php

intro to php

PHP (Hypertext Preprocessor) is a popular programming language used for web development. It is a server-side language, which means that it is executed on the server and the results are sent to the client (usually a web browser) in the form of HTML, CSS, and JavaScript.

One of the main advantages of PHP is that it is easy to learn and use, yet it is powerful enough to build large and complex web applications. It is also open-source and has a large community of developers contributing to its development and evolution.

To start using PHP, you will need to set up a web server with PHP installed. There are many options for doing this, such as installing a local server on your own computer, or using a cloud hosting provider. Once you have a server set up, you can create PHP files with a `.php` extension and use the `<?php` and `?>` tags to enclose your PHP code.

Here is a simple example of a PHP script that prints "Hello, World!" to the browser:

To execute this script, you would save it as a `.php` file (e.g. `hello.php`) and access it through your web server. The server would execute the PHP code and return the resulting HTML to the client, which would display the message "Hello, World!" in the browser.

There are many more features and capabilities of PHP, including support for working with databases, handling forms and user input, and creating dynamic content. To learn more about PHP, you can refer to online tutorials and resources, or take a course to get a more comprehensive understanding of the language.

What these tags specify in PHP : Can we run PHP from HTML file? If yes, how?

In PHP, the `<?php` and `?>` tags are used to enclose blocks of PHP code. The PHP interpreter will execute the code between these tags as PHP code.

Yes, it is possible to run PHP code from an HTML file. To do this, you will need to save the file with an `.php` extension rather than `.html`. Then, you can use the `<?php` and `?>` tags to enclose your PHP code within the HTML file, just as you would in a pure PHP file.

For example, consider the following HTML file that contains some PHP code:

To run this file, you would save it with a `.php` extension (e.g. `index.php`) and then access it through a web server that is configured to run PHP (such as Apache or Nginx with the PHP module installed). When the server receives a request for this file, it will execute the PHP code and return the resulting HTML to the client.

What is cookie and session tracking, how are they handled in PHP

In PHP, a cookie is a small piece of data that is stored on the client's computer by the web browser. It can be used to store information such as a user's preferences or login information. Cookies are sent back to the server with every request the client makes to the server, allowing the server to identify the client and maintain state.

A session is a way for a web server to store information about a user's interaction with a website. When a user visits a website, the web server creates a new session for the user and assigns a unique session ID to the user. This session ID is typically stored in a cookie on the user's computer, allowing the server to identify the user's session when the user makes subsequent requests to the server.

In PHP, cookies and sessions can be used to store and retrieve data. To set a cookie in PHP, you can use the `setcookie` function. For example:

```
setcookie('username', 'John', time() + 86400);
```

This will set a cookie with the name 'username' and the value 'John', and it will expire after 86400 seconds (24 hours).

To retrieve the value of a cookie in PHP, you can use the `$_COOKIE` superglobal array. For example:

```
echo $_COOKIE['username'];
```

To start a session in PHP, you can use the `session_start` function. For example:

`session_start();`

This will start a new session or resume an existing session. To store data in a session, you can use the `$_SESSION` superglobal array. For example:

`$_SESSION['username'] = 'John';`

This will store the value 'John' in the 'username' variable of the current session. To retrieve the value of a session variable in PHP, you can use the `$_SESSION` superglobal array. For example:

`echo $_SESSION['username'];`

Why PHP is known as scripting language? Write a program in PHP to calculate Square Root of a number?

PHP is known as a scripting language because it is typically used to write scripts that are executed on the server side of a web application. These scripts can be used to perform tasks such as generating dynamic web pages, accessing databases, and handling user input.

To calculate the square root of a number in PHP, you can use the `sqrt` function. Here is an example of a PHP script that calculates the square root of a number:

`<?php
// Get the number from the user
$number = \$_{P}OST['number']; //Calculatethesquarerootofthenumber\$square_{r}oot = sqrt($
number);

// Print the result
echo "The square root of $number is $square_root";
?>`

This script assumes that the user has submitted a form with a field named 'number', and it calculates the square root of the value entered by the user. The result is then printed to the screen.

Explain file operation in PHP.
In PHP, you can perform a variety of file operations such as reading, writing, and deleting files. Here are some common file operations in PHP:

1. Reading a file: You can use the `fopen` function to open a file, and then use the `fread` function to read the contents of the file. For example:

$handle = fopen(''file.txt'',''r'')$;contents = fread(
$handle, filesize(''file.txt'')); fclose$(handle);
echo $contents;

This code will open the file "file.txt", read the contents of the file, and then close the file. The contents of the file will then be printed to the screen.

2. Writing to a file: You can use the `fopen` function to open a file for writing, and then use the `fwrite` function to write to the file. For example:

$handle = fopen(''file.txt'',''w''); fwrite$(handle, "Hello, world!");
fclose($handle);

This code will open the file "file.txt" for writing, write the string "Hello, world!" to the file, and then close the file.

3. Appending to a file: You can use the `fopen` function to open a file for appending, and then use the `fwrite` function to write to the end of the file. For example:

$handle = fopen(''file.txt'',''a''); fwrite$(handle, "Hello, world!");
fclose($handle);

This code will open the file "file.txt" for appending, write the string "Hello, world!" to the end of the file, and then close the file.

4. Deleting a file: You can use the `unlink` function to delete a file. For example:

`unlink("file.txt");`

This code will delete the file "file.txt".

How can we create links in PHP pages?

The `link()` function in PHP is used to create a hard link to a file. A hard link is a reference to a file that is stored on the file system. When you create a hard link, you are creating another name for the same file. Any changes made to the file through

one link will be reflected in the other link, as they both refer to the same underlying file.
PHP link() Function
Example

Create a hard link:

To create a link in a PHP page, you can use the `a` (anchor) HTML element and specify the `href` attribute with the desired link destination. Here is an example of creating a link in a PHP page:

```
echo '<a href="http://www.example.com">Click here</a>';
```

This will create a link that says "Click here" and links to the URL "http://www.example.com".

What is the function of “foreach” construct in PHP?

The `foreach` construct in PHP is a looping structure that allows you to iterate over elements in an array. It is generally used to perform a certain action on each element of the array.

Here is the syntax for the `foreach` construct:
foreach ($array as $value) {
// code to be executed
}

The `foreach` construct loops through each element in the array and assigns the value of the element to the variable `$value`. You can then use the `$value` variable inside the loop to perform some action on each element of the array.

Here is an example of using the `foreach` construct to print the elements of an array:
$array = array(1, 2, 3, 4, 5);

foreach ($array as $value) {
echo $value . ' ';
}

// Output: 1 2 3 4 5

You can also use the `foreach` construct with an associative array, in which case the loop will assign both the key and the value of each element to variables. The syntax for this is as follows:

`foreach ($array as $key => $value){ // code to be executed }`

How do you create and access an array? Describe the various functions for dealing with arrays with example in PHP

To create an array in PHP, you can use the `array` function or use the short array syntax introduced in PHP 5.4. Here are some examples of creating arrays in PHP:

// Using the array function
$array = array(1, 2, 3, 4, 5);

// Using the short array syntax
$array = [1, 2, 3, 4, 5];

You can also create an associative array, which is an array with string keys rather than numeric keys. Here is an example of creating an associative array:

// Using the array function
$array = array('a' => 1, 'b' => 2, 'c' => 3);

// Using the short array syntax
$array = ['a' => 1, 'b' => 2, 'c' => 3];

To access an element of an array, you can use the square brackets notation and specify the index of the element you want to access. For example:

$array = [1, 2, 3, 4, 5];

echo $array[0]; // Output: 1
echo $array[3]; // Output: 4

For an associative array, you can use the keys to access the elements:
$array = ['a' => 1, 'b' => 2, 'c' => 3];

echo $array['a']; // Output: 1
echo $array['c']; // Output: 3

PHP provides a number of functions for dealing with arrays. Here are some examples of common array functions in PHP:

1. `count`: This function returns the number of elements in an array. For example:

```
$array = [1, 2, 3, 4, 5];

echo count($array);
// Output: 5
```

2. `sort`: This function sorts the elements of an array in ascending order. For example:

```
$array = [3, 1, 5, 4, 2];

sort($array);

print_r($array);
// Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
```

3. `array_reverse`: This function reverses the order of the elements in an array. For example:

```
$array = [1, 2, 3, 4, 5];

$ reversed_array = array_reverse($array);

print_r($reversed_array);

// Output: Array ( [0] => 5 [1] => 4 [2] => 3 [3] => 2 [4] => 1 )
```

4. `array_unique`: This function removes duplicate values from an array. For example:

```
$array = [1, 2, 3, 2, 4, 5, 5];

$ unique_array = array_unique($array);

print_r($unique_array);

// Output: Array ( [0] => 1 [1] => 2 [2] => 3 [4] => 4 [5] => 5 )
```

Explain the form handling in PHP for login form to check it for successful login or not

To handle a login form in PHP, you will need to do the following:

1. Create a form with a username field and a password field. You can use the `input` HTML element to create these fields.
2. Add a submit button to the form so that the user can submit the form. You can use the `button` HTML element for this.
3. When the form is submitted, the PHP script that handles the form submission should check the entered username and password against a list of valid credentials (such as a database of user accounts).
4. If the entered credentials are valid, the script should redirect the user to a protected page (such as a dashboard or homepage).
5. If the entered credentials are not valid, the script should display an error message and allow the user to try again.

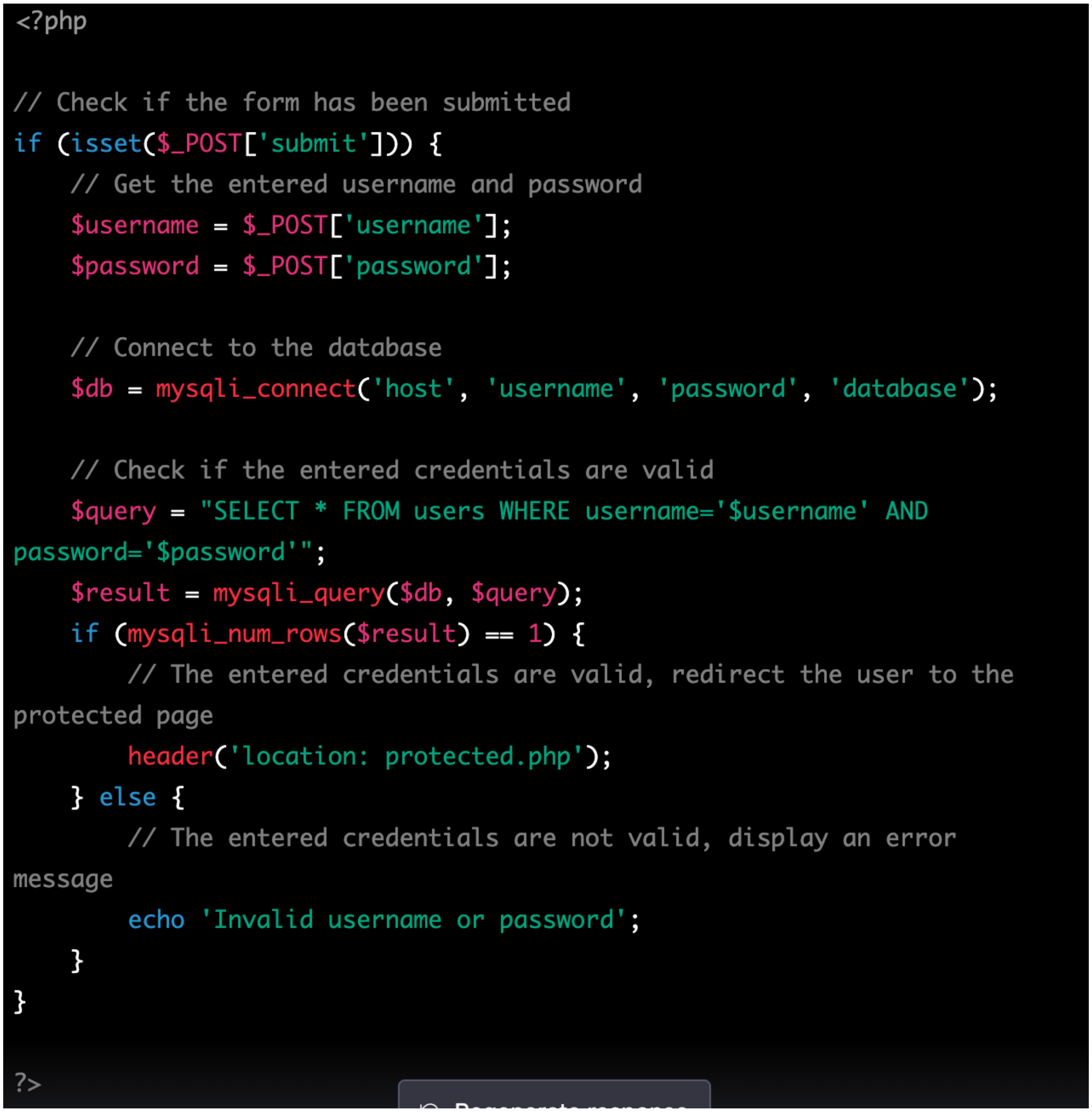
Here is an example of a PHP script that handles a login form:
`<?php
// Check if the form has been submitted
if (isset(_POST['submit'])) { // Get the entered username and password username =
$_POST['username']$;password = $_POST['password'];

// Connect to the database
$db = mysqli_connect('host', 'username', 'password', 'database');

// Check if the entered credentials are valid
$query = "SELECT * FROM users WHERE username = '$'username' AND password='
$password'"$;result = mysqli_query($db, \$query); if(mysqli_num_rows$(result) == 1) {
// The entered credentials are valid, redirect the user to the protected page

header('location: protected.php');
}
else {
// The entered credentials are not valid, display an error message
echo 'Invalid username or password';
}
}

?>`



```
<?php

// Check if the form has been submitted
if (isset($_POST['submit'])) {
    // Get the entered username and password
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Connect to the database
    $db = mysqli_connect('host', 'username', 'password', 'database');

    // Check if the entered credentials are valid
    $query = "SELECT * FROM users WHERE username='$username' AND
password='$password'";
    $result = mysqli_query($db, $query);
    if (mysqli_num_rows($result) == 1) {
        // The entered credentials are valid, redirect the user to the
protected page
        header('location: protected.php');
    } else {
        // The entered credentials are not valid, display an error
message
        echo 'Invalid username or password';
    }
}

?>
```

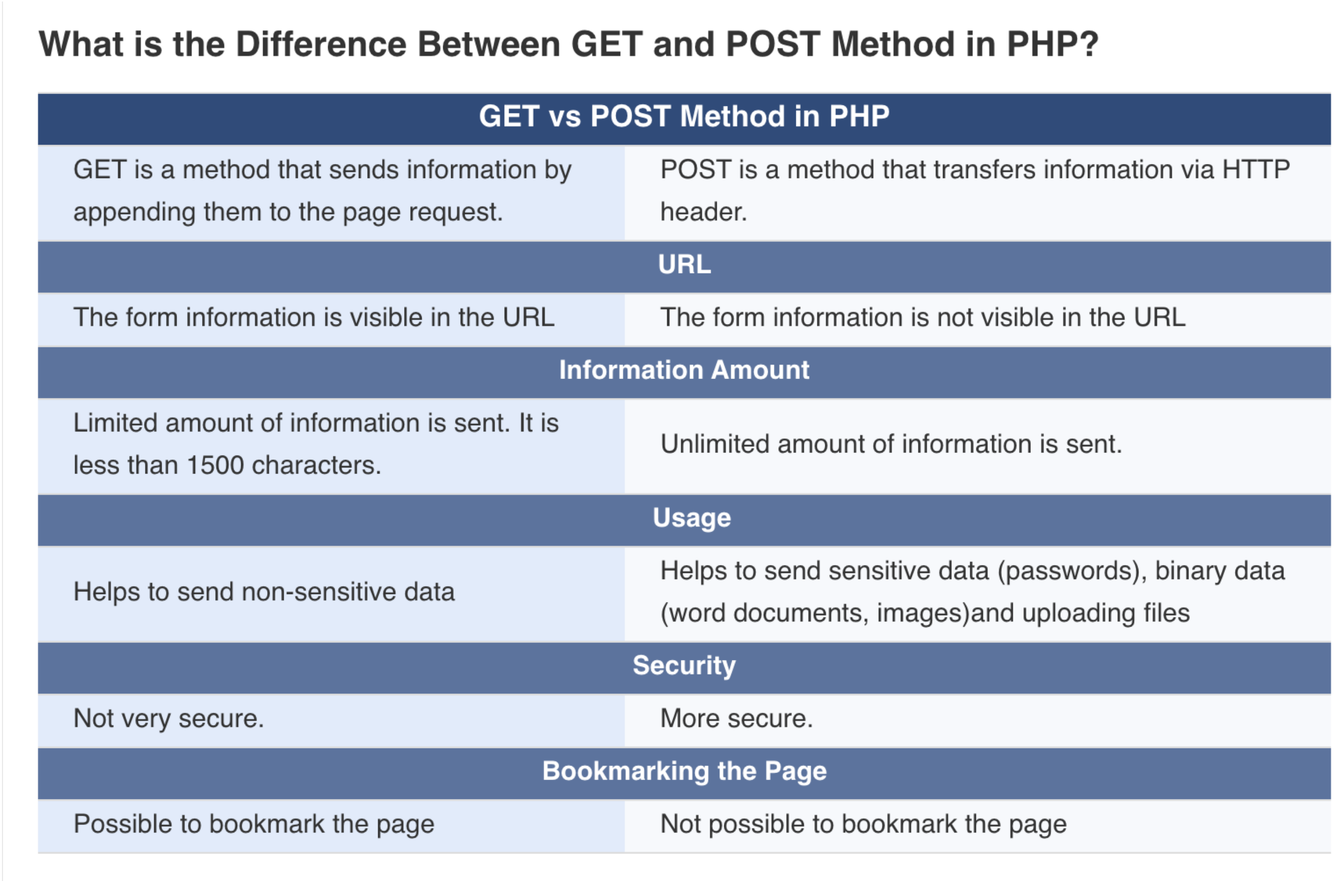
Username:

Password:

Submit

This script first checks if the form has been submitted using the `isset` function. If the form has been submitted, it gets the entered username and password and checks if they match any records in the `users` table of the database. If there is a match, the user is redirected to the protected page. If there is no match, an error message is displayed.

Differences between GET and POST methods? in php



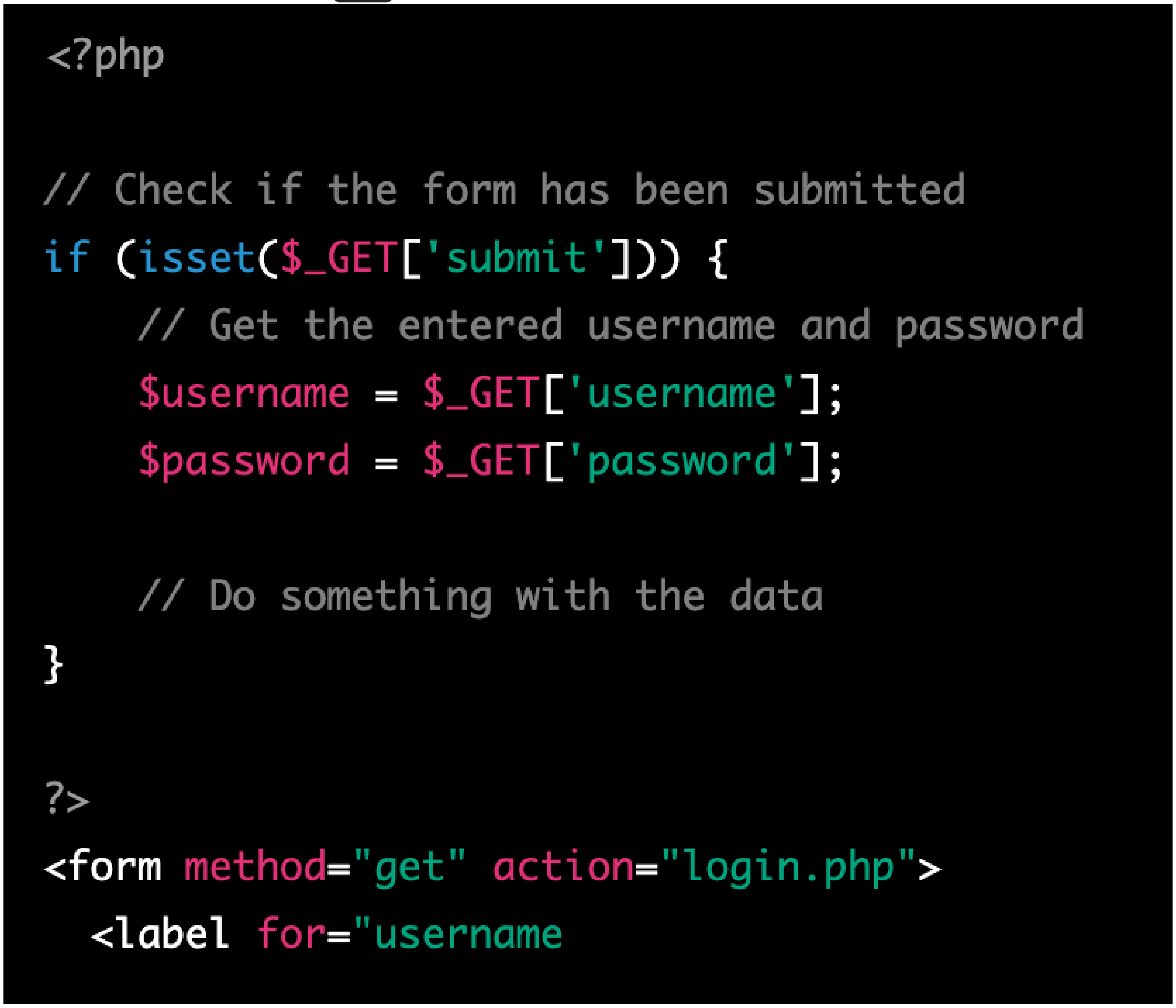
What is the Difference Between GET and POST Method in PHP?

GET vs POST Method in PHP	
GET is a method that sends information by appending them to the page request.	POST is a method that transfers information via HTTP header.
URL	
The form information is visible in the URL	The form information is not visible in the URL
Information Amount	
Limited amount of information is sent. It is less than 1500 characters.	Unlimited amount of information is sent.
Usage	
Helps to send non-sensitive data	Helps to send sensitive data (passwords), binary data (word documents, images)and uploading files
Security	
Not very secure.	More secure.
Bookmarking the Page	
Possible to bookmark the page	Not possible to bookmark the page

In PHP, the `GET` and `POST` methods are used to submit data to a server. There are several differences between these two methods:

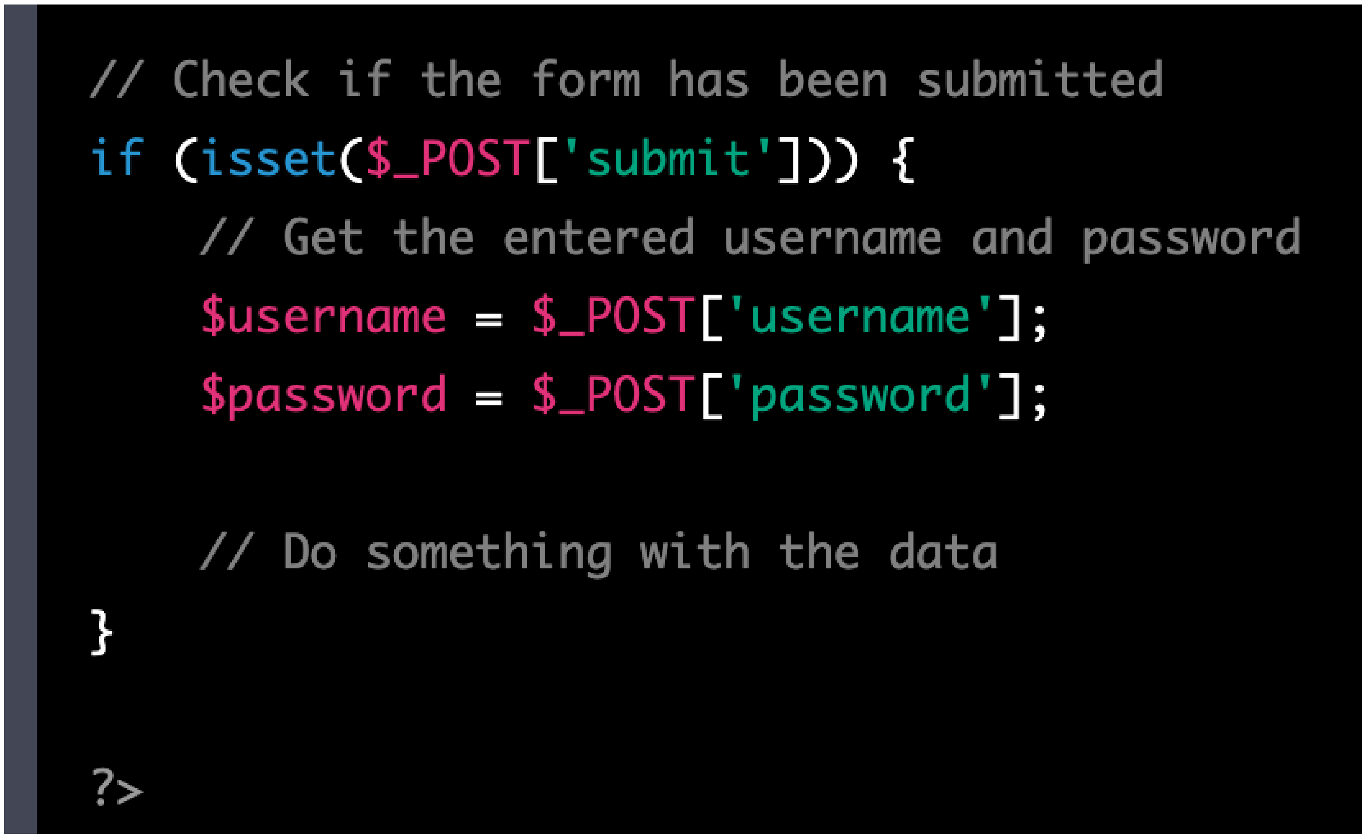
1. `GET` requests are sent through the URL of the request, while `POST` requests are sent in the body of the request.
2. `GET` requests are visible in the URL, while `POST` requests are not visible in the URL.
3. `GET` requests can be cached, while `POST` requests cannot be cached.
4. `GET` requests have a maximum size limit, while `POST` requests do not have a maximum size limit.
5. `GET` requests are typically used to retrieve data, while `POST` requests are typically used to submit data.

Here is an example of a `GET` request in PHP:



```
<?php

// Check if the form has been submitted
if (isset($_GET['submit'])) {
    // Get the entered username and password
    $username = $_GET['username'];
    $password = $_GET['password'];

    // Do something with the data
}

?>
<form method="get" action="login.php">
  <label for="username
```

post method



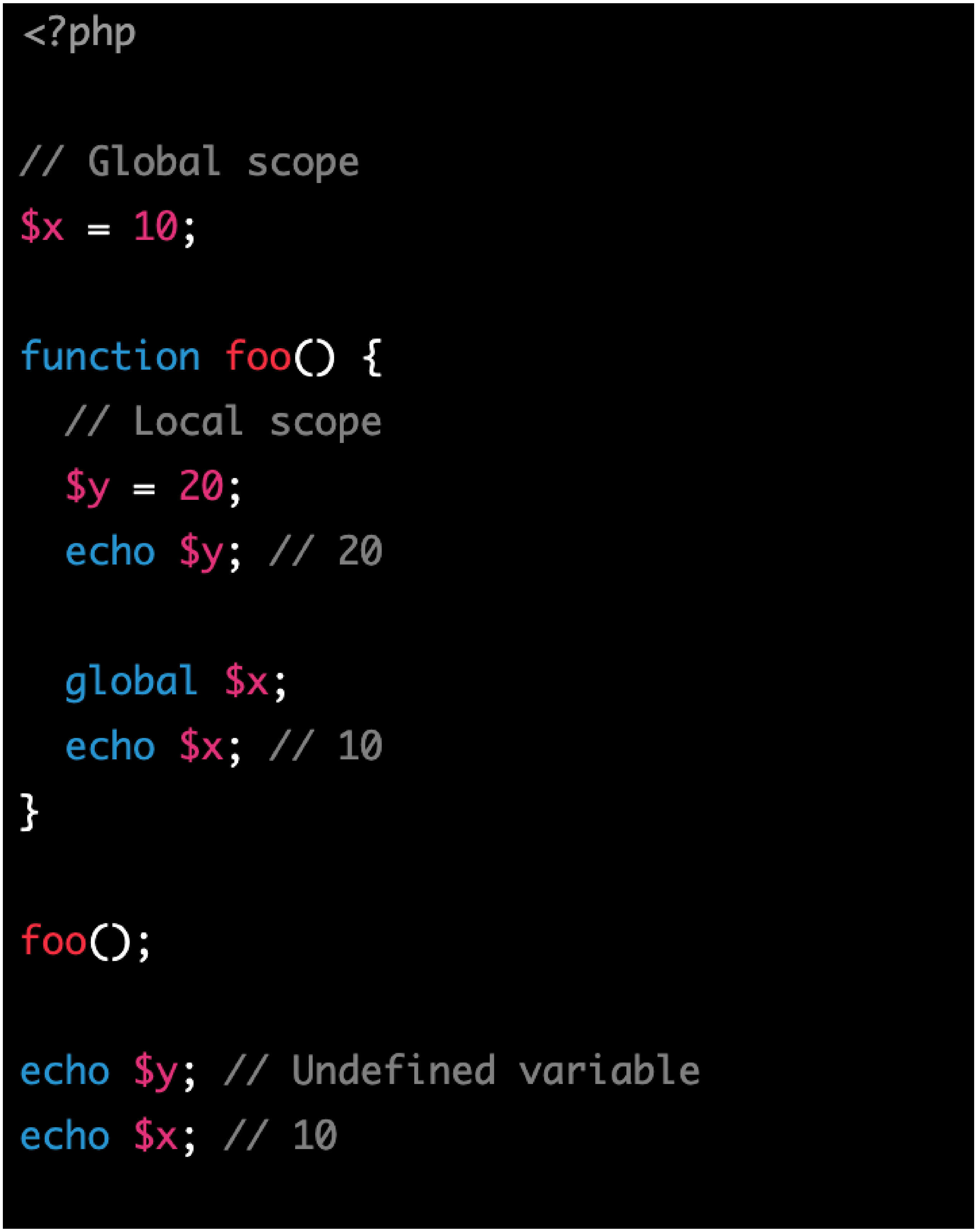
```php
// Check if the form has been submitted
if (isset($_POST['submit'])) {
    // Get the entered username and password
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Do something with the data
}

?>
```

Explain all possible types of scope of variable? in php

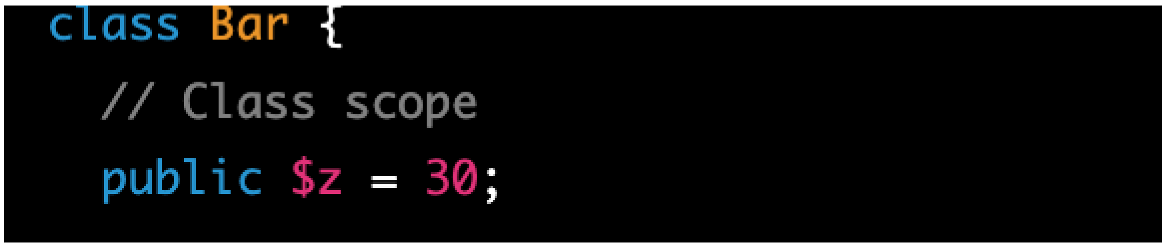
In PHP, there are four possible types of scope for variables:

1. Local scope: A variable declared within a function is local to that function and can only be accessed within that function.
2. Global scope: A variable declared outside of any function, or a variable declared with the `global` keyword within a function, is available globally and can be accessed from anywhere in the script.
3. Static scope: A variable declared with the `static` keyword within a function is static, meaning it retains its value between function calls.
4. Class scope: A variable declared within a class, but outside of any method, is a class property and is available to all methods of that class. A class property can also be declared with the `static` keyword, in which case it is shared among all instances of the class.

Here is an example that demonstrates the different scopes:



```php
<?php

// Global scope
$x = 10;

function foo() {
  // Local scope
  $y = 20;
  echo $y; // 20

  global $x;
  echo $x; // 10
}

foo();

echo $y; // Undefined variable
echo $x; // 10
```



```php
class Bar {
  // Class scope
  public $z = 30;
  public function baz() {
    // Local scope
    $w = 40;
    echo $w; // 40

    echo $this->z; // 30
  }

  public static function qux() {
    // Static scope
    static $v = 50;
    $v++;
    echo $v; // 51
  }
}

$bar = new Bar();
$bar->baz();

Bar::qux();
Bar::qux();

echo $w; // Undefined variable
echo $v; // Undefined variable
echo $bar->z; // 30
echo Bar::$z; // Undefined variable
```

Explain with example the use of substring function in php?

The `substr()` function in PHP is used to extract a portion of a string. It takes three arguments: the string, the starting position, and the length of the substring to be extracted.

Here is an example of how to use the `substr()` function:

```
<?php

$string = 'abcdefghijklmnopqrstuvwxyz';

// Extract the substring starting at position 3 with a length of 5
characters
$substring = substr($string, 3, 5);

echo $substring; // 'defgh'
```

Note that the `substr()` function is 0-indexed, meaning that the first character in the string is at position 0.

Write a program in HTML (Web-Page) having a form with firstname, lastname and a submit button. On clicking the button form validation should work (Use Javascript for that) and data entry should be saved in a database (PHP code needed, for insertion of data from "form" to database). Form should be styled using CSS. (Use any mode- inline, internal or external)?

First Name:

Last Name:

Submit

The form's action attribute specifies the PHP script that will process the form data. Here is an example of a PHP script that inserts the form data into a database:
php

// Connect to the database
$servername = ''localhost''$;username = "username";
$password = ''password''$;dbname = "myDB";

$conn = new mysqli($servername, $username, \$password, \$dbname); if($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}

// Get the form data
$firstname = \$_POST['firstname']$;lastname = $_POST['lastname'];

// Insert the data into the database
$sql = ''INSERT INTO users (firstname, lastname) VALUES ('$firstname', '
$lastname')''; if($conn->query($sql) === TRUE) {
echo "New record created successfully";
} else {
echo "Error: " . $sql . "
" . $conn->error;
}

$conn->close();

What do you understand about Numeric array, Associative array and Multidimensional array?

In PHP, an array is a data structure that stores a collection of values. There are three types of arrays in PHP: numeric arrays, associative arrays, and multidimensional arrays.

1. Numeric array: A numeric array is an array with a numerical index. The index starts at 0 and increments by 1 for each element. For example:
   $numeric_array = [1, 2, 3, 4, 5];

```
echo $numeric_array[0]; // 1
echo $numeric_array[1]; // 2
echo $numeric_array[2]; // 3
```

2. Associative array: An associative array is an array with string keys instead of numerical indices. For example:
   ```
   $associative_array = ['a' => 1, 'b' => 2, 'c' => 3];
   ```

```
echo $associative_array['a']; // 1
echo $associative_array['b']; // 2
echo $associative_array['c']; // 3
```

3. Multidimensional array: A multidimensional array is an array containing one or more arrays. For example:
   ```
   $multidimensional_array = [
   [1, 2, 3],
   [4, 5, 6],
   [7, 8, 9]
   ];
   ```

```
echo $multidimensional_array[0][0]; // 1
echo $multidimensional_array[1][1]; // 5
echo $multidimensional_array[2][2]; // 9
```