**PHP Practices**

**Outline:**

1. Continuing with JavaScript Form Validation
2. Tools required
3. Regular Expression
4. PHP Form Validation
5. Save data to Database
6. Challenge

# Continuing with JavaScript Form Validation

Have you completed the JavaScript code to verify and validate user input or to manipulate the style? Remember to separate HTML, CSS and JavaScript files.

A screenshot of a registration form

Description automatically generated

# Tools required

Text Editor for editing source code

Apache Web server

PHP Interpreter

Database server

# PHP Regular Expression

Regular Expressions are patterns used to match character combinations in strings or simply search pattern. Defining of regular expressions in JavaScript and PHP are similar but may not be fully compatible. You may check your regular expression using online application below

* <https://regexr.com/>
* <https://regex101.com/>
* <https://www.phpliveregex.com/>

Refer to the official documentation if you are using any web framework.

# PHP Form Validation

Form validation should occur at the server and client’s side. Previous form validation occurred at the server side where after the client had entered all necessary data and then pressed the Submit button. If the data that had been entered wrongly or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process and over burdening server.

Remember that JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. The form validation covers by JavaScript including

* **Basic Validation** - checking to make sure all required form fields is enter or no empty field. This would need just loop through each field in the form and check for data.
* **Data Format Validation** – checking user input data for correct format and value. This would need to put more logic to test correctness of data.

Now, you need to write PHP (server-side) code to do the same verification and validation as JavaScript (client-side). Following is an example code that you may refer to and later extended for the challenge.

1. Load following file in web browser.

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title> HTML form + PHP without validation</title>  </head>  <body>  <?php  $name = $email = $username = $password = ""; // define variables and set to empty values  if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  $name = $\_POST["fname"];  $email = $\_POST["femail"];  $username = $\_POST["fusername"];  $password = $\_POST["fpassword"];  }  ?>  <h2>PHP Form Validation Example</h2>  <form id="myForm" method="post" action="<?php echo $\_SERVER["PHP\_SELF"];?>">  Name: <input type="text" name="fname">  <br><br>  E-mail: <input type="text" name="femail">  <br><br>  Username: <input type="text" name="fusername">  <br><br>  Password: <input type="password" name="fpassword">  <br><br>  <input type="submit" name="btnSubmit" value="Register">  <input type="reset" value="Reset">  </form>  <?php  echo "<h2>Your Input:</h2>";  echo $name;  echo "<br>";  echo $email;  echo "<br>";  echo $username;  echo "<br>";  echo $password;  echo "<br>";  ?>  </body>  </html> |

1. Test by entering this line in any of the input field (in web browser).

|  |
| --- |
| <script>alert('hacked')</script>  or  <script>  var Name = "Unknown OS";  if (navigator.userAgent.indexOf("Win") != -1) alert("Window OS");  if (navigator.userAgent.indexOf("Mac") != -1) alert("Macintosh");  if (navigator.userAgent.indexOf("Linux") != -1) alert("Linus OS");  if (navigator.userAgent.indexOf("Android") != -1) alert("Android OS");  if (navigator.userAgent.indexOf("like Mac") != -1) alert("iOS");  </script> |

1. Go to the code and replace the <form> with this line.

|  |
| --- |
| <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>"> |

1. Re-test the code in web browser by entering the line in step 2. What do you notice?

Writing safe code is complicated where combination of defense techniques are necessary to prevent XSS. This example uses PHP htmlspecialchars() function to escape variables to tackle injection of JavaScript code snippets by converting special characters to HTML entities.

# Save data to Database

After user input data validation is completed, the now ‘clean’ data should be save into database. With PHP, this is easily achieve using

|  |
| --- |
| <?php  $databaseHost = ' '; // localhost – if using XAMPP  $databaseName = ‘ ’; // database name  $databaseUsername = ' '; // account username  $databasePassword = ' '; // account password  $mysqli = mysqli\_connect($databaseHost, $databaseUsername, $databasePassword, $databaseName);  mysqli\_query($mysqli, "INSERT INTO **TABLENAME**( list of column name = name attribute defined in HTML form) VALUES(list of PHP variables use to store user input data from HTML form)  or die("Could not execute the insert query.");  ?> |

But before this, you will need to

* Create a user account with username and password in XAMPP.
* Create database and table with corresponding column name that matches the name attribute defined in HTML form
* Link the user account with a specific database.

# Challenge

Now, you need to complete PHP code code to verify and validate user input data from HTML form. All the form elements will **automatically** be available to your PHP scripts. The form contains 14 input fields and two buttons (Register and Clear). When the user fills this form and click the ‘Register’ button, the “registration.php” file is executes.