~~htmlentities()~~ <? echo htmlspecialchars('<jitu>'); ?>
~~html_entity_decode()~~

output: <jitu>

On browser it will output the same as: <jitu>
(but not mistranslated by the browser)

htmlspecialchars()

Html special characters:
htmlspecialchars_decode()

Translations performed are:

'&' (ampersand)	'&'
'"' (double quote)	'"'
''' (single quote)	'''
'<' (less than)	'<'
'>' (greater than)	'>'

```
</HTML> ji'tu"ka
<script>alert('kutte');</script>
```

a textarea in form

Our midas.inc.php uses addslashes() function for each form value when form posts.

So that when query executes it takes as:

db_query("update table1 set name='ji\'tu\"ka' where id='1'");

it may be correctly as:

db_query("update table1 set name='ji\'tu\"' where id='1'");

In it sql understands that we do not finish query but it is just a single or double quote among query not a stopper.

But in database it saves as it:

ji'tu"ka or </HTML> ji'tu"ka
<script>alert('kutte');</script>

when we display these, we must translate the special html character in it so that site can not be hacked that is mistranslated.

ie.

echo "

Use ms_form_value() ⇒ to show or display value in form element
ms_display_value() ⇒ to display database value out of form as it is ⇒ when show or display value in fck editor.

array_key_exists(mixed $key, array $search) (bool)

Checks if the given key or index exists in the array

Example:

```
<?php
 $search_array = array('first' => 1, 'second' => 4);
 if(array_key_exists('first', $search_array))
 {
    echo "The 'first' element is in the array";
 } ?>
```

Output:

The 'first' element is in the array

bool method_exists (object $object, string $method_name)

Checks if the class method exists in the ~~class~~ given object

Example: class.MyClass.php (this page name)

```
class MyClass
{
    function MyMethod()
    { echo "jitu"; }
}
class Yourclass
{ function MyTest() {
    $a = new MyClass();
    if(method_exists($a, 'mymethod'))
    {
        echo "Jitendra";
    } }
}
```

testing.php

```
<?php
    include_once('class.MyClass.php');
    $b = new Yourclass();
    $b -> MyTest();
?>
```

Output: When running testing.php, we get:

Jitendra

bool in_array(mixed $n, array $h)

Searches h for n and returns true if it is found in the array, false otherwise.

Example:

```
<?php   $os = array("Mac", "Jric", "Antic");
        if(in_array("Jric", $os)
        { echo "Jitendra"; }
?>
```

Output:

Jitendra

array array_map(callback $callback, array $arr)

It returns an array containing all the elements of arr after applying the callback function.

Example:

```
<?php  function cube($n) { return ($n * $n * $n); }
       $a = array(1, 2, 3, 4);
       $b = array_map("cube", $a);
       print_r($b);
?>
```

Output: Array ([0] => 1, [1] => 8, [2] => 27, [3] => 64)

mixed array_shift(array &$array)

It shifts the first value of the array off and returns it. If array is empty (or is not an array), NULL will be returned.

Example:

```
<?php
$stack = array("ora", "baba", "app", "rasp");
$fruit = array_shift($stack);
print_r($stack);
?>
```

Output:

```
Array
(
    [0] => baba
    [1] => app
    [2] => rasp
)
```