F. Y. BCA

Subject – Web Development using PHP

Chapter 3 – Functions and Strings

Topics to be covered

- Introduction to string
- Built in string manipulation functions

INTRODUCTION

- A string is series of characters, where a character is the same as a byte. This means that PHP only supports a 256-character set, and hence does not offer native Unicode support.
- A string literal can be specified in four different ways:
  - Single quoted
  - Double quoted
  - Heredoc syntax
  - newdoc syntax(since PHP 5.3.0)

WAYS

- The simplest way to specify a string is to enclose it in single quotes
- A third way to delimit string is the heredoc syntax: <<<. After this operator, an identifier is provided, then a newline. The string itself follows, and then the same identifier again to close the quotation.
- Nowdocs are to single-quoted strings what heredocs are to double-quoted strings. A nowdoc is specified similarly to a heredoc, but *no parsing is done* inside a nowdoc

Single quoted

- This type of strings does not processes special char.

Example1:

```
<?php
 // single-quote strings
 $site  = 'Welcome to SBUP';
 echo $site;
?>
```

Example2:

```
<?php
    // single-quote strings
    $s = ‘SBUP';
    echo 'Welcome to $s';
?>
```

DOUBLE QUOTED

- Unlike single-quote strings, double-quote strings in PHP is capable of processing special characters.

Example 3:

```
<?php
    $name = "Krishna";
    echo "The name of the boy is $name \n";
    echo 'The name of the boy is $name';
?>
```

Example 4:

```
<?php
    // double-quote strings
    echo "Welcome to SBUP\n";
    $site = “SBUP";
    echo "Welcome to $site";
?>
```

Difference between single and double quote

- In PHP, weuse single quote to define a constant string, like 'a' , 'my name' , 'abc xyz' , while using double quote to define a string contain identifier like "a $b $c $d" .
- Example : echo "my $a";

USED SPECIAL CHARACTERS

- The character beginning with a backslash(“\”) are treated as escape sequences and are replaced with special characters. Here are few important escape sequences.
  - “\n” is replaced by a new line
  - “\t” is replaced by a tab space
  - “\$” is replaced by a dollar sign
  - “\r” is replaced by a carriage return
  - “\\” is replaced by a backslash
  - “\”” is replaced by a double quote
  - “\'” is replaced by a single quote
- The string starting with a dollar sign(“$”) are treated as variables and are replaced with the content of the variables.

BUILT-IN STRING FUNCTIONS

- **strlen() function -**This function is used to find the length of a string. This function accepts the string as argument and return the length or number of characters in the string.

- **Syntax:** strlen(*string*)
  where *string* is Required. Specifies the string to check

Example 5:

```
<?php
   echo strlen("Hello world!");
?>
```

Output: 12

Built-in String functions

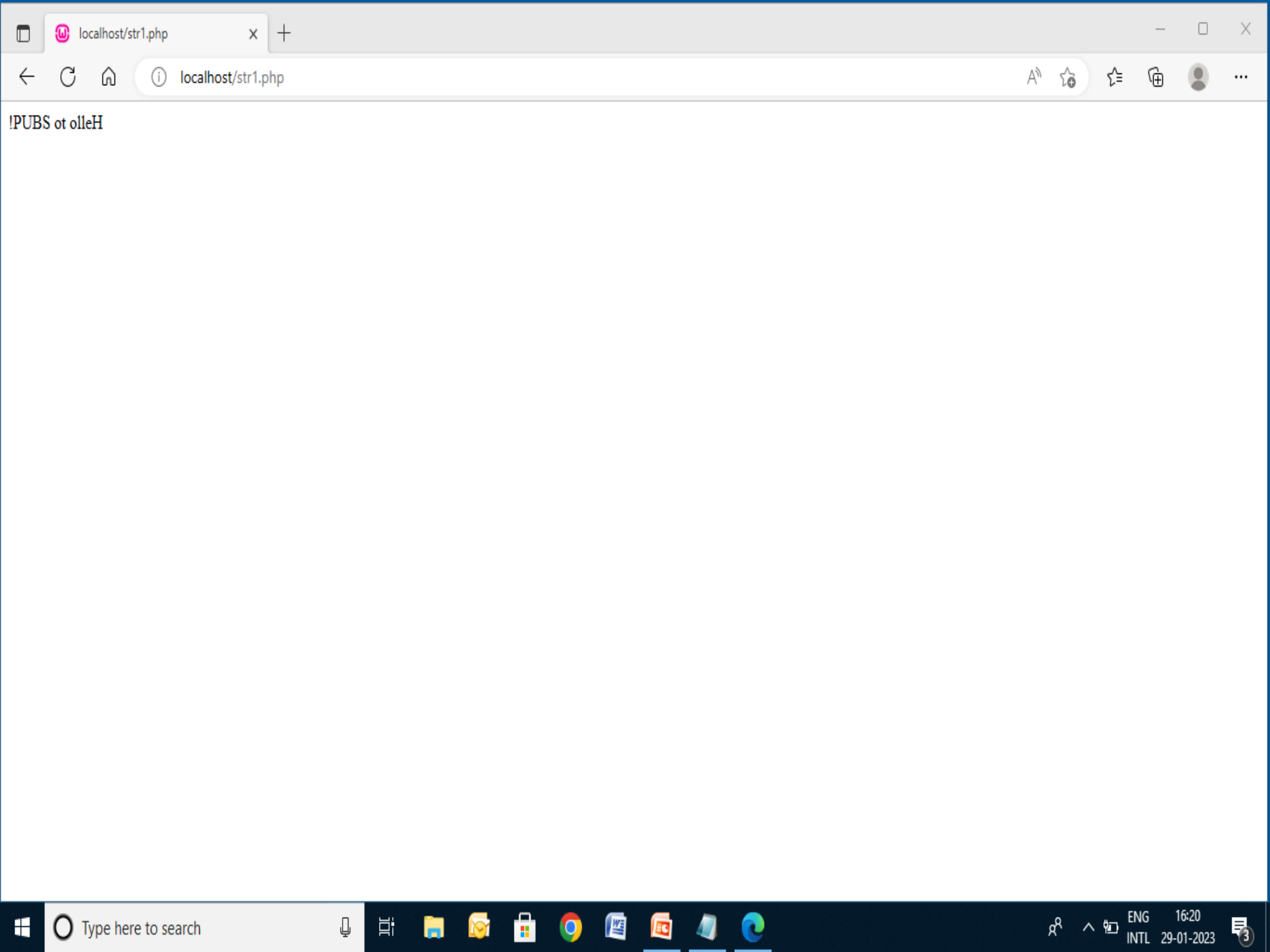
- **strrev() function**: This function is used to reverse a string. This function accepts a string as argument and returns its reversed string.

- Syntax: strrev(*string*)

Where *string* specifies the string to reverse

Example 6:

```
<?php
   echo strrev("Hello to SBUP!");
?>
```


localhost/str1.php
localhost/str1.php
!PUBS ot olleH
Type here to search
ENG
INTL
16:20
29-01-2023

Built-in String functions

- **str_replace() function:** This function takes three strings as arguments. The argument is the original string and the first argument is replaced by the second replaces all occurrences of the first argument in the original string by second argum

- Syntax: str_replace(*find,replace,string,count*)

Where,

find - Specifies the value to find

replace - Specifies the value to replace the value in *find*

string - Specifies the string to be searched

count - A variable that counts the number of replacements

Example 7:

```
<?php
  echo str_replace("Hello","Omkar","Hello World!"),"\n";
  echo str_replace("Hello","BOY","Hello Saksham!"),"\n";
?>
```

Built-in String functions

- **strpos() function** - Search For a Text Within a String
- Function searches for a specific text within a string.
- If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
- Syntax: strpos(*string*,*find*,*start*)

Where,

string - Specifies the string to find

find - Specifies the string to search

start - (Optional) Specifies where to begin the search. If *start* is a negative number, it counts from the end of the string.

Example 9:

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

BUILT-IN STRING FUNCTIONS

- **str_word_count() function** - counts number words in string.

- **Syntax:** str_word_count(*string,return,char*)

Where,

String - Specifies the string to check*return*Optional. Specifies the return value of the str_word_count() function.

Possible values:

0 - Default. Returns the number of words found

1 - Returns an array with the words from the string

2 - Returns an array where the key is the position of the word in the string, and value is the actual word

Char – *(*Optional) Specifies special characters to be considered as words.

Example 10:

```
<?php
   echo str_word_count("Hello world!");  // Output: 2
?>
```

BUILT-IN STRING FUNCTIONS

- **trim() function:** This function allows us to remove whitespaces or strings from both sides of a string.
  - ltrim() - Removes whitespace or other predefined characters from the left side of a string
  - rtrim() - Removes whitespace or other predefined characters from the right side of a string

- Syntax: trim(*string,charlist*);

Where,

string - Specifies the string to check

charlist – (Optional) Specifies which characters to remove from the string.

If omitted, all of the following characters are removed:
"\0" - NULL
"\t" - tab
"\n" - new line
"\x0B" - vertical tab
"\r" - carriage return
" " - ordinary white space

Example 10:

```
<?php
   echo trim("Hello World!", "Hed!");
 ?>
```

of all possible is created.

- **Syntax:** str_shuffle(*string*)

BUILT-IN STRING FUNCTIONS

Where,

String - Specifies the string to shuffle

Example 11:

```
<?php
   $str1="Hello!";
   echo str_shuffle($str1);
 ?>
```

Note : randomly shuffles all the characters of a string.

Built-in String functions

- **strcmp ()function**: compares two strings and returns integer value 0 or 1.
- If this function returns 0, the two strings are equal.

- Syntax: strcmp(*string1,string2*)

Where,
string1 - Specifies the first string to compare
string2 - Specifies the second string to compare

Example 12:

```
<?php
$var1 = "Hello";
$var2 = "hello";
if (strcmp($var1, $var2) !== 0)
 {
   echo"$var1 is not equal to $var2";
}
?>
```

Built-in String functions

- **strcmpi ()function**: compares two strings and returns integer value 0 or 1.
- If this function returns 0, the two strings are equal.
- Syntax: strcmpi(*string1,string2*)

Where,

string1 - Specifies the first string to compare

string2 - Specifies the second string to compare

Example 13:

```
<?php
$var1 = "Hello";
$var2 = "hello";
if (strcmpi($var1, $var2) !== 0)
 {
    echo '$var1 is not equal to $var2";
}
?>
```

String - Specifies the string to split

Length – (Optional) Specifies the length of each array element. Default is 1

BUILT-IN STRING FUNCTIONS

Example 14:

```
<?php
    $str = "Hello Friend";

    $arr1 = str_split($str);
    $arr2 = str_split($str, 3);

    print_r($arr1);
    echo "<br>";
    print_r($arr2);
?>
```

Output:

Array ([0] => H [1] => e [2] => l [3] => l [4] => o [5] => [6] => F [7] => r [8] => i [9] => e [10] => n [11])

Array ([0] => Hel [1] => lo [2] => Fri [3] => end)

Built-in String functions

- strcasecmp (string $str1 , string $str2) function
- Binary safe case-insensitive string comparison

- Syntax: strcmp(*string1,string2*)

Where,

string1 - Specifies the first string to compare

string2 - Specifies the second string to compare

Example 15:

```
<?php
    $var1 = "Hello";
    $var2 = "hello";
    if (strcasecmp($var1, $var2) == 0)
    {
        echo '$var1 is equal to $var2 ';
    }

?>
```

Built-in String Functions

- The substr_compare() function compares two strings from a specified start position.
- Syntax: substr_compare(*string1,string2,startpos,length,case*)

string1 - Specifies the first string to compare

string2 - Specifies the second string to compare

startpos - Specifies where to start comparing in string1.
If negative, it starts counting from the end of the string

length – *(*Optional) Specifies how much of string1 to compare

case – *(*Optional) A boolean value that specifies whether or not to perform a case-sensitive compare:
FALSE - Default. Case-sensitive
TRUE - Case-insensitive

```php
<?php
echo substr_compare("abcde","bc", 1, 2); // 0
echo substr_compare("abcde","de", -2, 2); // 0
echo substr_compare("abcde","bcg", 1, 2); // 0
echo substr_compare("abcde","BC", 1, 2, true); // 0
echo substr_compare("abcde","bc", 1, 3); // 1
echo substr_compare("abcde","cd", 1, 2); // -1
echo substr_compare("abcde","abc", 5, 1); // warning
?>
```

Output:

```
0
0
0
0
1
-1
-1
```

BUILT-IN STRING FUNCTIONS

- **str_pad() function** - returns the input string padded on the left, the right, or both sides to the specified padding length.
- If the optional argument pad_string is not supplied, the input is padded with spaces, otherwise it is padded with characters from pad_string up to the limit.
- **Syntax:** str_pad(*string,length,pad_string,pad_type*)

Where,

String - *S*pecifies the string to pad

Length - Specifies the new string length.

If this value is less than the original length of the string, nothing will be done

pad_string – *(*Optional) Specifies the string to use for padding. Default is whitespace

pad_type – *(*Optional) Specifies what side to pad the string.

Possible values:

STR_PAD_BOTH - Pad to both sides of the string.

If not an even number, the right side gets the extra padding

STR_PAD_LEFT - Pad to the left side of the string

STR_PAD_RIGHT - Pad to the right side of the string. This is default

Example 16:

```
<?php
$input = "Alien";
echo str_pad($input, 10)."<br>";
echo str_pad($input, 10, "*", STR_PAD_LEFT)."<br>";
echo str_pad($input, 10, "*", STR_PAD_BOTH)."<br>";
echo str_pad($input,  6, "*")."<br>";
echo str_pad($input,  3, "*")."<br>";
?>
```

Output:

Alien
*****Alien
Alien*
Alien*
Alien

Thank You