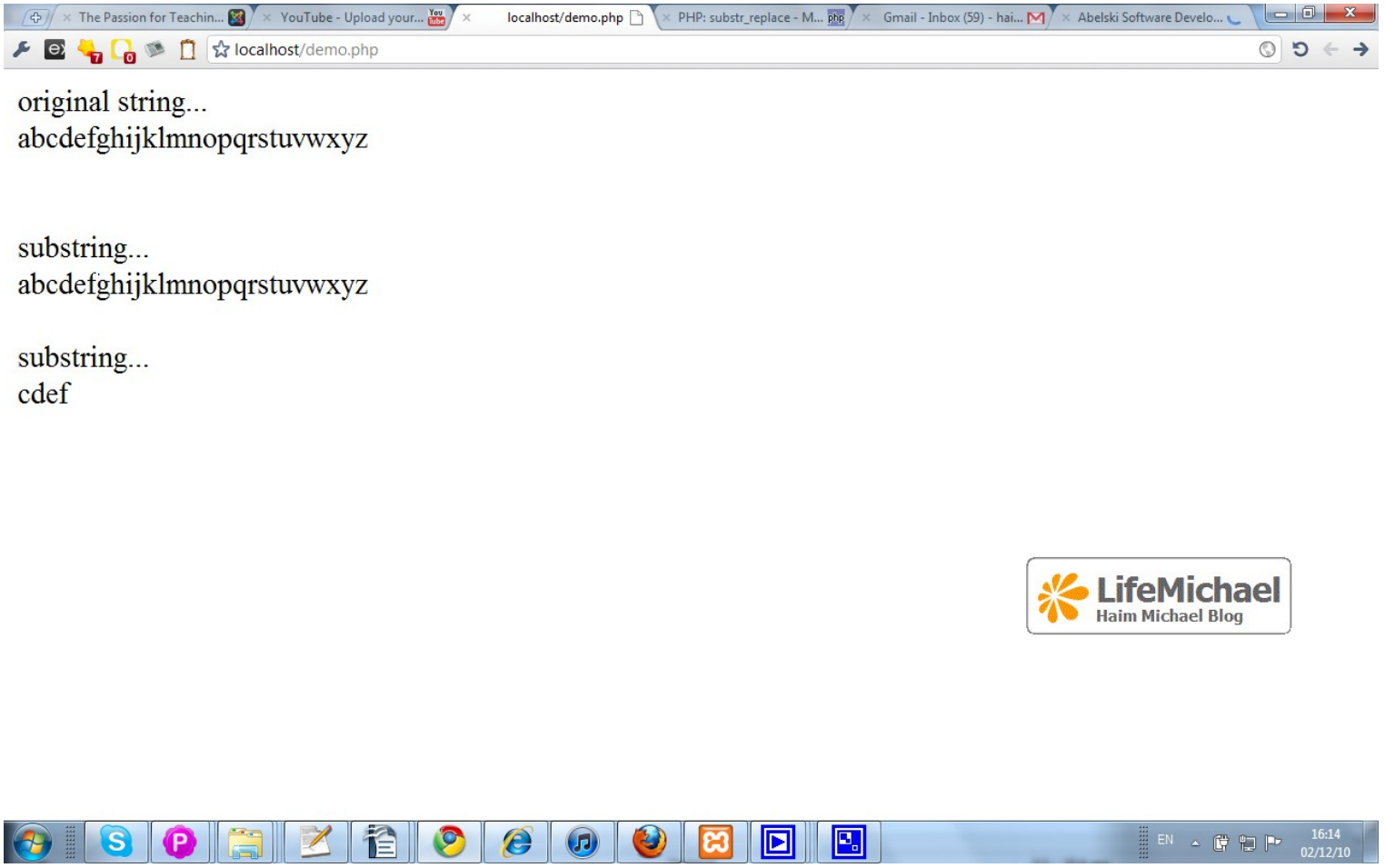
```
string substr (  string $string ,
                 int $start [, int $length ] )
```

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



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

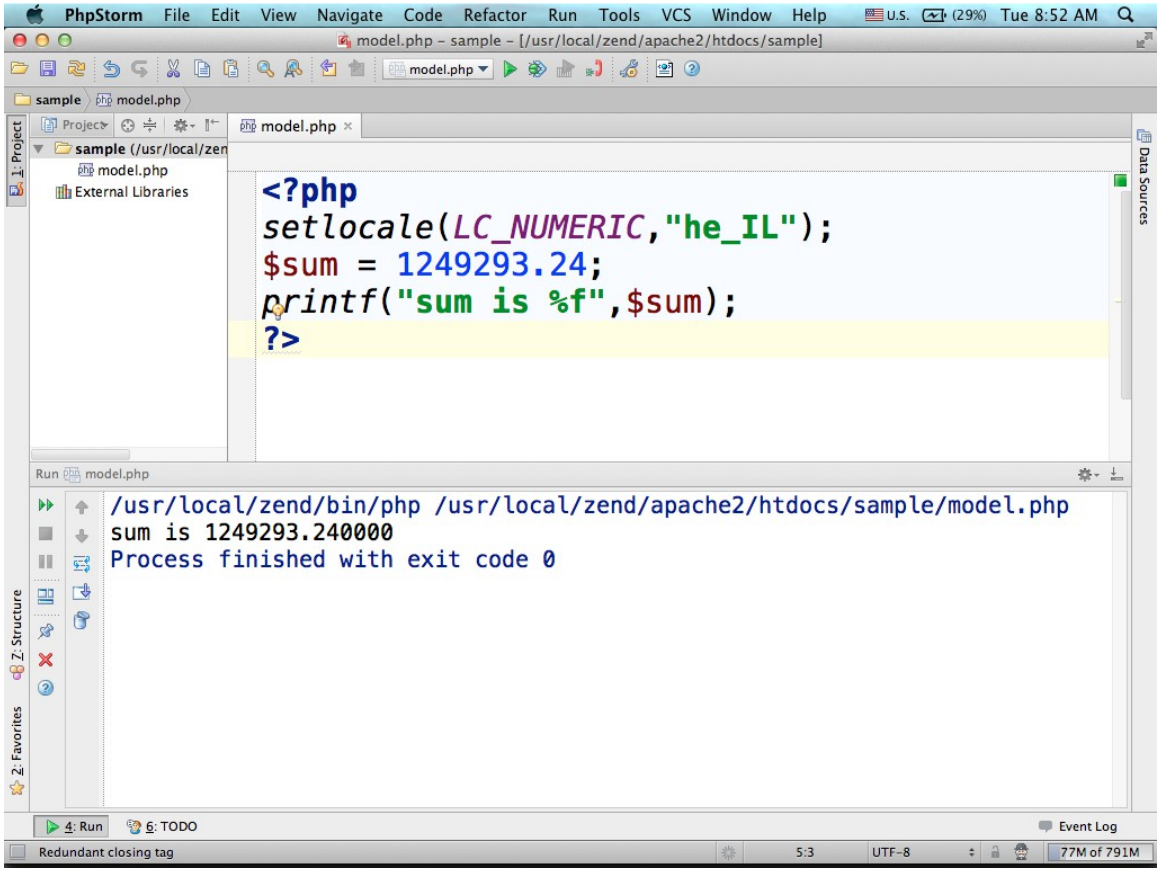
de_DE

fr_FR

The `setlocale()` Function

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

The `setlocale()` Function



The `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).

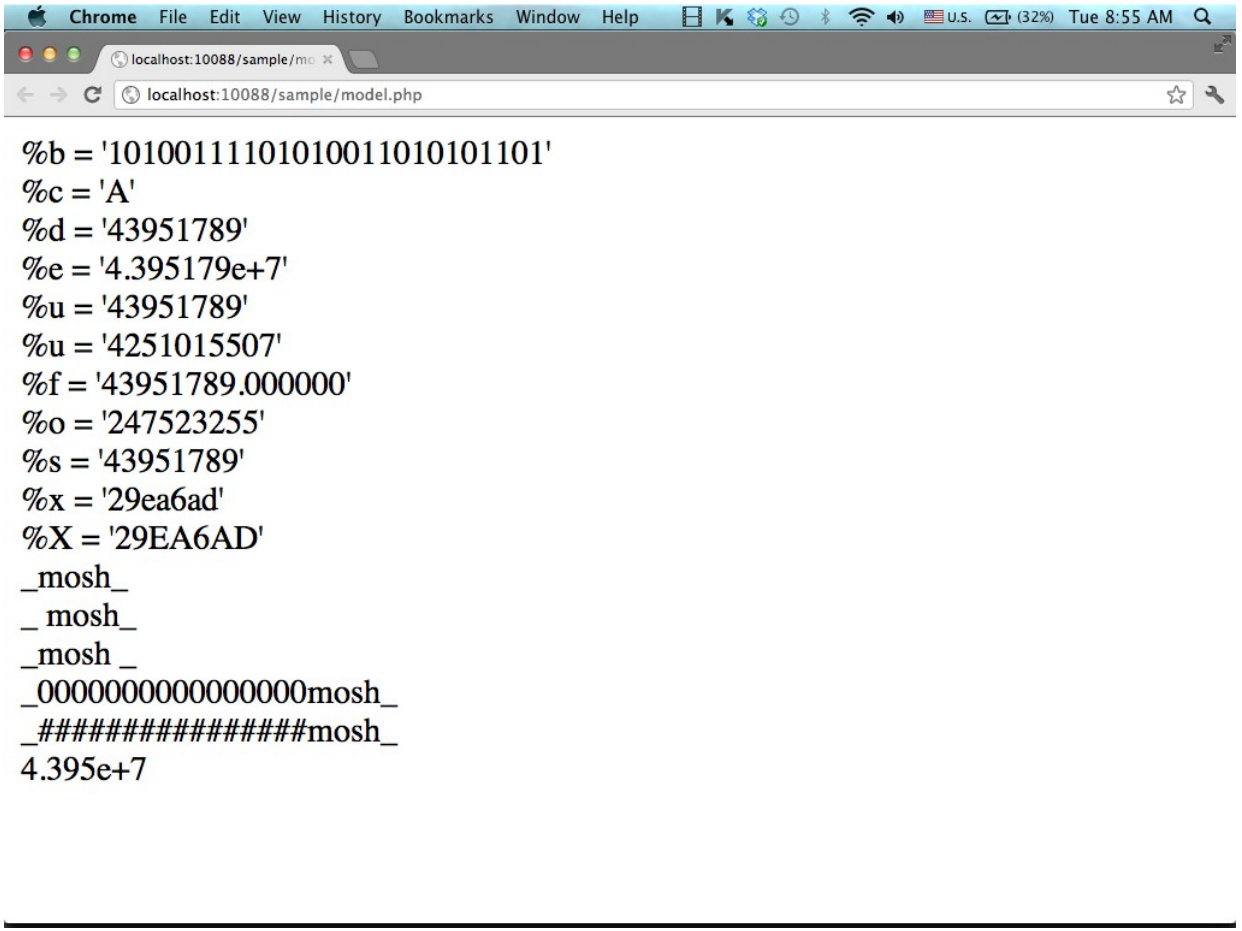
The `printf()` Function

```
<?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



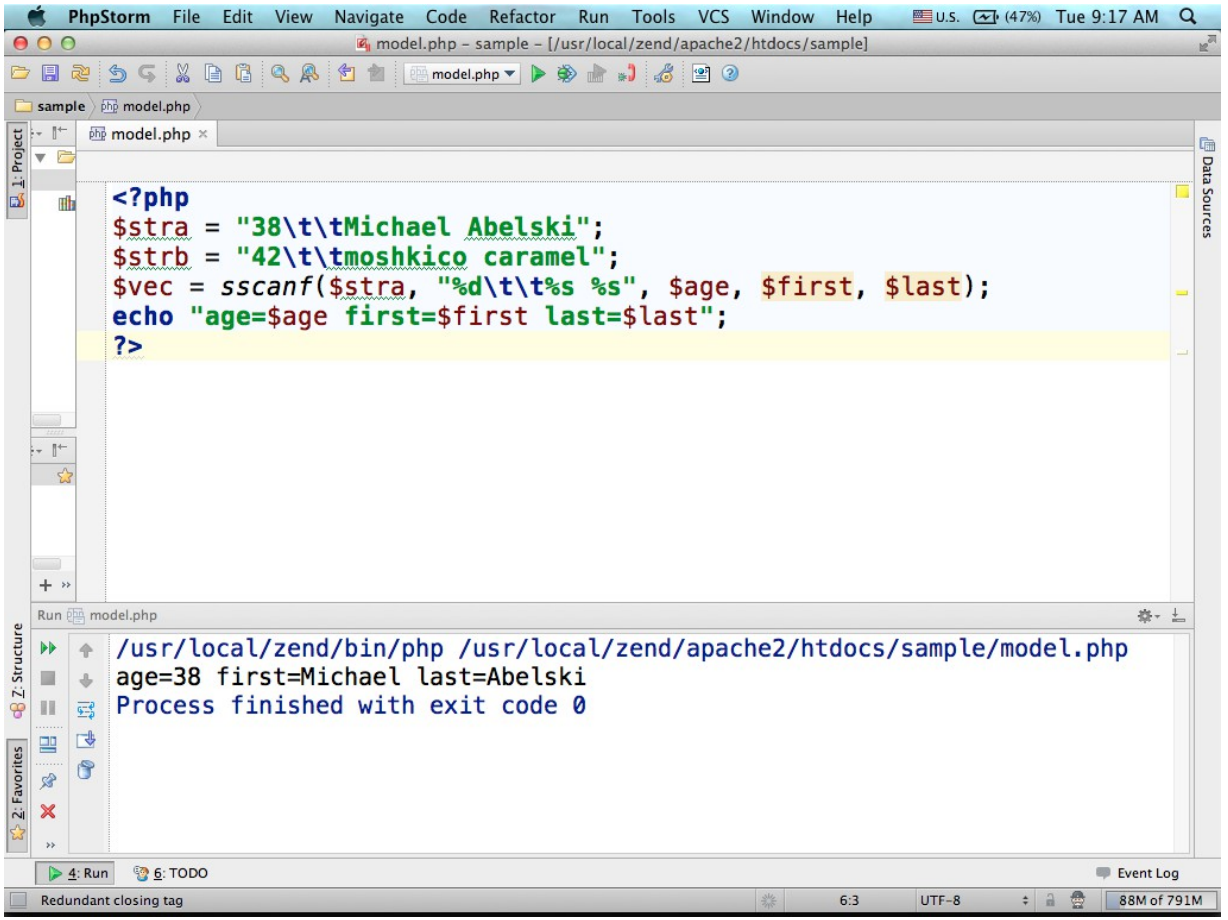
The `sscanf()` Function

- 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



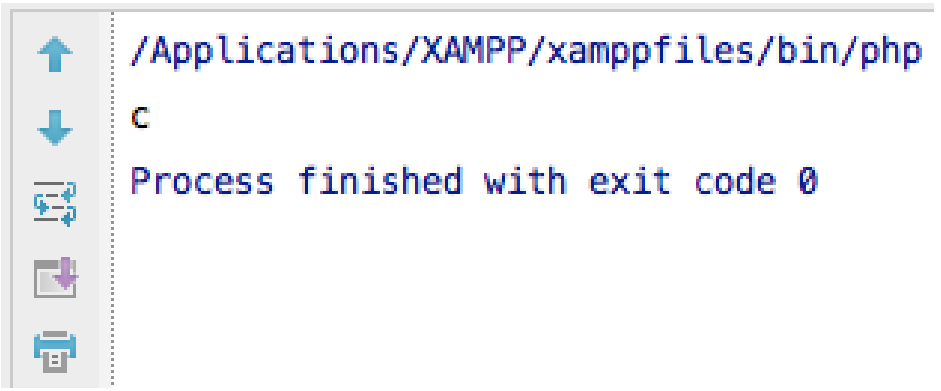
Strings Direct Dereferencing

- 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



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 and 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`".

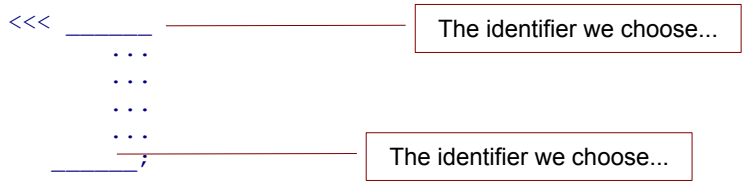
Embedded Variables

- 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 Heredoc

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


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";
?>
```


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 )
```

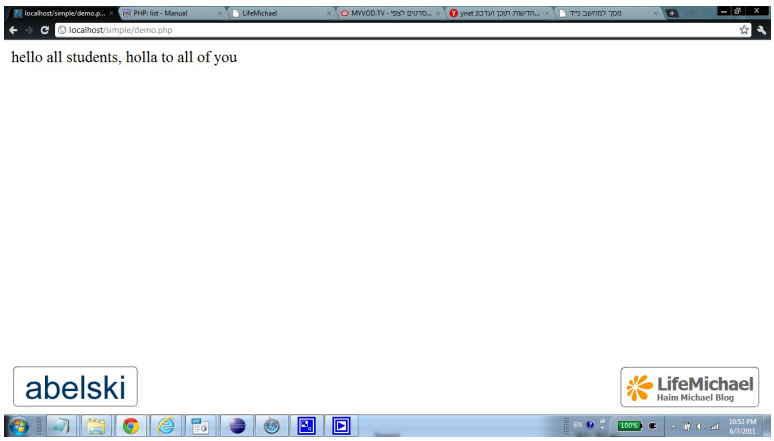
The `strtr()` Function

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

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

The `strtr()` Function



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]";
}
?>
```

YouTube

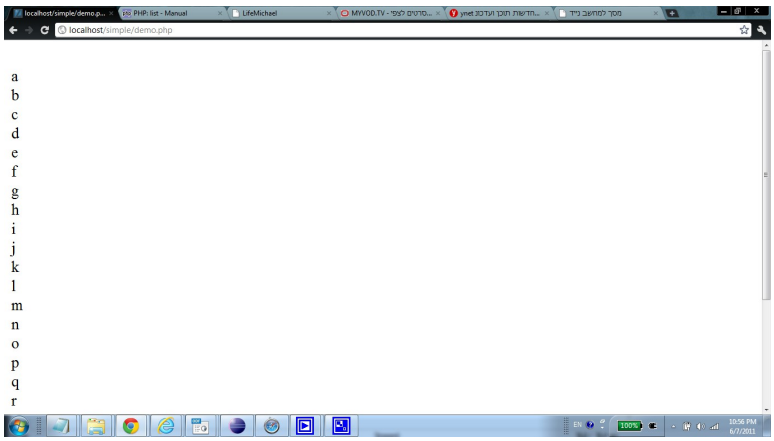
07/30/14 © Abelski eLearning 15

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



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

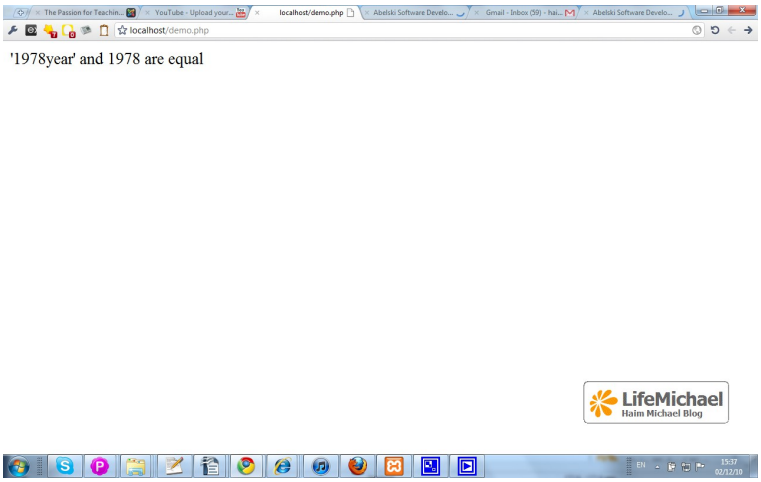
- Comparing strings using the == comparison operator is involved with a transparent conversion of textual strings to numeric values.

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

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



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.

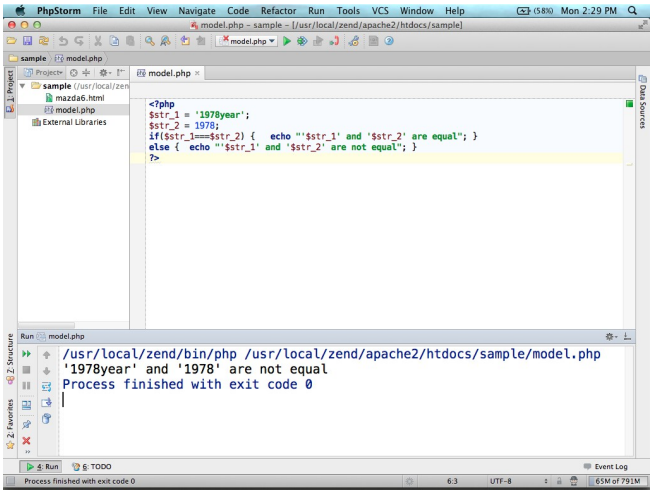
```
<?php
$str_1 = '1978year';
$str_2 = 1978;
if($str_1===$str_2) {  echo "'$str_1' and '$str_2' are equal"; }
else {  echo "'$str_1' and '$str_2' are not equal"; }
?>
```

'1978year' and '1978' are not equal

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



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 )
```

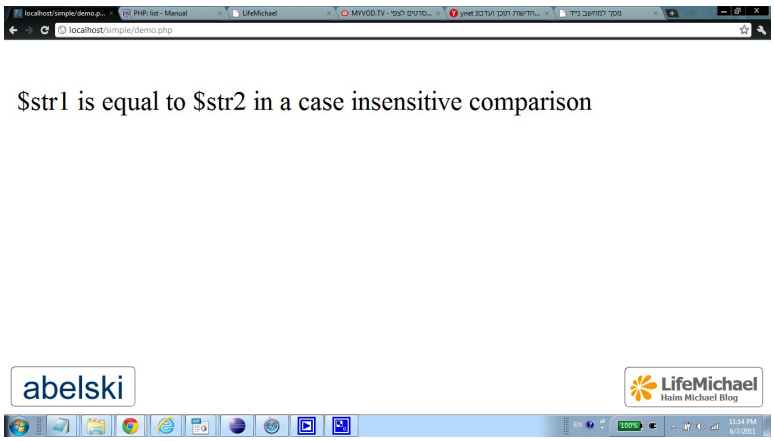
The `strcmp()` & `strcasecmp()` Functions

```
<?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';
}
?>
```

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



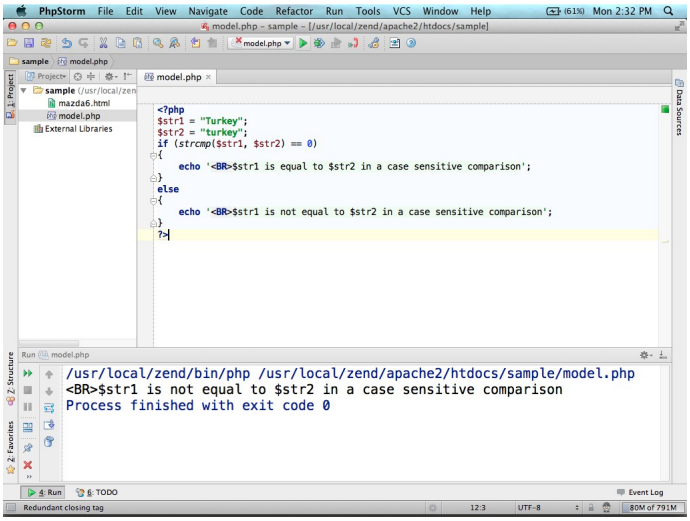
You can find this code sample (strcmp_sample.php) in this topic's samples folder.

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 `strncasecmp()` Function

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

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

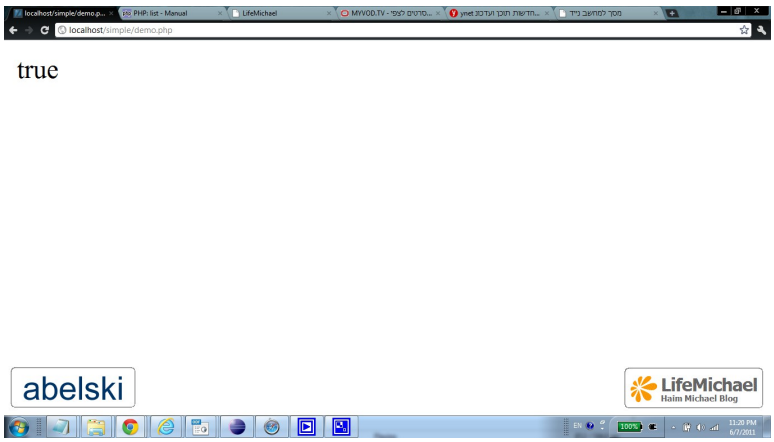
The `strncasecmp()` Function

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

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



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.

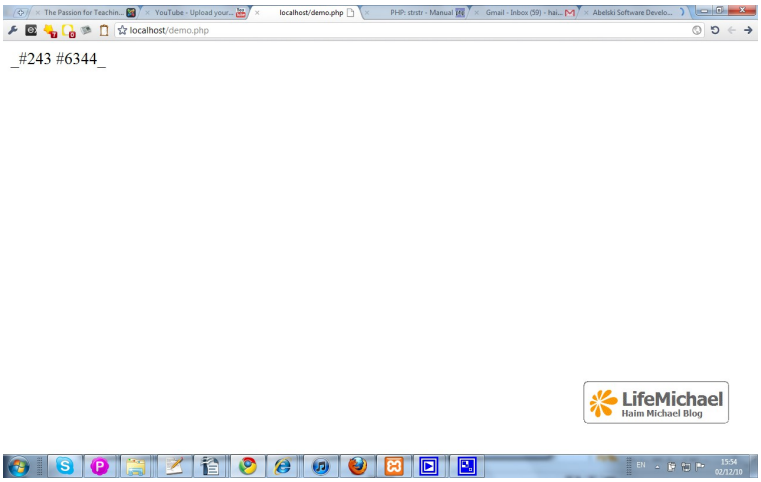
The `strpos()` & `strstr()` Functions

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

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



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;
?>
```

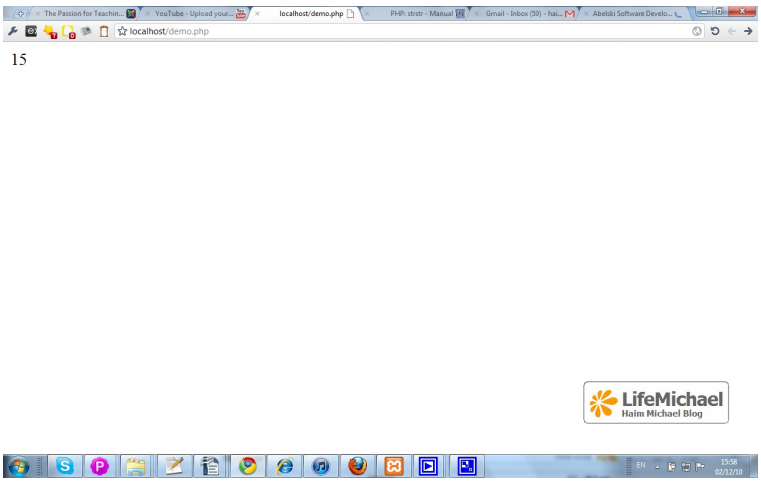
YouTube

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



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 `stripos()` & `stristr()` Functions

- The `stripos()` & `stristr()` do the same work `strpos()` & `strstr()` do... with one difference: The `stripos()` & `stristr()` 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.

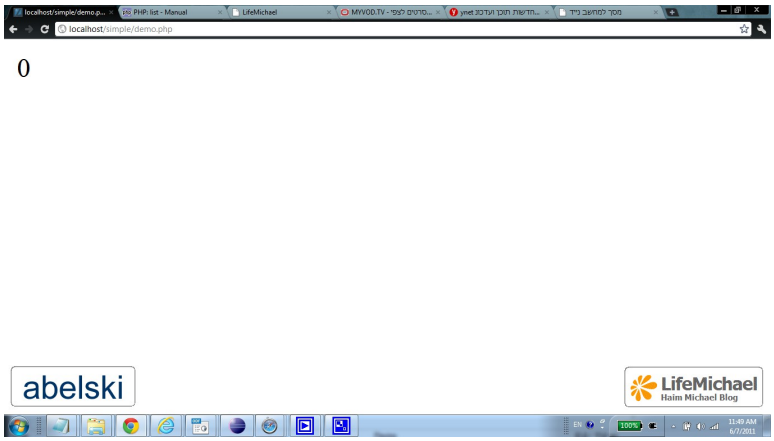
Both functions have optional parameters to specify where to start the search and the length of the string (from str1) that should be examined.

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

```php
<?php
$a = "i love java and php";
$b = "lov i e javaa";
$num = strcspn($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



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).

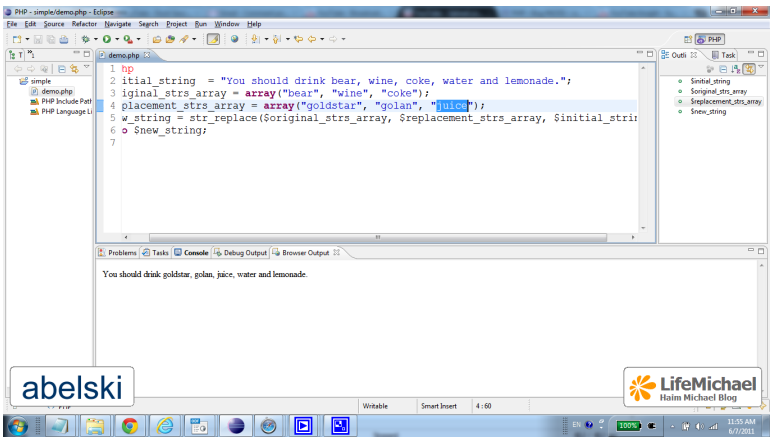
The `str_replace()` Functions

```php
<?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;
?>
```

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



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

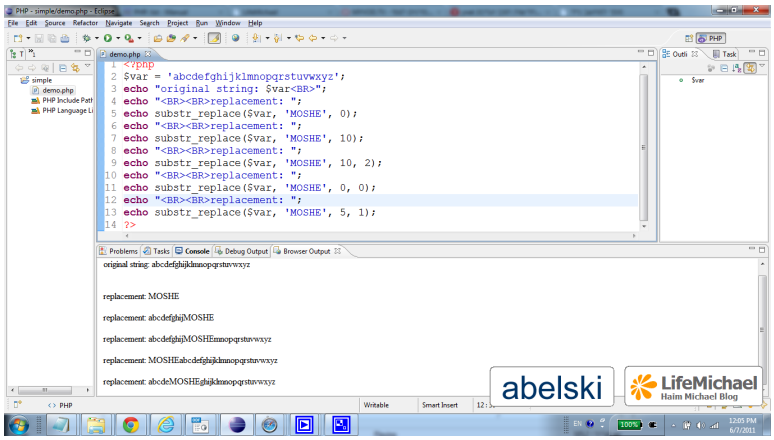
```php
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string: $var<BR>";
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 0);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 10);
echo "<BR><BR>replacement: ";
echo substr_replace($var, 'MOSHE', 10, 2);
echo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

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). I length is zero then this function will have the effect of inserting replacement into string at the given start offset.

The `substr_replace()` Function



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). I 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.

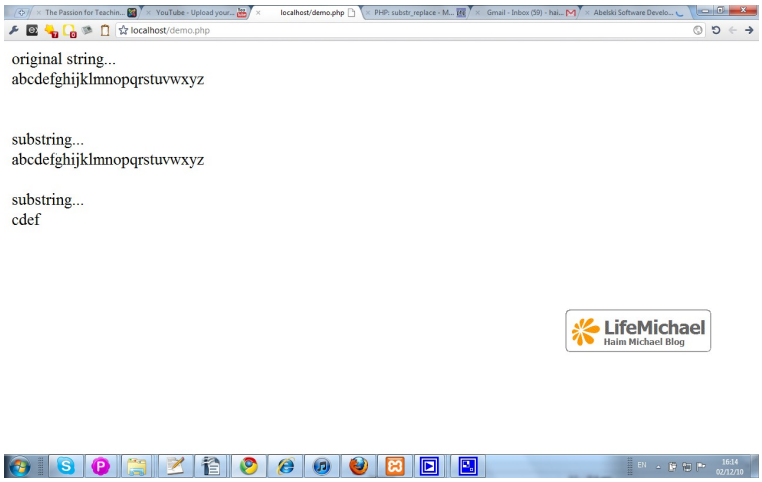
The `substr()` Function

```
<?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);
?>
```

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



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

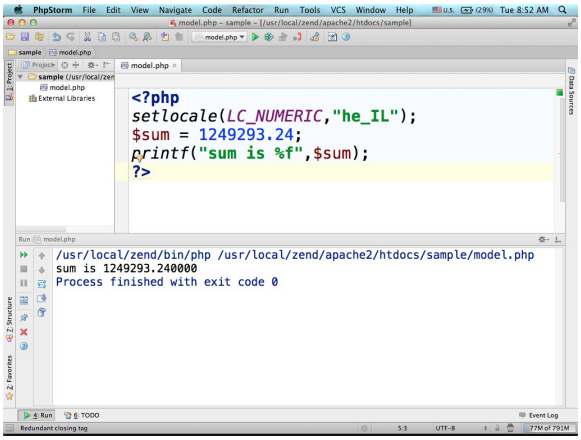
The $locale parameter can be any of the valid values according to http://www.w3.org/WAI/ER/IG/ert/iso639.htm.

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 `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);
```

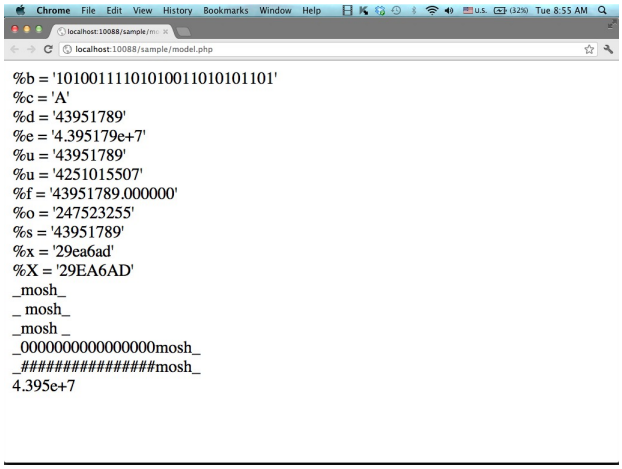
The `printf()` Function

```
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);
?>
```


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



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.

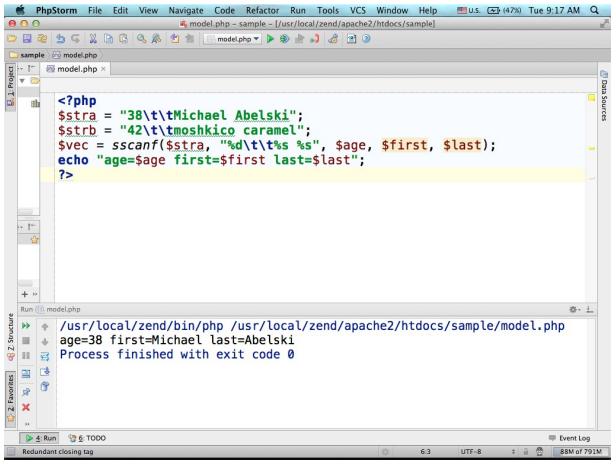
The `sscanf()` Function

```
<?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>";
?>
```

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



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

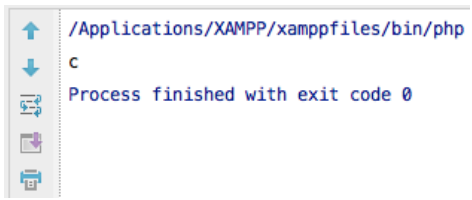
Strings Direct Dereferencing

- 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



The Output