**JOBSHEET 7**

**PHP Form Processing**



**Name**

Sherly Lutfi Azkiah Sulistyawati

**NIM**

2341720241

**Class**

2I

**Department**

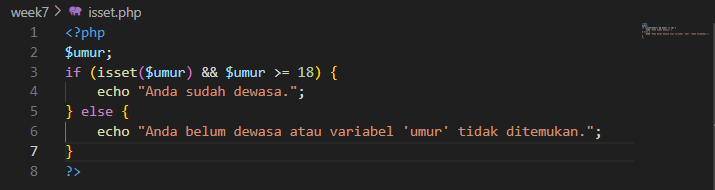
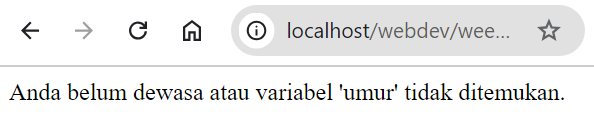
Information Technology

**Study Program**

D4 Informatics Engineering

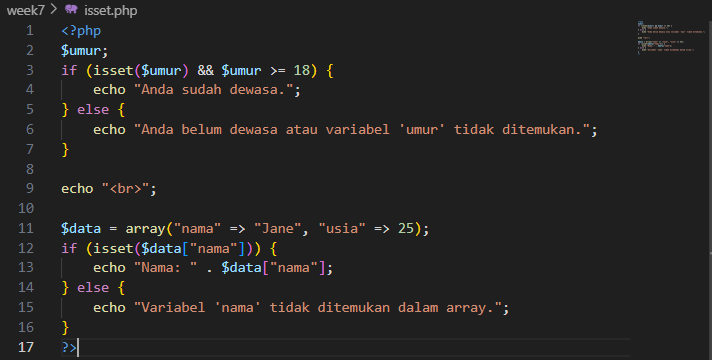
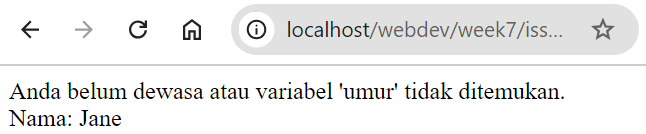
**Practical Section 1: Function isset()**

1. **What do you understand from using the isset on the file?**

 ****

The purpose of isset($umur) is to ensure that the variable $umur has been declared and holds a value other than null. If the variable is set, the code moves forward to the next condition, which checks if $umur is greater than or equal to 18. If the variable $umur is not set or holds no value, the else block will be executed, displaying the message: "Anda belum dewasa atau variabel 'umur' tidak ditemukan."

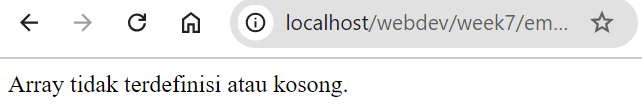
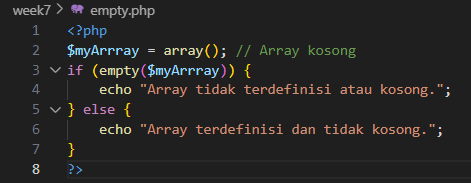
1. **Explain what you understand from the use of isset() in that file. Write your understanding below.**

**** ****

The isset() function in this code is used to check whether the key nama exists in the $data array and holds a value. If the key is present, the value associated with it is displayed ("Nama: Jane") using the echo statement. If the key nama is not found in the array, the code will display the message: "Variabel 'nama' tidak ditemukan dalam array."

**Practical Section 2: Function empty()**

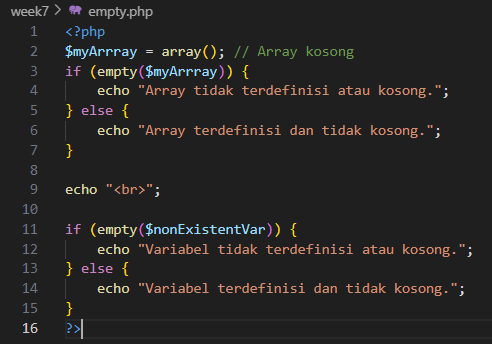
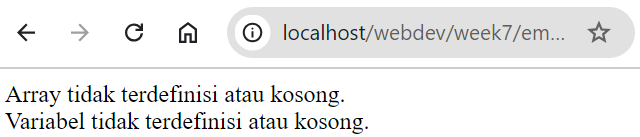
1. **What do you understand from the use of empty on the file? Write your understanding below.**

****

empty() checks if the array is either undefined or contains no elements. In this case, empty($myArrray) checks whether the variable $myArrray is empty or undefined.

Since $myArrray is declared as an empty array (array()), it is considered "empty." As a result, the message "Array tidak terdefinisi atau kosong." is displayed.

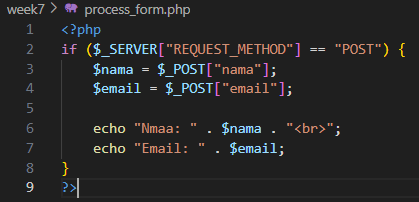
1. **Explain what you understand from the use of empty() in that file. Write your understanding below.**

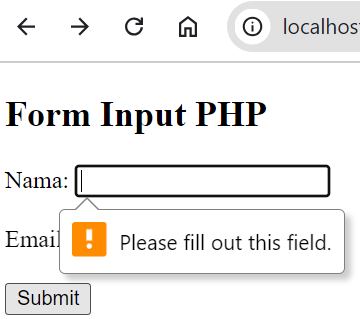
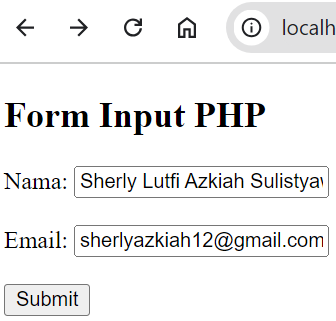
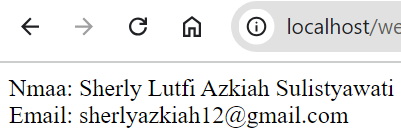
**** ****

empty($nonExistentVar) checks if $nonExistentVar is either not defined or holds an empty value. Since $nonExistentVar is not declared, empty() evaluates it as "empty." The message "Variabel tidak terdefinisi atau kosong." will be displayed. In both cases, the empty() function checks whether a variable is either not set or contains an empty (falsy) value. This function allows the code to handle variables that are missing or have no meaningful content without triggering warnings.

**Practical Section 3: PHP Input Form**

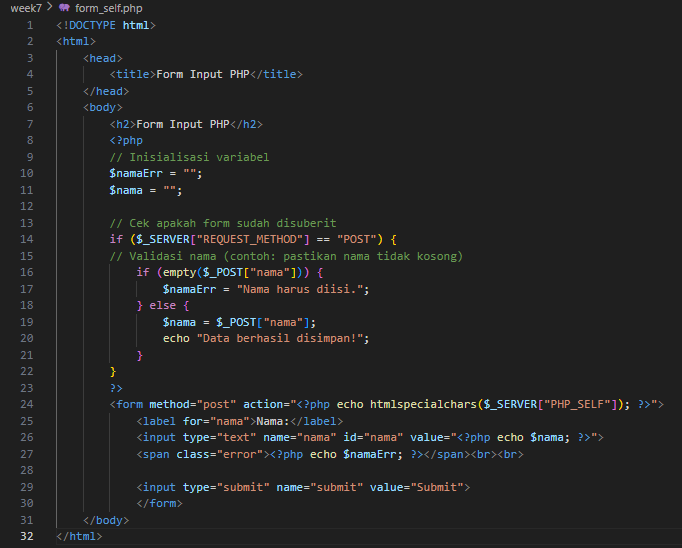
1. **Explain what happened and write your understanding below.**

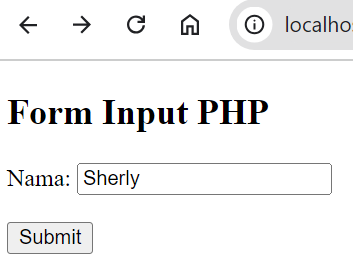
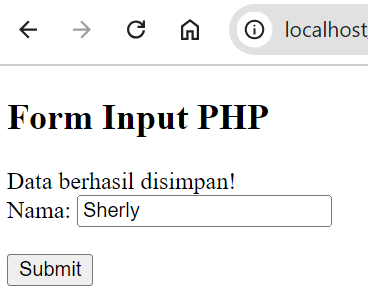
 ****

** **

form.html displays a form with fields for name and email. When the form is submitted, it sends the data to proses\_form.php, which processes the POST request and displays the submitted name and email.

1. **What do you understand from the use of forms in the file? Write your understanding below.**

****

**** ****

* PHP Initialization:

Variables like $namaErr and $nama are initialized to store error messages and the user's input, respectively.

* Form Submission Handling:
* The if ($\_SERVER["REQUEST\_METHOD"] == "POST") condition checks whether the form has been submitted using the POST method. This is crucial for handling form submissions safely.
* If the form is submitted, the script validates the input:

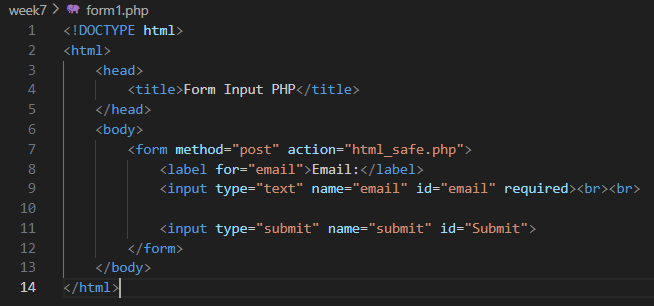
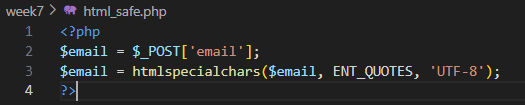
Validation: It checks if the nama field is empty. If it is, an error message is assigned to $namaErr. Otherwise, the input is captured in the $nama variable, and a success message ("Data berhasil disimpan!") is displayed.

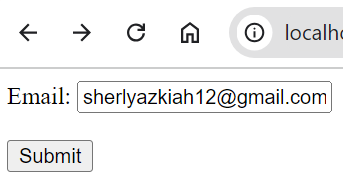
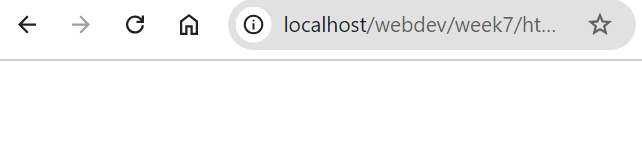
* Form Display:
* The <form> tag specifies the method (POST) and the action, which is set to <?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>. This means the form will submit data to the same page. Using htmlspecialchars() ensures that any special characters in the URL are converted to HTML entities, preventing XSS (Cross-Site Scripting) attacks.
* The <input> field for nama retains its value after submission by using value="<?php echo $nama; ?>". This provides a better user experience by allowing users to see what they entered after submission.
* Error messages are displayed next to the input field using a <span> tag with the class error.
* Submit Button:

The form includes a submit button that, when clicked, sends the input data to the server for processing.

**Practical Section 4: HTML Injection**

1. **Record here what you observed, give your explanation.**

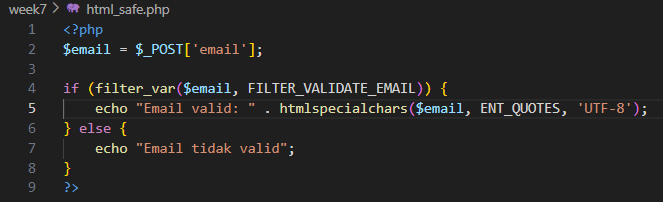
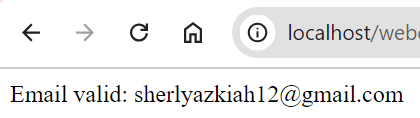
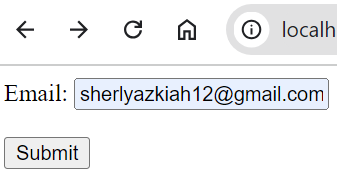
**** ****

**** ****

When I filled the form with an email address that contained HTML tags (e.g., <script>alert('XSS');</script>), the data is sent to html\_safe.php. Since htmlspecialchars() is used, the output would display the tags as plain text rather than executing them.

If you remove the htmlspecialchars() line and enter HTML tags in the input field, the browser will interpret and execute the HTML code. For example, entering <script>alert('XSS');</script> will cause the alert box to pop up in the browser, as the browser executes the script as part of the HTML.

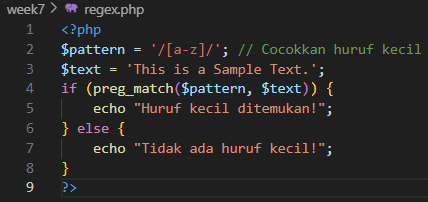
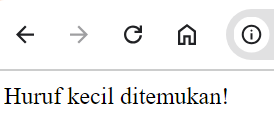
1. **Note here what you observe from the addition of the program code above.**

* Input Validation: The addition of the email input allows users to enter their email addresses. Using filter\_var() ensures that only valid email formats are accepted.
* User Feedback: If the email is valid, the system provides confirmation ("Email valid: ..."). If the email is invalid or empty, the user is informed with appropriate messages.
* Security Measures: By using htmlspecialchars() when displaying the valid email, the program prevents any potential XSS vulnerabilities by ensuring any special characters are encoded, making the output safe for display.

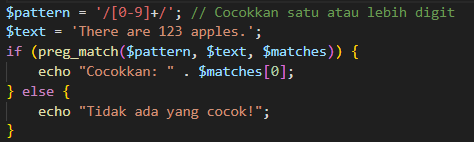
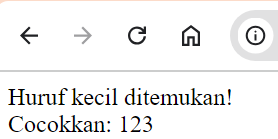
**Practical Section 5: The Use of Regex in PHP**

1. **Note here what you observe from the addition of the program code above.**

**** ****

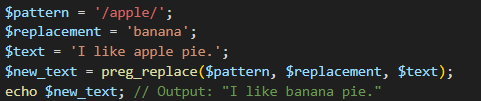
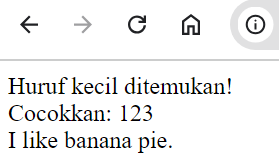
* The code uses a regular expression to check for the presence of any lowercase letters (from 'a' to 'z') in the variable $text, which contains the string "This is a Sample Text."
* Since the string includes lowercase letters ("is", "a", "ample", and "ext"), the preg\_match() function returns true, and the message "Huruf kecil ditemukan!" (which translates to "Lowercase letters found!") is displayed.
* If there were no lowercase letters in the string, the output would have been "Tidak ada huruf kecil!" (meaning "No lowercase letters!").

1. **Note here what you observe from the addition of the program code above.**

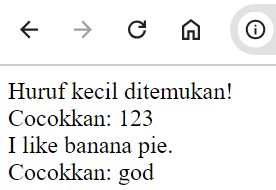
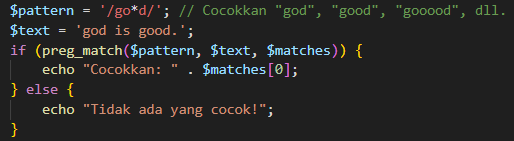
**** 

* The second part of the code checks for one or more digits in the string "There are 123 apples."
* Since the string contains the digits "123," the preg\_match() function finds a match and returns it in the $matches array. The output will show "Cocokkan: 123".

1. **Note here what you observe from the addition of the program code above.**

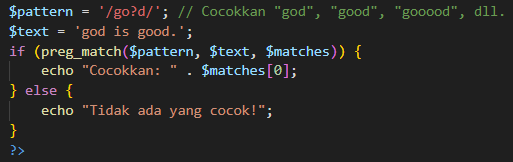
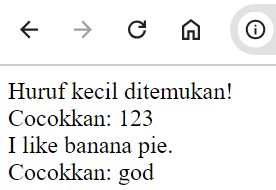
**** ****

The third part of the code uses preg\_replace() to replace the word "apple" with "banana" in the string "I like apple pie." The resulting string is "I like banana pie.", which is then displayed.

1. **Note here what you observe from the addition of the program code above.** 

The fourth part checks for the pattern /go\*d/, which matches "god", "good", "gooood", etc. In the string "god is good.", it finds "god" and displays "Cocokkan: god".

1. **Note here what you observe from the addition of the program code above.**

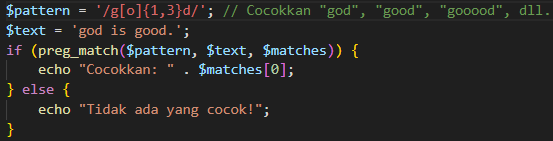
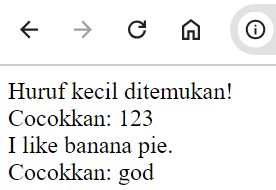
****

The change from \* to ? in the regex pattern modifies the matching behavior:

* go\*d (with \*) matches "god", "good", "gooood", etc., where o can appear zero or more times.
* go?d (with ?) matches "god" and "good", but it does not match "gooood" because the ? indicates that the preceding character (o) may appear zero or one time only.

Since the string "god is good." contains "god" and "good", both can be matched, but "gooood" will not be matched, and the output will still display "Cocokkan: god" since it was the first match found.

1. **Note here what you observe from the addition of the program code above.**

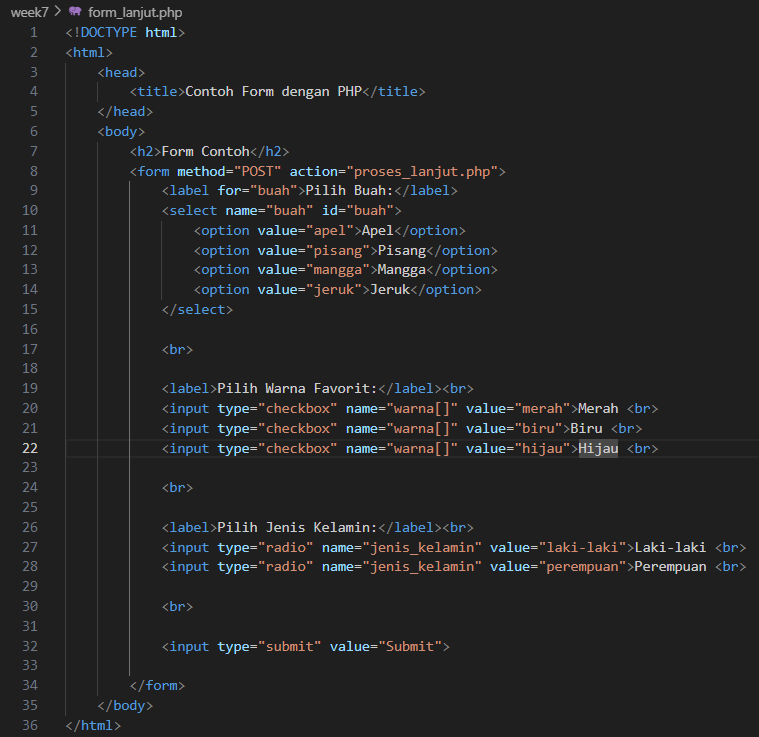
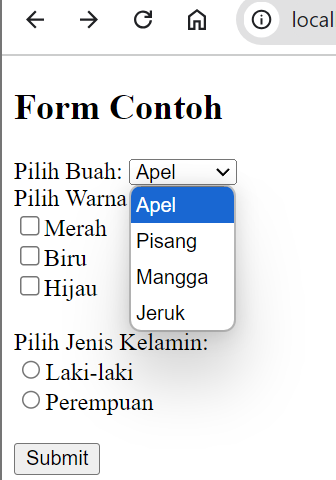
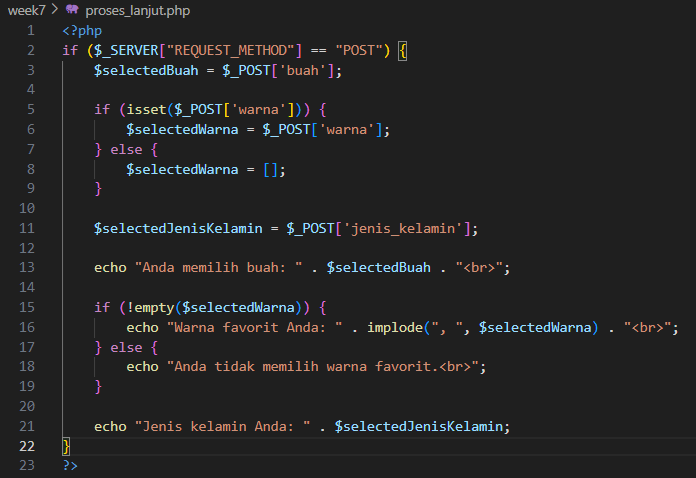
****The new pattern /g[o]{1,3}d/ checks for:

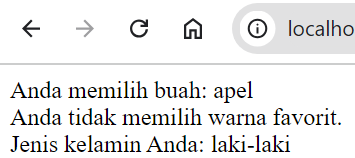
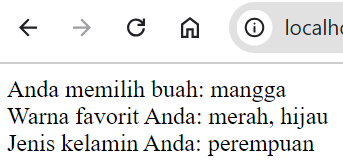
* The letter g
* Followed by one to three occurrences of the letter o
* Followed by the letter d

This means it will match "god", "good", and "gooood", but will also match "goood", "gooodd", and so forth, as long as the number of o is between 1 and 3.

**Practical Section 6: Advanced Form**

1. **Note here what you observe from the program code above.**

** **

****

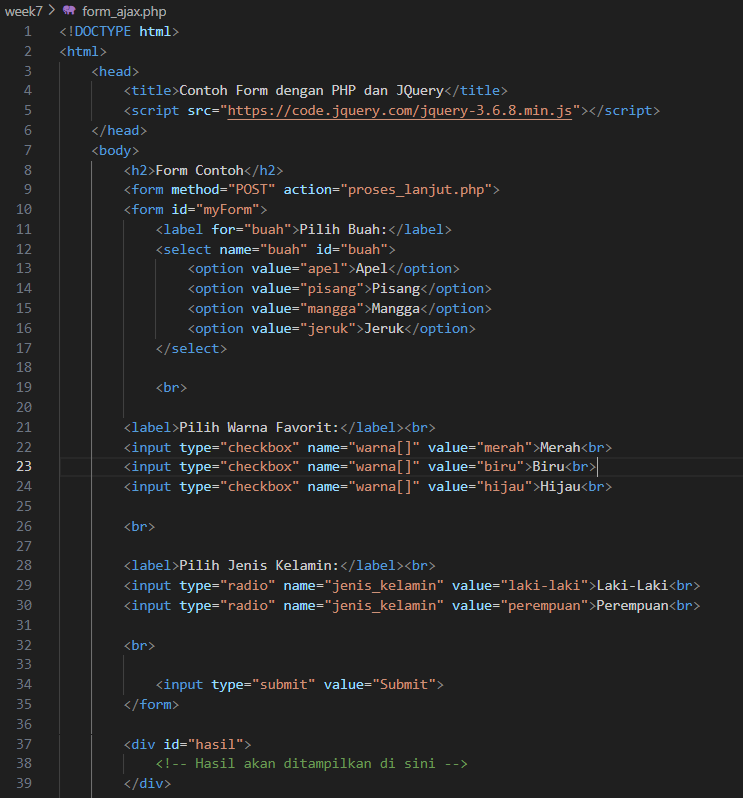
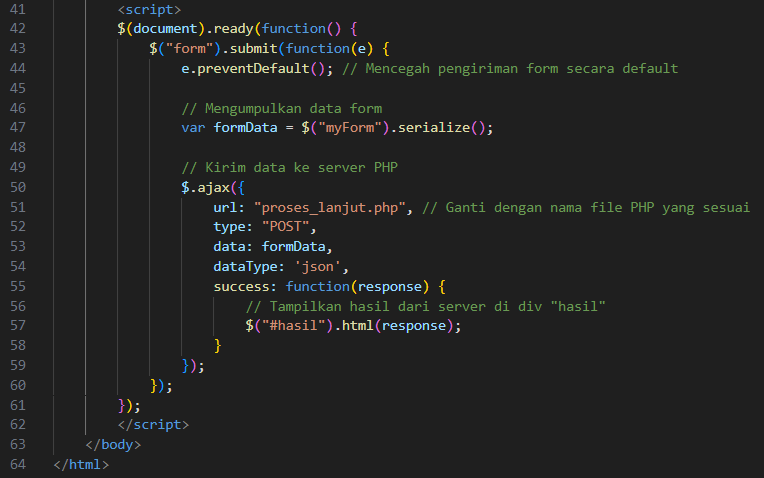
Explanation of HTML Form:

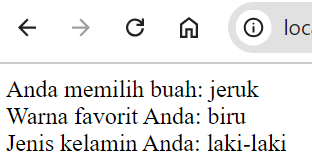
* Form Element:  
  The form uses the POST method to send data to proses\_lanjut.php for processing.
* Dropdown List:  
  The <select> tag contains four options for selecting a fruit (apel, pisang, mangga, jeruk).
* Checkboxes for Colors:  
  The user can select one or more favorite colors using checkboxes. The name="warna[]" attribute indicates that this is an array, allowing multiple selections.
* Radio Buttons for Gender:  
  The user can only select one gender (male or female) because radio buttons are mutually exclusive.
* Submit Button:  
  When the user clicks the "Submit" button, the form data is sent to proses\_lanjut.php for processing.

Explanation of PHP Script:

* Checking Form Submission:
  + The if ($\_SERVER["REQUEST\_METHOD"] == "POST") condition checks if the form was submitted using the POST method. This ensures the script only processes data when the form is submitted.
* Getting the Selected Fruit:
  + The line $selectedBuah = $\_POST['buah']; retrieves the selected fruit from the form using the POST method and stores it in the $selectedBuah variable.
* Checking and Getting the Selected Colors:
  + The code checks whether any colors were selected by using isset($\_POST['warna']).
  + If the warna[] checkboxes were selected, the selected values (an array) are assigned to $selectedWarna. If no colors were selected, $selectedWarna is set to an empty array ([]).
* Getting the Selected Gender:
  + The line $selectedJenisKelamin = $\_POST['jenis\_kelamin']; retrieves the selected gender value and stores it in the $selectedJenisKelamin variable.
* Displaying the Selected Data:
  + Selected Fruit:  
    The script displays the selected fruit using echo "Anda memilih buah: " . $selectedBuah . "<br>";.
  + Selected Colors:  
    If the user selected colors, the script uses implode(", ", $selectedWarna) to combine the selected color values into a string and display them. If no colors were selected, it displays a message saying, "Anda tidak memilih warna favorit."
  + Selected Gender:  
    The selected gender is displayed using echo "Jenis kelamin Anda: " . $selectedJenisKelamin;.

1. **Note here what you observe from the addition of the program code above.**

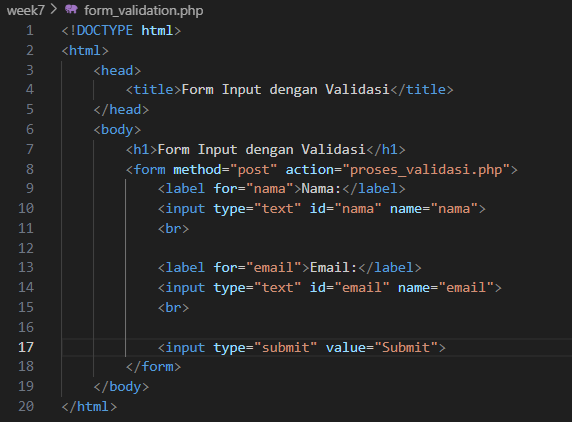
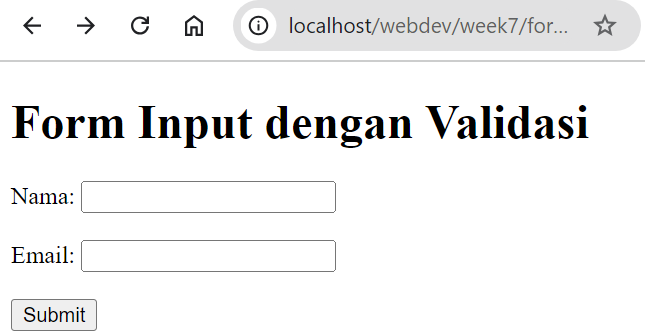
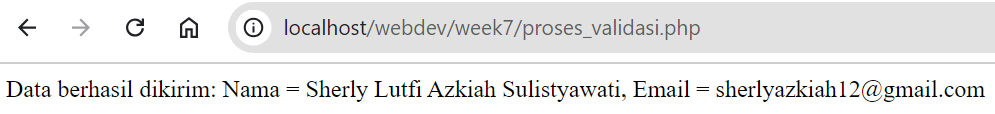
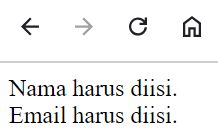
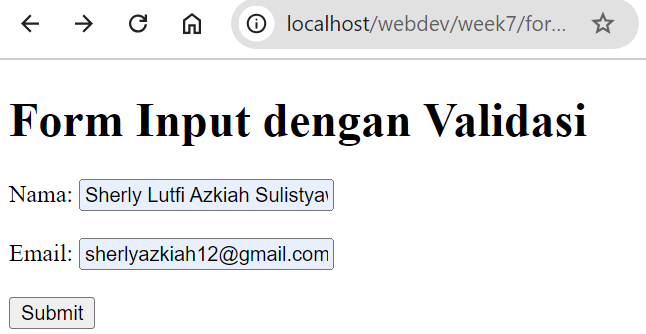
**** ****

**** ****

* Before AJAX: Normally, submitting a form would cause the page to reload, sending the data and then displaying the result.
* With AJAX: When you submit the form using this AJAX script, the page does not reload. Instead, the data is sent asynchronously to the server. The server processes the form data and sends back a response, which is dynamically displayed on the page in the #hasil div without a full page refresh.

**Practical Section 7: Form Validation**

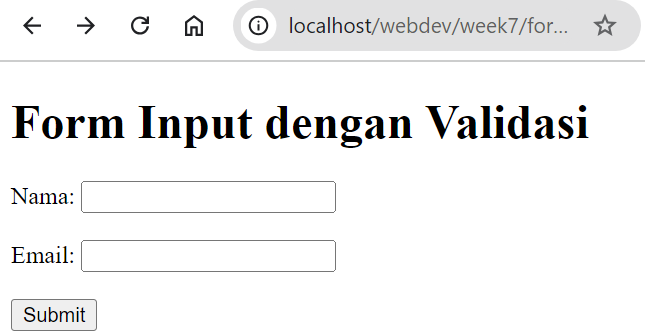
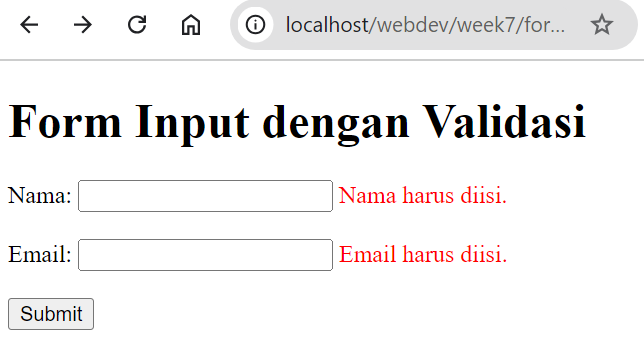
1. **Note here what you observe from the addition of the program code above.**

**  **

* The code above demonstrates a basic form validation process in PHP. It works by collecting user input for name and email through an HTML form. Once the form is submitted, the \*proses\_validasi.php\* script is executed, retrieving the submitted values for name and email. The script then uses an array called `$errors` to store any validation errors that may occur.
* To validate the input, the `empty()` function checks if the name or email fields are left blank. If either field is empty, an error message is added to the `$errors` array. Additionally, the `filter\_var()` function validates the email format, and if the email is invalid, another error message is stored in the array.
* If there are any errors, they are displayed to the user. However, if the input passes validation without errors, the script proceeds with further processing, such as saving the data to a database or sending an email, as suggested by the comment "Lanjutkan dengan pemrosesan data setelah validasi berhasil."
* This validation process ensures that users enter valid data before submitting the form, improving the user experience and preventing potential issues.

1. **Note here what you observe from the addition of the program code above.**

****

** **

jQuery Library Inclusion:

* The line <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script> includes the jQuery library, which is used for simplifying JavaScript code, especially for tasks like handling form submission and client-side validation.

Form Structure:

* The form remains mostly the same, with inputs for "Nama" and "Email". However, there are additional <span> tags (<span id="nama-error" style="color: red;"></span>) to display error messages next to the input fields.

Client-Side Validation with jQuery:

* When the form is submitted, the submit event is triggered: $("#myForm").submit(function(event) {...}).
* It checks if both the name and email fields are filled in. If either is empty, an error message is shown in the corresponding <span> tag, and the form is prevented from being submitted (event.preventDefault()).
* If all fields are valid, no errors are shown, and the form is submitted to proses\_validasi.php for server-side processing.

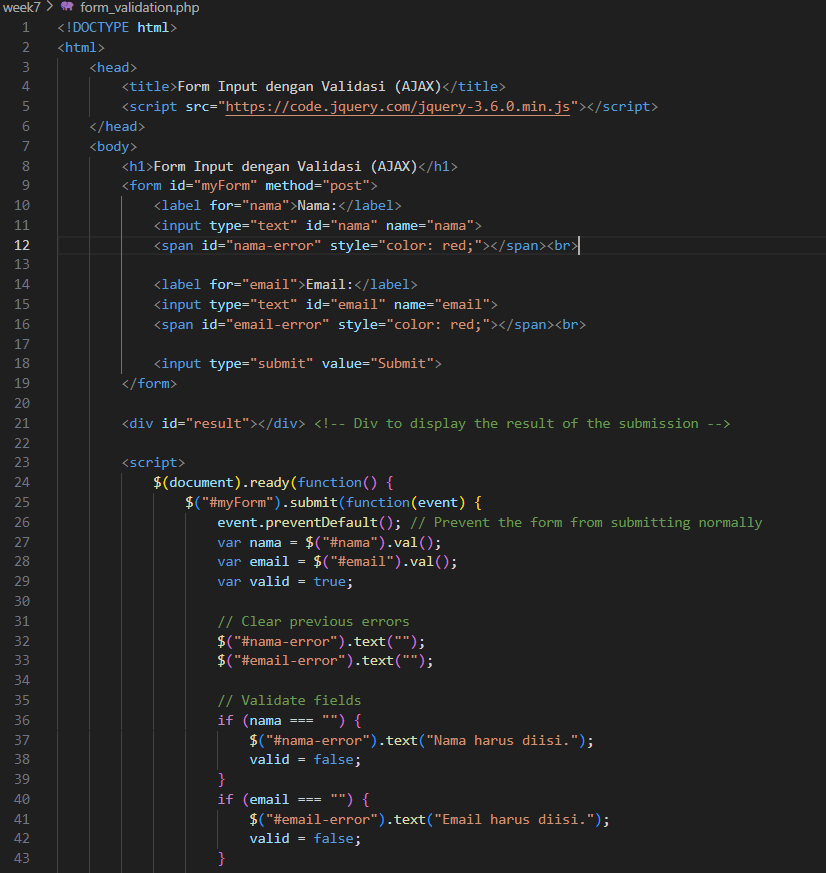
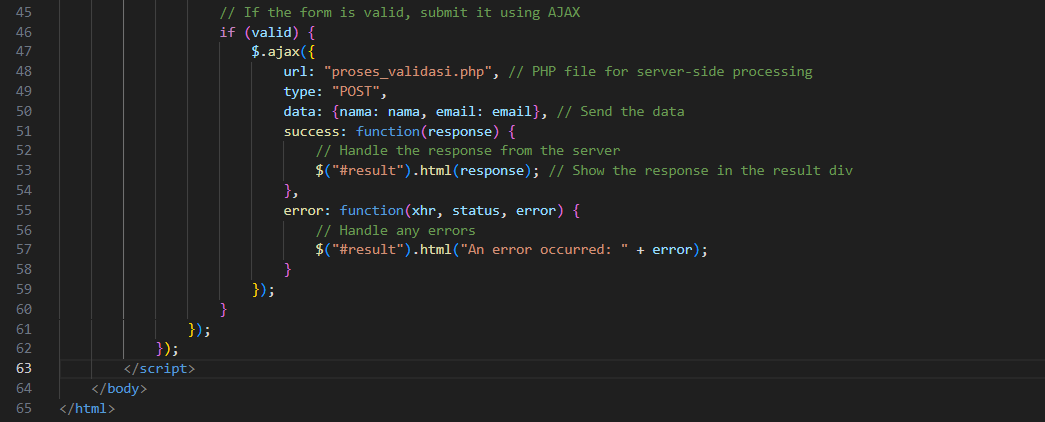
Error Display:

* If the name is empty, the error "Nama harus diisi." will be shown in the <span id="nama-error">.
* Similarly, if the email is empty, the error "Email harus diisi." will be shown in the <span id="email-error">.

Form Submission:

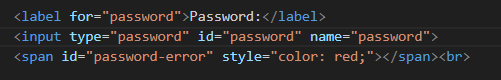
* If there are validation errors (i.e., if valid = false), the form submission is canceled. Otherwise, the form is submitted to the server as usual.

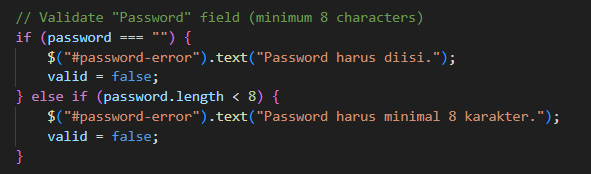
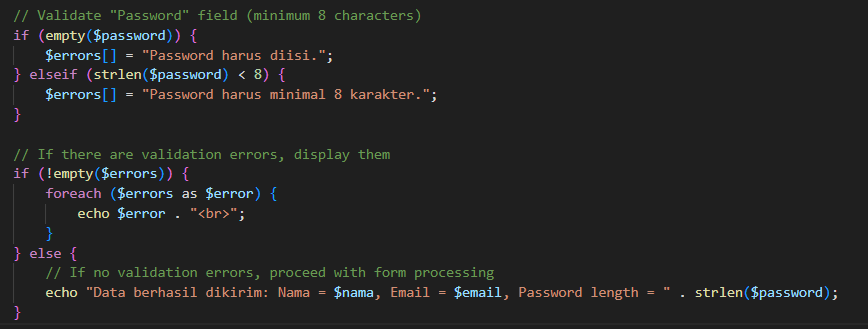
1. **Create a script for step 6 using ajax. Screen shoot the code and wrote here what you observe from the addition of the program code.**

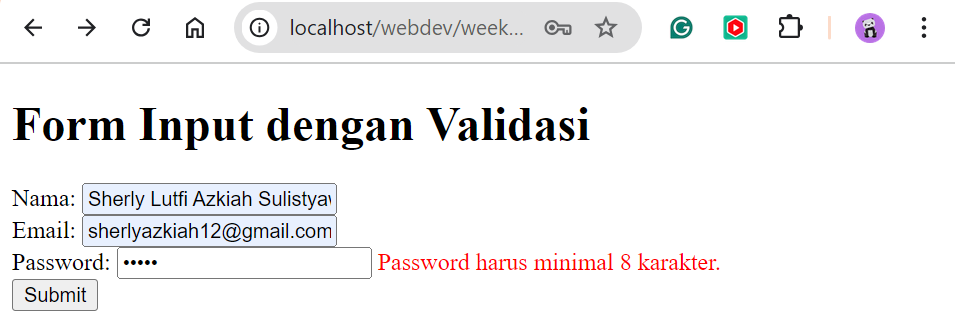
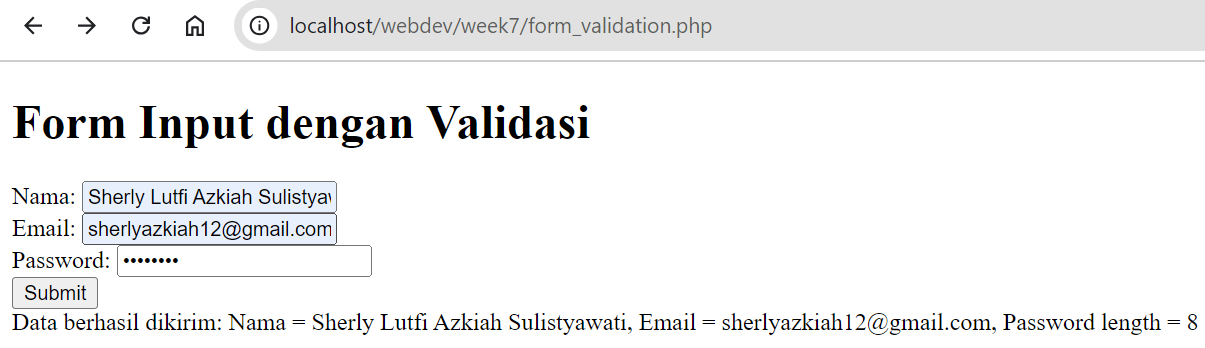
**** **** 

* The code uses jQuery's $.ajax() method to send the form data to proses\_validasi.php asynchronously.
* The preventDefault() method is used to prevent the default form submission behavior (reloading the page).
* The response from the server is displayed in the #result div.

1. **Add code for password validation with a minimum of 8 characters using jQuery and PHP. Screen shoot the code and note here what you observe from the addition of the program code**

****

**** ****

**** 

* The isValidPassword() function in JavaScript checks if the password is at least 8 characters long.
* The PHP code also validates the password length on the server-side.
* This combined validation provides better security and ensures that users enter strong passwords.

**GitHub Link:** [**https://github.com/sherlyazkiah/Web-Development/tree/main/week7**](https://github.com/sherlyazkiah/Web-Development/tree/main/week7)