Unit-IV
String Manipulation and Regular Expression
Lecture 2: Formatting Strings

Typically when we get data from an HTML form interface we need to tidy up user strings before using them.

- **Trimming Strings:** The first step in tidying up is to trim any excess whitespace from the string. It is never compulsory but it can be useful if you are going to store the string in a file or database, or you are going to compare it to other strings.
  PHP provides three useful functions for this purpose:

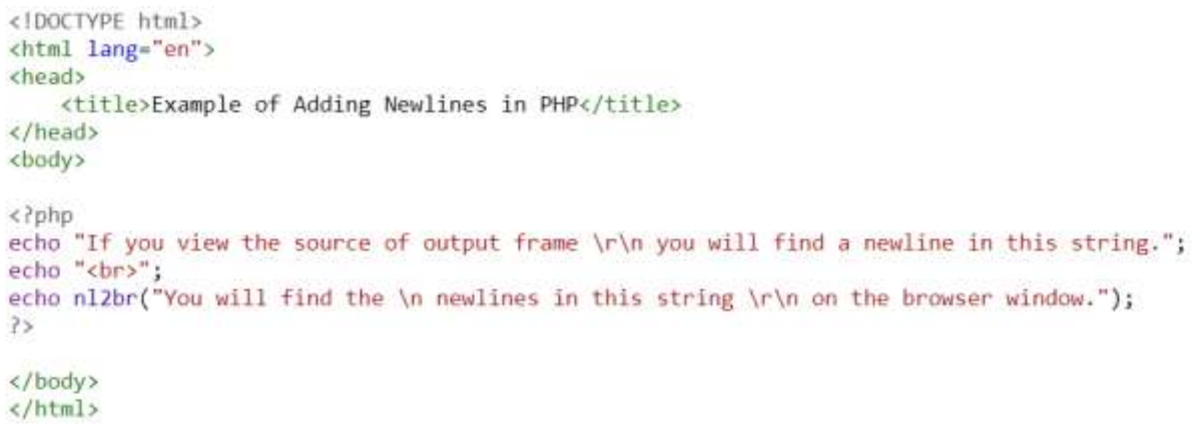
  e.g.

  ```
  $name = trim($name);
  $email = trim($email);
  ```

  - **trim()**
    - The trim() function strips whitespace from the start and end of a string and returns the resulting string.
    - The characters it strips by default are newlines and carriage returns (\n and \r), horizontal and vertical tabs (\t and \v), end of string characters (\0) and spaces.
  - **ltrim()**
    - The ltrim() removes whitespace from the start (or left) only.
  - **chop()**
    - The chop() removes whitespace from the end (or right) only.

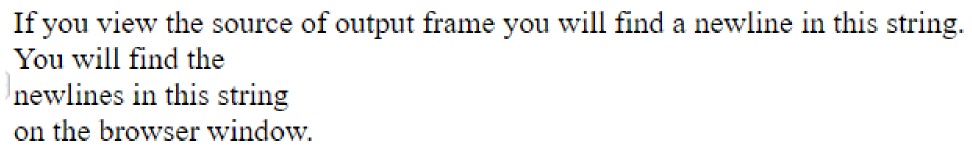
- **Formatting strings for Presentation:** PHP has a set of functions that you can use to reformat a string in different ways:
  - **Using HTML Formatting: the nl2br() Function:**
    - The nl2br() function takes a string as parameter and replaces all the newlines in it with the HTML <br> tag.
    - This is useful for echoing a long string to the browser.
    - HTML disregards plain whitespace, so if you don't filter the output through nl2br(), it will appear on a single line(except for newlines forced by the browser window).
    - You can use the PHP newline characters `\n` or `\r\n` to create a new line inside the source code. However, if you want the line breaks to be visible in the browser too, you can use the PHP `nl2br()` function which inserts HTML line breaks before all newlines in a string.

- **Example:**



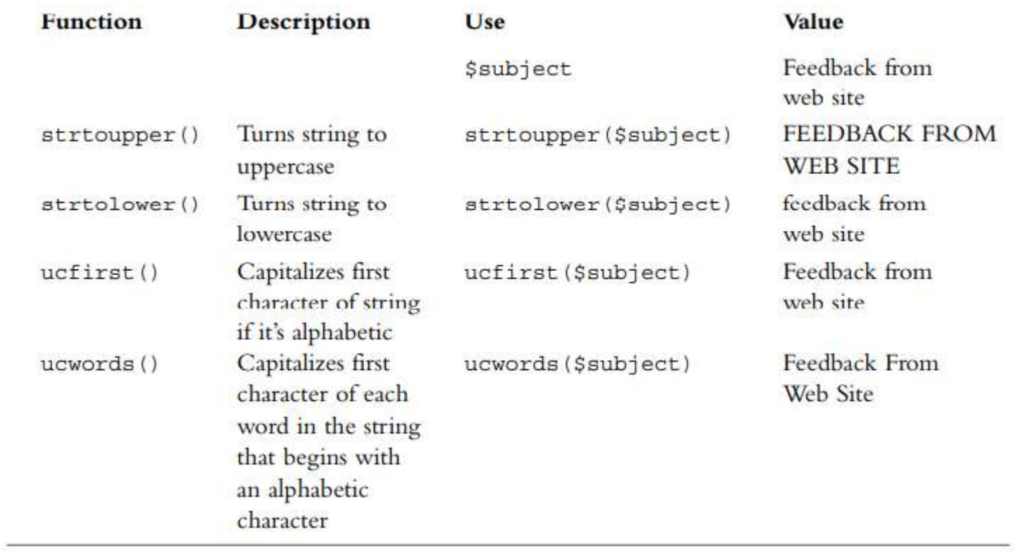
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example of Adding Newlines in PHP</title>
</head>
<body>

<?php
echo "If you view the source of output frame \r\n you will find a newline in this string.";
echo "<br>";
echo nl2br("You will find the \n newlines in this string \r\n on the browser window.");
?>

</body>
</html>
```

Output will be:



```
If you view the source of output frame you will find a newline in this string.
You will find the
newlines in this string
on the browser window.
```

- **Changing the case of a string:** One can also change the case of a string i.e. uppercase or lower case. e.g. consider we have a variable $subject containing the string "Feedback from the website". The various functions we can use to change the case are as below:



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

- **Formatting Strings for Storage: AddSlashes() and StripSlashes()**

  Certain characters are perfectly valid as part of a string but can cause problems, particularly when inserting data into a database because the database could interpret these characters as control characters. The problematic ones are quotes (single and double), backslashes(\) and the NUL character.

  There should be a way to escape these characters so that databases such as MySQL can understand that we meant a literal special character rather than a control sequence. To escape these characters, add a backslash in front of them.

  e.g. “ (double quote) becomes \”.

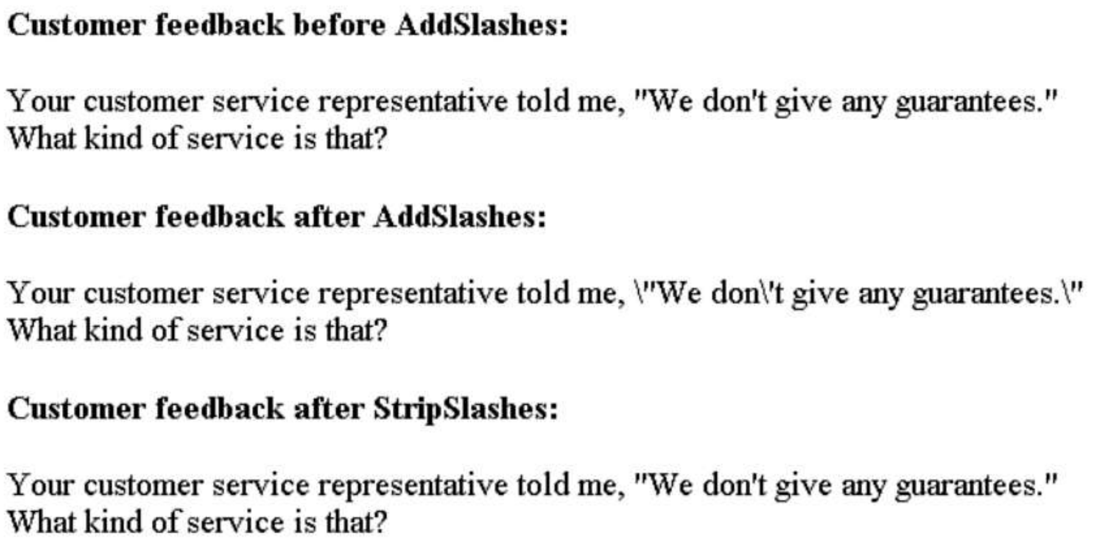
  PHP provides two functions specifically designed for escaping characters. Before you write any strings into a database, you should reformat them with AddSlashes().

  e.g. $feedback = AddSlashes($feedback);

  AddSlashes() take a string as parameter and returns the formatted string. Now, the string will be stored in the database with the slashes in it.

  When you retrieve the string, you will need to remember to take the slashes out for that PHP provides the function StripSlashes().

  $feedback = StripSlashes($feedback);



```
Customer feedback before AddSlashes:

Your customer service representative told me, "We don't give any guarantees."
What kind of service is that?

Customer feedback after AddSlashes:

Your customer service representative told me, \"We don\'t give any guarantees.\"
What kind of service is that?

Customer feedback after StripSlashes:

Your customer service representative told me, "We don't give any guarantees."
What kind of service is that?
```

Magic quotes was a feature of PHP, wherein strings are automatically escaped i.e. special characters are prefixed with a backslash-before being passed on. This feature was removed in PHP5.4 due to security concerns.