User Defined and Built-in Functions in PHP

Introduction

In PHP, functions are blocks of code that perform a specific task. There are two types of functions in PHP: built-in functions and user-defined functions. Built-in functions are pre-defined functions that come with PHP and can be used in any PHP script without the need for additional code. On the other hand, user-defined functions are created by the programmer to perform specific tasks that may not be available in the built-in functions. In this lesson, we will explore both built-in and user-defined functions in PHP and learn how to create, call and use them in our programs.

Objectives of the Study

The objectives of studying User Defined and Built-in Functions in PHP are:

1. To understand the concept of functions in PHP and how they can be used to group and organize code for better readability and reusability.
2. To learn about the different types of functions in PHP, including built-in functions provided by PHP and user-defined functions created by developers.
3. To develop the skills needed to create and use user-defined functions in PHP, including passing arguments and returning values.
4. To explore the wide range of built-in functions available in PHP, including functions for working with arrays, strings, dates and times, and more.
5. To be able to identify appropriate built-in functions for specific programming tasks, and to be able to use them effectively in PHP code.

Lesson Proper

What is a function in PHP?

In PHP, a function is a block of code that can be called and executed multiple times in a program. Functions help in organizing the code and make it more modular, reusable, and easier to understand. PHP has many built-in functions that perform various tasks, such as manipulating strings, performing mathematical operations, and working with arrays. Additionally, PHP allows users to define their own functions that can be used throughout their programs.

A function in PHP typically consists of the following parts:

- Function name: This is the name that is used to call the function.
- Parameters: These are optional values that can be passed to the function.
- Function body: This is the code that is executed when the function is called.

- Return value: This is the value that the function returns when it is finished executing. It can be any data type, including arrays and objects.
- Function scope: This refers to the accessibility of the function within the script. Functions can be defined to be either global or local in scope.

Syntax:

```
function functionName($param1, $param2, ...)
{
 // Function code goes here
 return $result; // Optional
}
```

In this syntax:

- function is the keyword that tells PHP that you are defining a function.
- functionName is the name of the function. You can name it whatever you like, as long as it follows the same rules as variable names in PHP.
- $param1, $param2, etc. are the parameters (or arguments) that the function accepts. These are optional, and you can have as many or as few as you need. If you have more than one parameter, separate them with commas.
- The curly braces {} contain the code that the function executes when it is called.
- return is a keyword that specifies what the function should return when it is called. This is optional, and if you omit it, the function will not return anything. If you do include a return statement, you can return a value or a variable.

How to call a function?

To call a function in PHP, you simply need to write the name of the function followed by parentheses. If the function takes any arguments, you can pass them inside the parentheses. Here is an example:

```php
<?php
function display_message($message)
{
    echo "$message";
}
Display_message ("Hello iNetTutor!"); // Output: Hello iNetTutor!
?>
```

The PHP code above defines a function called "display_message" that takes a single parameter called "$message". When this function is called and passed an argument, it simply outputs that argument using the "echo" statement.

In the second line, the function is called with the argument "Hello iNetTutor!", so the output of the function call is "Hello iNetTutor!".

Naming Convention

Here are some guidelines for naming conventions when creating functions in PHP:

1. Use only alphanumeric characters and underscores in function names. Avoid using special characters, spaces, and hyphens.
   Example: calculate_area()
2. Function names should be descriptive of what the function does. Use a verb-noun combination that describes the function's action and result.
   Example: calculate_area()
3. Use camelCase to separate words in function names. The first word should start with a lowercase letter, while subsequent words should start with uppercase letters.
   Example: calculateArea()
4. Avoid using reserved keywords or function names that are already defined in PHP. This can lead to conflicts and unexpected results.
   Example: do not name your function "echo" or "print".
5. Be consistent with naming conventions throughout your code.
   Example: If you use camelCase for function names, stick with that convention for all your functions.
6. Use underscores to separate words in a function name if you are following the snake_case naming convention.
   Example: calculate_area()
7. Here's an example of a function name that follows these guidelines: calculateArea() or calculate_area()

Code example

```
1. <?php
2. function display_message($message)
3. {
4.     echo "$message";
5. }
6. display_message ("Hello iNetTutor!"); // Output: Hello iNetTutor!
7. display_message ("Hello World!"); // Output: Hello World!
8. display_message ("Happy Birthday!"); // Output: Happy Birthday!
9. ?>
```

The given code defines a function named display_message that accepts one parameter $message. This function simply outputs the value of $message using echo statement. The function is then called three times with different string arguments passed to it as $message parameter, which results in displaying

the message passed as an argument to the function. So, the output of the code is three lines of text, each containing the string message passed to the function separately.

Another Example

```
1.  <!DOCTYPE html>
2.  <html>
3.  <body>
4.
5.  <form action="" method="post">
6.    Enter your message: <input type="text" name="message" required>
7.    <input type="submit">
8.  </form>
9.
10. <?php
11.
12. function display_message($message)
13. {
14.     echo "$message";
15. }
16.
17. if ($_SERVER["REQUEST_METHOD"] == "POST") {
18.     $user_message = $_POST["message"];
19.     display_message($user_message);
20. }
21. ?>
22.
23. </body>
24. </html>
```

Line 5-8

This code is an HTML form that prompts the user to enter a message in a text input field. The form uses the POST method to submit the data to the server, and the action attribute is set to an empty string, which means that the form will be submitted to the current page. The input element has a required attribute, which means that the user must enter a value in the field before submitting the form. Finally, there is a submit button that the user can click to submit the form.

Line 10-21

This is the scope of the PHP script embedded in the HTML file.

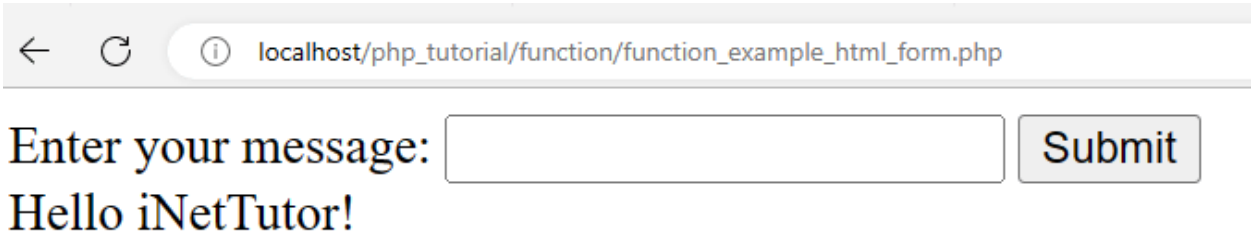
Line 12-15

This code defines a PHP function named display_message. The function takes one parameter called $message, which is used to specify the message to be displayed. The function uses the echo statement to display the value of the $message parameter. The $message parameter can be any string value, and it will be displayed as is when the function is called.

Line 17-20

This code checks if the form has been submitted using the HTTP POST method. If the form has been submitted, it retrieves the value of the "message" input field using the $_POST superglobal array and assigns it to the $user_message variable. The display_message() function is then called with the $user_message variable as its argument. This function displays the value of the $user_message variable on the webpage.

Output:



Built-in Functions in PHP

PHP provides a vast library of built-in functions that can be used to perform various tasks, such as manipulating strings, performing mathematical operations, working with arrays, and handling dates and times, among others. These functions are available for use in any PHP script and do not require any special installation or configuration.

Some example of built-in functions in PHP:

- String Functions: PHP provides a rich set of functions for working with strings, such as strlen(), substr(), strpos(), str_replace(), and strtolower(), among others.
- Mathematical Functions: PHP includes a number of mathematical functions, such as round(), sqrt(), rand(), and ceil(), among others, that can be used to perform various mathematical operations.
- Array Functions: PHP provides a wide range of functions for working with arrays, such as array(), array_push(), array_pop(), array_search(), and array_reverse(), among others.
- Date and Time Functions: PHP includes a number of functions for working with dates and times, such as date(), time(), strtotime(), and mktime(), among others.
- File System Functions: PHP also provides a set of functions for working with files and directories, such as file_get_contents(), file_put_contents(), mkdir(), and rmdir(), among others.

By using these built-in functions, developers can save a lot of time and effort in coding, as they do not need to write code from scratch for performing various tasks. Moreover, these functions have been tested and optimized for performance and are therefore reliable and efficient.

Summary

The PHP programming language provides developers with a wide range of built-in functions, allowing them to perform various tasks without having to write extensive code. Additionally, developers can create their own functions, which can be reused multiple times throughout their codebase. The blog post and lesson on User Defined and Built-in Functions in PHP cover the basics of functions, including their definition, syntax, and how to call them. It also includes guidelines on naming conventions, passing parameters, and returning values. Furthermore, the lesson shows examples of using functions with HTML forms, including data sanitization to improve the security of the application. By understanding the fundamentals of functions in PHP, developers can write more efficient and maintainable code.

Meta Description

Learn about user-defined and built-in functions in PHP with this guide. Understand the basics of creating and calling functions, as well as common built-in functions in PHP. Get code examples and best practices to improve your skills in PHP programming.