What is PHP?

* PHP is an acronym for "PHP: Hypertext Preprocessor"
* PHP is a widely-used, open source scripting language
* PHP scripts are executed on the server
* PHP is free to download and use

**PHP is an amazing and popular language!**

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!  
It is deep enough to run the largest social network (Facebook)!  
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code
* PHP code is executed on the server, and the result is returned to the browser as plain HTML
* PHP files have extension ".php"

What Can PHP Do?

* PHP can generate dynamic page content
* PHP can create, open, read, write, delete, and close files on the server
* PHP can collect form data
* PHP can send and receive cookies
* PHP can add, delete, modify data in your database
* PHP can be used to control user-access
* PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

* PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* PHP is compatible with almost all servers used today (Apache, IIS, etc.)
* PHP supports a wide range of databases
* PHP is free. Download it from the official PHP resource: [www.php.net](http://www.php.net/)
* PHP is easy to learn and runs efficiently on the server side

Variables are "containers" for storing information.

## Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

### **Example**

<?php  
$txt = "Hello world!";  
$x = 5;  
$y = 10.5;  
?>

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for PHP variables:

* A variable starts with the $ sign, followed by the name of the variable
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive ($age and $AGE are two different variables)

## PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

You will learn more about strict and non-strict requirements, and data type declarations in the [PHP Functions](https://www.w3schools.com/php/php_functions.asp) chapter.

## PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

* local
* global
* static

## Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

### **Example**

Variable with global scope:

<?php  
$x = 5; // global scope  
  
function myTest() {  
    // using x inside this function will generate an error  
    echo "<p>Variable x inside function is: $x</p>";  
}   
myTest();  
  
echo "<p>Variable x outside function is: $x</p>";  
?>

## PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

### **Example**

<?php  
function myTest() {  
    static $x = 0;  
    echo $x;  
    $x++;  
}  
  
myTest();  
myTest();  
myTest();  
?>

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

**Note:** The variable is still local to the function.

# **PHP echo and print Statements**

With PHP, there are two basic ways to get output: echo and print.

In this tutorial we use echo or print in almost every example. So, this chapter contains a little more info about those two output statements.

## PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

## The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

**Display Text**

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

### **Example**

<?php  
echo "<h2>PHP is Fun!</h2>";  
echo "Hello world!<br>";  
echo "I'm about to learn PHP!<br>";  
echo "This ", "string ", "was ", "made ", "with multiple parameters.";  
?>

**Display Variables**

The following example shows how to output text and variables with the echo statement:

### **Example**

<?php  
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
$x = 5;  
$y = 4;  
  
echo "<h2>" . $txt1 . "</h2>";  
echo "Study PHP at " . $txt2 . "<br>";  
echo $x + $y;  
?>

## PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource

## PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

### **Example**

<?php   
$x = "Hello world!";  
$y = 'Hello world!';  
  
echo $x;  
echo "<br>";   
echo $y;  
?>

## PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

* An integer must have at least one digit
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var\_dump() function returns the data type and value:

### **Example**

<?php   
$x = 5985;  
var\_dump($x);  
?>

## PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var\_dump() function returns the data type and value:

### **Example**

<?php   
$x = 10.365;  
var\_dump($x);  
?>

## PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

$x = true;  
$y = false;

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

## PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var\_dump() function returns the data type and value:

### **Example**

<?php   
$cars = array("Volvo","BMW","Toyota");  
var\_dump($cars);  
?>

You will learn a lot more about arrays in later chapters of this tutorial.

## PHP Object

An object is a data type which stores data and information on how to process that data.

In PHP, an object must be explicitly declared.

First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

### **Example**

<?php  
class Car {  
    function Car() {  
        $this->model = "VW";  
    }  
}  
  
// create an object  
$herbie = new Car();  
  
// show object properties  
echo $herbie->model;  
?>

## PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

**Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

### **Example**

<?php  
$x = "Hello world!";  
$x = null;  
var\_dump($x);  
?>

## PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

* if statement - executes some code if one condition is true
* if...else statement - executes some code if a condition is true and another code if that condition is false
* if...elseif...else statement - executes different codes for more than two conditions
* switch statement - selects one of many blocks of code to be executed

## PHP - The if Statement

The if statement executes some code if one condition is true.

### **Syntax**

if (*condition*) { *code to be executed if condition is true*;  
}

### **Example**

Output "Have a good day!" if the current time (HOUR) is less than 20:

<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
}  
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_if)

## PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

### **Syntax**

if (*condition*) {  
    *code to be executed if condition is true;*  
} else {  
  *code to be executed if condition is false;*}

### **Example**

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>

## PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

### **Syntax**

if (*condition*) {  
    *code to be executed if this condition is true;*} elseif (*condition*) {  
  *code to be executed if first condition is false and this condition is true;*} else {  
    *code to be executed if all conditions are false;*}

### **Example**

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

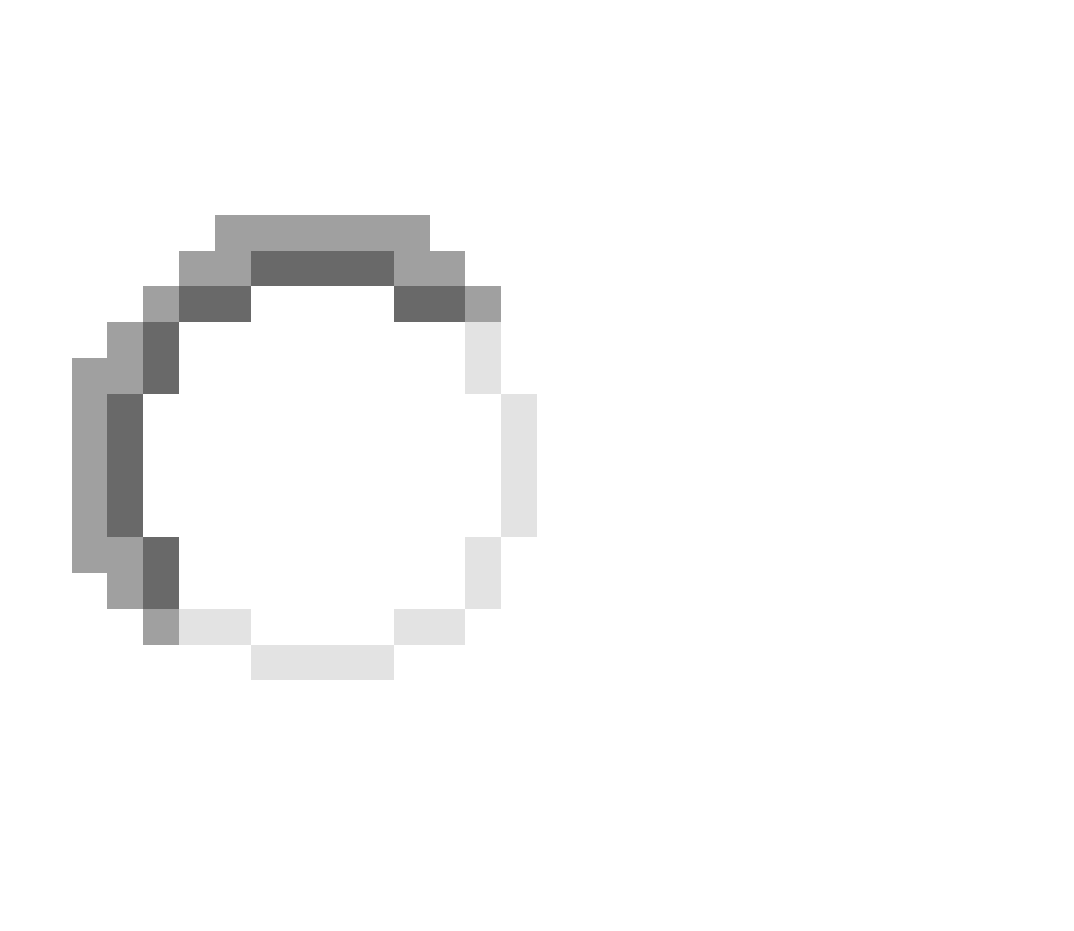
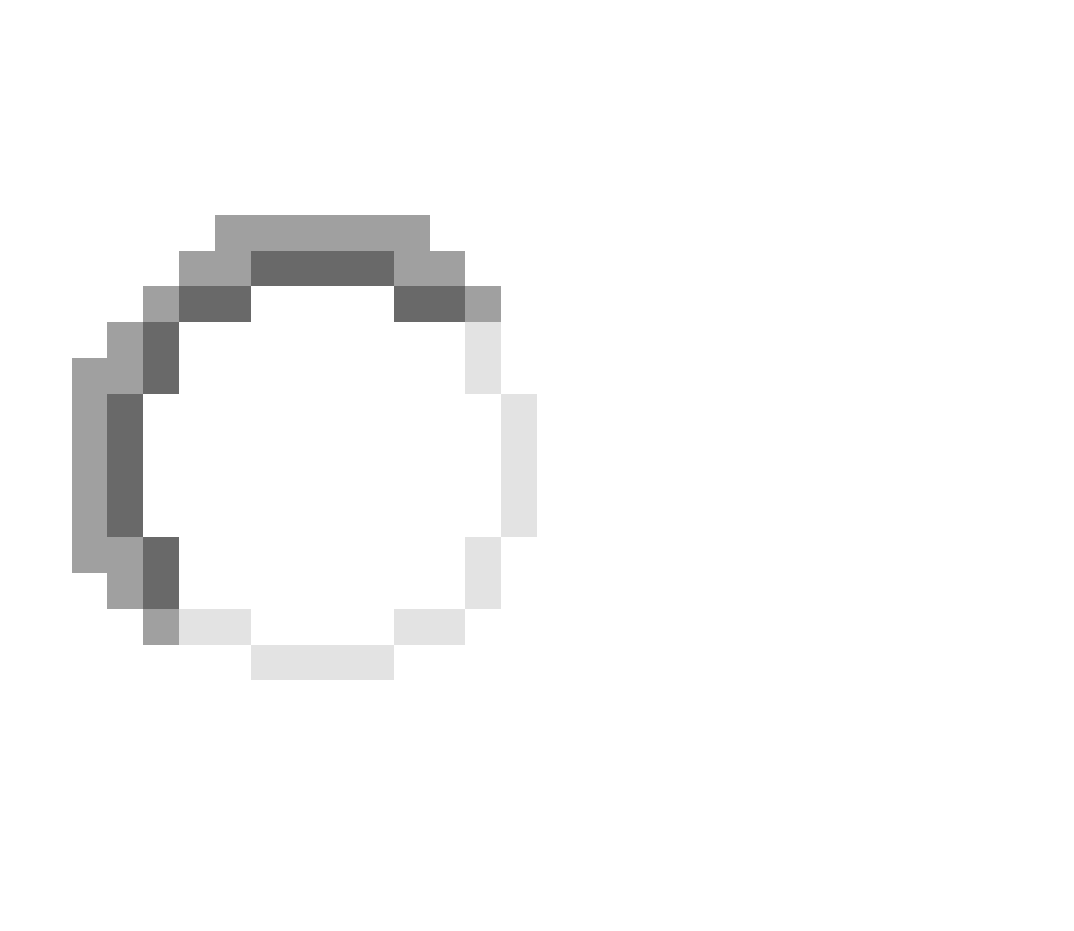
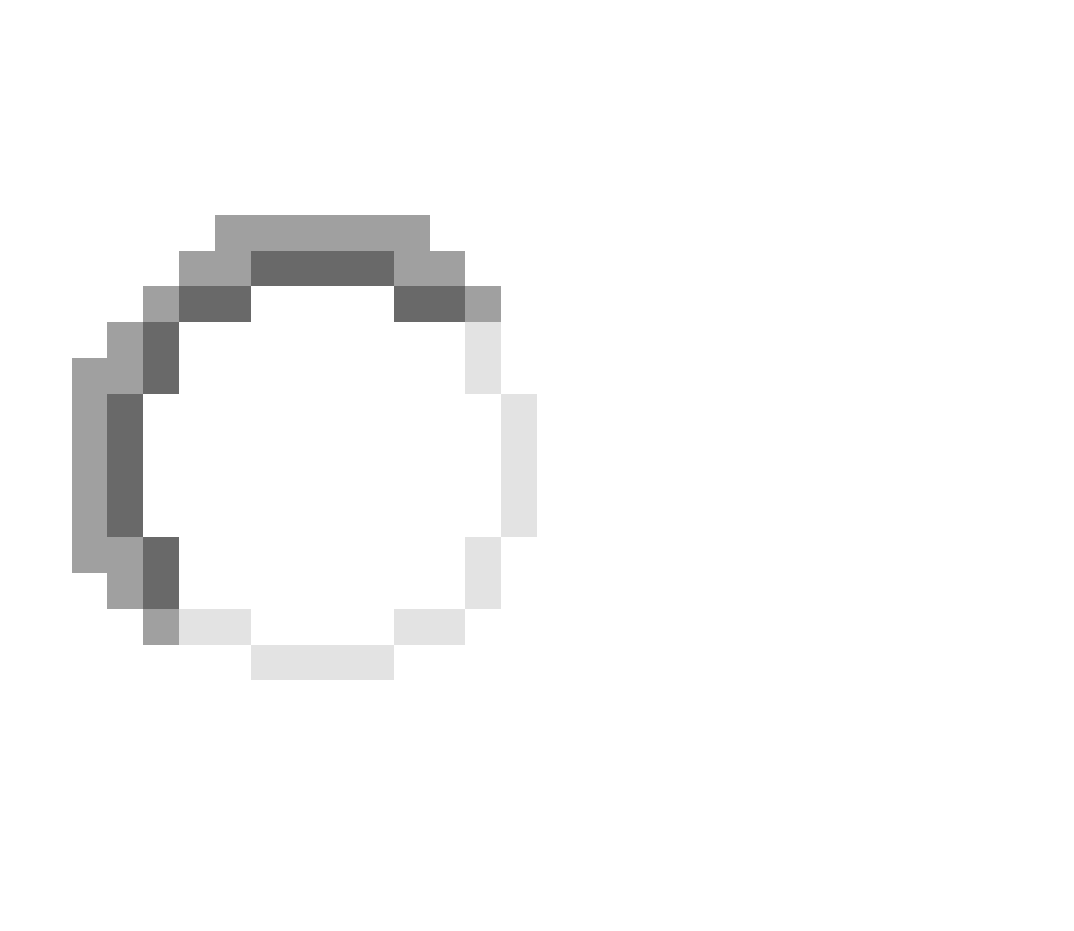
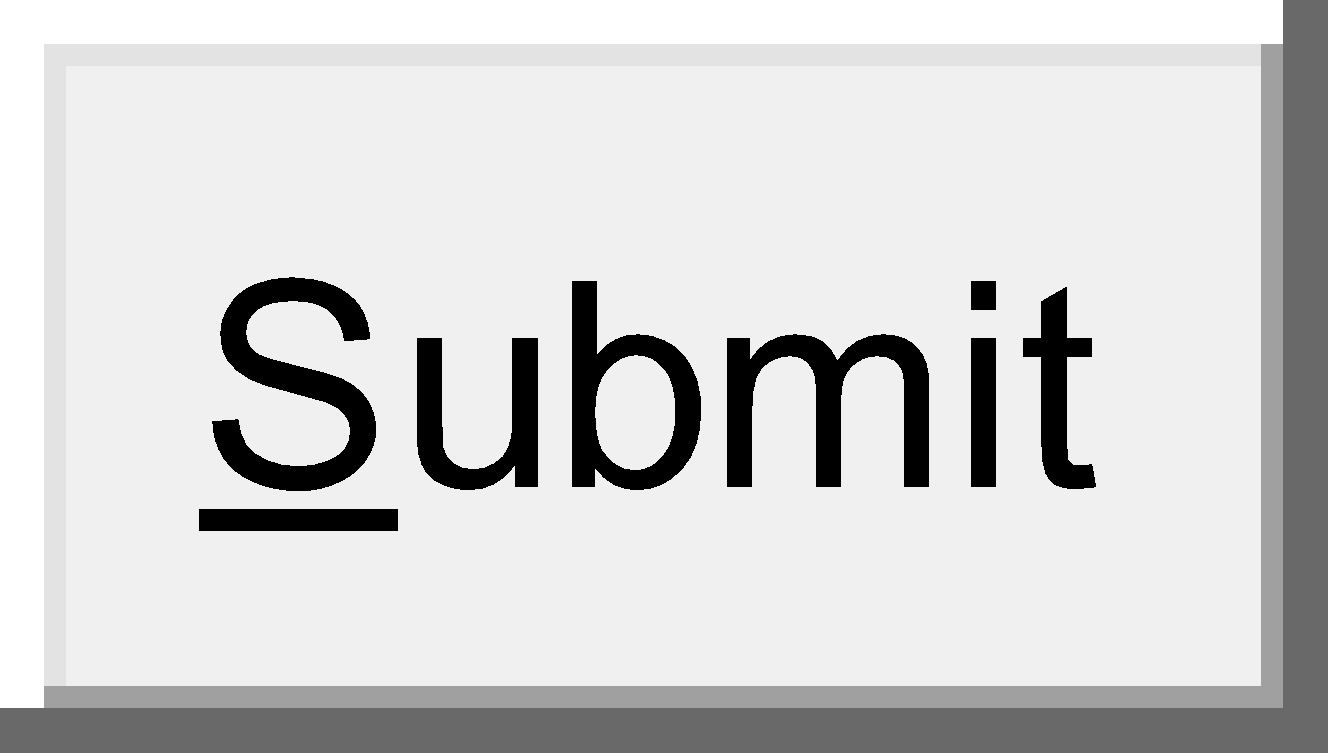
<?php  
$t = date("H");  
  
if ($t < "10") {  
    echo "Have a good morning!";  
} elseif ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>

Form Handling and Validation:

## PHP Form Validation Example

\* required field

Top of Form

Name: \*   
  
E-mail: \*   
  
Website:   
  
Comment:   
  
Gender: Female Male Other \*   
  


Bottom of Form

## Your Input:

<!DOCTYPE HTML>    
<html>  
<head>  
<style>  
.error {color: #FF0000;}  
</style>  
</head>  
<body>    
  
<?php  
// define variables and set to empty values  
$nameErr = $emailErr = $genderErr = $websiteErr = "";  
$name = $email = $gender = $comment = $website = "";  
  
if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  
  if (empty($\_POST["name"])) {  
    $nameErr = "Name is required";  
  } else {  
    $name = test\_input($\_POST["name"]);  
  }  
    
  if (empty($\_POST["email"])) {  
    $emailErr = "Email is required";  
  } else {  
    $email = test\_input($\_POST["email"]);  
  }  
      
  if (empty($\_POST["website"])) {  
    $website = "";  
  } else {  
    $website = test\_input($\_POST["website"]);  
  }  
  
  if (empty($\_POST["comment"])) {  
    $comment = "";  
  } else {  
    $comment = test\_input($\_POST["comment"]);  
  }  
  
  if (empty($\_POST["gender"])) {  
    $genderErr = "Gender is required";  
  } else {  
    $gender = test\_input($\_POST["gender"]);  
  }  
}  
  
function test\_input($data) {  
  $data = trim($data);  
  $data = stripslashes($data);  
  $data = htmlspecialchars($data);  
  return $data;  
}  
?>  
  
<h2>PHP Form Validation Example</h2>  
<p><span class="error">\* required field</span></p>  
<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">    
  Name: <input type="text" name="name">  
  <span class="error">\* <?php echo $nameErr;?></span>  
  <br><br>  
  E-mail: <input type="text" name="email">  
  <span class="error">\* <?php echo $emailErr;?></span>  
  <br><br>  
  Website: <input type="text" name="website">  
  <span class="error"><?php echo $websiteErr;?></span>  
  <br><br>  
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>  
  <br><br>  
  Gender:  
  <input type="radio" name="gender" value="female">Female  
  <input type="radio" name="gender" value="male">Male  
  <input type="radio" name="gender" value="other">Other  
  <span class="error">\* <?php echo $genderErr;?></span>  
  <br><br>  
  <input type="submit" name="submit" value="Submit">    
</form>  
  
<?php  
echo "<h2>Your Input:</h2>";  
echo $name;  
echo "<br>";  
echo $email;  
echo "<br>";  
echo $website;  
echo "<br>";  
echo $comment;  
echo "<br>";  
echo $gender;  
?>  
  
</body>  
</html>