



Name : Evan Diantha Fafian  
Class : SIB 2G  
Absent : 09  
NIM : 2341760163

### Topics

1. Concept of *arrays* in PHP programming
2. Concept of *functionss* in PHP programming

### Objective

Students are expected to:

1. Understand the concept of arrays in PHP programming
2. Understand the concept of functions in PHP programming

## INTRODUCTION

### Introduction to Arrays and Functions

Arrays and functions are fundamental concepts in programming that help organize and optimize code.

#### ➤ Arrays

Array, or a list, is one of the data types. An array is not a basic data type like an integer or a boolean, but rather a data type that consists of a collection of other data types. Arrays make it easier to group data, save on writing, and make variable usage more efficient. An array is a data structure that holds a collection of elements, *usually* of the same data type, under a single variable. Arrays simplify the handling of large sets of data, allowing multiple values to be stored and accessed easily using an index or key. In PHP, arrays are classified into three types:

- Indexed Arrays: Arrays with numeric indexes.
- Associative Arrays: Arrays with named keys that assign a value to each key.
- Multidimensional Arrays: Arrays that contain other arrays as their elements.

#### ➤ Functions

A function is a block of code designed to perform a specific task, which can be reused multiple times throughout the program. Functions reduce redundancy, make code more readable, and increase efficiency. By breaking down repetitive tasks like database querying or mathematical calculations into functions, the program becomes modular, and maintenance becomes easier.

### Practical Section 1. Indexed Array

A PHP indexed array is an array where the elements are stored with numeric indexes, starting from 0 by default. Each element in the array is associated with an index number, which is used to access or reference that element.

```
<?php
// Creating an indexed array
$variable = array("Value0", "Value1", "Value2", "Value3");

// Accessing the elements of the indexed array
echo $variable[0]; // Outputs: Value0
echo $variable[1]; // Outputs: Value1

// Adding a new element
$variable[] = "Value4"; // Adds Value4 to the array
?>
```

It can also be written as follows:

```
<?php
// Creating an indexed array
$variable[0] = "Value0";
$variable[1] = "Value1";
$variable[2] = "Value2";
$variable[3] = "Value3";

// Accessing the elements of the indexed array
echo $variable[0]; // Outputs: Value0
echo $variable[1]; // Outputs: Value1

// Adding a new element
$variable[] = "Value4"; // Adds Value4 to the array
?>
```

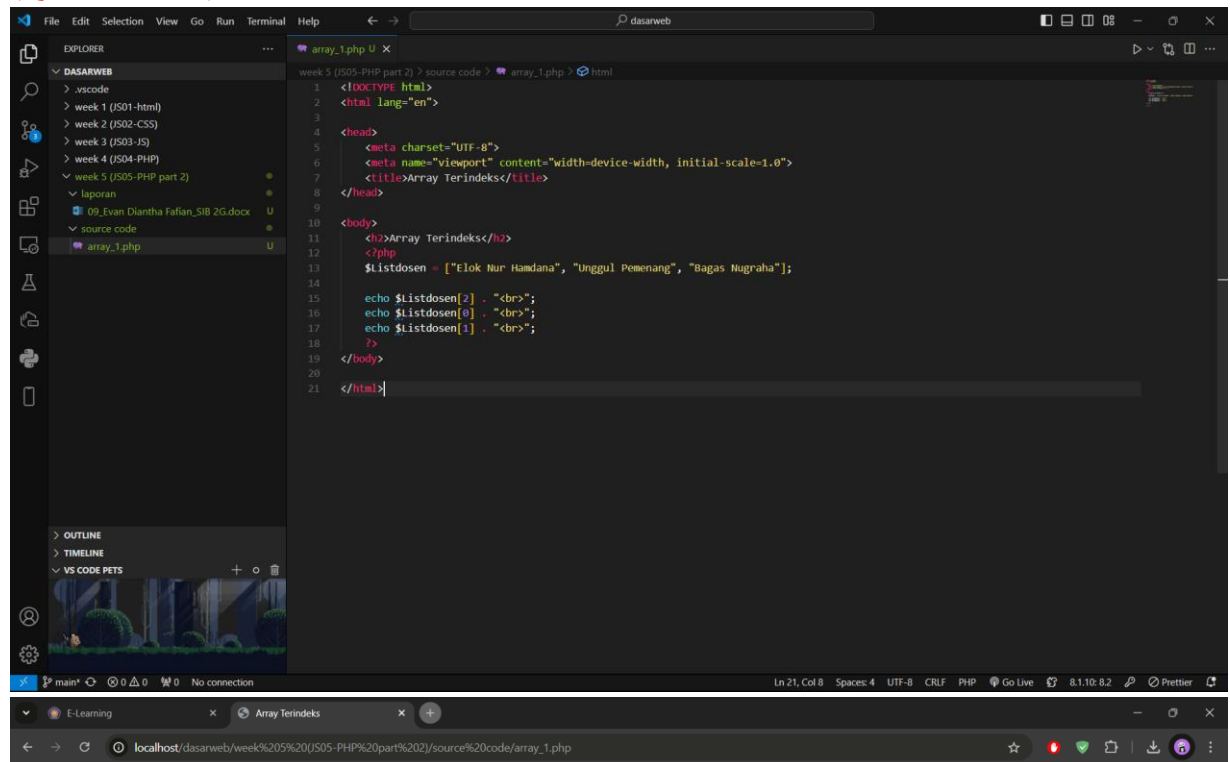
In this form, the array is created by directly assigning values to specific indexes without using the `array()` function. PHP automatically assigns the next available index if a new element is added without specifying an index, as shown when adding "Value4".

Follow these steps to understand indexed arrays in PHP:

Step	Description
1	<p>Create a new file named <code>array_1.php</code> inside the <code>JS05_PHP-2</code> directory, then type the following code:</p> <pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;/head&gt; &lt;body&gt; &lt;h2&gt;Array Terindeks&lt;/h2&gt; &lt;?php     \$Listdosen=["Elok Nur Hamdana","Unggul Pamenang", "Bagas Nugraha"];      echo \$Listdosen[2] . "&lt;br&gt;";     echo \$Listdosen[0] . "&lt;br&gt;";     echo \$Listdosen[1] . "&lt;br&gt;";  ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>
2	<p>Save the file and run your program in the browser. Type this link to your browser  <a href="localhost/dasarWeb/JS05_PHP-2/array_1.php">localhost/dasarWeb/JS05_PHP-2/array_1.php</a></p>
3	<p>Observe the output displayed</p>

To display an array, in addition to using indexes, we can also use loops. Try displaying the output of the program above using a *loop*.

(Question No.1)



The screenshot shows a development environment with VS Code and a web browser. In VS Code, the file `array_1.php` is open, showing the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Array Terindeks</title>
8 </head>
9
10 <body>
11   <h2>Array Terindeks</h2>
12   <pre>
13     $listdosen = ["Elok Nur Hamdana", "Unggul Pemenang", "Bagas Nugraha"];
14
15     echo $listdosen[2] . "<br>";
16     echo $listdosen[0] . "<br>";
17     echo $listdosen[1] . "<br>";
18   </pre>
19 </body>
20
21 </html>
```

The web browser shows the output of the script, titled "Array Terindeks", displaying the names on separate lines:

```
Bagas Nugraha
Elok Nur Hamdana
Unggul Pemenang
```

- This will display each name in the \$Listdosen array.

## Practical Section 2. Associative Array

A PHP associative array is an array where the keys are not numeric but instead are strings, allowing you to associate specific values with meaningful keys. This makes it easier to access and manipulate data based on the key name rather than a numerical index.

The components of an associative array consist of key-value pairs. The key indicates the position where the value is stored. PHP uses the arrow symbol ( $\Rightarrow$ ) to assign values to keys. Here is the syntax for writing an associative array:

```
<?php
// Creating an indexed array
$variable = array(
    'Key0' => 'value0',
    'Key1' => 'value1',
    'Key2' => 'value2',
    'Key3' => 'value3'
);

// Accessing the elements of the indexed array
echo $variable['Key0']; // Outputs: Value0
echo $variable['Key1']; // Outputs: Value1

// Adding a new element
$variable['Key4'] = "Value4"; // Adds Value4 to the array
?>
```

It can also be written as follows:

```
<?php
// Creating an indexed array
$variable['Key0'] = 'value0';
$variable['Key1'] = 'value1';
$variable['Key2'] = 'value2';
$variable['Key3'] = 'value3';

// Accessing the elements of the indexed array
echo $variable['Key0']; // Outputs: Value0
echo $variable['Key1']; // Outputs: Value1

// Adding a new element
$variable['Key4'] = "Value4"; // Adds Value4 to the array
?>
```

Follow these steps to understand associative arrays in PHP:

Step	Description
1	Create a new file named <b>array_2.php</b> inside the JS05_PHP-2 directory, then type the following code:

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title></title>
</head>
<body>
  <?php
    $Dosen = [
      'nama' => 'Elok Nur Hamdana',
      'domisili' => 'Malang',
      'jenis_kelamin' => 'Perempuan' ];

    echo "Nama : {$Dosen ['nama']} <br>";
    echo "Domisili : {$Dosen ['domisili']} <br>";
    echo "Jenis Kelamin : {$Dosen ['jenis_kelamin']} <br>";

  ?>
</body>
</html>

```

2

Save the file and run your program in the browser. Type this link to your browser  
[localhost/dasarWeb/JS05\\_PHP-2/array\\_2.php](localhost/dasarWeb/JS05_PHP-2/array_2.php)

Observe the output displayed. Then, add table styling to the output to make it more visually appealing. Here's the updated code with a simple table style.

(Question No.2)

\*For styling, you can choose either internal or external CSS.

3

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4
5  <head>
6    <meta charset="UTF-8">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <style>
9      th
10         width: 20%;
11         margin: 20px auto;
12    </style>
13    <title></title>
14  </head>
15
16  <body>
17    <table border="1">
18      <tr>
19        <th>Nama</th>
20        <th>Domisili</th>
21        <th>Jenis Kelamin</th>
22      </tr>
23      <?php
24        $Dosen = [
25          'nama' => 'Elok Nur Hamdana',
26          'domisili' => 'Malang',
27          'jenis_kelamin' => 'Perempuan'
28        ];
29
30        echo "<tr>";
31        echo "<td>{$Dosen['nama']}</td>";
32        echo "<td>{$Dosen['domisili']}</td>";
33        echo "<td>{$Dosen['jenis_kelamin']}</td>";
34        echo "</tr>";
35      ?>
36    </table>
37  </body>
38
39  </html>

```

Nama	Domisili	Jenis Kelamin
Elok Nur Hamdana	Malang	Perempuan

### Practical Section 3. Multidimensional Array

A multidimensional array in PHP is an array that contains one or more arrays as its elements. This means that each element in the array can be an array itself, allowing you to store data in a grid or matrix-like structure. Multidimensional arrays are often used to represent complex data structures like tables, matrices, or databases.

Example of a Multidimensional Array in PHP:

```
<?php
// Creating an multidimensional array
$variable = array(
    array('value00', 'value01', 'value02'),
    array('value10', 'value11', 'value12'),
    array('value20', 'value21', 'value22')
);

// Accessing the elements
echo $variable[0][0]; // Outputs: value00
echo $variable[1][1]; // Outputs: value11
echo $variable[2][2]; // Outputs: value22
?>
```

Or it can also be written as follows:

```

<?php
// Creating a multidimensional array
$students = array(
    array("name" => "John", "age" => 20, "grade" => "A"),
    array("name" => "Sarah", "age" => 19, "grade" => "B"),
    array("name" => "Mike", "age" => 21, "grade" => "A")
);

// Accessing elements of the multidimensional array
echo $students[0]["name"]; // Outputs: John
echo $students[1]["name"]; // Outputs: Sarah
echo $students[2]["name"]; // Outputs: Mike
echo $students[0]["age"]; // Outputs: 20
echo $students[1]["age"]; // Outputs: 19
echo $students[2]["age"]; // Outputs: 21
?>

```

Follow these steps to understand multidimensional arrays in PHP:

Step	Description
1	<p>Create a new file named <b>style.css</b> inside the JS05_PHP-2 directory, then type the following code:</p> <pre> 1  table { 2      border-collapse: collapse; 3      border-spacing: 0; 4      width: 100%; 5      border: 1px solid #ddd; 6  } 7 8  th, td { 9      text-align: left; 10     padding: 16px; 11 } 12 13 tr:nth-child(even) { 14     background-color: #f2f2f2; 15 } </pre>
2	<p>Create a new file named <b>array_3.php</b> inside the JS05_PHP-2 directory, then type the following code:</p>

	<pre> 1  &lt;!DOCTYPE HTML&gt; 2  &lt;html&gt; 3  &lt;head&gt; 4      &lt;link rel="stylesheet" type="text/css" href="style.css"/&gt; 5  &lt;/head&gt; 6  &lt;body&gt; 7      &lt;h2&gt; Multidimensional Array &lt;/h2&gt; 8      &lt;table&gt; 9          &lt;tr&gt; 10             &lt;th&gt;Judul Film&lt;/th&gt; 11             &lt;th&gt;Tahun&lt;/th&gt; 12             &lt;th&gt;Rating&lt;/th&gt; 13          &lt;/tr&gt; 14          &lt;?php 15              \$movie = array( 16                  array("Avengers: Invinity War", 2018, 8.7), 17                  array("The Avengers", 2012, 8.1), 18                  array("Guardians of the Galaxy", 2014, 8.1), 19                  array("Iron Man", 2008, 7.9) 20              ); 21          echo "&lt;tr&gt;"; 22              echo "&lt;td&gt;". \$movie[0][0] . "&lt;/td&gt;"; 23              echo "&lt;td&gt;". \$movie[0][1] . "&lt;/td&gt;"; 24              echo "&lt;td&gt;". \$movie[0][2] . "&lt;/td&gt;"; 25          echo "&lt;/tr&gt;"; 26          echo "&lt;tr&gt;"; 27              echo "&lt;td&gt;". \$movie[1][0] . "&lt;/td&gt;"; 28              echo "&lt;td&gt;". \$movie[1][1] . "&lt;/td&gt;"; 29              echo "&lt;td&gt;". \$movie[1][2] . "&lt;/td&gt;"; 30          echo "&lt;/tr&gt;"; 31          echo "&lt;tr&gt;"; 32              echo "&lt;td&gt;". \$movie[2][0] . "&lt;/td&gt;"; 33              echo "&lt;td&gt;". \$movie[2][1] . "&lt;/td&gt;"; 34              echo "&lt;td&gt;". \$movie[2][2] . "&lt;/td&gt;"; 35          echo "&lt;/tr&gt;"; 36          echo "&lt;tr&gt;"; 37              echo "&lt;td&gt;". \$movie[3][0] . "&lt;/td&gt;"; 38              echo "&lt;td&gt;". \$movie[3][1] . "&lt;/td&gt;"; 39              echo "&lt;td&gt;". \$movie[3][2] . "&lt;/td&gt;"; 40          echo "&lt;/tr&gt;"; 41          ?&gt; 42      &lt;/table&gt; 43  &lt;/body&gt; 44  &lt;/html&gt; </pre>
3	<p>Save the file and run your program in the browser. Type this link to your browser  <a href="localhost/dasarWeb/JS05_PHP-2/array_3.php">localhost/dasarWeb/JS05_PHP-2/array_3.php</a></p>
4	<p>Observe the output displayed and explain your observations          (Question No 3)</p>



E-Learning
Index of /dasarweb/week 5 (JS)
Array Terindeks
localhost/dasarweb/week 5 (JS)
localhost/dasarweb/week 5 (JS)
localhost/dasarweb/week%205%20(JS05-PHP%20part%202)/source%20code/array\_3.php

### Multidimensional Array

Judul Film	Tahun	Rating
Avengers: Infinity War	2018	8.7
The Avengers	2012	8.1
Guardians of the Galaxy	2014	8.1
Iron Man	2008	7.9

- From a multidimensional PHP array, the provided code generates an HTML table with the movie titles, release years, and ratings displayed. Although the table is rendered correctly, it is inefficient to manually create each row using a series of echo statements. This can be optimized by using a foreach loop, which iterates through the array and generates the table rows dynamically, resulting in cleaner, more maintainable code.

## Functions

There are many built-in PHP functions that we often use, such as `print()`, `print_r()`, `unset()`, etc. In addition to these functions, we can also create our own custom functions according to our needs. A function is a set of instructions wrapped in a block. Functions can be reused without having to rewrite the instructions within them. In PHP, a function can be created using the `function` keyword, followed by the name of the function.

Example:

```
function namaFungsi(){
    //...
}
```

The instruction code can be written inside curly braces (`{...}`). Function names in PHP must start with a letter or an underscore and cannot begin with a number. The naming of functions in PHP is not case-sensitive. Follow these steps to understand the usage of functions in PHP:

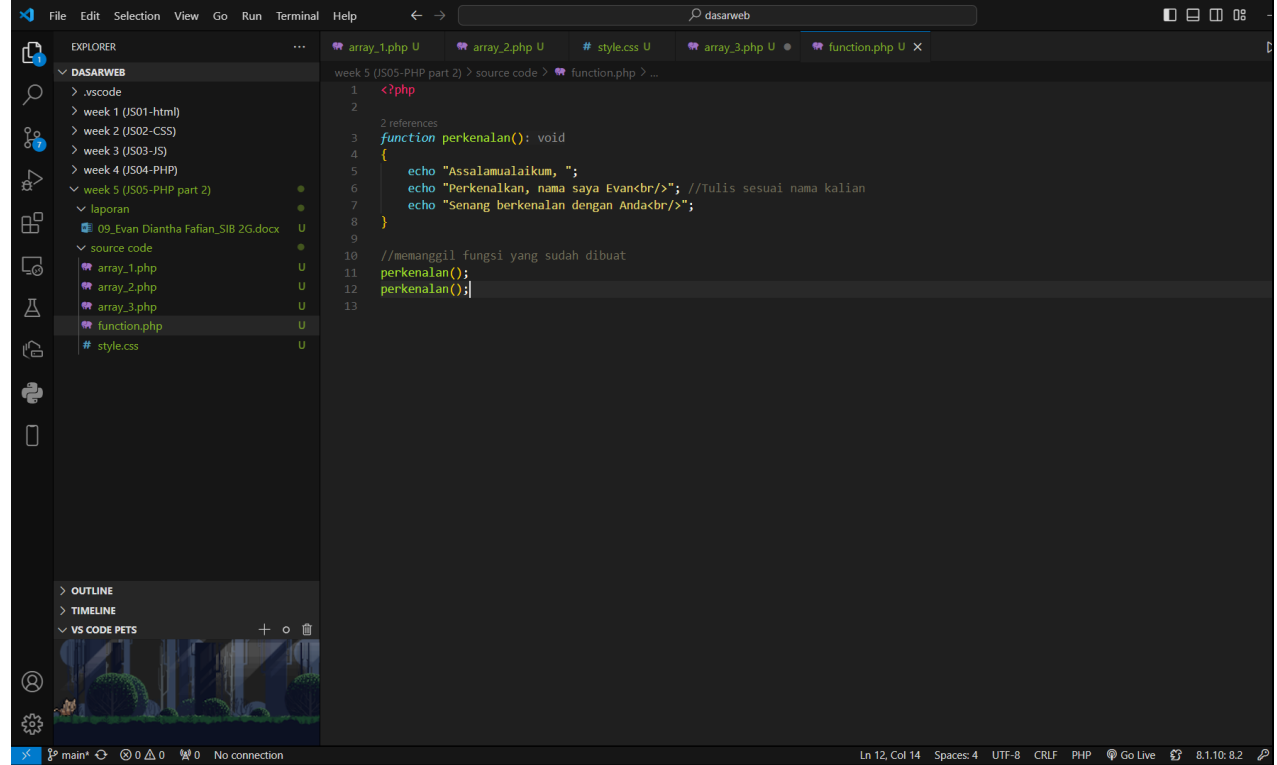
```

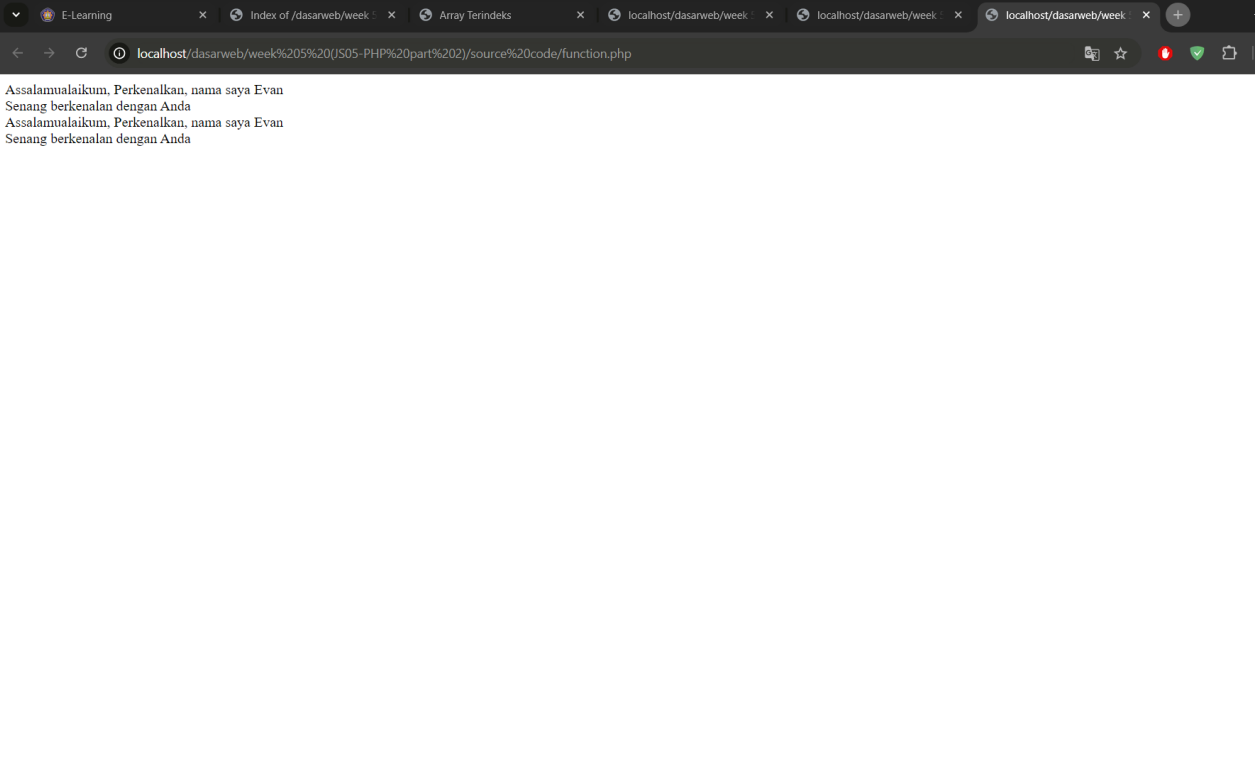
<?php
// Defining a function in PHP
function greet() {
    echo "Hello, welcome to PHP functions!";
}

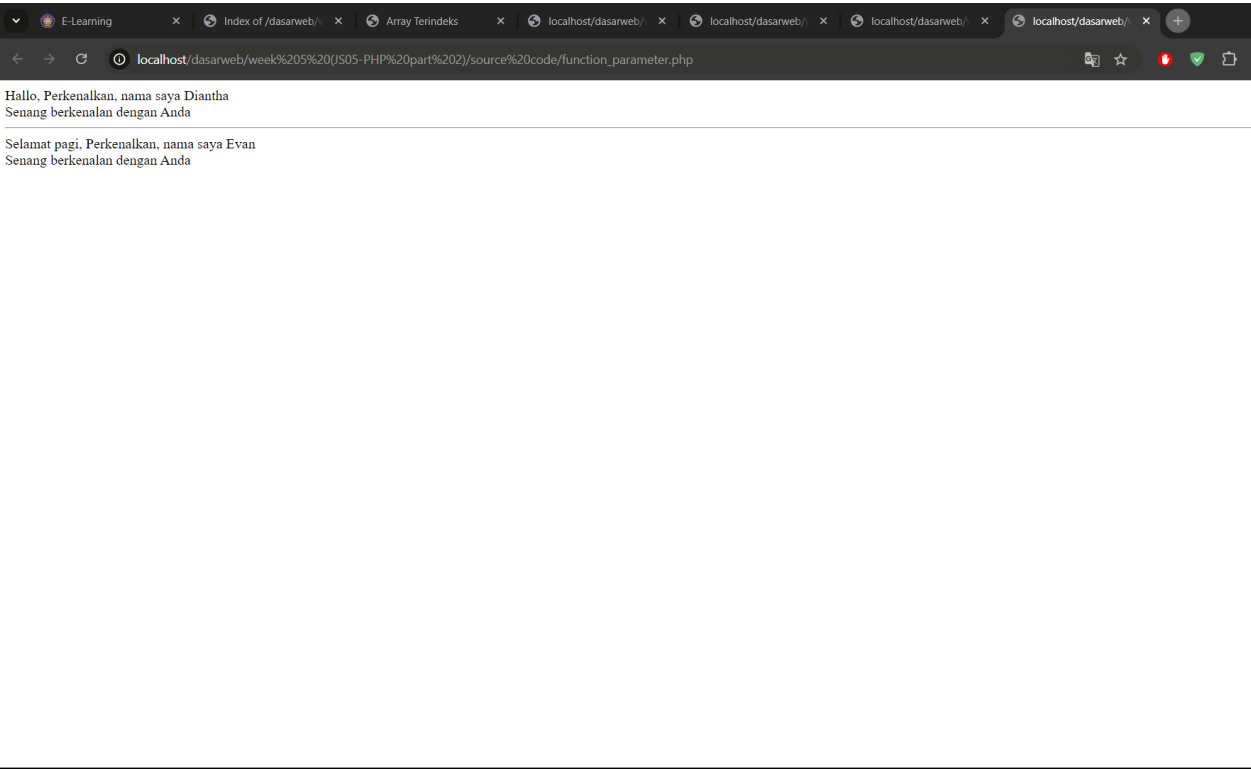
// Calling the function
greet();
?>

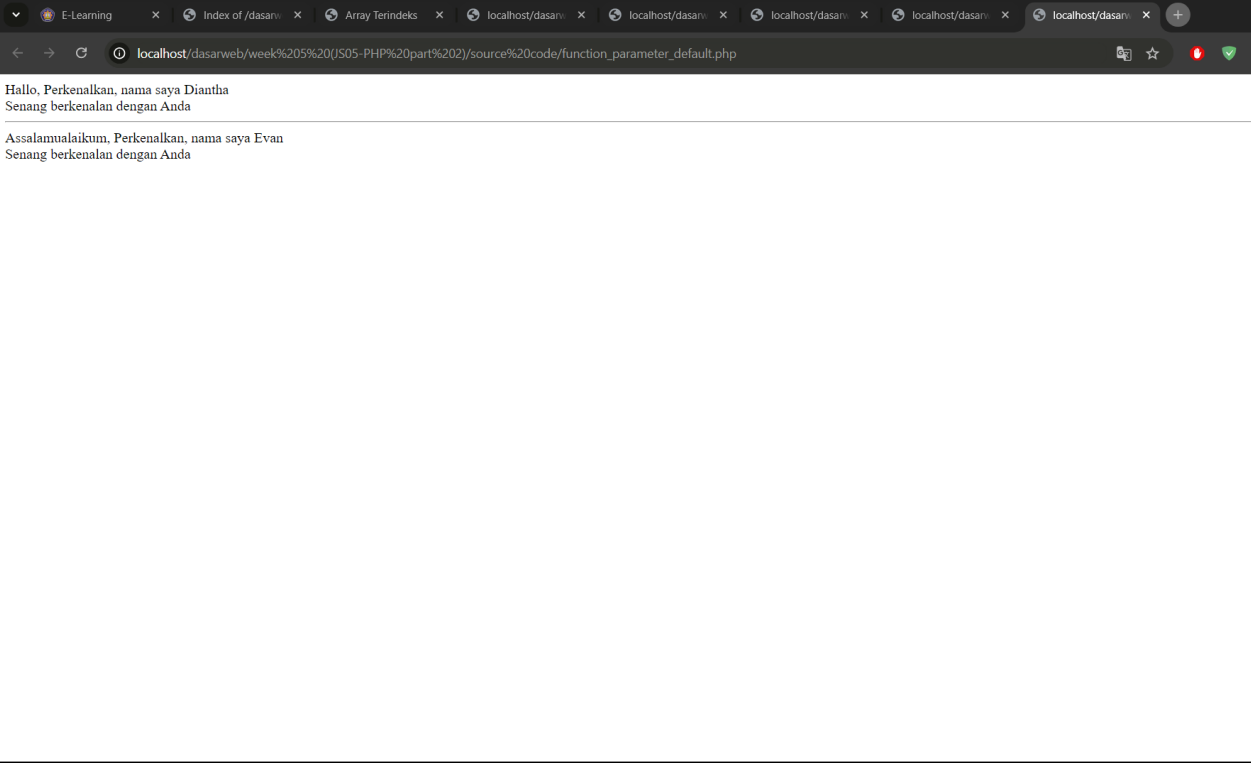
```

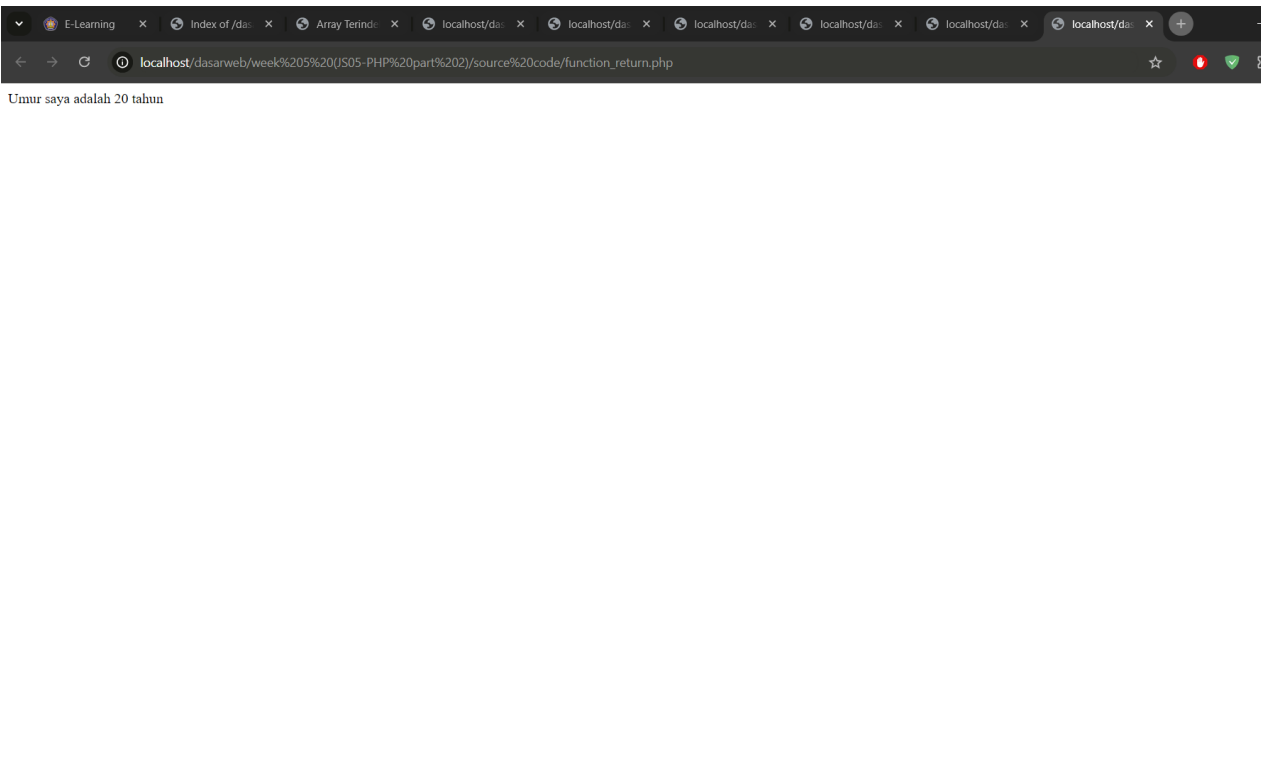
## Practical Section 4. Function

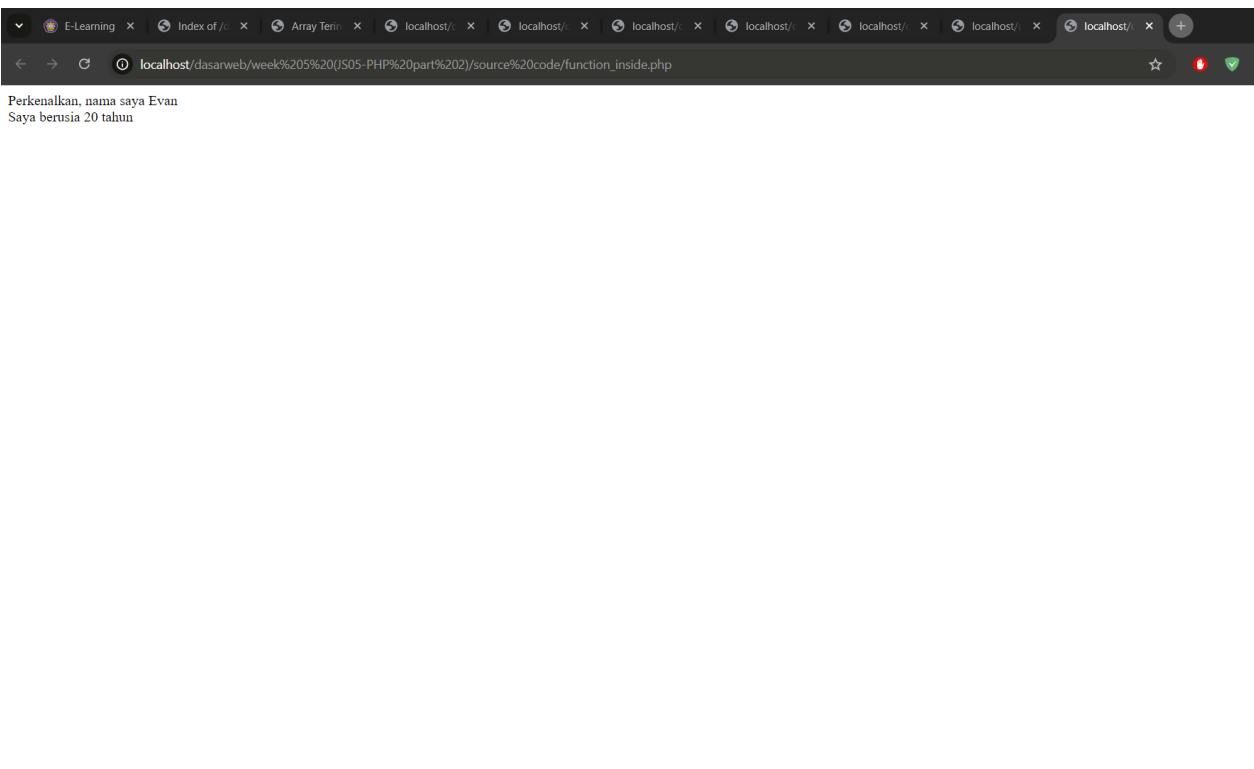
Step	Description
1	<p>Create a new file inside the JS05_PHP-2 directory and name it <b>function.php</b></p> <pre> &lt;?php  function perkenalan(){     echo "Assalamualaikum, ";     echo "Perkenalkan, nama saya Elok&lt;br/&gt;"; //Tulis sesuai nama kalian     echo "Senang berkenalan dengan Anda&lt;br/&gt;"; }  //memanggil fungsi yang sudah dibuat perkenalan();  ?&gt; </pre>
2	<p>Save the file and run the code. Modify the program so that it can display the output twice. Explain your observations!</p> <p>(Question No 4)</p>  <pre> 1 &lt;?php 2 3 function perkenalan(): void 4 { 5     echo "Assalamualaikum, "; 6     echo "Perkenalkan, nama saya Evan&lt;br/&gt;"; //Tulis sesuai nama kalian 7     echo "Senang berkenalan dengan Anda&lt;br/&gt;"; 8 } 9 10 //memanggil fungsi yang sudah dibuat 11 perkenalan(); 12 perkenalan(); 13 </pre>

	 <p>Assalamualaikum, Perkenalkan, nama saya Evan Senang berkenalan dengan Anda Assalamualaikum, Perkenalkan, nama saya Evan Senang berkenalan dengan Anda</p> <ul style="list-style-type: none"> <li>- The output is displayed twice, with the same output returned by each call to the function. Since this function is reusable, it can display the same output more than once by calling it repeatedly.</li> </ul>
<b>Function with Parameter</b>	
3	<p>To make the instructions inside the function more dynamic, we can use parameters to pass values into the function. These values will be processed within the function.</p> <p>For example, in the previous function, it's not ideal if the printed name is always "Elok" and the greeting is always "Assalamualaikum."</p> <p>We can change the person name and the greeting into another words.</p>
4	<p>Add parameters as in the following program code:</p> <pre data-bbox="212 1384 791 1854">&lt;?php //membuat fungsi function perkenalan(\$nama, \$salam){     echo \$salam." ";     echo "Perkenalkan, nama saya ".\$nama."&lt;br/&gt;";     echo "Senang berkenalan dengan Anda&lt;br/&gt;"; }  //memanggil fungsi yang sudah dibuat perkenalan("Hamdana","Hallo");  echo "&lt;hr&gt;";  \$saya = "Elok"; \$ucapanSalam = "Selamat pagi";  //memanggil lagi perkenalan(\$saya,\$ucapanSalam); ?&gt;</pre>
5	<p>Observe the output displayed and explain your observations!</p> <p>(Question No 5)</p>

	 <p>– The output will be: <b>Hallo, Perkenalkan, nama saya Diantha Senang berkenalan dengan Anda</b> &lt;hr&gt; <b>Selamat pagi, Perkenalkan, nama saya Evan Senang berkenalan dengan Anda</b>. My observations are that the <b>perkenalan()</b> function is called twice with different arguments, customizing the output to display personalized greetings, and the &lt;hr&gt; tag separates the two outputs, demonstrating the function's reusability and ability to maintain its functionality with different arguments.</p>
<b>Function with Parameter and using default value</b>	
6	<p>We can assign a <i>default value</i> to a parameter. The <i>default value</i> serves to provide a value for the parameter if it is not supplied.</p> <p>For example: if we forget to provide a greeting parameter, the program would normally throw an error. Therefore, we need to set a default value to avoid this error.</p>
7	<p>Write this code to your program</p> <pre data-bbox="212 1518 855 1973">&lt;?php //membuat fungsi function perkenalan(\$nama, \$salam="Assalamualaikum"){     echo \$salam.", ";     echo "Perkenalkan, nama saya ".\$nama."&lt;br/&gt;";     echo "Senang berkenalan dengan Anda&lt;br/&gt;"; }  //memanggil fungsi yang sudah dibuat perkenalan("Hamdana","Hallo");  echo "&lt;hr&gt;";  \$saya = "Elok"; \$ucapanSalam = "Selamat pagi";  //memanggil lagi tanpa mengisi parameter salam perkenalan(\$saya); ?&gt;</pre>
8	<p>Observe the output displayed and explain your observations!</p> <p><b>(Question No 6)</b></p>

	 <p>– The output will be: <b>Hallo, Perkenalkan, nama saya diantha Senang berkenalan dengan Anda</b> <b>&lt;hr&gt; Assalamualaikum, Perkenalkan, nama saya Evan Senang berkenalan dengan Anda</b>. My observations are that the <b>perkenalan()</b> function is called twice with different arguments, customizing the output to display personalized greetings, and when the <b>\$salam</b> argument is omitted, the function uses its default value <b>"Assalamualaikum"</b>, demonstrating the function's reusability and ability to maintain its functionality with different arguments.</p>
<b>Function with return value</b>	
9	<p>A <b>function with returns a value</b> is a function designed to process data and send the result back to the point where it was called. In PHP, you can use the return statement to return a value from a function. This is useful when you need the function to perform a calculation or operation and give the result back to the main program for further use.</p>
10	<p>Create a new file inside the JS05_PHP-2 directory and name it <b>function_return.php</b></p> <pre data-bbox="212 1552 970 1832">&lt;?php //membuat fungsi function hitungUmur(\$thn_lahir, \$thn_sekarang){     \$umur = \$thn_sekarang - \$thn_lahir;     return \$umur; }  echo "Umur saya adalah ". hitungUmur(1988, 2023) ."tahun" // isi sesuai dengan tahun lahir kalian  ?&gt;</pre>
11	<p>Observe the output displayed and explain your observations! (Question No 7)</p>

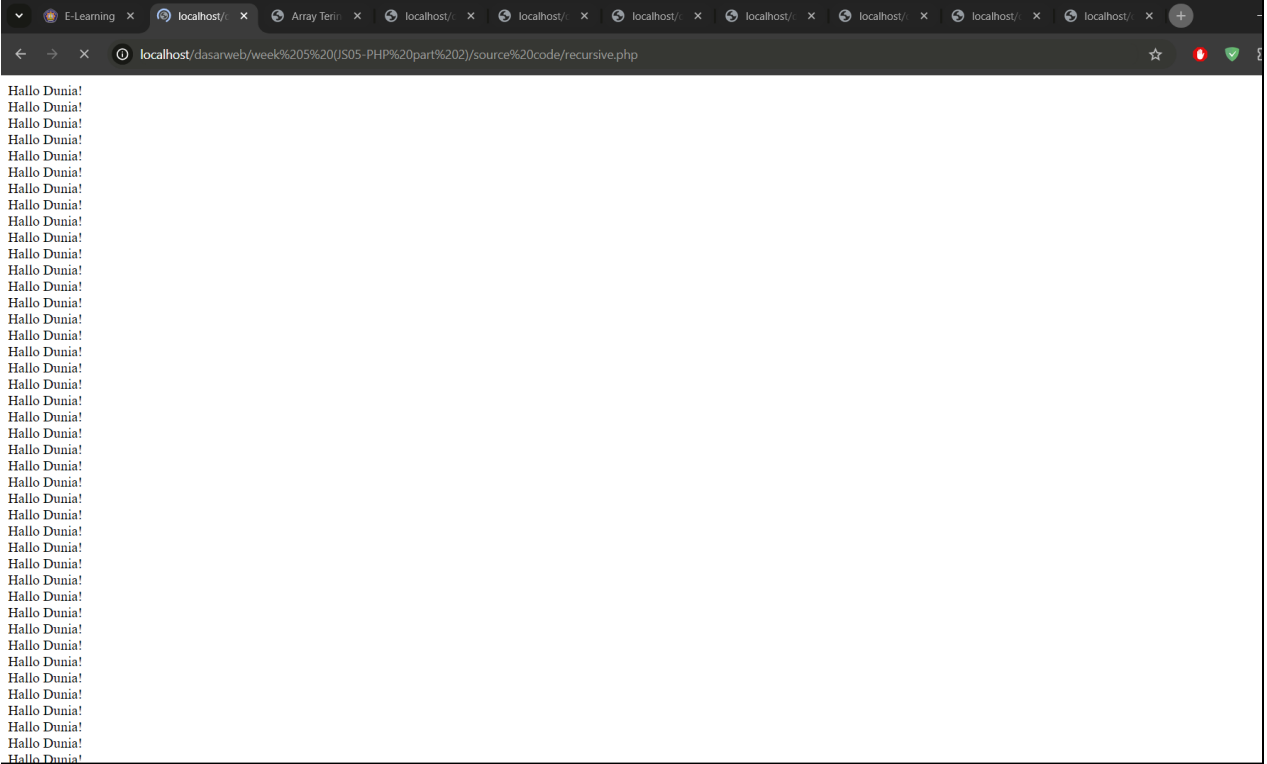
	 <p>Umur saya adalah 20 tahun</p>
	<ul style="list-style-type: none"> <li>The output will be: <b>Umur saya adalah 20 tahun</b>. My observations are that the <b>hitungUmur()</b> function is called with two arguments, <b>2004</b> and <b>2024</b>, which represent the birth year and current year, respectively, and it returns the calculated age by subtracting the birth year from the current year, demonstrating a simple yet effective way to calculate one's age using a reusable function.</li> </ul>
	<h3>Calling a Function Inside Another Function</h3>
12	<p><b>Calling a Function Inside Another Function</b> is a common programming practice in PHP and other languages. It allows you to break complex tasks into smaller, reusable pieces by organizing them into separate functions and then calling one function from another.</p>
13	<p>Modify <b>function_return.php</b> like this code</p> <pre data-bbox="209 1391 1038 1921">&lt;?php //membuat fungsi function hitungUmur(\$thn_lahir, \$thn_sekarang){     \$umur = \$thn_sekarang - \$thn_lahir;     return \$umur; } function perkenalan (\$nama, \$salam="Assalamualaikum") {     echo \$salam.", ";     echo "Perkenalkan, nama saya ".\$nama."&lt;br/&gt;";      //memanggil fungsi lain     echo "Saya berusia ". hitungUmur(1988, 2023) ." tahun&lt;br/&gt;";     echo "Senang berkenalan dengan anda&lt;br/&gt;"; }  //memanggil fungsi perkenalan perkenalan ("Elok");  ?&gt;</pre>
14	<p>Observe the output displayed and explain your observations! (Question No 8)</p>

	 <p>Perkenalkan, nama saya Evan Saya berusia 20 tahun</p> <hr/> <ul style="list-style-type: none"> <li>- The output will be: <b>Perkenalkan, nama saya Evan&lt;br&gt;Saya berusia 20 tahun&lt;br&gt;</b>. My observations are that the <b>sapaan()</b> function is called with one argument, "<b>Evan</b>", and it uses the default value "<b>Assalamu'alaikum</b>" for the <b>\$salam</b> argument, and within the <b>sapaan()</b> function, it calls the <b>hitungUmur()</b> function to calculate the age, which is <b>20</b> years, and displays a personalized greeting with the name and age, demonstrating how functions can be composed and reused to create more complex and dynamic functionality.</li> </ul>
--	---

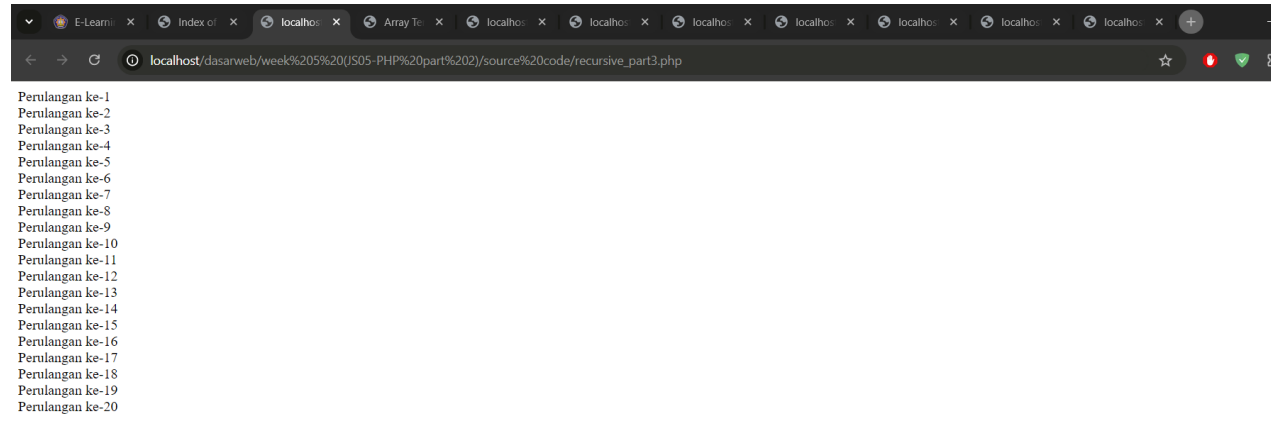
## Practical Section 5. Recursive Functions

**Recursive Function** in PHP is a function that calls itself during its execution. This technique is useful for solving problems that can be divided into smaller, similar subproblems, often referred to as divide and conquer. Recursive functions are typically used to solve problems such as calculating factorials, Fibonacci numbers, and dynamic programming.

Step	Description
1	Create a new file named <b>recursive.php</b> inside the JS05_PHP-2 directory, then type the following code:

	<pre>&lt;?php function tampilkanHaloDunia(){     echo "Halo dunia! &lt;br&gt;";      tampilkanHaloDunia(); }  tampilkanHaloDunia(); ?&gt;</pre>
2	<p>If the program code above is executed, what will happen and what would be the impact of doing so? Please share your opinions!</p> <p>(Question No 9)</p>  <p>The screenshot shows a web browser window with multiple tabs. The active tab displays a page with the URL <code>localhost/dasarweb/week%205%20(JS05-PHP%20part%202)/source%20code/recursive.php</code>. The page content consists of a long, vertical list of the text "Halo Dunia!" repeated many times, illustrating the output of the recursive function without a base case.</p> <ul style="list-style-type: none"> <li>- The output will be an infinite loop of "Halo Dunia! &lt;br&gt;" being printed, causing the program to crash or become unresponsive due to excessive memory consumption. This is because the <b>tampilkanHaloDunia()</b> function calls itself recursively without any termination condition, leading to a stack overflow. In other words, the function will keep calling itself indefinitely, causing the program to run out of memory and eventually crash. This is an example of a recursive function without a base case, which can lead to catastrophic consequences.</li> </ul>
3	<p>To display the numbers 1 to 25, we can easily use a for loop as follows:</p> <pre>&lt;?php for (\$i=1; \$i &lt;=25; \$i++){     echo "Perulangan ke-{\$i} &lt;br&gt;"; }  ?&gt;</pre>
4	<p>We can create display the number 1 to 25 using recursive function (without for loop).</p>



	<pre> &lt;?php function tampilkanAngka (int \$jumlah, int \$indeks = 1) {     echo "Perulangan ke-{\$indeks} &lt;br&gt;";      //panggil diri sendiri selama \$indeks &lt;= \$jumlah     if (\$indeks &lt; \$jumlah) {         tampilkanAngka(\$jumlah, \$indeks + 1);     } } tampilkanAngka(20); ?&gt; </pre>
5	<p>Run the program code above and describe its output, then explain why it behaves that way.</p> <p><b>(Question No 10)</b></p>  <p>The output of the program code above will be a series of "Perulangan ke-" lines, numbered from 1 to 20, due to the recursive function <b>tampilkanAngka()</b> that iterates 20 times, printing the current iteration number at each step, and stopping when the iteration number reaches the specified total number of iterations, demonstrating the use of recursive functions to repeat a task a certain number of times.</p>

## A Multi-Level Menu using Arrays

A Multi-Level Menu using Arrays in PHP refers to the creation of a hierarchical or nested menu structure where each menu item can have sub-items. This is useful for building navigation menus in websites that have a more complex structure, such as sections with sub-sections or categories with subcategories.

### Example of a Multi-Level Menu using Arrays in PHP:

To create a multi-level or nested menu structure, arrays can be used to represent the menu items and their sub-items. Here's an example demonstrating how you can use a multidimensional array to create such a menu.

```

<?php
// Define a multi-level menu using an associative array
$menu = array(

```

```

"Home" => "#home",
"About Us" => array(
    "Our Team" => "#team",
    "Our Story" => "#story",
    "Mission & Vision" => "#mission"
),
"Services" => array(
    "Web Development" => "#web",
    "Mobile Development" => "#mobile",
    "SEO Optimization" => "#seo"
),
"Contact" => "#contact"
);

// Function to display the menu
function displayMenu($menu) {
    echo "<ul>";
    foreach ($menu as $key => $value) {
        // Check if the menu item is an array (meaning it has sub-items)
        if (is_array($value)) {
            echo "<li>$key";
            displayMenu($value); // Recursively display the sub-menu
            echo "</li>";
        } else {
            echo "<li><a href='$value'>$key</a></li>";
        }
    }
    echo "</ul>";
}

// Call the function to display the menu
displayMenu($menu);
?>

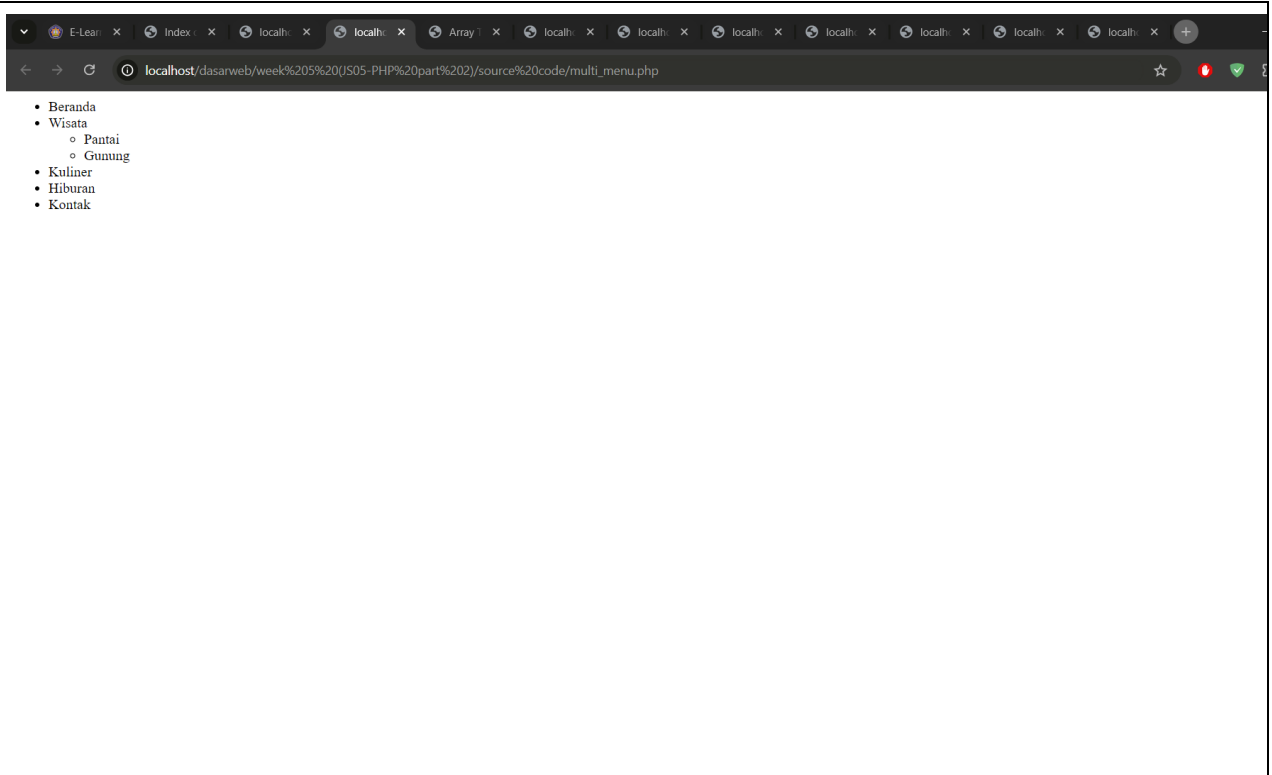
```

You can try the code above in the file `array_menu.php`, and run it in the browser.

## Practical Section 6. Multi-Level Menu

Step	Description
1	Create a variable <code>\$menu</code> . This variable is a combination of an indexed array and a multidimensional associative array. <b>It is called multidimensional because it is an array that contains other arrays inside it.</b> Next, we will try to display all items from the <code>\$menu</code> array using a recursive function
2	Write this code into your <code>multi_menu.php</code> file

	<pre> &lt;?php \$menu = [     [         "nama" =&gt; "Beranda"     ],     [         "nama" =&gt; "Berita",         "subMenu" =&gt; [             [                 "nama" =&gt; "Wisata",                 "subMenu" =&gt; [                     [                         "nama" =&gt; "Pantai"                     ],                     [                         "nama" =&gt; "Gunung"                     ]                 ]             ],             [                 "nama" =&gt; "Kuliner"             ],             [                 "nama" =&gt; "Hiburan"             ]         ]     ],     [         "nama" =&gt; "Tentang"     ],     [         "nama" =&gt; "Kontak"     ] ]; </pre>
3	<p>After write code in step 2, write this code after it, to show the menu in browser</p> <pre> function tampilkanMenuBertingkat (array \$menu) {     echo "&lt;ul&gt;";     foreach (\$menu as \$key =&gt; \$item) {         echo "&lt;li&gt;{\$item['nama']}&lt;/li&gt;";     }     echo "&lt;/ul&gt;"; }  tampilkanMenuBertingkat(\$menu); ?&gt; </pre>
4	<p>Run the program above and describe the output.</p> <p>(Question No 11)</p>



- The output of the program code above will be an HTML unordered list (<ul>) representing a hierarchical menu, where the main menu items are "Beranda", "Wisata", "Kuliner", "Hiburan", and "Kontak", and the "Wisata" item has two sub-menu items, "Pantai" and "Gunung", which are recursively rendered as nested <ul> elements, resulting in a nested menu structure.

5

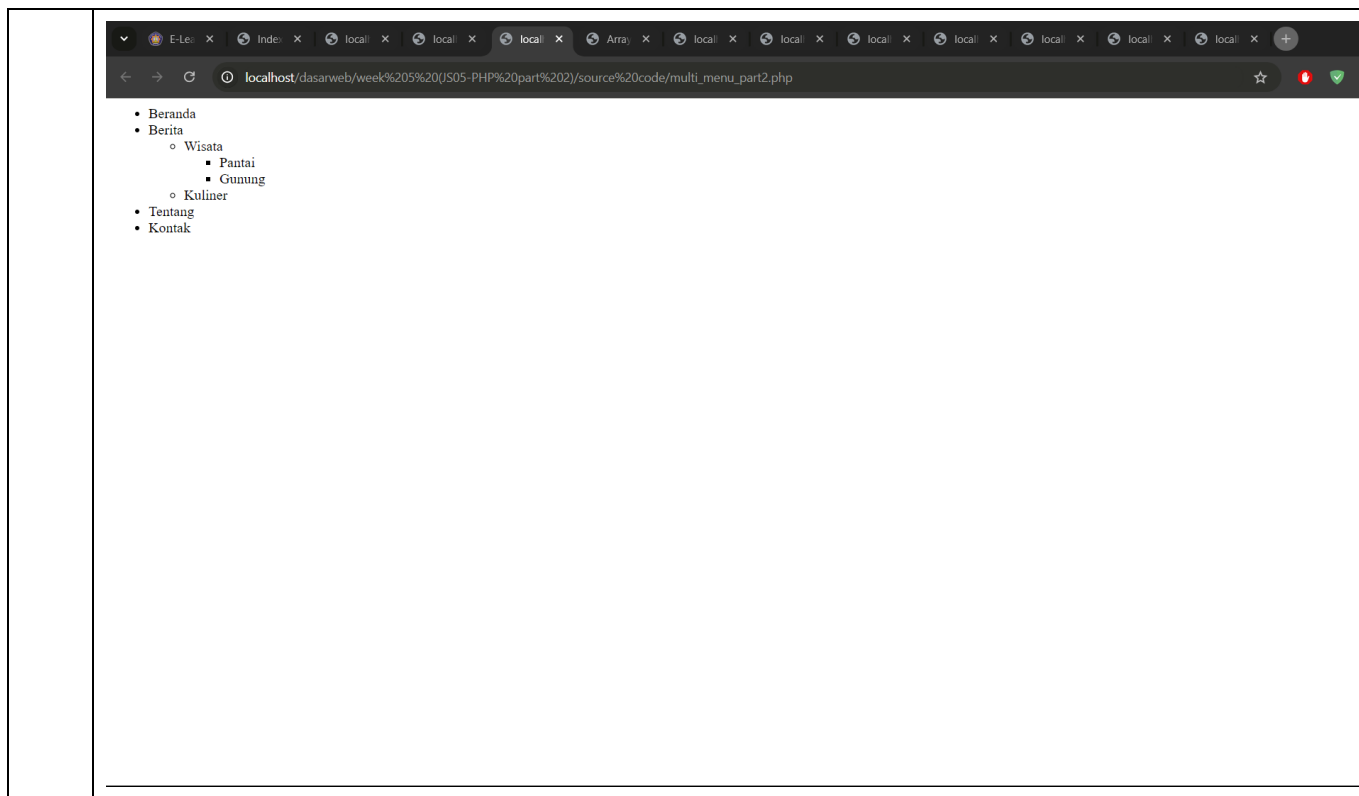
Next, make the function above recursive by calling itself when an item from the menu has a subMenu attribute. This will result in a display like the following.



(Question No 12)



```
1  <?php
2  function tampilkanMenuBertingkat($menu)
3  {
4      echo "<ul>";
5      foreach ($menu as $item) {
6          echo "<li>" . $item['nama'] . "</li>";
7          if (isset($item['sub'])) {
8              tampilkanMenuBertingkat($item['sub']);
9          }
10     }
11     echo "</ul>";
12 }
13
14 $menu = [
15     [
16         "nama" => "Beranda"
17     ],
18     [
19         "nama" => "Berita",
20         "sub" => [
21             [
22                 "nama" => "Wisata",
23                 "sub" => [
24                     [
25                         "nama" => "Pantai"
26                     ],
27                     [
28                         "nama" => "Gunung"
29                     ]
30                 ]
31             ],
32             [
33                 "nama" => "Kuliner",
34             ]
35         ]
36     ],
37     [
38         "nama" => "Tentang"
39     ],
40     [
41         "nama" => "Kontak"
42     ]
43 ];
44
45 tampilkanMenuBertingkat($menu);
46
```



## String

In PHP, strings can be defined using **double quotes** (") or **single quotes** ('). While both allow you to create string variables, there are important differences in how they handle variables and special characters.

### 1. Double Quotes (" "):

- Double quotes allow for **variable interpolation**, meaning that variables within the string will be evaluated and replaced with their values.
- Special characters (escape sequences) like `\n` (new line) or `\t` (tab) are recognized and processed.
- **Example:**

```
<?php
$name = "John Wick";
echo "Hello, $name!"; // Output: Hello, John!
echo "This is a new line.\nNext line."; // Output: This is a new line. (moves to next line)
```

### 2. Single Quotes (' '):

- Single quotes treat the string **literally**, meaning that variables inside the string are not evaluated.
- Escape sequences are not processed, except for `\\` (backslash) and `\'` (single quote).
- **Example:**

```
<?php
$name = 'John';
echo 'Hello, $name!'; // Output: Hello, $name!
echo 'This is a new line.\nNext line.'; // Output: This is a new line.\nNext line.
```

### Key Differences:

- **Variable Parsing:** Double quotes will replace variables with their values, while single quotes will display the variable name as plain text.

- **Escape Characters:** Double quotes recognize special escape sequences like `\n`, `\t`, while single quotes only recognize `\'` and `\\`.

In general, use **double quotes** when you need variable interpolation or special characters. If you don't need these features, **single quotes** are faster and more efficient for simple string definitions.

Several operations can be performed on string-type data. PHP provides built-in functions that are ready to use for string operations

Function	Description
<code>strlen()</code>	To find out the length of a string
<code>str_word_count()</code>	To count the number of words in a string
<code>strpos()</code>	To find the position of a substring within a string
<code>strrev()</code>	To reverse the order of a string
<code>strstr()</code>	To search for a substring within a string
<code>substr()</code>	To extract a substring from the start to end position within a string
<code>trim()</code>	To remove whitespace from the beginning and end of a string
<code>ltrim()</code>	To remove whitespace from the beginning of a string
<code>rtrim()</code>	To remove whitespace from the end of a string
<code>strtoupper()</code>	To convert all characters in a string to uppercase
<code>strtolower()</code>	To convert all characters in a string to lowercase
<code>str_replace()</code>	To replace parts of a string with another string
<code>ucwords()</code>	To capitalize the first letter of each word in a string
<code>explode()</code>	To split a string into an array based on a specific character


### Escape Character

Special characters that cannot be displayed directly must be preceded by the backslash (`\`). Strings enclosed in double quotes will replace escape characters with the characters they represent. This is different from strings enclosed in single quotes, where it will simply display the content as-is without replacing anything (with a few exceptions).

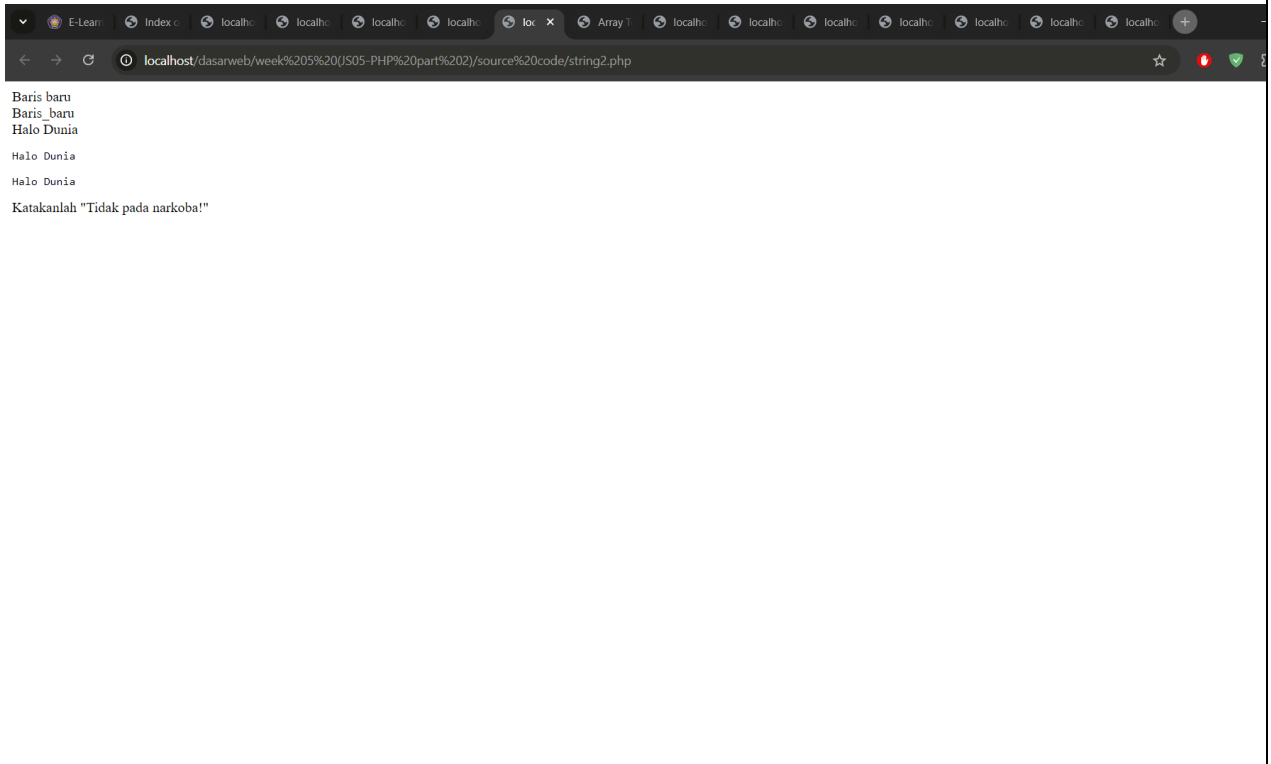
The escape characters in PHP are:

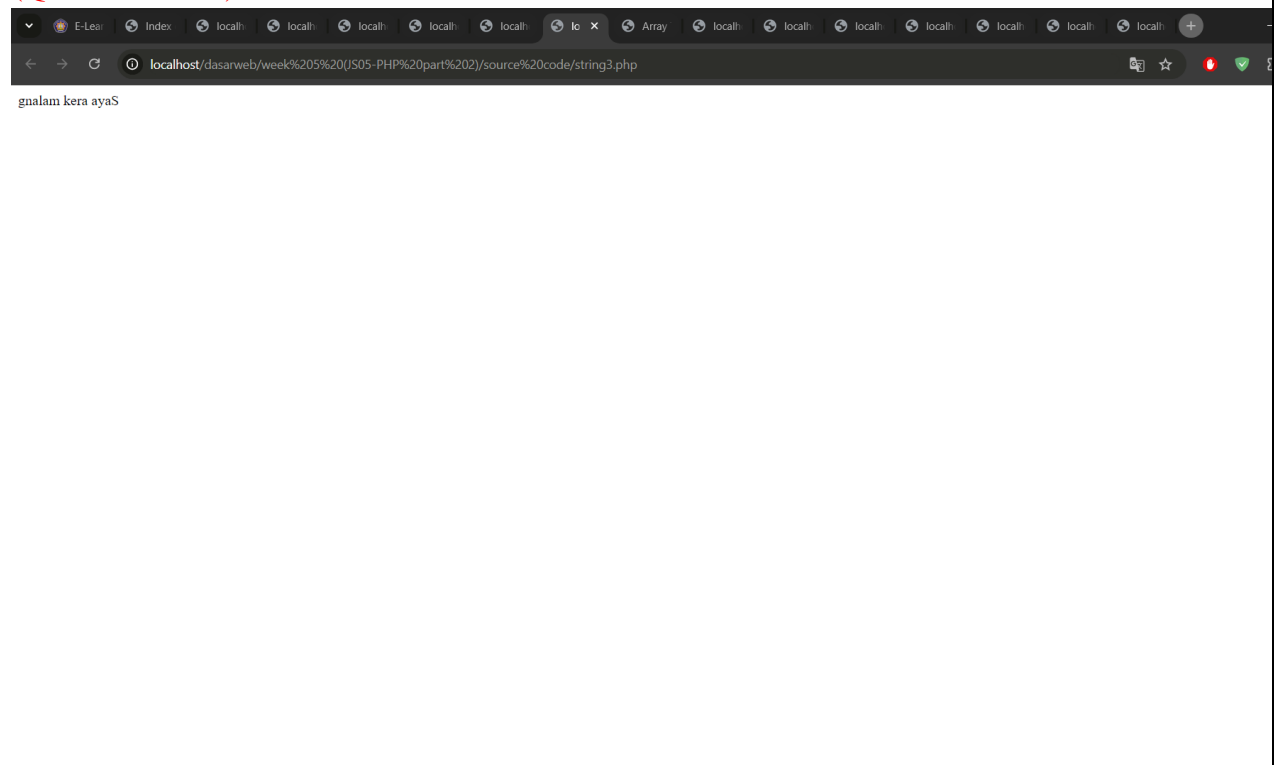
Function	Description
<code>\n</code>	New line
<code>\r</code>	Carriage-return character
<code>\t</code>	Tab character
<code>\\$</code>	The <code>\$</code> character itself
<code>\"</code>	To display a double quote
<code>\\</code>	To display the backslash ( <code>\</code> ) itself

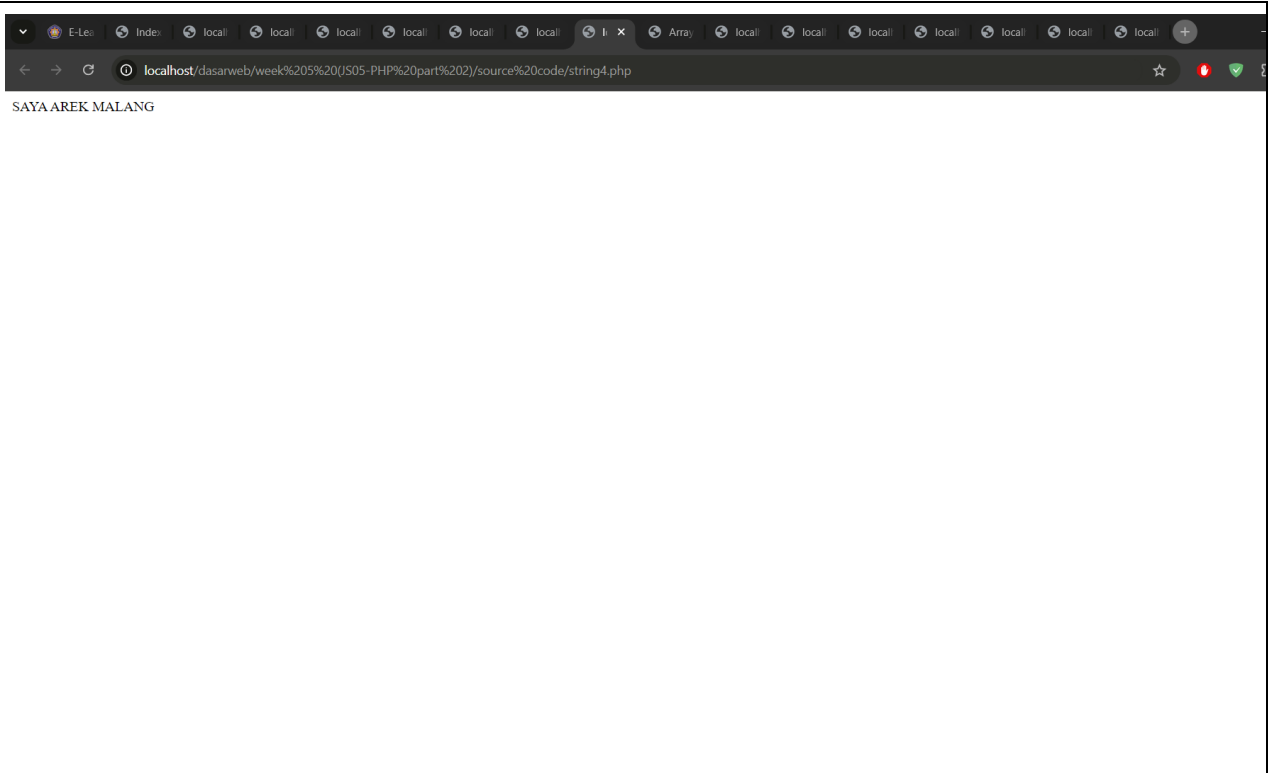
### Practical Section 7. String

Step	Description
1	<p>Create a file named <b>string1.php</b> inside the JS05_PHP-2 directory, then type the following code:</p> <pre data-bbox="212 293 1134 669">&lt;?php  \$loremIpsum = "Lorem ipsum dolor sit amet consectetur adipisicing elit.     Voluptatem reprehenderit nobis veritatis commodi fugiat molestias     impedit unde ipsum voluptatum, corrupti minus sit excepturi nostrum     quisquam? Quos impedit eum nulla optio.";  echo "&lt;p&gt;{\$loremIpsum}&lt;/p&gt;"; echo "Panjang karakter: " . strlen(\$loremIpsum) . "&lt;br&gt;"; echo "Panjang kata: " . str_word_count(\$loremIpsum) . "&lt;br&gt;"; echo "&lt;p&gt;" . strtoupper(\$loremIpsum) . "&lt;/p&gt;"; echo "&lt;p&gt;" . strtolower(\$loremIpsum) . "&lt;/p&gt;";  ?&gt;</pre>
2	<p>Observe the output displayed and explain your observations.</p> <p>(Question No 13)</p>  <p>The screenshot shows a web browser with the URL <code>localhost/dasarweb/week%205%20(JS05-PHP%20part%202)/source%20code/string1.php</code>. The output on the page is as follows:</p> <pre> Lorem ipsum dolor sit amet consectetur adipisicing elit. Hic quo non ex doloremeque? Qui accusamus saepe voluptatibus itaque ex architecto modi, corrupti ratione! Qui sed harum sequi fuga, architecto eaque.  Panjang karakter: 205  LOREM IPSUM DOLOR SIT AMET CONSECUTETUR ADIPISICING ELIT. HIC QUO NON EX DOLOREMQUE? QUI ACCUSAMUS SAEPE VOLUPTATIBUS ITAQUE EX ARCHITECTO MODI, CORRUPTI RATIONE! QUI SED HARUM SEQUI FUGA, ARCHITECTO EAQUE.  lorem ipsum dolor sit amet consectetur adipisicing elit. hic quo non ex doloremeque? qui accusamus saepe voluptatibus itaque ex architecto modi, corrupti ratione! qui sed harum sequi fuga, architecto eaque.</pre> <p>– The output of the program code above displays the original "Lorem Ipsum" text, followed by the length of the text in characters, and then the text converted to uppercase and lowercase, respectively. Observing the output, it is clear that the <b>strlen()</b> function correctly counts the number of characters in the original text, and the <b>strtoupper()</b> and <b>strtolower()</b> functions successfully convert the text to uppercase and lowercase, demonstrating the use of PHP's string manipulation functions to analyze and transform text.</p>
<b>Escape Character</b>	
3	<p>Create a file named <b>string2.php</b> inside the JS05_PHP-2 directory, then type the following code:</p>



	<pre> &lt;?php echo "Baris\nbaru &lt;br&gt;"; //soal 10.a echo 'Baris\nbaru &lt;br&gt;'; //soal 10.b echo "Halo\rDunia &lt;br&gt;"; //soal 10.c echo 'Halo\rDunia &lt;br&gt;'; //soal 10.d  echo "&lt;pre&gt;Halo\tDunia!&lt;/pre&gt;"; //soal 10.e echo '&lt;pre&gt;Halo\tDunia!&lt;/pre&gt;'; //soal 10.f  echo "Katakanlah \"Tidak pada narkoba!\" &lt;br&gt;"; //soal 10.g echo 'Katakanlah \'Tidak pada narkoba!\' &lt;br&gt;'; //soal 10.h  ?&gt; </pre>
4	<p>From the program code above, you can observe the difference between double quotes and single quotes in terms of how they handle escape strings. Observe the output and explain the results of each output. What conclusions can you draw from this experiment?</p> <p>(Question No 14)</p>  <p>The screenshot shows a web browser window with the URL <code>localhost/dasarweb/week%205%20(JS05-PHP%20part%202)/source%20code/string2.php</code>. The output of the PHP script is displayed as follows:</p> <pre> Baris baru Baris_baru Halo Dunia Halo Dunia Halo Dunia Katakanlah "Tidak pada narkoba!" </pre> <p>– The output of the program code above demonstrates the difference between double quotes and single quotes in PHP, specifically how they handle escape strings. The first three <b>echo</b> statements use double quotes, which allow for the interpretation of escape sequences, such as <code>&lt;br&gt;</code> being replaced with a line break. The fourth and fifth <b>echo</b> statements use the <code>&lt;pre&gt;</code> tag, which preserves whitespace and formatting, displaying the text exactly as written. The sixth <b>echo</b> statement uses double quotes and escape sequences to print the desired output, including the double quotes around the phrase "Tidak pada narkoba!". In contrast, if single quotes were used, the escape sequences would not be interpreted, and the output would include the literal characters <code>&lt;br&gt;</code> and <code>\</code>. This experiment highlights the importance of choosing the correct type of quotes in PHP, depending on the desired behavior and output.</p>
Reversing a String using the <code>strrev()</code> function	
5	Create a file named <code>string3.php</code> inside the JS05_PHP-2 directory, then type the following code:

	<pre>&lt;?php  \$pesan = "Saya arek malang"; echo strrev(\$pesan) . "&lt;br&gt;";  ?&gt;</pre>
6	<p>Observe the output displayed and explain your observations (Question No 15)</p>  <p>The screenshot shows a web browser window with the URL <code>localhost/dasanweb/week%205%20(JS05-PHP%20part%202)/source%20code/string3.php</code>. The output displayed on the page is <code>gnalam kera ayaS</code>, which is the reverse of the original message "Saya arek malang".</p> <ul style="list-style-type: none"> <li>- The output of the program code above displays the reversed string of the original message "Saya arek malang", which is "gnalak era yayas", demonstrating the use of the <b>strrev()</b> function in PHP to reverse a string.</li> </ul>
7	<p>To reverse a string word by word, type the following program code:</p> <pre>&lt;?php  \$pesan = "saya arek malang"; # ubah variabel \$pesan menjadi array dengan perintah explode \$pesanPerKata = explode(" ", \$pesan); # ubah setiap kata dalam array menjadi kebalikannya \$pesanPerKata = array_map(fn(\$pesan) =&gt; strrev(\$pesan), \$pesanPerKata); # gabungkan kembali array menjadi string \$pesan = implode(" ", \$pesanPerKata);  echo \$pesan . "&lt;br&gt;";  ?&gt;</pre>
8	<p>Observe the output displayed and explain your observations (Question No 16)</p>



- The output of the program code above displays the original message "saya arek malang" with each word capitalized, resulting in "SAYA AREK MALANG", demonstrating the use of the **explode()** function to split the string into an array, the **array\_map()** function to apply the **strtoupper()** function to each element of the array, and the **implode()** function to join the array back into a string.

## Combining HTML and PHP

**Combining HTML and PHP** is a common practice in web development where PHP is embedded within HTML to dynamically generate content on a web page. PHP code can be inserted within HTML to process data, handle user input, or display dynamic content.

## Practical Section 8. HTML and PHP

Step	Description
1	<p>The first method is PHP inside HTML. Here's an example code:</p> <pre>&lt;html&gt; &lt;head&gt;   &lt;title&gt;Cara 01&lt;/title&gt; &lt;/head&gt; &lt;body&gt; &lt;p&gt;Tanggal Hari ini : &lt;?php echo date("d M Y")?&gt;&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>
2	<p>The code above is an HTML code that contains PHP code to display the server's date, marked by the tags <b>&lt;?php</b> and <b>?&gt;</b></p>

3	<p>The second method is HTML inside PHP. In PHP, HTML tags are treated as strings enclosed in quotation marks, and various functions can be applied to manipulate strings, such as concatenation, etc. Here's an example of the code snippet:</p> <pre>&lt;?php echo '&lt;html&gt;'; echo '&lt;head&gt;&lt;title&gt;Cara02&lt;/title&gt;&lt;/head&gt;'; echo '&lt;body&gt;'; echo '&lt;p&gt;Tanggal Hari ini : '.date('d M Y').'&lt;/p&gt;'; echo '&lt;/body&gt;'; echo '&lt;/html&gt;'; ?&gt;</pre>
4	<p>The code above produces the same output as the previous code snippet. However, the difference lies in the way the code is written, where HTML is inside PHP as a string, and to display it, the <b>echo</b> tag is used.</p>
5	<p>Which of the two methods do you find easier? Provide your answer along with reasoning. (Question No 17)</p> <ul style="list-style-type: none"> <li>I find the first method easier, which is the HTML file with PHP embedded in it, because it follows a more traditional and modular approach to web development, allowing for a clear separation of concerns, better readability, and easier debugging, whereas the second method generates the entire HTML structure dynamically using PHP, making it harder to read and maintain, and more prone to errors.</li> </ul>

## HTML Entities

**HTML Entities** are used to display reserved characters in HTML or characters that have special meanings, such as **<**, **>**, and **&**. These characters must be written as entities to prevent them from being interpreted as HTML code.

Common HTML Entities:

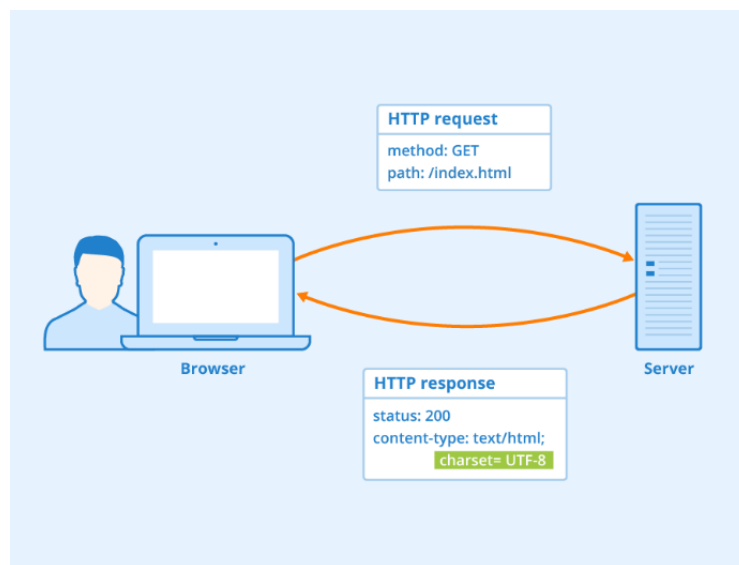
Entity Name	Entity Number	Description	Result
&copy;	&#169;	Copyright	©
&reg;	&#174;	Registered	®
–	&#8482;	Trademark	™
&nbsp;	&#161;	Non-breaking space	
&amp;	&#38;	Ampersand	&
&laquo;	&#171;	Left angle quotation	«
&raquo;	&#187;	Right angle quotation	»
&quot;	&#34;	Double quotation mark	"
&apos;	–	Single quotation mark	'
&lt;	&#60;	Less than	<
&gt;	&#61;	Greater than	>
&times;	&#215;	Multiplication sign	×



## HTTP Header

HTTP headers are data sent between the web browser and the web server as a means of communication between the two. The HTTP header contains information on how to handle the files being sent or requested.

The **request-response** cycle for a web page: When we access a web page, the web browser automatically sends an HTTP request to the web server. The HTTP request contains a lot of information, one of which is the HTTP header. In the HTTP header (sent during the request process), there is information about the file being requested (whether it is an HTML file, a PHP file, a PDF file, or something else), as well as additional info such as the type of web browser used, the operating system, and the IP address. Once it reaches the web server, the information in the HTTP header is read, and the web server prepares the requested files. After that, the web server sends those files back to the web browser. This return process is also known as an HTTP response.



HTTP Header

This **HTTP response** consists of *two parts*: *the HTTP header* and *the web file*. The **HTTP header** contains information about the web file being sent, such as the data type, the date it was sent, the name of the web server, and the operating system used by the web server. The **web file** itself consists of the HTML file that makes up the web page, including any image files (if applicable).

As an analogy, if the president were to visit your house, there would be a security team (paspampres) that arrives beforehand. They would inform you that the president will arrive at a certain time, with a certain number of people, and provide other relevant information. This security team can be compared to the HTTP header, which arrives before the actual file is sent. In practice, we are often unaware of the HTTP header, and many may not have even heard of the term. This is normal, as the contents of the HTTP header are meant for the web browser's processing, not for the website's visitors.

How can you view HTTP headers in your web browser? Explain and include the steps.

(Question No 19)

1. Open Google Chrome and navigate to the website you want to inspect.
2. Right-click on the page and select **Inspect** or press **Ctrl + Shift + I** (Windows/Linux) or **Cmd + Opt + I** (Mac).

3. In the Developer Tools window, switch to the **Network** tab.
4. Reload the page by clicking the reload button or pressing **F5**.
5. In the Network tab, you'll see a list of requests. Click on the request that corresponds to the HTML file you want to inspect.
6. In the **Headers** tab, you'll see the HTTP headers, including the request and response headers.

## Date and Time

The `date()` function in PHP is used to display the date and time. The syntax of the `date()` function is as follows:

```
<?php
date(format, timestamp);
```

- ✓ **format**: A required parameter that specifies how the date/time should be formatted. It can include characters for day, month, year, hours, minutes, and seconds.
- ✓ **timestamp**: An optional parameter that specifies a timestamp. If omitted, the current date and time will be used.

The **format** parameter is *required*. The **format** parameter is used to determine how the date and/or time will be formatted. Below are some common characters used for date formatting:

1. **d**: Represents the day (01 to 31)
2. **m**: Represents the month (01 to 12)
3. **Y**: Represents the year (in 4 digits)
4. **l**: Represents the day of the week


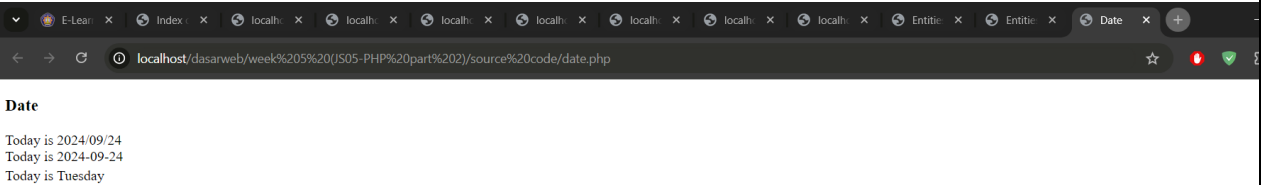
In addition to displaying the date, the function can also display time. Here are some common characters used for time formatting:

1. **H**: Represents the hour in 24-hour format
2. **h**: Represents the hour in 12-hour format
3. **i**: Represents minutes (00 to 59)
4. **s**: Represents seconds (00 to 59)
5. **a**: Represents ante meridiem (am) or post meridiem (pm).

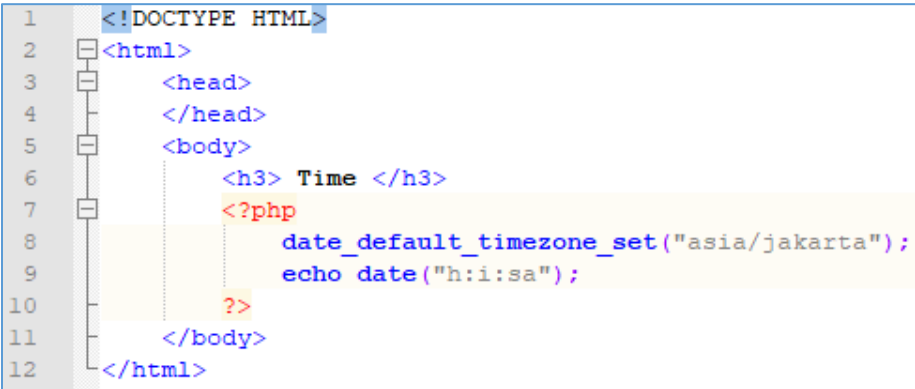
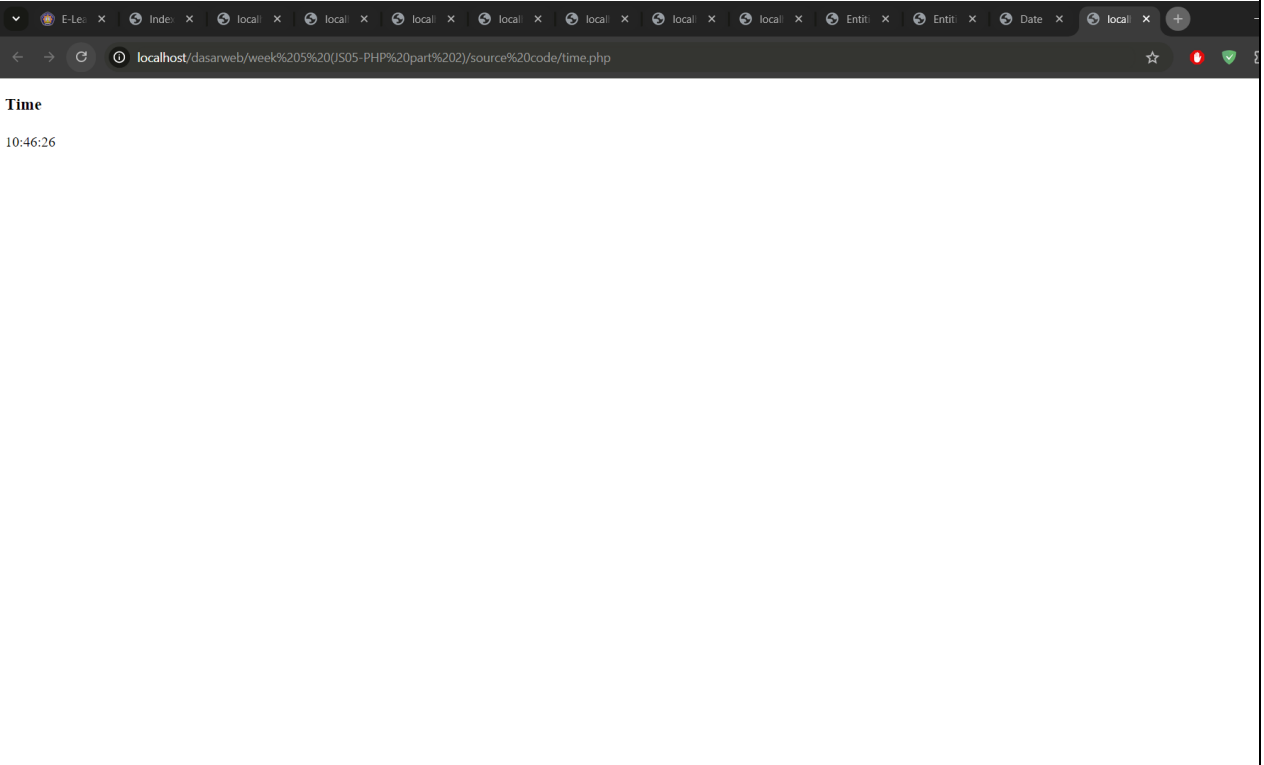
## Practical Section 10. Date

Follow these steps to understand how to use the `date()` function:

Step	Description
1	Create a new file named <code>date.php</code> inside the JS05_PHP-2 directory, then type the following code:

	 <pre> 1  &lt;!DOCTYPE HTML&gt; 2  &lt;html&gt; 3      &lt;head&gt; 4      &lt;/head&gt; 5      &lt;body&gt; 6          &lt;h3&gt; Date &lt;/h3&gt; 7          &lt;?php 8              echo "Today is " . date("Y/m/d") . "&lt;br&gt;"; 9              echo "Today is " . date("Y.m.d") . "&lt;br&gt;"; 10             echo "Today is " . date("Y-m-d") . "&lt;br&gt;"; 11             echo "Today is " . date("l"); 12         ?&gt; 13     &lt;/body&gt; 14 &lt;/html&gt; </pre>
2	Save the file and run the program
3	<p>Observe the output displayed and explain your observations! (Question No 19)</p>  <p>The output displays a HTML page with a title "Date" and a heading "Date", followed by three lines of text that dynamically display the current date using PHP's <b>date()</b> function, with the first line showing the date in the format "Year/Month/Day", the second line showing the date in the format "Year-Month-Day", and the third line showing the day of the week in full (e.g. "Monday", "Tuesday", etc.), demonstrating the use of PHP to generate dynamic content on a web page.</p>
4	Create a new file named <b>time.php</b> inside the JS05_PHP-2 directory, then type the following code:



	 <pre> 1  &lt;!DOCTYPE HTML&gt; 2  &lt;html&gt; 3      &lt;head&gt; 4      &lt;/head&gt; 5      &lt;body&gt; 6          &lt;h3&gt; Time &lt;/h3&gt; 7          &lt;?php 8              date_default_timezone_set("asia/jakarta"); 9              echo date("h:i:sa"); 10         ?&gt; 11      &lt;/body&gt; 12  &lt;/html&gt; </pre>
5	Save the file and run the program
6	<p>Observe the output displayed and explain your observations! (Question No 20)</p>  <p>The output displays a heading "Time" followed by the current time "10:46:26".</p> <ul style="list-style-type: none"> <li>- The output displays a HTML page with a heading "Time", followed by a line of text that dynamically displays the current time in the format "Hour:Minute:Second" in the Asia/Jakarta timezone, demonstrating the use of PHP's <b>date()</b> function and <b>date_default_timezone_set()</b> function to set the timezone and display the current time on a web page.</li> </ul>

## PHP Superglobal Variables

PHP Superglobal Variables are predefined variables in PHP that are accessible from anywhere within a script. They are available globally, meaning they can be used in any function, class, or file without the need to declare them as global. Superglobals are used to handle different types of data, such as form inputs, server details, session information, and more.

Superglobal variables store a lot of important and useful data that we can use to complete the projects we are working on. There are 9 superglobal variables in PHP.

## Practical Section 11. Superglobal Variables

### 1. `$_SERVER`

The first and most important variable is the `$_SERVER` variable. It is an associative array that provides various kinds of information about the request captured by the server. The data includes headers, paths, script locations, and more.

The values stored in the `$_SERVER` variable are provided by the web server, which means there is no specific guarantee that every web server we use will provide all the standard data available.

To find out what values are available in the `$_SERVER` variable, we can execute the following command and save it in `global_server.php`

```
<?php
// Display all available information in the $_SERVER array
echo '<pre>';
print_r($_SERVER);
echo '</pre>';
?>
```

Run the program code above, then explain the output from each echo command.

(Soal no.22)

- The output displays a formatted printout of the `$_SERVER` array, which contains a wealth of information about the server environment, including HTTP headers, server variables, and execution scripts. The `echo '<pre>';` command is used to format the output in a readable manner, and the `print_r($_SERVER);` command prints the entire `$_SERVER` array, displaying key-value pairs of various server variables, such as `HTTP_HOST`, `HTTP_USER_AGENT`, `SERVER_NAME`, `SCRIPT_FILE_NAME`, and many others, providing a comprehensive overview of the server's configuration and the current request.

Here are some examples of data from the `$_SERVER` variable that will often be needed:

No	Variable	Description
1	<code>\$_SERVER['PHP_SELF']</code>	Contains the name of the file currently being executed, taken from the document root.
2	<code>\$_SERVER['SERVER_ADDR']</code>	The IP address of the server where the file is being executed.
3	<code>\$_SERVER['SERVER_NAME']</code>	The hostname of the server where the PHP file is being executed. The hostname is usually the PC's name within the network. If the PHP script is run on a Virtual Host, the virtual host name will be used as the server name.
4	<code>\$_SERVER['SERVER_PROTOCOL']</code>	The communication protocol currently in use, such as HTTP or HTTPS. Example: 'HTTP/0.1'.

No	Variable	Description
5	<code>\$_SERVER['REQUEST_METHOD']</code>	Contains the request method of the PHP file being executed, such as GET, POST, PUT, DELETE, OPTIONS.
6	<code>\$_SERVER['QUERY_STRING']</code>	Returns the query string of the PHP file being executed. For example, if the user accesses <a href="http://localhost/halo-dunia?nama=Budi&amp;umur=20&amp;asal=Surabaya">http://localhost/halo-dunia?nama=Budi&amp;umur=20&amp;asal=Surabaya</a> , this variable will return the value <code>nama=Budi</code> <code>umur=20</code> <code>asal=Surabaya</code>
7	<code>\$_SERVER['DOCUMENT_ROOT']</code>	The document root directory of the PHP file being executed, returned based on server settings.
8	<code>\$_SERVER['HTTP_HOST']</code>	Returns the host content, such as headers (if available).
9	<code>\$_SERVER['HTTP_REFERER']</code>	The URL of the page that referred to the current page being executed. If none, the value is empty.
10	<code>\$_SERVER['HTTP_USER_AGENT']</code>	Contains information about the user making the request, including browser, language, and operating system. Example: Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586).
11	<code>\$_SERVER['REMOTE_ADDR']</code>	The IP address of the user accessing the PHP page.
12	<code>\$_SERVER['SCRIPT_FILENAME']</code>	The absolute path name of the file being executed.
13	<code>\$_SERVER['REQUEST_URI']</code>	The URI of the file being executed. Example: "/php/halo-dunia".

## 2. `$_GET`

The `$_GET` variable is an associative array that contains values from the query string. For example, Create a file `global_get.php`, and write code as follows:

```
<?php
$nama = @$_GET['nama']; //tanda @ agar tidak ada peringatan error
ketika key-nya kosong
$usia = @$_GET['usia']; //tanda @ agar tidak ada peringatan error
ketika key-nya kosong

echo "Halo {$nama}! Apakah benar anda berusia {$usia} tahun?";
?>
```

Run this url on your browser

[localhost/dasarWeb/JS05\\_PHP-2/global\\_get.php?nama=Elok&usia=37](localhost/dasarWeb/JS05_PHP-2/global_get.php?nama=Elok&usia=37)

What output is produced, observe, and explain the result.

(Question No 23)

- The output produced is: "Halo Evan! Apakah anda berusia 20 tahun?". This is because the PHP script retrieves the values of **nama** and **usia** from the URL query string using the `$_GET` superglobal array, and then uses string interpolation to insert these values into the output string. The @ symbol is used to suppress any error warnings that might occur if the **nama** or **usia** keys are not present in the `$_GET` array. In this case, since the URL query string contains

both **nama=Evan** and **usia=20**, the script successfully retrieves these values and displays the personalized greeting message.

### 3. `$_POST`

The `$_POST` variable is similar to the `$_GET` variable. However, the data is **not passed through** the query string in the URL, but rather in the **body of the request**. Additionally, the request method used must be POST.

Create a file `global_post.php`, and write code as follows:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

Run this url on your browser

`localhost/dasarWeb/JS05_PHP-2/global_post.php`

Submit the form and what output is produced? Observe and explain the result.

(Question No 24)

- The output produced is dependent on the user's input. If the user submits the form without entering a name, the output will be "Name is empty". If the user enters a name, the output will be the name itself. For example, if the user enters "Evan", the output will be "Evan".

### 4. `$_SESSION`

The `$_SESSION` variable is an associative array that stores user session data. This variable can be used to store a logged-in user for a specific session. It can also be used to store cart data in an online store. By default, the session lifetime in PHP is 1440 seconds or 24 minutes.

### 5. `$_COOKIE`

Similar to `$_SESSION`, the `$_COOKIE` variable can be used to store data related to the user, such as login information, cart details in an online store, and so on. The difference is that a cookie is a small

file stored in the user's browser. This file is sent every time the browser makes a request to the server. The lifespan of a cookie is generally longer than that of a session.

## 6. `$_REQUEST`

Variabel `$_REQUEST` adalah array asosiatif yang menyimpan gabungan nilai dari variabel `$_GET`, `$_POST`, dan `$_COOKIE` yang kesemuanya berhubungan dengan data yang dikirim bersamaan dengan *request* user.

Create a file `global_request.php`, and write code as follows:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

Run this url on your browser

[localhost/dasarWeb/JS05\\_PHP-2/global\\_request.php](localhost/dasarWeb/JS05_PHP-2/global_request.php)

Submit the form and what output is produced? Observe and explain the result. And what is the difference with the global variable `$_POST`?

(Question No 25)

- If the user enters a name, the output will be the name itself. This is because the PHP script uses the `$_SERVER["REQUEST_METHOD"]` variable to check if the form has been submitted using the POST method, retrieves the value of the `fname` input field using the `$_POST` superglobal array, and checks if the `$name` variable is empty using the `empty()` function. The difference between `$_REQUEST` and `$_POST` is that `$_REQUEST` contains the contents of `$_GET`, `$_POST`, and `$_COOKIE`, whereas `$_POST` only contains the data sent in the request body of an HTTP POST request, making `$_POST` more appropriate for this script since it's submitted using the POST method.

## 7. `$_FILES`

The `$_FILES` variable is an associative array that stores data about files uploaded by the user in a single request using the `POST` or `PUT` method.

## 8. `$_ENV`

The `$ _ENV` variable is an associative array that contains data about the environment in which the PHP script is running. The `$ _ENV` variable is provided by the shell that runs the PHP script, so its values can vary depending on the operating system used.

In modern PHP frameworks like Laravel, the `$ _ENV` variable is also used to store environment-related information, such as the database name, database password, and other values necessary for configuring the framework.

## 9. `$GLOBALS`

The `$GLOBALS` variable is an associative array that stores all global variables defined when the program is running. The `$GLOBALS` variable is a PHP superglobal used to access global variables from anywhere within a PHP script (including inside functions or methods).

Create a file `global_globals.php`, and write code as follows:

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

Run this url on your browser

[localhost/dasarWeb/JS05\\_PHP-2/global\\_globals.php](localhost/dasarWeb/JS05_PHP-2/global_globals.php)

Submit the form and what output is produced? Observe and explain the result.

(Question No 26)

- The output produced is 100, because the PHP script defines two variables `$x` and `$y` with values 75 and 25, respectively, and the `addition()` function uses the `$GLOBALS` superglobal array to access and modify the global variables `$x` and `$y` within the function scope, calculating the sum of `$x` and `$y` and assigning it to the global variable `$z`, which is then outputted using the `echo` statement.

## References:

- 1) Nixon, Robin. (2018). Learning PHP, MySQL, JavaScript, CSS & HTML: A Step-by-step Guide to Creating Dynamic Websites, 5<sup>th</sup> Edition. O'Reilly Media, Inc.
- 2) Forbes, Alan. (2012). The Joy of PHP: A Beginners's Guide to Programming Interactive Web Applications with PHP and MySQL, 5<sup>th</sup> Edition. Plum Island Publishing