String Manipulation

Why manipulate strings?

In previous classes, we learned how to create and use strings.

Using strings and manipulating them is one of the essential skills you need as a programmer
(not just in PHP but in all languages).

For example, if a user fills out a form on your site, then you must verify and validate that data.

Example : converting letters to uppercase or lowercase, checking an email address to check if all parties are present,
checking the user's browser, reducing spaces from the text entered in a text box.

All these elements are the manipulation of strings.

For starters, let's see how to change the case of a string.

Change the case

Imagine that on your site you have a contact form with a text box for the subject of the message.
Most of the time, users will type something similar:

'request information on a product'

But when you receive an email from your site, you like to have the object in capital letters. Like this:

'REQUEST INFORMATION ON A PRODUCT'

Your job is to convert the entire string to uppercase.
For that, you will use the function: **strtoupper ().**

Example:

```
<? Php
   $ text = "request for information about a
                          product";
   $ text = strtoupper ($ text);
?>
```

To convert the entire string to lowercase, you will use the function: **strtolower ().**

Let's go back to our contact form. There is also a text box for the user to fill in his name.
Most of the time, users will type something similar:

'john doe'

Instead of this:

'John Doe'

Your job is to convert the first letters of each word to uppercase.

For that, you will use the function: **ucwords ().**

Example:

```
<? Php
   $ text = "john doe";
   $ text = ucwords ($ text);
?>
```

If you just want to put the first letter of a string in uppercase (for example, a sentence), you will use **ucfirst ()**

**/! \ Remember, you have to surround the variable or the text you want to modify with the parentheses of the function /! **

Counting the number of characters

We can count the number of characters of a string with the function: **strlen ().**

Example:

```
<? Php
   $ text = "example";
   $ count = strlen ($ text);

   echo $ count;
   // OUTPUT: 7
?>
```

Delete characters

Return to our contact form. There is also a text box for the user to enter his email address.

Sometimes the user enters his email address this way:

'address@gmail.com-'

We can see that he made a typo and typed his email address with a dash at the end.

To delete a character at the beginning or end of a string, use the function: **trim ().**

Example:

```
<? Php
   $ mail = "address@gmail.com-";
   $ mail_valid = trim ($ mail, '-');
?>
```

This function takes two parameters: the string on which to operate and the character to be deleted.

By default, if you do not specify the character to delete, this function will delete the SPACES.

You can enter several characters to delete.

Searching for a string

Often used more often, thefunction **strpos ()** allows you to search for one string in another.
Let's take the example of our email address.

We want to know the position of @ in the string:

```
<? Php
   $ mail = "address@gmail.com";
   $ pos_mail = strpos ($ mail, "@");
?>
```

Two arguments: the string on which to search and the string to search.

In this way, we know the position of the @ but we also check that the symbol exists. Indeed, if the character to search is not found (the function returns false), it is that the email address is not valid.

Replace a string

To replace a character or string, there is the function **str_replace ()**

The syntax is:

str_replace (search, replace, subject)

- 'search' is the string to search / replace.
- 'replace' is the string by which to replace 'search'
- 'subject' is the string on which to perform the operation

Example:

```
<? php

   echo str_replace ('world', 'simon', 'hello world');

   // OUPUT: Hello Simon;
?>
```

Converting a String to Array

In PHP, we can use a function to divide a string into segments and convert it into a table.

For that, we use the function: **explode ()**

Example:

```
<? Php
   $ kebab = "tomato salad onion carrot";
   $ ingredients = explode ("", $ kebab);

   / * OUTPUT:
   * array ([0] => "salad",
   * [1] => "tomato",
   * [2] => "onion",
   * [3] => "carrot");
   * /
?>
```

The function **explode ()** needs two arguments: the string that we want to convert and the separator used to separate each segment.

Convert an Array to a String

Unlike the function we just saw (explode), we can gather / merge the elements of an array into a string.
For this we use the function: **implode ()**

Example:

```
<? Php

   $ ingredients = array ([0] => "salad",
                          [1] => "tomato",
                          [2] => "onion",
                          [3] => "carrot");

   $ kebab = implode (",", $ ingredients);


    // OUPUT: "salad, tomato, onion, carrot";
?>
```

String functions

In PHP, and other programming languages, there are a lot of functions that you can use on strings.

We have just seen some functions that will be useful but it is only a small part.

If you want to learn more, browse the official documentation on php.net or search Google when you have a particular need on strings.