PHP Part-1

(These notes are the result of self-learning and W3c pages)

1. ## Intro:

   U have to use a server using >>> php -S localhost:4000
2. To view our site go to browser and type:
   localhost:4000/foldername/.php file name
   Remember the folder must be created in users>91935>folder name
3. **The "echo"** command is the same as printf in C and is used to print something.
   Examples:

```
 1.

 <html>
<head>
</head>
<body>
<?php
echo("Hello jaka");
 ?>
</body>
2.

</html>
<html>
<head>
</head>
<body>
<?php
echo "<h1>Hello jake</h1>"; //Inside the “” html code can be put
echo “<hr>”;
echo”<p1>Jake is sup</p1>”
 ?>
</body>
</html>
```

4. ## Variables

1. Declaration
$varname=initialization
Eg:

```
<html>
<head>
</head>
<body>
<?php
$Username="jake";
echo "<h1>Hello $Username</h1>";
 ?>
</body>
</html>
```

5. Data Types

1.String
$phrase=”Hello jake”
2.Floats
$flo=90.8989
3.Intergers
$inkk=30
4. Boolean(true or false)
$ismale=false
5.Null

6. Strings

1. strtolower()

Eg:

```
<html>
<head>
</head>
<body>
<?php
$Username="JAke";
echo strtolower($Username);
 ?>
</body>
</html>
```

2. strtoupper()

3.strlen()

Eg:
strlen($phrase)

4.Accessing string elements:

Eg:
```
echo $phrase[0];
```
Output:
Prints ‘J’ on the development server

Eg:
```
$phrase[0]=M
```
Output:
Prints MAke on page

4.**str_replace**(“Word in the tring to be replaced”,”Word that needs to replace”)

5. Substring -> substr($phrase,index number,length)

Eg;
```
<html>
<head>
</head>
<body>
<?php
$Username="JAke from Dhaka";
echo substr($Username,5,9);
 ?>
</body>
</html>
```

Output:
```
from Dhak
```

For more PHP String functions

6.Working with numbers

1. Echoing math ops:

    Eg:
    echo 9+5   output:14
    Note: $num++ is increment operator

2. Absolute value

    abs()
    Eg;
    $num=-223
    echo abs($num)

    Output:
    223

3. Power

    pow(number,power)

    Eg:
    pow(2,4) is the same as 2^4

4. Square root

    sqrt(144) gives 12

5. Comparing numbers:

    max(2,10) prints 10 (has more magnitude and same sign)

6. Ceil

    ceil(4.3) is the same as 5

    For more MATH functions

7.Taking user inputs

We will make use of HTML forms.
Syntax:
<form action=”Name of PHP file” method=”get”></form>
The method specifies what we want to do with the form (get means to get input).

Example of forms by W3c:

```
<!DOCTYPE html>
<html>
<body>

<h2>Text input fields</h2>

<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>

<p>Note that the form itself is not visible.</p>

<p>Also note that the default width of text input fields is 20
characters.</p>

</body>
</html>
```

W3c HTML forms
W3c PHP form handeling

Forms can be divided into 2 parts:

1. Space to enter info and submit
2. PHP file to retrieve info to the host

!!!!! 1. Space to enter info and submit button !!!!!

- Enclosed by <form></form>
  We have to specify the specific PHP where this form is intended to save the info entered. This is done using an “action” attribute inside the form tag. i.e.
  <form action=”site.php” method=”get”></form>
  The method for taking the info is “get” (post method can also be used).
- The user has to be given an idea about what has to be added in the box. This can be given using <label></label> tags.
  Example: if an email has to be taken then <label>email</label>
- An input tag is used to take input from the user.
  And the “type” attribute is used to specify the type of data intended to be taken.
  “type” attribute may have the following values:
  1. text-when we take a text as an input
  2. password: get user passwords
  3. radio-When we want a checklist as an input
  4. submit-submit button is made available. “value” attribute is used to specify what has to be written inside the submit button.
     <input type=”submit” value=”submit me”></input>

!!!!! 2. Info handling using PHP !!!!!

Mainly two PHP methods are used for this purpose.

1. $_GET
2. $_POST

   Eg:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo
$_POST["email"]; ?>
```

```
</body>
</html>
```

If the user enters his name and email then that would be echoed out.
Note: **Developers prefer POST for sending form data.**

8.GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls, and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class, or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

9.When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

10.When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover, POST supports advanced functionality such as support for multi-part binary input while uploading files to the server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

8. Building a basic calculator

Taking numbers and adding them

```
<html>
<head>
</head>
<body>
<form action="site.php" method="get">
<label>Enter first number</label><br>
<input type="number" name="num1">
<label>Enter the second number</label><br>
<input type="number" name="num2"><br>
<input type="submit" value="add them">
</form>
<!--retrieving the info entered using PHP-->
The sum of numbers is <?php echo $_GET["num1"]+$_GET["num2"]; ?>
</body>
</html>
```

9. Building a basic mad libs game:

```
<html>
<head>
</head>
<body>
<form action="site.php" method="get">
<label>Enter a color</label><br>
<input type="text" name="color"><br>
<label>Enter a plural</label><br>
<input type="text" name="plural"><br>
<label>Enter your favorite celebrity</label><br>
<input type="text" name="celebrity"><br>
<input type="submit" value="MAD LIB ME">
```

```
</form>
<?php
$color=$_GET["color"];
$plural=$_GET["plural"];
$celebrity=$_GET["celebrity"];
echo "Roses are $color <br>";
echo "$plural are blue <br>";
echo "I love $celebrity";

 ?>
</body>
</html>
```

9. PHP form validation
Data forms can be like:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
</form>
```

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a superglobal variable that returns the filename of the currently executing script. So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

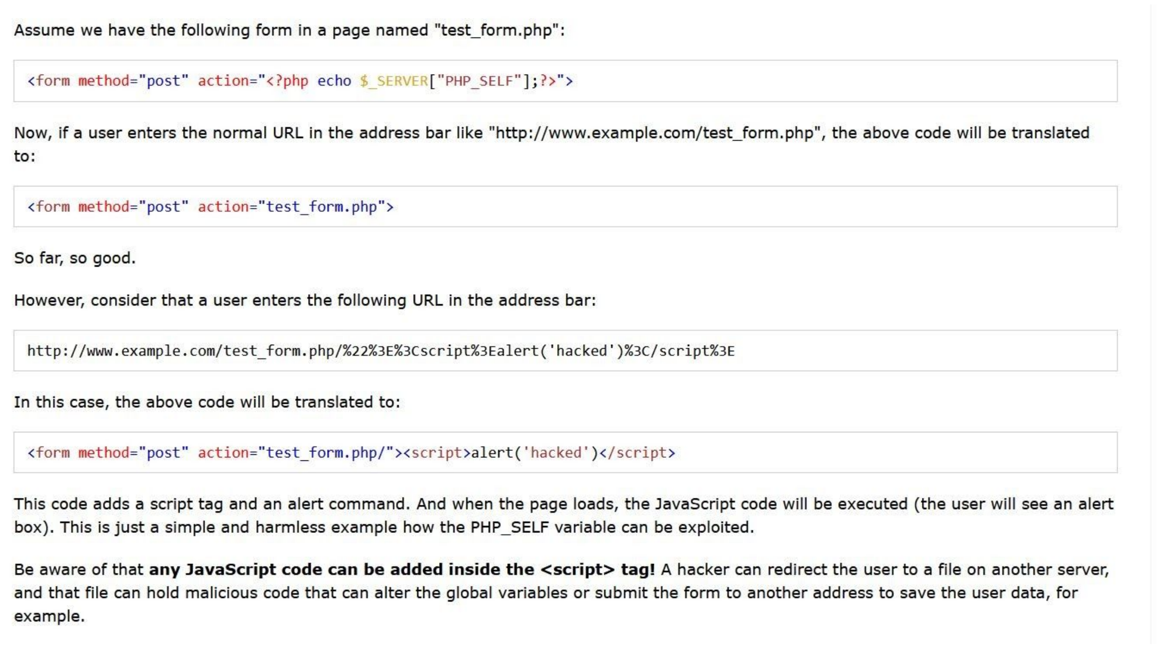
The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

11.Big Note on PHP Form Security

The $_SERVER["PHP_SELF"] variable can be used by hackers!

If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross-Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.



Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

So far, so good.

However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.

Be aware of that **any JavaScript code can be added inside the <script> tag!** A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

12.How To Avoid $_SERVER["PHP_SELF"] Exploits?

$_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post"
action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

Since, all the attached script has been converted to html chars i.e. < and > got converted to their HTML forms &lt and &gt, this hack is of no use.

The exploit attempt fails, and no harm is done!

An example of a program to take user data to avoid hack:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">


Name: <input type="text" name="name">

<span class="error">* <?php echo $nameErr;?></span>

<br><br>

E-mail:

<input type="text" name="email">

<span class="error">* <?php echo $emailErr;?></span>

<br><br>

Website:

<input type="text" name="website">

<span class="error"><?php echo $websiteErr;?></span>

<br><br>

Comment: <textarea name="comment" rows="5" cols="40"></textarea>

<br><br>

Gender:

<input type="radio" name="gender" value="female">Female

<input type="radio" name="gender" value="male">Male

<input type="radio" name="gender" value="other">Other

<span class="error">* <?php echo $genderErr;?></span>

<br><br>

<input type="submit" name="submit" value="Submit">


</form>
```

```php
<?php

// define variables and set to empty values

$nameErr = $emailErr = $genderErr = $websiteErr = "";

$name = $email = $gender = $comment = $website = "";


if ($_SERVER["REQUEST_METHOD"] == "POST") {

  if (empty($_POST["name"])) {

          $nameErr = "Name is required";

  } else {

          $name = test_input($_POST["name"]);

  }


  if (empty($_POST["email"])) {

          $emailErr = "Email is required";

  } else {

          $email = test_input($_POST["email"]);

  }


  if (empty($_POST["website"])) {

          $website = "";

  } else {

          $website = test_input($_POST["website"]);

  }


  if (empty($_POST["comment"])) {
          $comment = "";
  } else {
          $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
          $genderErr = "Gender is required";
  } else {
          $gender = test_input($_POST["gender"]);
  }
}
?>
```

Note: We have made use of the test_ input method in the if-else statements making sure, that if the user doesn’t enter a value, they are prompted to do so.

13.Validating emails and names:

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
          $nameErr = "Name is required";
  } else {
          $name = test_input($_POST["name"]);
```

```
        // check if name only contains letters and whitespace

        if (!preg_match("/^[a-zA-Z ]*$/",$name)) {

        $nameErr = "Only letters and white space allowed";

        }

  }


  if (empty($_POST["email"])) {

        $emailErr = "Email is required";

  } else {

        $email = test_input($_POST["email"]);

        // check if e-mail address is well-formed

        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {

        $emailErr = "Invalid email format";

        }

  }


  if (empty($_POST["website"])) {

        $website = "";

  } else {

        $website = test_input($_POST["website"]);

        // check if URL address syntax is valid (this regular expression also allows
dashes in the URL)

        if
(!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/
%=~_|]/i",$website)) {

        $websiteErr = "Invalid URL";

        }
```

```
 }

 if (empty($_POST["comment"])) {
        $comment = "";
 } else {
        $comment = test_input($_POST["comment"]);
 }

 if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
 } else {
        $gender = test_input($_POST["gender"]);
 }
}
?>
```

14. Arrays

Arrays are mutable i.e. we can change the values whenever we feel like.

eg:

```
<?php
$friends=array("Mike","Jason","Dwight","Muthu");
echo "$friends[2]"
 ?>
```

Output: Dwight

Since, arrays are mutable we can use,

$freinds[4]=” Kevin”;

15. More on checkboxes:

eg:

```
<html>

<head>

</head>

<body>

<form target="site.php" method="post">

<label>Apple</label>

<input type="checkbox" name="fruits[]" value="apple"}<br>

<label>Pears</label>

<input type="checkbox" name="fruits[]" value="pears"}<br>

<label>Banana</label>

<input type="checkbox" name="fruits[]" value="banana"}<br>

<input type="submit" value="submit">

</form>

<?php

$fruits = $_POST["fruits"];

echo $fruits[1]

 ?>

</body>

</html>
```