មេរៀន ទី១ ការណែនាំអំពី PHP

PHP តំណាងអោយ Hypertext Preprocessor ។ PHP គឺជាភាសាស្គ្រីបផ្នែកខាងម៉ាស៊ីនបម្រើប្រភពបើកចំហរដែលមានប្រជាប្រិយភាពនិងត្រូវបានគេប្រើយ៉ាងទូលំទូលាយដើម្បីសរសេរទំព័របណ្តាញដែលបានបង្កើតយ៉ាងស្វាហាប់។ ដំបូងឡើយ PHP ត្រូវបានបង្កើតឡើងដោយ Rasmus Lerdorf ក្នុងឆ្នាំ ១៩៩៤ ។ ដំបូងឡើយវាត្រូវបានគេស្គាល់ថាជាទំព័រដើមផ្ទាល់ខ្លួន។

ស្គ្រីប PHP ត្រូវបានប្រតិបត្តិលើម៉ាស៊ីនមេហើយលទ្ធផលត្រូវបានបញ្ជូនទៅកម្មវិធីរុករកអ៊ីនធឺណិតជា HTML ធម្មតា។ PHP អាចត្រូវបានរួមបញ្ចូលជាមួយនឹងចំនួនមូលដ្ឋានទិន្នន័យដែលពេញនិយមរួមមាន MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Sybase ជាដើម។ ជំនាន់ចុងក្រោយរបស់ PHP គឺ ៧។ កូដទាំងអស់នៅក្នុងឯកសារបង្រៀននេះត្រូវបានសាកល្បងនិងធ្វើឱ្យមានសុពលភាពប្រឆាំងនឹងការចេញផ្សាយរបស់ PHP ៧ ថ្មីបំផុត។

PHP ជាភាសាដែលមានថាមពលខ្លាំងណាស់តែងាយស្រួលរៀននិងប្រើ។ ដូច្នេះកត់ចំណាំគេហទំព័រនេះហើយបន្តទៅមុខទៀត។

ការបង្រៀន PHP របស់យើងនឹងជួយអ្នកឱ្យរៀនមូលដ្ឋានគ្រឹះនៃភាសាស្គ្រីប PHP ចាប់ពីកំរិតមូលដ្ឋានដល់ប្រធានបទជឿនលឿនមួយដំហានម្តង ៗ ។ ប្រសិនបើអ្នកជាអ្នកចាប់ផ្តើមដំបូងចូរចាប់ផ្តើមពីមូលដ្ឋានហើយឈានទៅមុខបន្តិចម្តង ៗ ដោយរៀនបន្តិចជារៀងរាល់ថ្ងៃ។

១.១ អ្វីដែលអ្នកអាចធ្វើបានជាមួយ PHP

មានអ្វីជាច្រើនទៀតដែលអ្នកអាចធ្វើជាមួយកម្មវិធី PHP ដូចជា៖

- អ្នកអាចបង្កើតទំព័រនិងឯកសារថាមវន្ត
- អ្នកអាចបង្កើតបើកអានសរសេរនិងបិទឯកសារនៅលើម៉ាស៊ីនមេ
- អ្នកអាចប្រមូលទិន្នន័យពីទំរង់គេហទំព័រដូចជាព័ត៌មានអ្នកប្រើប្រាស់អ៊ីម៉ែលលេខទូរស័ព្ទ។ ល។
- អ្នកអាចផ្ញើអ៊ីមែលទៅអ្នកប្រើប្រាស់គេហទំព័ររបស់អ្នក
- អ្នកអាចផ្ញើនិងទទួលខូឃីស៍ដើម្បីតាមដានអ្នកចូលមើលគេហទំព័ររបស់អ្នក
- អ្នកអាចរក្សាទុកលុបនិងកែប្រែព័ត៌មាននៅក្នុងឃ្លាំងទិន្នន័យរបស់អ្នក
- អ្នកអាចដាក់កម្រិតការចូលប្រើគេហទំព័ររបស់អ្នកដោយគ្មានការអនុញ្ញាត
- អ្នកអាចអ៊ិនគ្រីបទិន្នន័យសម្រាប់ការបញ្ជូនដោយសុវត្ថិភាពតាមអ៊ីនធឺណិត។

បញ្ជីមិនបានបញ្ចប់នៅទីនេះទេមានរឿងគួរឱ្យចាប់អារម្មណ៍ជាច្រើនទៀតដែលអ្នកអាចធ្វើបានជាមួយ PHP ។ អ្នកនឹងរៀនអំពីចំណុចទាំងអស់នេះលម្អិតនៅក្នុងជំពូកបន្ទាប់។

១.២ គុណសម្បត្តិរបស់ PHP លើភាសាផ្សេងទៀត

ប្រសិនបើអ្នកស៊ាំជាមួយភាសាផ្នែកខាងម៉ាស៊ីនបម្រើ(S e r v e r - s i d e)ផ្សេងទៀតដូចជា ASP.NET ឬJava, អ្នកអាចនឹងត្រូវបានឆ្ងល់ថាអ្វីដែលធ្វើឱ្យកម្មវិធី PHP ពិសេសដូច្នេះ។ មានគុណសម្បត្តិជាច្រើនហេតុអ្វីបានជាមនុស្សម្នាក់គួរតែជ្រើសរើសយក PHP៖

- **Easy to learn:** PHP ងាយស្រួលរៀននិងប្រើ។ សម្រាប់អ្នកសរសេរកម្មវិធីដំបូងដែលទើបតែចាប់ផ្តើមអភិវឌ្ឍបណ្តាញអ៊ិនធឺរណែតជាញឹកញាប់ត្រូវបានគេចាត់ទុកថាជាជំរើសនៃភាសាដែលត្រូវរៀន
- **Open source:** PHP គឺជាគម្រោងប្រភពបើកចំហ។ វាត្រូវបានបង្កើតនិងថែរក្សាដោយសហគមន៍អ្នកអភិវឌ្ឍន៍ទូទាំងពិភពលោកដែលធ្វើឱ្យកូដប្រភពរបស់វាអាចទាញយកនិងប្រើប្រាស់បានដោយសេរី។

- **Portability:** PHP ដំណើរការលើប្រព័ន្ធប្រតិបត្តិការផ្សេងៗដូចជា Microsoft Windows, Linux, Mac OS, ។ល។ ហើយវាត្រូវគ្នាជាមួយស្ទើរតែរាល់សឺរីវដែលប្រើសព្វថ្ងៃ Apache, IIS, ។ល។
- **Fast Performance:** ស្គ្រីបដែលសរសេរក្នុង PHP ជាធម្មតាដំណើរការឬដំណើរការលឿនជាងអក្សរដែលសរសេរជាភាសាស្គ្រីបផ្សេងទៀតដូចជាអេអេសអេស Ruby Python ចាវ៉ាជាដើម។
- **Vast Community:** ចាប់តាំងពីកម្មវិធី PHP ត្រូវបានគាំទ្រដោយសហគមន៍ទូទាំងពិភពលោកការស្វែងរកជំនួយឬឯកសារទាក់ទងនឹង PHP តាមអ៊ិនធរណេតគឺងាយស្រួលបំផុត

តើអ្នកដឹងទេថាគេហទំព័រធំ ៗ មួយចំនួនដូចជាហ្វេសប៊ុកយ៉ាហ៊ូហ្គូគ្រីនិងវិគីភីឌាត្រូវបានបង្កើតឡើងដោយប្រើ PHP ។ ភាគច្រើននៃប្រព័ន្ធគ្រប់គ្រងមាតិកាសំខាន់ៗ (CMS) ដូចជា WordPress, Drupal, Joomla និង Magento ត្រូវបានបង្កើតឡើងនៅក្នុង PHP ផងដែរ។

១.៣ តើឯកសារបង្រៀនេះមានអ្វីខ្លះ?

This PHP tutorial series covers all the fundamental programming concepts, including data types, operators, creating and using variables, generating outputs, structuring your code to make decisions in your programs or to loop over the same block of code multiple times, creating and manipulating strings and arrays, defining and calling functions, and so on.

Once you're comfortable with the basics, you'll move on to next level that explains the concept file system, sessions and cookies, dates and times, as well as how to send email from your script, handling and validating forms, perform data filtration and handling errors in PHP.

Finally, you'll explore some advanced concepts like classes and objects, parsing JSON data, pattern matching with regular expressions, exception handling as well as how to use PHP to manipulate data in MySQL database and create useful features like user login system, Ajax search, etc.

មេរៀន ទី២ ការចាប់ផ្តើមកម្មវិធី PHP

២.១.ការចាប់ផ្តើមជាមួយ PHP

Install Wampserver or XAMPP on your PC to quickly create web applications with Apache, PHP and a MySQL database.

២.២ ការបង្កើត(Setting Up) a Local Web Server

ស្គ្រីប PHP ប្រតិបត្តិនៅលើម៉ាស៊ីនមេគេហទំព័រដែលដំណើរការ PHP ។ ដូច្នេះមុនពេលអ្នកចាប់ផ្តើមសរសេរកម្មវិធី PHP ណាមួយអ្នកត្រូវដំឡើងកម្មវិធីដូចខាងក្រោមនៅលើកុំព្យូទ័ររបស់អ្នក។

- The Apache Web server
- The PHP engine
- The MySQL database server

អ្នកអាចតំឡើងវារៀងៗខ្លួនឬជ្រើសរើសកញ្ចប់ដែលបានកំនត់ទុកជាមុនសំរាប់ប្រព័ន្ធប្រតិបត្តិការរបស់អ្នកដូចជាលីនុចនិងវីនដូ។ កញ្ចប់ដែលបានកំនត់ទុកជាមុនដែលពេញនិយមគឺ XAMPP និង WampServer ។

Wampserver គឺជាបរិស្ថានអភិវឌ្ឍបណ្តាញប្រព័ន្ធប្រតិបត្តិការ Windows មួយ។ វាអនុញ្ញាតឱ្យអ្នកបង្កើតកម្មវិធីគេហទំព័រជាមួយ Apache2, PHP និងមូលដ្ឋានទិន្នន័យ MySQL ។ វាក៏នឹងផ្តល់នូវឧបករណ៍រដ្ឋបាល MySQL ផងដែរ PhpMyAdmin ដើម្បីគ្រប់គ្រងមូលដ្ឋានទិន្នន័យរបស់អ្នកយ៉ាងងាយស្រួលដោយប្រើកម្មវិធីអ៊ីនធឺណិត។

គេហទំព័រផ្លូវការសម្រាប់ការទាញយកនិងតំឡើងការណែនាំសំរាប់ WampServer ៖

http://www.wampserver.com/en/

What is a WAMP Server?

WAMP stands for Windows (operating system), Apache (webserver), MySQL (database), and PHP (coding language). WAMP can be compared to two other close variants, namely:

- LAMP (Linux, Apache, MySQL, and PHP) for Linux Systems and,
- MAMP (Mac, Apache, MySQL, and PHP) for Mac OS operating systems

WAMP, therefore, is a localhost server for Windows Operating Systems that lets you manage, create, test, and develop websites without having to use a remote web server.

What do we mean here?

Let´s take *WordPress* as an example. If you want to build a website on the *WordPress* platform, for example, you can use the WAMP server to create a local environment that allows you to work offline.
Here you can experiment with code, plugins, design, as well as test the different features of your website before taking it live.

It´s important to note that Apache, PHP, and MySQL are all open-source software. This means that you can install and configure each service individually.
However, this process is not easy, even for skilled developers. Luckily, a WAMP server automates the configuration process giving you a local working environment in a matter of minutes.

With that in mind, now let´s look at **how to install a WAMP server**.
Step 1: Installing and Setting Up WAMP Server on Windows Computer

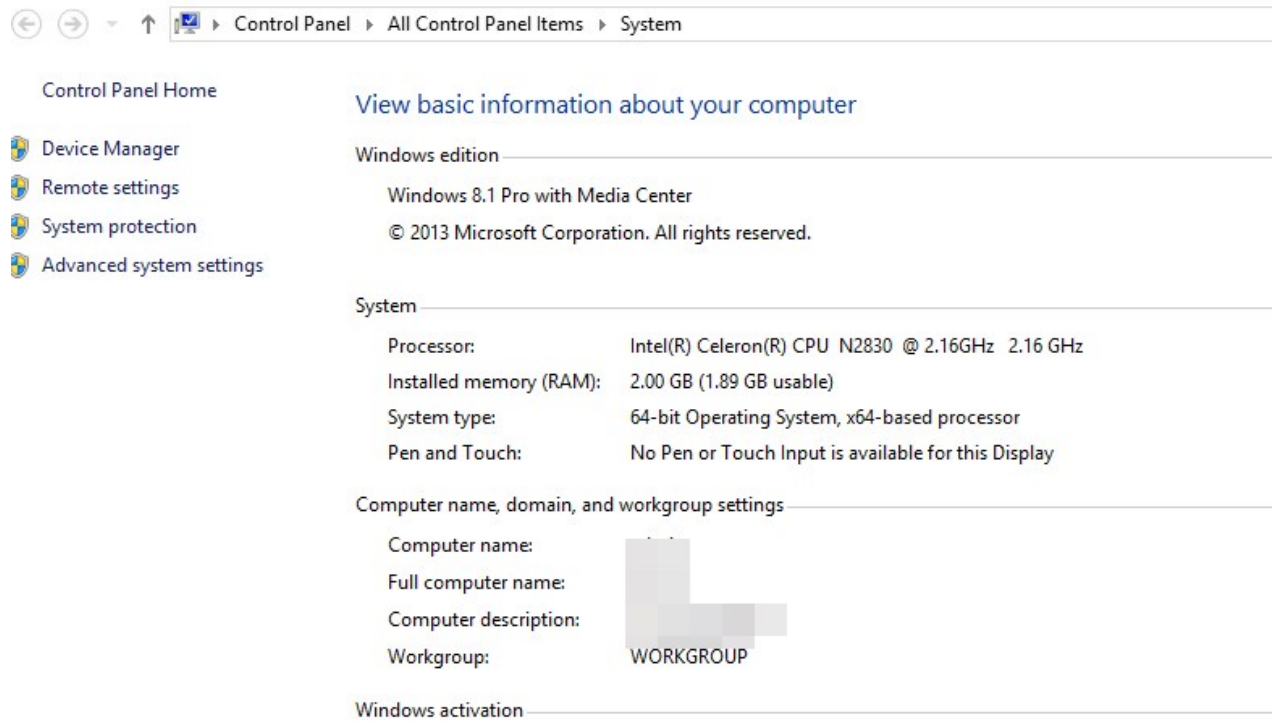
The first step is to download the latest version of the installer file for the WAMP server on to your Windows PC.
To do this:

Visit the **WAMP**server website.

On the web page, you'll see the "**START USING WAMPSERVER**" button. Click on it.

The button will automatically take you to the "**DOWNLOADS**" section where you'll be greeted by two versions of the WAMP server namely, **WAMP SERVER 64 BITS (X64)** and **WAMPSERVER 32 BITS (X86)**.

Choose the installer file that matches your Windows operating system. If you are not sure whether you are using a 62-bit system or a 32-bit system, just go to **This PC** >> **Open Control Panel** >> **System**. Alternatively, you can just right-click on **This PC** (My **Computer**) >> **Properties.**



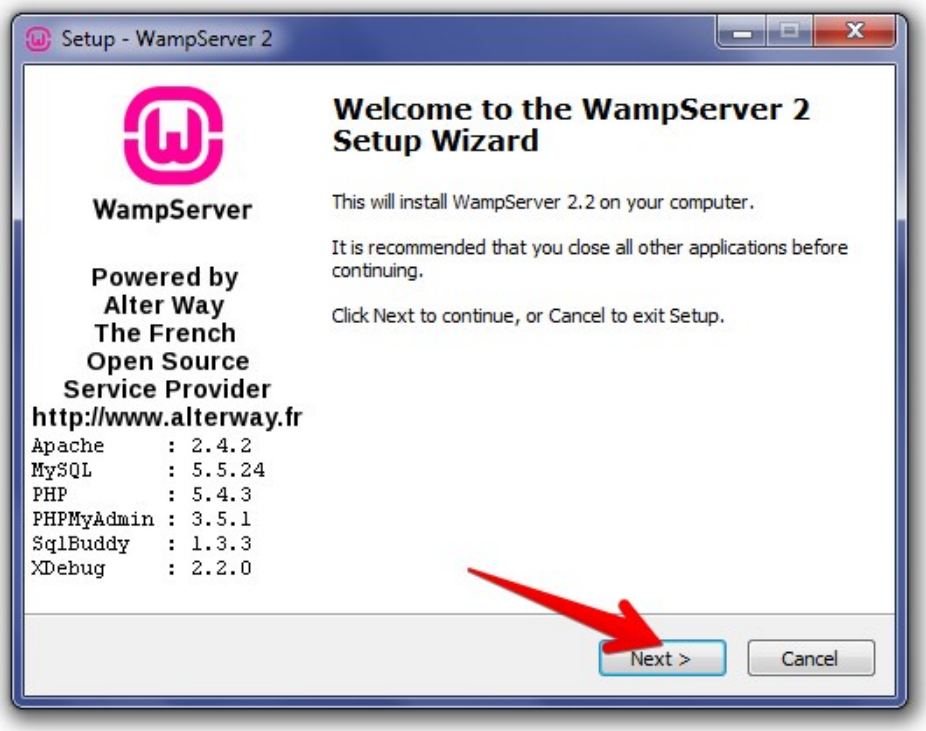
Once the download process is complete, follow the following simple steps:

Go to the **Downloads** folder and locate the WAMP server installer file.

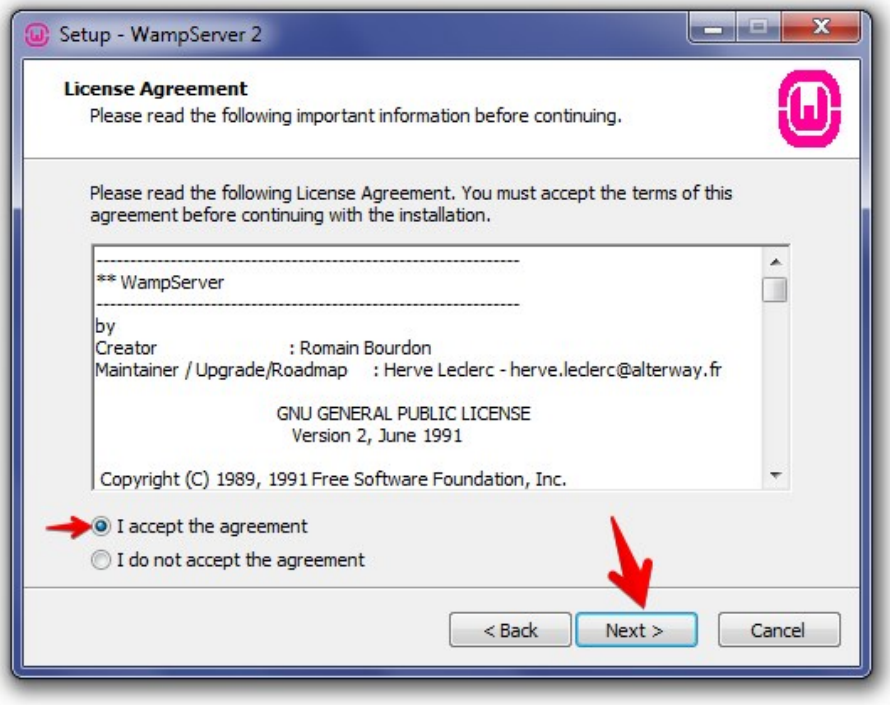
Double-click the WAMP server installer file. A security window will pop up, asking you whether you are sure of whether you want to run the file.

Click "**Run**" to initiate the installation process.

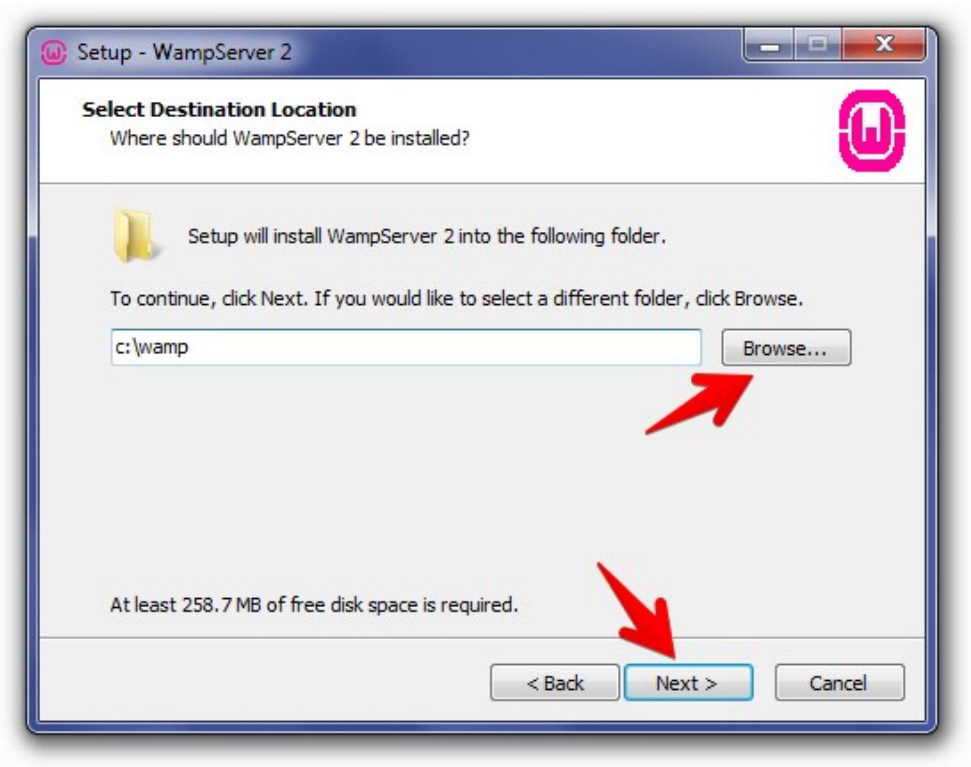
A WAMP Server wizard screen will pop up. Select "**Next**" to continue with the installation process.



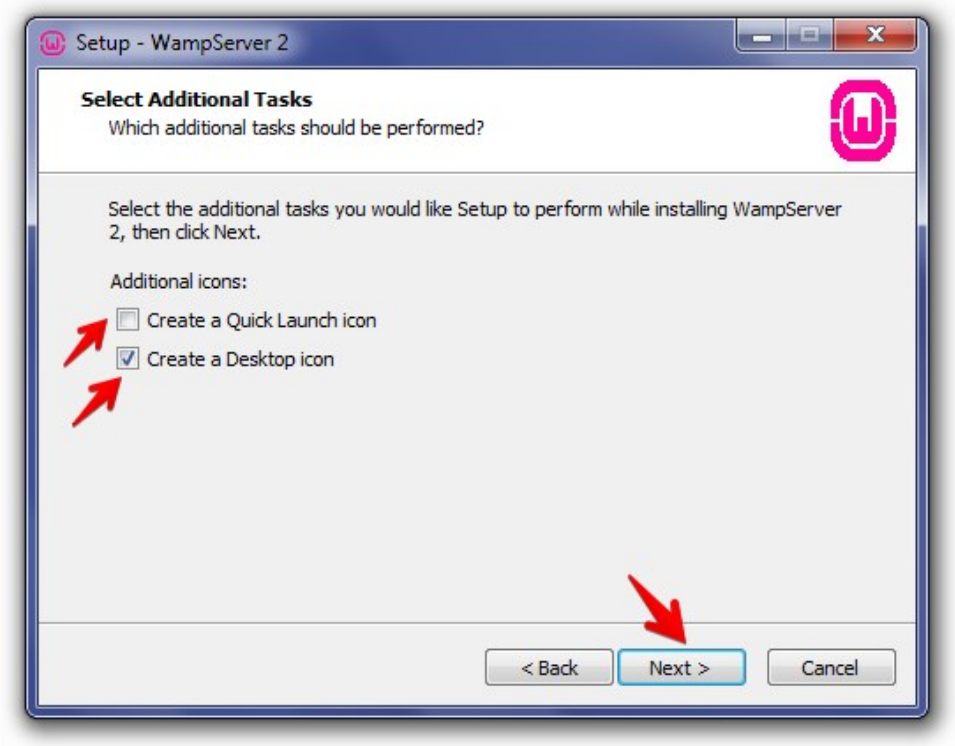
On the next screen, you´ll see a License Agreement. Read through it and then check "**I accept the agreement**" at the bottom of the screen". Now click "**Next.**"



On the next screen, you´ll be presented with a Select Destination Location screen. Choose a folder where you would like to install the WAMP server. In most cases, it´s just good to leave it as it is. Now click "**Next**" to continue.

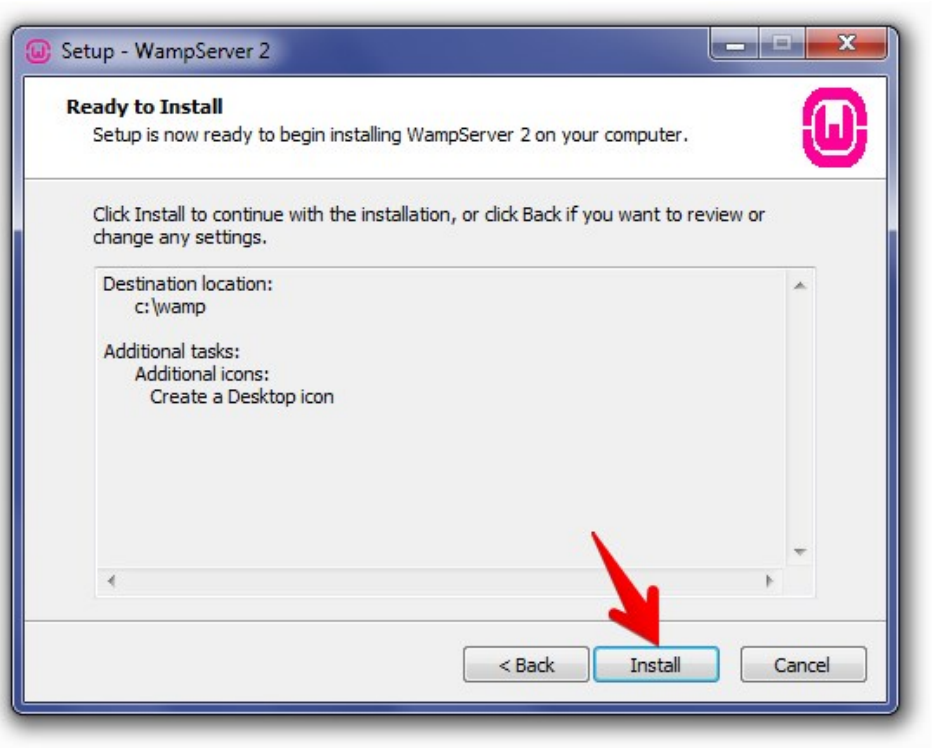


A "**Select Additional Tasks**" screen will follow. Here you can choose to have a desktop icon created, or a quick launch icon added. Make your selection and click on "Next."

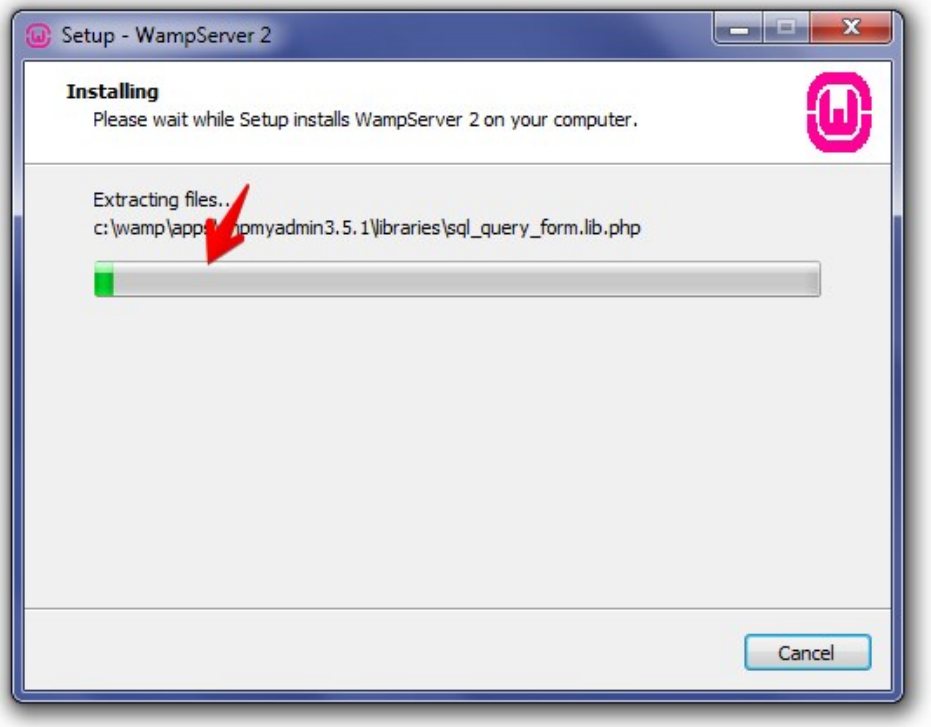


At this juncture, the setup process is just a click away. If you are sure you do not need to make any changes in your previous selections, click "**Install**" and the **extraction** process

will start automatically.



Once the extraction process is complete, you'll be required to choose your default browser. In most cases, the default file browser will be Internet Explorer, but you can change it to Safari, Opera, Mozilla, or Chrome.



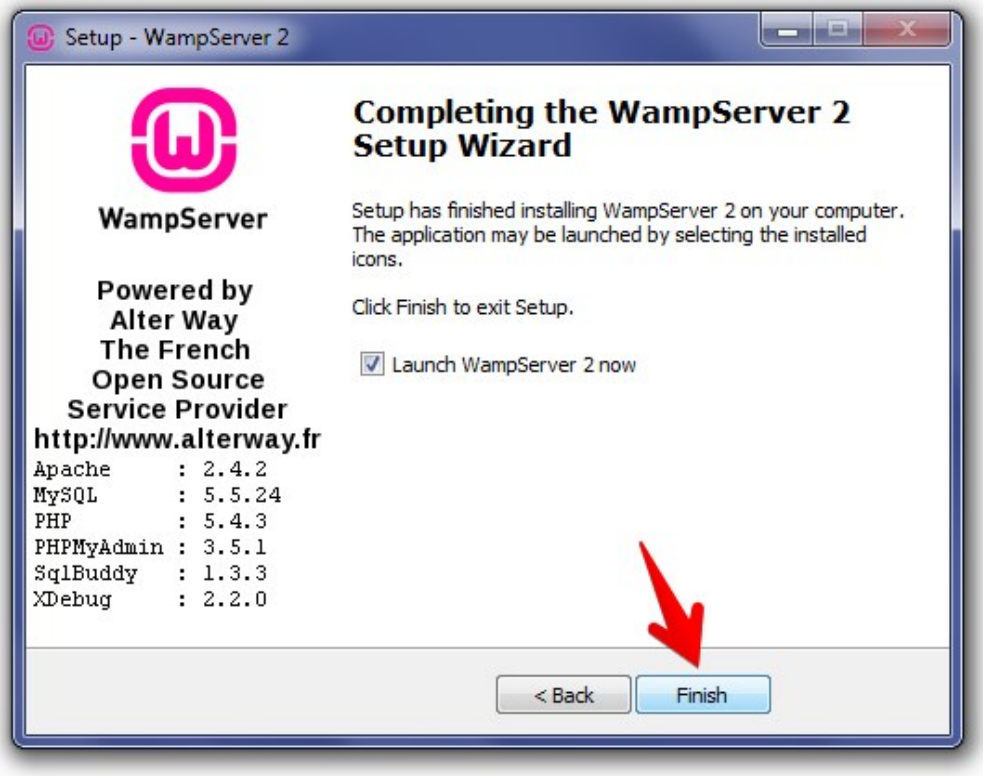
Your default browser should have a corresponding .exe file. For example, safari.exe, opera.exe, firefox.exe, or chome.exe. Select **Open** to continue.

You´ll see a Windows security alert indication that firewall has blocked some of the program´s features. Check to allow Apache HTTP to communicate on a public or private network.

Choose to **Allow Access.**

A setup screen with a green bar will appear to show the status of your installation. Once the bar is fully green, another screen will appear, asking you for PHP mail parameters. Leave the SMTP server as "**localhost**" and enter your preferred email address. Click on **Next**.

A completing installation set up wizard screen will appear. Check on the box adjacent to "Launch Wamp Server" now and click **Finish**.



At this stage, you should be able to see a WAMP server icon at the system tray of your Windows PC taskbar. You should know that the WAMP server is working correctly if the icon is green. A red icon indicates an issue with Apache and MySQL and an orange icon is an indication of a problem with one of the services.

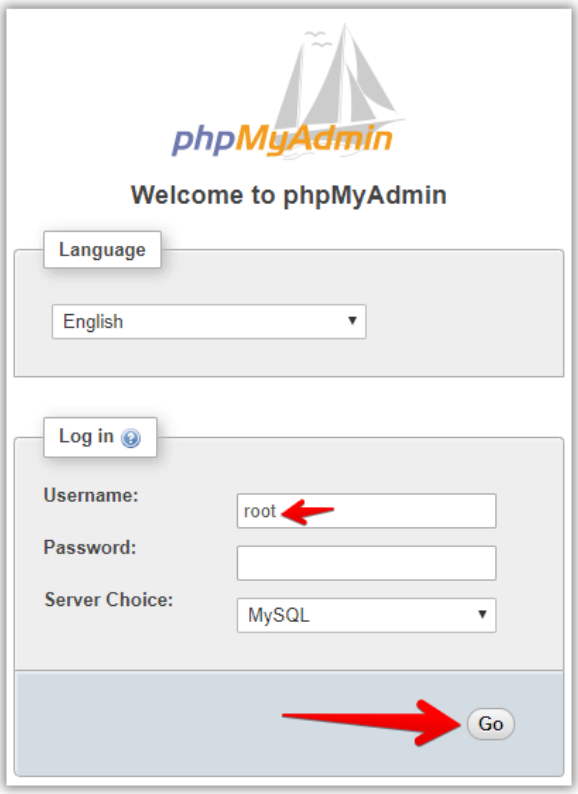
Step 2: Setting Up a Database for WordPress

Now that you've finished installing WAMP, the next step is to create a MySQL database. Now launch WAMP and you'll see a green icon at the systray. Click on phpMyAdmin. Your default web browser will show displaying the phpMyAdmin login screen.

You'll be required to enter your username and password. Enter the following details:

Username: **root**

Leave the password area field blank and click on the "Go" button. This will automatically log you in.



Now click on Databases in phpMyAdmin to create your database name for WordPress such as WP_Project. Click on "**Create**" and your database will be ready. You can now install Word-Press.

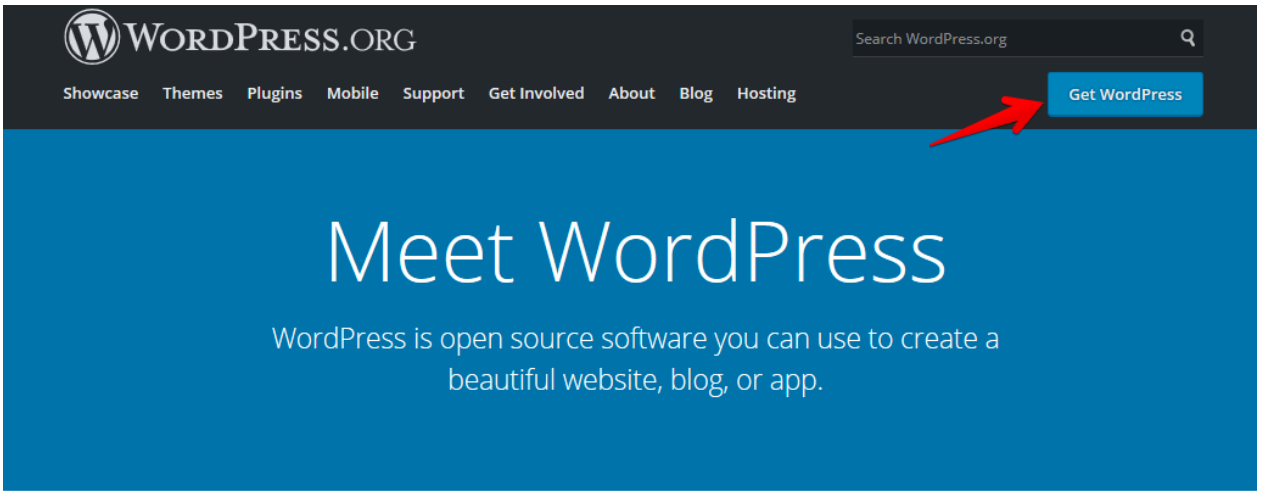
Step 3: Install WordPress on WAMP Server

There are two versions of WordPress. One is Hosted WordPress from WordPress.com and the other is self-hosted WordPress from wordpress.org.

In this case, we will be talking about the self-hosted WordPress, which is a free, open-source CMS for website building and maintenance. Now, to download WordPress, vis-

it www.wordpress.org.

Click on "**Get WordPress.**" The file will download as a zip archive.



Beautiful designs, powerful features, and the freedom to build anything you want. WordPress is both free and priceless at the same time.

Extract the contents of the file and copy or move the WordPress folder. In our case, our WAMP server file was installed in **C:\wamp64**, so we pasted our WordPress folder in **C:\wamp64 \www** folder.

To avoid confusion in the later stages of your project, you can choose to give the WordPress folder a name that you will remember such as mywpproject, testwp, etc. Keep in mind that this will be your local WordPress site URL.

For example, your URL will appear something like http://localhost/folder-name. For the sake of this tutorial, we used the default WordPress folder whose URL is http://localhost/wordpress.

Now follow the following steps:

Enter the following address into your default web browser: http://localhost/wordpress.

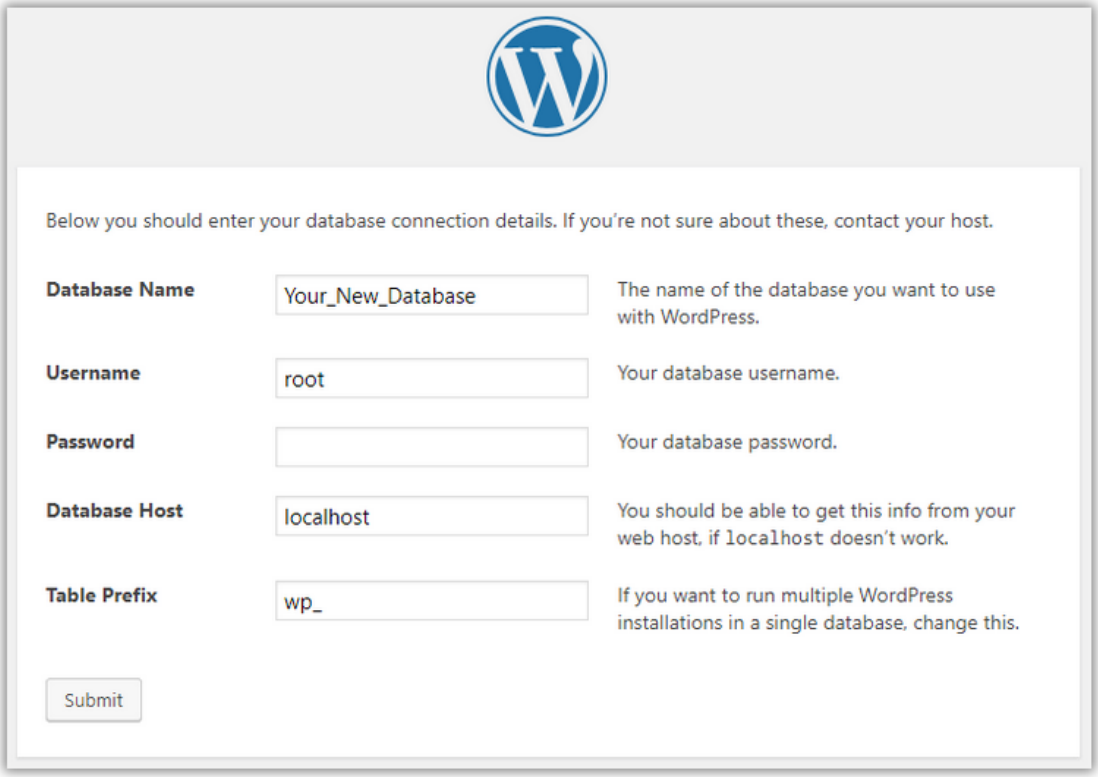
Choose a language and create a configuration file.

Click on "**Let's go!**"

You'll be required to enter your database name, username, password, host, and table prefix.

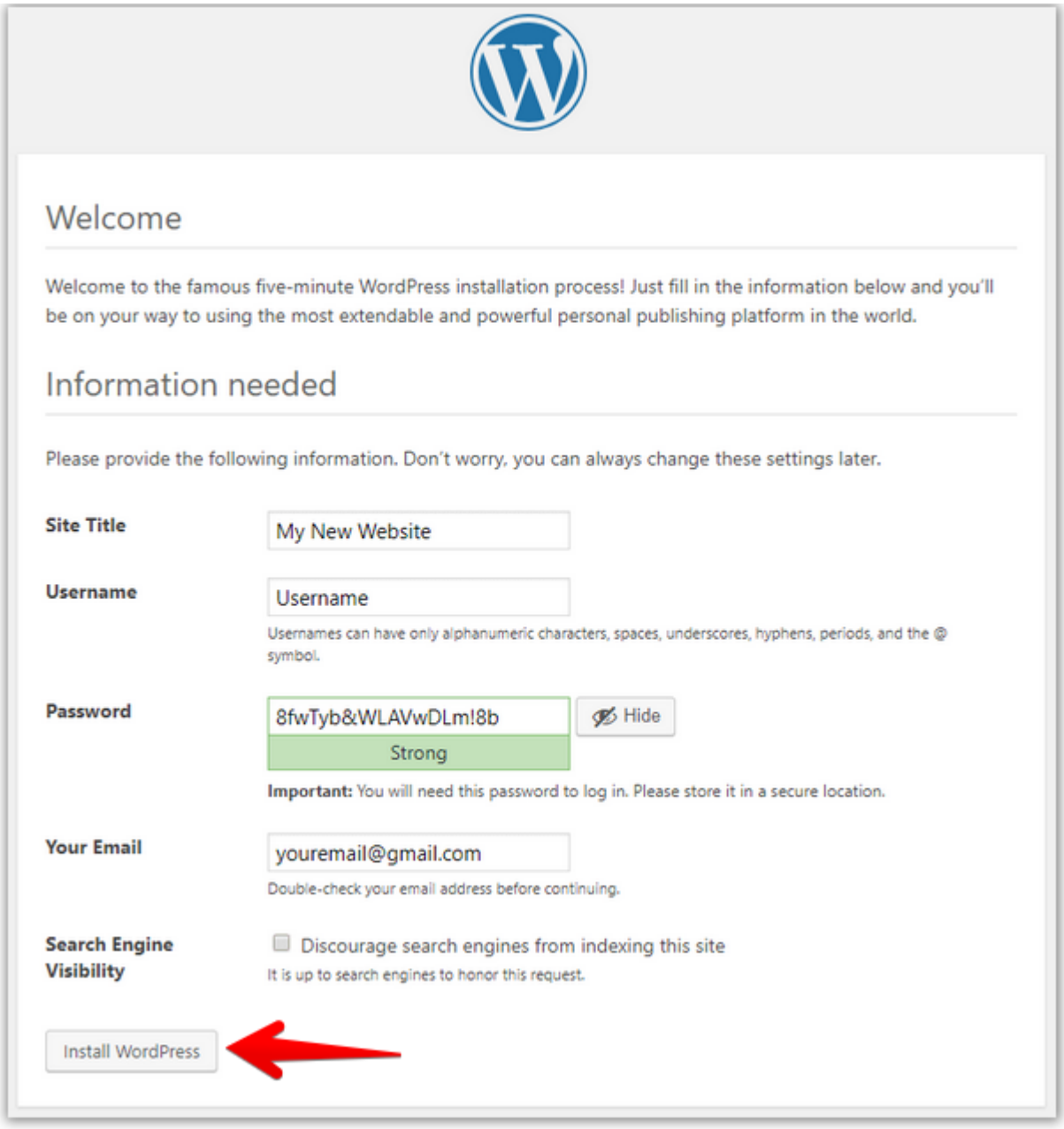
In our case, we entered the following credentials.

Database Name: WP_Project (Our preferred database name)
Username: root (default MySQL password)
Password: (leave this field blank)
Host: localhost
Table prefix: wp_



After you've entered the above credentials, click **Submit.** A configuration file will be created in the backend. You'll then be met by another screen indicating that WordPress has successfully been connected to your database.

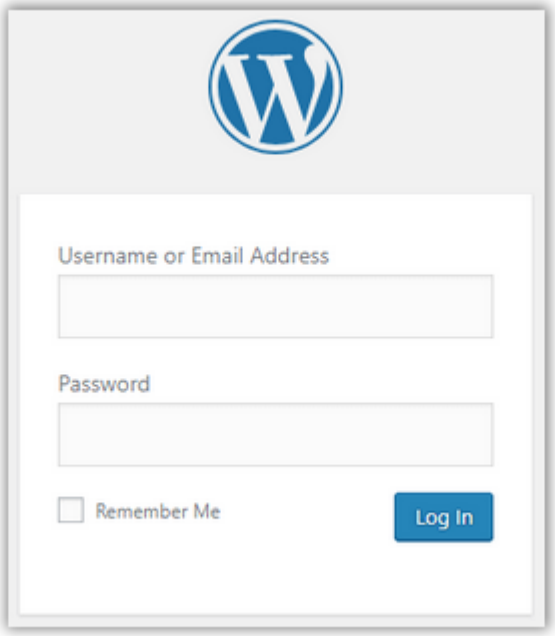
Now click on "**Run the installation.**"



The next screen will ask for your information regarding your new WordPress website. These include the site title, username, password, and email address.

Enter the credentials and hit "**Install WordPress**."

Once the installation is complete, you'll be greeted by a success message. You can now move forward and log into your WordPress dashboard.



If you've already done that, Congratulations! Because you just installed WordPress using WAMP.

Wrapping Up

Having a local server such as WAMP helps create a near-live environment where you can test, design, and experiment on different features such as plugins and get a gist of how your site will look like.

You can also experiment on different website building platforms such as Word-Press, Drupal, etc., and get to learn how the two platforms compare.

២.៣ ការបង្កើតស្គ្រីប PHP ដំបូងរបស់អ្នក

ឥឡូវអ្នកបានដំឡើង WampServer ដោយជោគជ័យនៅលើកុំព្យូទ័ររបស់អ្នកហើយ។ នៅក្នុងផ្នែកនេះយើងនឹងបង្កើតស្គ្រីប PHP សាមញ្ញបំផុតដែលបង្ហាញអត្ថបទ "សួស្តីពិភពលោក!" នៅក្នុងបង្អួចកម្មវិធីរុករក។

ចុចលើរូបតំណាង WampServer នៅកន្លែងណាមួយនៅលើរបារភារកិច្ចរំនែដូរបស់អ្នកហើយជ្រើសរើសយក "www directory". ម៉្យាងទៀតអ្នកអាចចូលមើលថតឯកសារ "www" តាមរយៈការបើកមើល C: \ wamp \ www ។ បង្កើតថតរងនៅក្នុងថត" www" សូមនិយាយថា" គំរោង" ។

ឥឡូវបើកកម្មវិធីនិពន្ធកូដដែលអ្នកចូលចិត្តហើយបង្កើតឯកសារ PHP ថ្មីបន្ទាប់មកវាយលេខកូដខាងក្រោម :

```
<?php
// Display greeting message
echo "Hello, world!";
?>
```

ឥឡូវរក្សាទុកឯកសារនេះជា "hello.php" នៅក្នុងថតគំរោងរបស់អ្នក (ដែលមានទីតាំងនៅ C: \ wamp \ www \ project) ហើយមើលលទ្ធផលនៅក្នុង browser របស់អ្នកតាមរយៈការចូលទៅកាន់ URL នេះ៖ http: //localhost/project/hello.php ។ ម៉្យាងទៀតអ្នកអាចចូលប្រើឯកសារ "hello.php" តាមរយៈការជ្រើសរើសជំរើសក្នុងម៉ាស៊ីនមូលដ្ឋានហើយបន្ទាប់មកជ្រើសថតគំរោងពីមឺនុយ WampSever នៅលើរបារភារកិច្ច។

PHP អាចត្រូវបានបង្កប់នៅក្នុងគេហទំព័រ HTML ធម្មតា។ នោះមានន័យថានៅក្នុងឯកសារ HTML របស់អ្នកអ្នកអាចសរសេរសេចក្តីថ្លែងរបស់ PHP ដូចដែលបានបង្ហាញក្នុងឧទាហរណ៍តាមដាន៖

```
<!DOCTYPE HTML>
<html>
<head>
    <title>PHP Application</title>
</head>
<body>
<?php
// Display greeting message
```

```
    echo 'Hello World!';
    ?>
    </body>
    </html>
```

You will learn what each of these statements means in upcoming chapters.

មេរៀន ទី៣ Syntax របស់ PHP

៣.១ Standard PHP Syntax

ស្គ្រីប PHP ចាប់ផ្តើមដោយ <? php ហើយបញ្ចប់ដោយស្លាក?> ។

អ្នកកំណត់ព្រំដែន PHP <? php និង?> ក្នុងឧទាហរណ៍ខាងក្រោមគ្រាន់តែប្រាប់ម៉ាស៊ីន PHP ឱ្យចាត់ទុកប្លុកកូដរុំព័ទ្ធជាកូដ PHP ជាជាងកូដ HTML ធម្មតា។

```
<?php
// Some code to be executed
echo "Hello, world!";
?>
```

រាល់សេចក្តីថ្លែងការណ៍របស់ PHP បញ្ចប់ដោយសញ្ញា (;) - នេះប្រាប់ម៉ាស៊ីន PHP ថាចុងបញ្ចប់នៃសេចក្តីថ្លែងការណ៍បច្ចុប្បន្នបានដល់ហើយ។

៣.២ Embedding PHP within HTML

ឯកសារ PHP គឺជាឯកសារអត្ថបទធម្មតាដែលមានផ្នែកបន្ថែម .php ។ នៅខាងក្នុងឯកសារ PHP អ្នកអាចសរសេរ HTML ដូចអ្នកធ្វើនៅក្នុងទំព័រ HTML ធម្មតាក៏ដូចជាបង្កប់កូដ PHP សំរាប់ការប្រតិបត្តិផ្នែកខាងម៉ាស៊ីនមេ។

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>A Simple PHP File</title>
</head>
<body>
  <h1><?php echo "Hello, world!"; ?></h1>
</body>
</html>
```

ឧទាហរណ៍ខាងលើបង្ហាញពីវិធីដែលអ្នកអាចបង្កប់កូដ PHP ក្នុង HTML ដើម្បីបង្កើតទំព័របណ្តាញថាមវន្ត។ ប្រសិនបើអ្នកមើលកូដប្រភពនៃទំព័របណ្តាញលទ្ធផលនៅក្នុងកម្មវិធីរុករករបស់អ្នកភាពខុសគ្នាតែមួយគត់ដែលអ្នកនឹងឃើញគឺកូដ PHP <? php echo "សួស្តីពិភពលោក!"; ?> ត្រូវបានជំនួសដោយលទ្ធផល "សួស្តីពិភពលោក!"

តើមានអ្វីកើតឡើងនៅទីនេះ? នៅពេលអ្នកដំណើរការលេខកូដនេះម៉ាស៊ីន PHP បានបកស្រាយសេចក្តីណែនាំរវាងស្លាក <? php ...?> ហើយទុកអ្វីដែលនៅដដែល។ នៅចុងបញ្ចប់ម៉ាស៊ីនបម្រើគេហទំព័រផ្ទេលទ្ធផលចុងក្រោយត្រលប់ទៅកម្មវិធីរុករករបស់អ្នកដែលមានទាំងស្រុងនៅក្នុង HTML ។

៣.៣ PHP Comments

សេចក្តីអត្ថាធិប្បាយគឺជាអត្ថបទធម្មតាដែលត្រូវបានមិនអើពើដោយម៉ាស៊ីន PHP ។ គោលបំណងនៃមតិយោបល់គឺធ្វើឱ្យក្រមអាចអានបានកាន់តែច្រើន។ វាអាចជួយអ្នកអភិវឌ្ឍន៍ផ្សេងទៀត (ឬអ្នកនៅពេលអនាគតនៅពេលដែលអ្នកកែលេខកូដប្រភពដើម) ឱ្យយល់ពីអ្វីដែលអ្នកកំពុងព្យាយាមធ្វើជាមួយ PHP ។

កម្មវិធី PHP គាំទ្របន្ទាត់តែមួយក៏ដូចជាមតិយោបល់ពហុបន្ទាត់។ ដើម្បីសរសេរមតិយោបល់បន្ទាត់តែមួយចាប់ផ្តើមទាំងបន្ទាត់ដាច់ ។ ពីរ (//) ឬសញ្ញាហាស (#) ។ ឧទាហរណ៍:

```php
<?php
// This is a single line comment
# This is also a single line comment
echo "Hello, world!";

?>
```

ទោះយ៉ាងណាដើម្បីសរសេរយោបល់ពហុបន្ទាត់ចាប់ផ្តើមសេចក្តីអធិប្បាយដោយសញ្ញា (- *) និងបញ្ចប់មតិដោយសញ្ញាផ្គាយមួយអមដោយសញ្ញា (* /) ដូចនេះ៖

```php
<?php
This is a multiple line comment block
that spans across more than
one line
*/
echo "Hello, world!";

?>
```

៣.៤ Case Sensitivity in PHP

ឈ្មោះអថេរនៅក្នុង PHP គឺប្រកាន់អក្សរតូចធំ។ ជាលទ្ធផលអថេរ $color, $Color និង $COLOR ត្រូវបានចាត់ទុកជាអថេរបីផ្សេងគ្នា។

```
<?php
// Assign value to variable
$color = "blue";

// Try to print variable value
echo "The color of the sky is " . $color . "<br>";
echo "The color of the sky is " . $Color . "<br>";
echo "The color of the sky is " . $COLOR . "<br>";
?>
```

ប្រសិនបើអ្នកព្យាយាមដំណើរការកូដឧទាហរណ៍ខាងលើវានឹងបង្ហាញតែតម្លៃអថេរប៉ុណ្ណោះ

$color and produce the "Undefined variable" warning for the variable $Color and $COLOR.

ទោះជាយ៉ាងណាពាក្យគន្លឹះមុខងារនិងឈ្មោះថ្នាក់គឺមិនសមហេតុផលទេ។ ជាលទ្ធផលហៅឈ្មោះ gettype() or GETTYPE() បង្ហាញលទ្ធផលដូចគ្នា។

```
<?php
// Assign value to variable
$color = "blue";

// Get the type of a variable
echo gettype($color) . "<br>";
echo GETTYPE($color) . "<br>";
?>
```

ប្រសិនបើអ្នកព្យាយាមដំណើរការកូដឧទាហរណ៍ខាងលើទាំងមុខងារ gettype () និង GETTYPE () ផ្តល់នូវលទ្ធផលដូចគ្នាដែលជាខ្សែអក្សរ(string)។

មេរៀនទី៤ Variables និងConstants នៃ PHP

៤.១ What is Variable in PHP

Variables are used to store data, like string of text, numbers, etc. Variable values can change over the course of a script. Here're some important things to know about variables:

- In PHP, a variable does not need to be declared before adding a value to it. PHP automatically converts the variable to the correct data type, depending on its value.
- After declaring a variable it can be reused throughout the code.
- The assignment operator (=) used to assign value to a variable.

In PHP variable can be declared as: $var_name = value;

```
<?php
// Declaring variables
$txt = "Hello World!";
$number = 10;

// Displaying variables value
echo $txt;  // Output: Hello World!
echo $number; // Output: 10
?>
```

In the above example we have created two variables where first one has assigned with a string value and the second has assigned with a number. Later we've displayed the variables values in the browser using the echo statement. The PHP echo statement is often used to output data to the browser. We will learn more about this in upcoming chapter.

៤.២ Naming Conventions for PHP Variables

These are the following rules for naming a PHP variable:

- All variables in PHP start with a $ 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 in PHP can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- A variable name cannot contain spaces.

Variable names in PHP are case sensitive, it means $x and $X are two different variables. So be careful while defining variable names

៤.៣ តើ Constant ក្នុង PHPជាអ្វី?

A constant is a name or an identifier for a fixed value. Constant are like variables, except that once they are defined, they cannot be undefined or changed (except magic constants).

Constants are very useful for storing data that doesn't change while the script is running. Common examples of such data include configuration settings such as database username and password, website's base URL, company name, etc.

Constants are defined using PHP's define() function, which accepts two arguments: the name of the constant, and its value. Once defined the constant value can be accessed at any time just by referring to its name. Here is a simple example:

```
<?php
// Defining constant
define("SITE_URL", "https://www.tutorialrepublic.com/");

// Using constant
echo 'Thank you for visiting - ' . SITE_URL;
?>
```

The output of the above code will be:

Thank you for visiting - https://www.tutorialrepublic.com/

The PHP echo statement is often used to display or output data to the web browser. We will learn more about this statement in the next chapter.

By storing the value in a constant instead of a variable, you can make sure that the value won't get changed accidentally when your application runs.

៤.៤ ការដាក់ឈ្មោះ Conventions for PHP Constants

Name of constants must follow the same rules as variable names, which means a valid constant name must starts with a letter or underscore, followed by any number of letters, numbers or underscores with one exception: the $ prefix is not required for constant names.

By convention, constant names are usually written in uppercase letters. This is for their easy identification and differentiation from variables in the source code.

មេរៀនទី៥ Echo និងPrint Statement នៃ PHP

នៅក្នុងឯកសារបង្រៀននេះអ្នកនឹងរៀនពីរបៀបប្រើ echo PHP និង print statement ដើម្បីបង្ហាញលទ្ធផលនៅក្នុងកម្មវិធីរុករកអ៊ីនធឺណិត។

៥.១ The PHP echo Statement

The echo statement can output one or more strings. In general terms, the echo statement can display anything that can be displayed to the browser, such as string, numbers, variables values, the results of expressions etc.

Since echo is a language construct not actually a function (like if statement), you can use it without parentheses e.g. echo or echo(). However, if you want to pass more than one parameter to echo, the parameters must not be enclosed within parentheses.

៥.២ Display Strings of Text

The following example will show you how to display a string of text with the echo statement:

```
<?php
// Displaying string of text
echo "Hello World!";
?>
```

The output of the above PHP code will look something like this:

Hello World!

៥.៣ Display HTML Code

The following example will show you how to display HTML code using the echo statement:

```
<?php
// Displaying HTML code
echo "<h4>This is a simple heading.</h4>";
echo "<h4 style='color: red;'>This is heading with style.</h4>";
?>
```

The output of the above PHP code will look something like this:

This is a simple heading.

This is heading with style.

៥.៤ Display Variables

The following example will show you how to display variable using the echo statement:

```php
<?php
// Defining variables
$txt = "Hello World!";
$num = 123456789;
$colors = array("Red", "Green", "Blue");

// Displaying variables
echo $txt;
echo "<br>";
echo $num;
echo "<br>";
echo $colors[0];
?>
```

The output of the above PHP code will look something like this:

Hello World!

123456789

Red

៥.៥ The PHP print Statement

You can also use the print statement (an alternative to `echo`) to display output to the browser. Like echo the print is also a language construct not a real function. So you can also use it without parentheses like: `print` or `print()`.

Both `echo` and `print` statement works exactly the same way except that the `print` statement can only output one string, and always returns 1. That's why the `echo` statement considered marginally faster than the `print` statement since it doesn't return any value.

៥.៦ Display Strings of Text

The following example will show you how to display a string of text with the print statement:

```
<?php
// Displaying string of text
print "Hello World!";
?>
```

The output of the above PHP code will look something like this:

Hello World!

៥.៧ Display HTML Code

The following example will show you how to display HTML code using the print statement:

```
<?php
// Displaying HTML code
print "<h4>This is a simple heading.</h4>";
print "<h4 style='color: red;'>This is heading with style.</h4>";
?>
```

The output of the above PHP code will look something like this:

This is a simple heading.

This is heading with style.

៥.៨ Display Variables

The following example will show you how to display variable using the print statement:

```
<?php
// Defining variables
$txt = "Hello World!";
$num = 123456789;
$colors = array("Red", "Green", "Blue");

// Displaying variables
print $txt;
print "<br>";
print $num;
print "<br>";
print $colors[0];
?>
```

The output of the above PHP code will look something like this:

Hello World!

123456789

Red

មេរៀនទី៦ ប្រភេទទិន្នន័យ(Data Types)

១. ប្រភេទទិន្នន័យ - Data Types in PHP

The values assigned to a PHP variable may be of different data types including simple string and numeric types to more complex data types like arrays and objects.

PHP supports total eight primitive data types: Integer, Floating point number or Float, String, Booleans, Array, Object, resource and NULL. These data types are used to construct variables. Now let's discuss each one of them in detail.

២.ចំនួនគត់- PHP Integers

Integers are whole numbers, without a decimal point (..., -2, -1, 0, 1, 2, ...). Integers can be specified in decimal (base 10), hexadecimal (base 16 - prefixed with 0x) or octal (base 8 - prefixed with 0) notation, optionally preceded by a sign (- or +).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP Integers</title>
</head>
<body>

    <?php
    $a = 123; // decimal number
    var_dump($a);
    echo "<br>";

    $b = -123; // a negative number
    var_dump($b);
    echo "<br>";

    $c = 0x1A; // hexadecimal number
    var_dump($c);
    echo "<br>";

    $d = 0123; // octal number
    var_dump($d);
    ?>
```

```
</body>
</html>
```

Since PHP 5.4+ you can also specify integers in binary (base 2) notation. To use binary notation precede the number with 0b (e.g. $var = 0b11111111;).

៣.សេរីនៃតួអក្សរ PHP Strings

Strings are sequences of characters, where every character is the same as a byte.

A string can hold letters, numbers, and special characters and it can be as large as up to 2GB (2147483647 bytes maximum). The simplest way to specify a string is to enclose it in single quotes (e.g. 'Hello world!'), however you can also use double quotes ("Hello world!").

```php
<?php
$a = 'Hello world!';
echo $a;
echo "<br>";

$b = "Hello world!";
echo $b;
echo "<br>";

$c = 'Stay here, I\'ll be back.';
echo $c;
?>
```

៣.១ What is String in PHP

A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all. The simplest way to create a string is to enclose the string literal (i.e. string characters) in single quotation marks ('), like this:

```
$my_string = 'Hello World';
```

You can also use double quotation marks ("). However, single and double quotation marks work in different ways. Strings enclosed in single-quotes are treated almost literally, whereas the strings delimited by the double quotes replaces variables with the string representations of their values as well as specially interpreting certain escape sequences. The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Here's an example to clarify the differences between single and double quoted strings:

```
<?php
$my_str = 'World';
echo "Hello, $my_str!<br>";       // Displays: Hello World!
echo 'Hello, $my_str!<br>';       // Displays: Hello, $my_str!

echo '<pre>Hello\tWorld!</pre>'; // Displays: Hello\tWorld!
echo "<pre>Hello\tWorld!</pre>"; // Displays: Hello   World!
echo 'I\'ll be back';            // Displays: I'll be back
?>
```

៣.២ Manipulating PHP Strings

PHP provides many built-in functions for manipulating strings like calculating the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others. Here are the examples of some of these functions.

៣.៣ Calculating the Length of a String

The strlen() function is used to calculate the number of characters inside a string. It also includes the blank spaces inside the string.

```
<?php
```

```
$my_str = 'Welcome to Tutorial Republic';

// Outputs: 28
echo strlen($my_str);
?>
```

៣.៤ Counting Number of Words in a String

The str_word_count() function counts the number of words in a string.

```
<?php
$my_str = 'The quick brown fox jumps over the lazy dog.';

// Outputs: 9
echo str_word_count($my_str);
?>
```

៣.៥ Replacing Text within Strings

The str_replace() replaces all occurrences of the search text within the target string.

```
<?php
  $my_str = 'If the facts do not fit the theory, change the
  facts.';

  // Display replaced string
  echo str_replace("facts", "truth", $my_str);
?>
```

The output of the above code will be:

If the truth do not fit the theory, change the truth.

You can optionally pass the fourth argument to the str_replace() function to know how many times the string replacements was performed, like this.

```
<?php
  $my_str = 'If the facts do not fit the theory, change the
  facts.';

  // Perform string replacement
  str_replace("facts", "truth", $my_str, $count);

  // Display number of replacements performed
  echo "The text was replaced $count times.";
?>
```

The output of the above code will be:

The text was replaced 2 times.

៣.៦ Reversing a String

The strrev() function reverses a string.

```
<?php
$my_str = 'You can do anything, but not everything.';

// Display reversed string
echo strrev($my_str);
?>
```

The output of the above code will be:

.gnihtyreve ton tub ,gnihtyna od nac uoY

៤.ចំនួនទស្សភាគ PHP Floating Point Numbers or Doubles

Floating point numbers (also known as "floats", "doubles", or "real numbers") are decimal or fractional numbers, like demonstrated in the example below.

```
<?php
$a = 1.234;
var_dump($a);
echo "<br>";

$b = 10.2e3;
var_dump($b);
echo "<br>";

$c = 4E-10;
var_dump($c);
?>
```

៥. ប៊ូលីនPHP Booleans

Booleans are like a switch it has only two possible values either 1 (true) or 0 (false).

```
<?php
// Assign the value TRUE to a variable
$show_error = true;
var_dump($show_error);
?>
```

៦.អារ៉ៃ-PHP Arrays

An array is a variable that can hold more than one value at a time. It is useful to aggregate a series of related items together, for example a set of country or city names.

An array is formally defined as an indexed collection of data values. Each index (also known as the key) of an array is unique and references a corresponding value.

```
<?php
$colors = array("Red", "Green", "Blue");
var_dump($colors);
echo "<br>";

$color_codes = array(
    "Red" => "#ff0000",
    "Green" => "#00ff00",
    "Blue" => "#0000ff"
);
var_dump($color_codes);
?>
```

You will learn more about arrays in PHP Array tutorial.

៧.PHP Objects

An object is a data type that not only allows storing data but also information on, how to process that data. An object is a specific instance of a class which serve as templates for objects. Objects are created based on this template via the new keyword.

Every object has properties and methods corresponding to those of its parent class. Every object instance is completely independent, with its own properties and methods, and can thus be manipulated independently of other objects of the same class.

Here's a simple example of a class definition followed by the object creation.

```
<?php
// Class definition
class greeting{
    // properties
    public $str = "Hello World!";
```

```
    // methods
    function show_greeting(){
        return $this->str;
    }
}

// Create object from class
$message = new greeting;
var_dump($message);
?>
```

The data elements stored within an object are referred to as its properties and the information, or code which describing how to process the data is called the methods of the object

៨.PHP NULL

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data. NULL is the only possible value of type null.

```
<?php
$a = NULL;
var_dump($a);
echo "<br>";

$b = "Hello World!";
$b = NULL;
var_dump($b);
```

```
?>
```

When a variable is created without a value in PHP like `$var;` it is automatically assigned a value of null. Many novice PHP developers mistakenly considered both `$var1 = NULL;` and `$var2 = "";` are same, but this is not true. Both variables are different — the `$var1` has null value while `$var2` indicates no value assigned to it.

៩.PHP Resources

A resource is a special variable, holding a reference to an external resource.

Resource variables typically hold special handlers to opened files and database connections.

```
<?php
// Open a file for reading
$handle = fopen("note.txt", "r");
var_dump($handle);
echo "<br>";

// Connect to MySQL database server with default setting
$link = mysqli_connect("localhost", "root", "");
var_dump($link);
?>
```

មេរៀនទី៧ ប្រមាណវិធី Operators

ក្នុងមេរៀននេះអ្នកនឹងរៀនពីរបៀបគណនាជាមួយនឹងអថេរ ហើយនឹងតម្លៃនៃប្រមាណវិធី។

១.What is Operators in PHP

ប្រតិបត្តិករ(Operators)គឺជានិមិត្តសញ្ញាដែលប្រាប់ឱ្យដំណើរការរបស់ PHP អនុវត្តសកម្មភាពជាក់លាក់។ ឧទាហរណ៍និមិត្តសញ្ញាបន្ថែម (+) គឺជាប្រតិបត្តិករដែលប្រាប់ PHP ឱ្យបន្ថែមអថេរឬតម្លៃពីរ ចំណែកសញ្ញាធំជាង (>) គឺជាសញ្ញាប្រមាណវិធីដែលប្រាប់ PHP ឲ្យ ប្រៀបធៀបតម្លៃពីរ។

តារាងខាងក្រោមពិពណ៌នាអំពីសញ្ញាប្រមាណវិធីផ្សេងៗគ្នាដែលប្រើក្នុង PHP ។

២.PHP Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of PHP's arithmetic operators:

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

The following example will show you these arithmetic operators in action:

```
<?php
$x = 10;
$y = 4;
echo($x + $y); // Outputs: 14
echo($x - $y); // Outputs: 6
echo($x * $y); // Outputs: 40
echo($x / $y); // Outputs: 2.5
echo($x % $y); // Outputs: 2
?>
```

៣.PHP Assignment Operators

The assignment operators are used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	$x = $y	$x = $y
+=	Add and assign	$x += $y	$x = $x + $y
-=	Subtract and assign	$x -= $y	$x = $x - $y
*=	Multiply and assign	$x *= $y	$x = $x * $y
/=	Divide and assign quotient	$x /= $y	$x = $x / $y
%=	Divide and assign modulus	$x %= $y	$x = $x % $y

The following example will show you these assignment operators in action:

```php
<?php
  $x = 10;
  echo $x; // Outputs: 10

  $x = 20;
  $x += 30;
  echo $x; // Outputs: 50

  $x = 50;
  $x -= 20;
  echo $x; // Outputs: 30

  $x = 5;
  $x *= 25;
  echo $x; // Outputs: 125

  $x = 50;
  $x /= 10;
  echo $x; // Outputs: 5

  $x = 100;
  $x %= 15;
  echo $x; // Outputs: 10
?>
```

៤.PHP Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
<	Less than	$x < $y	True if $x is less than $y
>	Greater than	$x > $y	True if $x is greater than $y
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

The following example will show you these comparison operators in action:

```php
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z);   // Outputs: boolean true
var_dump($x === $z);  // Outputs: boolean false
var_dump($x != $y);   // Outputs: boolean true
var_dump($x !== $z);  // Outputs: boolean true
var_dump($x < $y);    // Outputs: boolean true
var_dump($x > $y);    // Outputs: boolean false
var_dump($x <= $y);   // Outputs: boolean true
var_dump($x >= $y);   // Outputs: boolean false
?>
```

៥.PHP Incrementing and Decrementing Operators

The increment/decrement operators are used to increment/decrement a variable's value.

Operator	Name	Effect
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

The following example will show you these increment and decrement operators in action:

```
<?php
$x = 10;
echo ++$x; // Outputs: 11
echo $x;   // Outputs: 11

$x = 10;
echo $x++; // Outputs: 10
echo $x;   // Outputs: 11

$x = 10;
echo --$x; // Outputs: 9
echo $x;   // Outputs: 9

$x = 10;
echo $x--; // Outputs: 10
echo $x;   // Outputs: 9
?>
```

៦.PHP Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true

xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $$x or $y is true
!	Not	!$x	True if $x is not true

The following example will show you these logical operators in action:

```
<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))){
    echo "$year is a leap year.";
} else{
    echo "$year is not a leap year.";
}
?>
```

៧.PHP String Operators

There are two operators which are specifically designed for strings.

Operator	Description	Example	Result
.	Concatenation	$str1 . $str2	Concatenation of $str1 and $str2
.=	Concatenation assignment	$str1 .= $str2	Appends the $str2 to the $str1

The following example will show you these string operators in action:

```
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!

$x .= $y;
echo $x; // Outputs: Hello World!
?>
```

៨.PHP Array Operators

The array operators are used to compare arrays:

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	True if $x and $y have the same key/value pairs
===	Identity	$x === $y	True if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	True if $x is not equal to $y
<>	Inequality	$x <> $y	True if $x is not equal to $y
!==	Non-identity	$x !== $y	True if $x is not identical to $y

The following example will show you these array operators in action:

```php
<?php
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
$z = $x + $y; // Union of $x and $y
var_dump($z);
var_dump($x == $y);    // Outputs: boolean false
var_dump($x === $y);   // Outputs: boolean false
var_dump($x != $y);    // Outputs: boolean true
var_dump($x <> $y);    // Outputs: boolean true
var_dump($x !== $y);   // Outputs: boolean true
?>
```

៩.PHP Spaceship Operator PHP 7

PHP 7 introduces a new spaceship operator (`<=>`) which can be used for comparing two expressions. It is also known as combined comparison operator.

The spaceship operator returns `0` if both operands are equal, `1` if the left is greater, and `-1` if the right is greater. It basically provides three-way comparison as shown in the following table:

Operator	<=> Equivalent
$x < $y	($x <=> $y) === -1
$x <= $y	($x <=> $y) === -1 \|\| ($x <=> $y) === 0
$x == $y	($x <=> $y) === 0
$x != $y	($x <=> $y) !== 0
$x >= $y	($x <=> $y) === 1 \|\| ($x <=> $y) === 0
$x > $y	($x <=> $y) === 1

The following example will show you how spaceship operator actually works:

```php
<?php
// Comparing Integers
echo 1 <=> 1; // Outputs: 0
echo 1 <=> 2; // Outputs: -1
echo 2 <=> 1; // Outputs: 1

// Comparing Floats
echo 1.5 <=> 1.5; // Outputs: 0
echo 1.5 <=> 2.5; // Outputs: -1
echo 2.5 <=> 1.5; // Outputs: 1

// Comparing Strings
echo "x" <=> "x"; // Outputs: 0
echo "x" <=> "y"; // Outputs: -1
echo "y" <=> "x"; // Outputs: 1
?>
```

មេរៀន ទី៨ ការគ្រប់គ្រងឃ្លាបញ្ជា
(Control Statement)

១. ឃ្លាបញ្ជាលក្ខ័ណ្ឌ If...Else Statements

ដូចជាភាសាសរសេរកម្មវិធីភាគច្រើនដែរកម្មវិធី PHP ក៏អនុញ្ញាតឱ្យអ្នកសរសេរកូដដែលអនុវត្តសកម្មភាពផ្សេងៗគ្នាដោយផ្អែកលើលទ្ធផលនៃលក្ខខណ្ឌសាកល្បងឡូជីខល(logical) ឬប្រៀបធៀបនៅពេលរត់(run time)។ នេះមានន័យថាអ្នកអាចបង្កើតលក្ខខ័ណ្ឌសាកល្បងក្នុងទំរង់នៃកន្សោមដែលវាយតំលៃថាពិត(true) រឺមិនពិត(false) ហើយផ្អែកលើលទ្ធផលទាំងនេះអ្នកអាចធ្វើសកម្មភាពជាក់លាក់។

មានសេចក្តីថ្លែងការណ៍ជាច្រើននៅក្នុង PHP ដែលអ្នកអាចប្រើដើម្បីធ្វើការសម្រេចចិត្ត:

- ឃ្លារបញ្ជា **if** statement
- ឃ្លារបញ្ជា **if...else** statement
- ឃ្លារបញ្ជា **if...elseif....else** statement
- ឃ្លារបញ្ជា **switch...case** statement

យើងនឹងស្វែងយល់ពីសេចក្តីថ្លែងទាំងនេះនីមួយៗនៅក្នុងផ្នែកបន្ទាប់។

១.១ ឃ្លារបញ្ជា if Statement

The *if* statement is used to execute a block of code only if the specified condition evaluates to true. This is the simplest PHP's conditional statements and can be written like:

```
if(condition){
    // Code to be executed
}
```

The following example will output "Have a nice weekend!" if the current day is Friday:

```
<?php
```

```
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
}
?>
```

១.២ ឃ្លារបញ្ជា if...else Statement

You can enhance the decision making process by providing an alternative choice through adding an *else* statement to the *if* statement. The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false. It can be written, like this:

```
if(condition){
    // Code to be executed if condition is true
} else{
    // Code to be executed if condition is false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!"

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} else{
    echo "Have a nice day!";
}
?>
```

១.៣ ឃ្លារបញ្ជា if...elseif...else Statement

The *if...elseif...else* a special statement that is used to combine multiple *if...else* statements.

```
if(condition1){
    // Code to be executed if condition1 is true
} elseif(condition2){
    // Code to be executed if the condition1 is false and
condition2 is true
} else{
```

```
    // Code to be executed if both condition1 and
condition2 are false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday, otherwise it will output "Have a nice day!"

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} elseif($d == "Sun"){
    echo "Have a nice Sunday!";
} else{
    echo "Have a nice day!";
}
?>
```

You will learn about PHP switch-case statement in the next chapter.

១.៤ ប្រតិបត្តិករ Ternary

The ternary operator provides a shorthand way of writing the *if...else* statements. The ternary operator is represented by the question mark (?) symbol and it takes three operands: a condition to check, a result for true, and a result for false.

To understand how this operator works, consider the following examples:

```
<?php
 if($age < 18){
     echo 'Child'; // Display Child if age is less than 18
 } else{
     echo 'Adult'; // Display Adult if age is greater than or
 equal to 18
 }
?>
```

Using the ternary operator the same code could be written in a more compact way:

```
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

The ternary operator in the example above selects the value on the left of the colon (i.e. 'Child') if the condition evaluates to true (i.e. if $age is less than 18), and selects the value on the right of the colon (i.e. 'Adult') if the condition evaluates to false.

Code written using the ternary operator can be hard to read. However, it provides a great way to write compact if-else statements.

១.៥ ប្រមាណវិធី Null Coalescing ក្នុង PHP 7

PHP 7 introduces a new null coalescing operator (??) which you can use as a shorthand where you need to use a ternary operator in conjunction with isset() function. To uderstand this in a better way consider the following line of code. It fetches the value of $_GET['name'], if it does not exist or NULL, it returns 'anonymous'.

```
<?php
$name = isset($_GET['name']) ? $_GET['name'] : 'anonymous';
?>
```

Using the null coalescing operator the same code could be written as:

```
<?php
$name = $_GET['name'] ?? 'anonymous';
?>
```

As you can see the later syntax is more compact and easy to write.

២. ឃ្លារបញ្ជាSwitch…Case Statements

ឃ្លារបញ្ជា If…Else Vs Switch…Case

The switch-case statement is an alternative to the if-elseif-else statement, which does almost the same thing. The switch-case statement tests a variable against a series of values until it finds a match, and then executes the block of code corresponding to that match.

```
switch(n){
    case label1:
        // Code to be executed if n=label1
        break;
    case label2:
        // Code to be executed if n=label2
        break;
    ...
    default:
```

```
        // Code to be executed if n is different from all labels
}
```

Consider the following example, which display a different message for each day.

```
<?php
$today = date("D");
switch($today){
    case "Mon":
        echo "Today is Monday. Clean your house.";
        break;
    case "Tue":
        echo "Today is Tuesday. Buy some food.";
        break;
    case "Wed":
        echo "Today is Wednesday. Visit a doctor.";
        break;
    case "Thu":
        echo "Today is Thursday. Repair your car.";
        break;
    case "Fri":
        echo "Today is Friday. Party tonight.";
        break;
    case "Sat":
        echo "Today is Saturday. Its movie time.";
        break;
    case "Sun":
        echo "Today is Sunday. Do some rest.";
        break;
    default:
        echo "No information available for that day.";
        break;
}
?>
```

The switch-case statement differs from the if-elseif-else statement in one important way. The switch statement executes line by line (i.e. statement by statement) and once PHP finds a case statement that evaluates to true, it's not only executes the code corresponding to that case statement, but also executes all the subsequent case statements till the end of the switch block automatically.

To prevent this add a break statement to the end of each case block. The break statement tells PHP to break out of the switch-case statement block once it executes the code associated with the first true case.

មេរៀន ទី៩ សេរីឥន្ទន៍យ
(Array)

១.តើសេរីឥន្ទន៍យក្នុង PHP ជាអ្វី?

Arrays គឺជាអថេរស្មុគស្មាញដែលអនុញ្ញាតឱ្យយើងរក្សាទុកតម្លៃច្រើនជាងមួយរឺក្រុមនៃតំលៃក្រោមឈ្មោះអថេរតែមួយ។ ឧបមាថាអ្នកចង់ទុកពណ៌នៅក្នុងស្ក្រីប PHP របស់អ្នក។ ការរក្សាទុកពណ៌ម្តងមួយក្នុងមួយអថេរអាចមើលទៅដូចនេះ៖

```
<?php
    $color1 = "Red";
    $color2 = "Green";
    $color3 = "Blue";
?>
```

ប៉ុន្តែអ្វីដែល, ប្រសិនបើអ្នកចង់រក្សាទុកឈ្មោះរដ្ឋឬទីក្រុងនៃប្រទេសមួយនៅក្នុងអថេរហើយពេលនេះមិនមែនត្រឹមតែបីទេដែលអាចរាប់រយ។ វាពិតជាលំបាកគួរឱ្យធុញទ្រាន់និងគំនិតអាក្រក់ក្នុងការរក្សាទុកឈ្មោះទីក្រុងនីមួយៗក្នុងអថេរដាច់ដោយឡែក។

២. ប្រភេទនៃ អារេ ក្នុង PHP - Types of Arrays in PHP

យើងអាចបង្កើត Array បានបីយ៉ាង ។ ខាងក្រោមគឺជាប្រភេទនៃ Array៖

- **Indexed array** — An array with a numeric key.
- **Associative array** — An array where each key has its own specific value.
- **Multidimensional array** — An array containing one or more arrays within itself.

២.១ Indexed Arrays

An indexed or numeric array stores each array element with a numeric index. The following examples shows two ways of creating an indexed array, the easiest way is:

```php
<?php
// Define an indexed array
$colors = array("Red", "Green", "Blue");
?>
```

In an indexed or numeric array, the indexes are automatically assigned and start with 0, and the values can be any data type

This is equivalent to the following example, in which indexes are assigned manually:

```php
<?php
$colors[0] = "Red";
$colors[1] = "Green";
$colors[2] = "Blue";
?>
```

២.២ Associative Arrays

In an associative array, the keys assigned to values can be arbitrary and user defined strings. In the following example the array uses keys instead of index numbers:

```php
<?php
// Define an associative array
$ages = array("Peter"=>22, "Clark"=>32, "John"=>28);
?>
```

The following example is equivalent to the previous example, but shows a different way of creating associative arrays:

```
<?php
    $ages["Peter"] = "22";
    $ages["Clark"] = "32";
    $ages["John"] = "28";
?>
```

២.៣ Multidimensional Arrays

The multidimensional array is an array in which each element can also be an array and each element in the sub-array can be an array or further contain array within itself and so on. An example of a multidimensional array will look something like this:

```
<?php
// Define a multidimensional array
$contacts = array(
    array(
        "name" => "Peter Parker",
        "email" => "peterparker@mail.com",
    ),
    array(
        "name" => "Clark Kent",
        "email" => "clarkkent@mail.com",
    ),
    array(
        "name" => "Harry Potter",
        "email" => "harrypotter@mail.com",
    )
);
// Access nested value
echo "Peter Parker's Email-id is: " . $contacts[0]["email"];
?>
```

៣. Viewing Array Structure and Values

You can see the structure and values of any array by using one of two statements — var_dump() or print_r(). The print_r() statement, however, gives somewhat less information. Consider the following example:

```
<?php
// Define array
$cities = array("London", "Paris", "New York");

// Display the cities array
print_r($cities);
?>
```

The print_r() statement gives the following output:

```
Array ( [0] => London [1] => Paris [2] => New York )
```

This output shows the key and the value for each element in the array. To get more information, use the following statement:

```
<?php
// Define array
$cities = array("London", "Paris", "New York");

// Display the cities array
var_dump($cities);
?>
```

This var_dump() statement gives the following output:

```
array(3) { [0]=> string(6) "London" [1]=> string(5) "Paris" [2]=> string(8) "New York" }
```

This output shows the data type of each element, such as a string of 6 characters, in addition to the key and value. In the next chapter you will learn how to sort array elements.

You will learn how to loop through the values of an array in the later chapter.

៤. អនុគមន៍ នៃអារេ - PHP Array Functions

Here is a complete list of array functions belonging to the latest PHP 7. These functions are the part of the PHP core so you can use them within your script without any further installation.

Also, deprecated function such as each() is not included in the list.

array_change_key_case()	Changes the case of all keys in an array to lowercase or uppercase.
array_chunk()	Splits an array into chunks of arrays.
array_column()	Returns the values from a single column in the input array.
array_combine()	Creates an array by using one array for keys and another for its values.
array_count_values()	Counts all the distinct values of an array.
array_diff()	Compare arrays values, and returns the differences.
array_diff_assoc()	Compare arrays values, with additional key check, and returns the differences.
array_diff_key()	Compare arrays keys, and returns the differences.
array_diff_uassoc()	Compare arrays values, with additional key check using a user-defined key comparison function, and returns the differences.
array_diff_ukey()	Compare array keys, using a user-defined key comparison function, and returns the differences.
array_fill()	Fills an array with values.
array_fill_keys()	Fills an array with values, specifying keys.

array_filter()	Filters elements of an array using a user-defined function.
array_flip()	Flips or exchanges all keys with their associated values in an array.
array_intersect()	Compare arrays values, and returns the matches.
array_intersect_assoc()	Compare arrays values, with additional key check, and returns the matches.
array_intersect_key()	Compare arrays keys, and returns the matches.
array_intersect_uassoc()	Compare arrays values, with additional key check using a user-defined key comparison function, and returns the matches.
array_intersect_ukey()	Compare arrays keys, using a user-defined key comparison function, and returns the matches.
array_key_exists()	Checks if the specified key exists in the array.
array_key_first()	Gets the first key of an array.
array_key_last()	Gets the last key of an array.
array_keys()	Returns all the keys or a subsets of the keys of an array.
array_map()	Sends the elements of the given arrays to a user-defined function, which may use it to returns new values.
array_merge()	Merges one or more arrays into one array.
array_merge_recursive()	Merges one or more arrays into one array recursively.
array_multisort()	Sorts multiple or multi-dimensional arrays.
array_pad()	Inserts a specified number of items, with a specified value, to an array.

array_pop()	Removes the last element of an array, and returns the value of the removed element.
array_product()	Calculates the product of the values in an array.
array_push()	Inserts one or more elements to the end of an array.
array_rand()	Returns one or more random keys from an array.
array_reduce()	Reduce the array to a single value by using a user-defined callback function.
array_replace()	Replaces the values of the first array with the values from following arrays.
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively.
array_reverse()	Returns an array with elements in reverse order.
array_search()	Searches an array for a given value and returns the corresponding key if successful.
array_shift()	Removes the first element from an array, and returns the value of the removed element.
array_slice()	Extract a slice from an array.
array_splice()	Removes a portion of the array and replace it with something else.
array_sum()	Calculates the sum of values in an array.
array_udiff()	Compares arrays values using a user-defined comparison callback function, and returns the differences.

array_udiff_assoc()	Compares arrays values using a user-defined comparison callback function, with additional key check, and returns the differences.
array_udiff_uassoc()	Compares the keys and values of arrays using two separate user-defined key and value comparison callback functions, and returns the differences.
array_uintersect()	Compares arrays values using a user-defined comparison callback function, and returns the matches.
array_uintersect_assoc()	Compares arrays values using a user-defined comparison callback function, with additional key check, and returns the matches.
array_uintersect_uassoc()	Compares the keys and values of arrays using two separate user-defined key and value comparison callback functions, and returns the matches.
array_unique()	Removes duplicate values from an array.
array_unshift()	Adds one or more elements to the beginning of an array.
array_values()	Returns all the values of an array.
array_walk()	Applies a user-defined function to each element of an array.
array_walk_recursive()	Applies a user-defined function recursively to each element of an array.
array()	Creates an array.
arsort()	Sorts an associative array by value, in reverse or descending order.
asort()	Sorts an associative array by value, in ascending order.

compact()	Creates array containing variables and their values.
count()	Returns the number of elements in an array.
current()	Returns the current element in an array.
end()	Sets the internal pointer of an array to its last element.
extract()	Import variables into the current symbol table from an array.
in_array()	Checks if a value exists in an array.
key()	Fetches a key from an array.
krsort()	Sorts an associative array by key, in reverse or descending order.
ksort()	Sorts an associative array by key, in ascending order.
list()	Assign variables as if they were an array.
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm.
natsort()	Sorts an array using a "natural order" algorithm.
next()	Advance the internal array pointer of an array.
pos()	Returns the current element in an array. Alias of current() function.
prev()	Rewind the internal array pointer.
range()	Creates an array containing a range of elements.
reset()	Sets the internal pointer of an array to its first element.
rsort()	Sorts an array in reverse or descending order.

shuffle()	Shuffles or randomizes the order of the elements in an array.
sizeof()	Returns the number of elements in an array. Alias of count() function.
sort()	Sorts an array in ascending order.
uasort()	Sorts an array using a user-defined comparison function and maintain index association.
uksort()	Sorts an array by keys using a user-defined comparison function.
usort()	Sorts an array by values using a user-defined comparison function.

៤.១ PHP array_change_key_case() Function

The array_change_key_case() function is used to change the case of all keys in an array to lowercase or uppercase. Numbered indices are left as is.

The following table summarizes the technical details of this function.

Return Value:	Returns an array with its keys lower or uppercased, or FALSE if array is not an array.
Version:	PHP 4.2+

Syntax

The basic syntax of the array_change_key_case() function is given with:

```
array_change_key_case(array, case)
```

The following example shows the array_change_key_case() function in action.

```
<?php
// Sample array
$persons = array("Harry"=>22, "Clark"=>32, "John"=>28);

// Changing keys to uppercase
print_r(array_change_key_case($persons, CASE_UPPER));
?>
```

Parameters

The `array_change_key_case()` function accepts the following parameters.

Parameter	Description
array	Required. Specifies the array to work on.
case	Optional. Specifies the case. Possible values are: `CASE_LOWER` – Changes the keys to lowercase. This is default. `CASE_UPPER` – Changes the keys to uppercase.

If you do not specify the case parameter inside the array_change_key_case() function, all keys are converted to lowercase letter, because CASE_LOWER is the default case value.

More Examples

Here're some more examples showing how `array_change_key_case()` function basically works:

In the following example the array keys are converted to lowercase letters:

```
<?php
// Sample array
$persons = array("Harry"=>22, "Clark"=>32, "John"=>28);

// Changing keys to lowercase
print_r(array_change_key_case($persons));
?>
```

If two or more keys will be same after running `array_change_key_case()` (e.g. "keY" and "kEY"), the value that is later in the array will override the previous ones.

```
<?php
// Sample array
$persons = array("Harry"=>22, "Clark"=>32, "harry"=>28);
```

```
// Changing keys to lowercase
print_r(array_change_key_case($persons));
?>
```

៤.២ PHP array_chunk() Function

Description

The array_chunk() function splits an array into chunks.

The following table summarizes the technical details of this function.

Return Value:	Returns a multidimensional numerically indexed array, starting with zero, with each dimension containing size elements.
Version:	PHP 4.2+

Syntax

The basic syntax of the `array_chunk()` function is given with:

```
array_chunk(array, size, preserve_keys)
```

The following example shows the `array_chunk()` function in action.

```
<?php
// Sample array
$colors = array("red", "green", "blue", "orange", "yellow",
"black");

// Split colors array into chunks
print_r(array_chunk($colors, 2));
?>
```

Parameters

The `array_chunk()` function accepts the following parameters.

Parameter	Description
array	Required. Specifies the array to work on.
size	Required. A positive integer (greater than 0) which specifies the size of each chunk.

If you do not specify the preserve_keys parameter inside the array_chunk() function the chunks will be reindexed numerically, because the default value of this parameter is FALSE.

preserve_keys	Optional. Specifies whether to preserve the origial keys or not. When set to `TRUE` the keys will be preserved. Default is `FALSE` which will reindex the chunk numerically.

More Examples

Here're some more examples showing how `array_chunk()` function basically works:

The following example will split an array into chunks of two while preserving the original keys:

```
<?php
	// Sample array
	$alphabets = array("a"=>"apple", "b"=>"ball", "c"=>"cat",
	"d"=>"dog");

	// Split alphabets array into chunks
	print_r(array_chunk($alphabets, 2, true));
?>
```

៤.៣ PHP array_combine() Function

The `array_combine()` function creates an array by using one array for keys and another for its values.

The following table summarizes the technical details of this function.

Return Value:	Returns the combined array, `FALSE` if the number of elements for each array isn't equal.
Changelog:	Versions before PHP 5.4.0 issues `E_WARNING` and returns `FALSE` for empty arrays.
Version:	PHP 5+

Syntax

The basic syntax of the array_combine() function is given with:

```
array_combine(keys, values)
```

The following example shows the array_combine() function in action.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Combining Two Arrays in PHP</title>
</head>
<body>
<pre>

    <?php
        // Sample arrays
        $array1 = array("a", "b", "c", "d");
        $array2 = array("apple", "ball", "cat", "dog");

        // Combining both arrays
        print_r(array_combine($array1, $array2));
    ?>

</pre>
</body>
</html>
```

Parameters

The array_combine() function accepts the following parameters.

Parameter	Description
keys	Required. Specifies the array of keys to be used.
values	Required. Specifies the array of values to be used.

Both the arrays you want to combine using the *array_combine()* function must have equal number of elements, otherwise it returns FALSE.

More Examples

Here're some more examples showing how `array_combine()` function basically works:

If the array you want to use for keys has duplicate values, the later value will prevail as key in the combined array, as shown in the following example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>When Keys Array has Duplicate Values in PHP</title>
</head>
<body>
<pre>

<?php
// Sample arrays
$array1 = array("a", "a", "b", "c");
$array2 = array(1, 2, 3, 4);

// Combining both arrays
print_r(array_combine($array1, $array2));
?>

</pre>
</body>
</html>
```

៤.៤PHP array_merge() Function

Description

The array_merge() function merge one or more arrays into one array.

This function merges the elements of one or more arrays together in such a way that the values of one are appended to the end of the previous one. It returns a new array with merged elements.

The following table summarizes the technical details of this function.

Return Value:	Returns the merged array.
Changelog:	Since PHP 5.0, this function only accept parameters of type array.
Version:	PHP 4+

Syntax

The basic syntax of the `array_merge()` function is given with:

```
array_merge(array1, array2, ...)
```

The following example shows the `array_merge()` function in action.

```
<?php
// Sample arrays
$array1 = array("red", "green", "blue");
$array2 = array("apple", "banana");

// Merging the two indexed array
$result = array_merge($array1, $array2);
print_r($result);
?>
```

Similarly, you can merge two or more associative arrays as shown in the example below:

```
<?php
// Sample arrays
$array1 = array("a" => "apple", "b" => "ball", "c" => "cat");
$array2 = array("x" => "xylophone", "y" => "yacht", "z" => "zebra");

// Merging the two associative arrays
$result = array_merge($array1, $array2);
print_r($result);
?>
```

It is also possible to merge indexed and associative array, as shown here:

```php
<?php
// Sample arrays
$array1 = array(4, 6, 12, 18);
$array2 = array("a"=>"apple", "b"=>"ball", "c"=>"cat");

// Merging two indexed arrays
$result = array_merge($array1, $array2);
print_r($result);
?>
```

Parameters

The `array_merge()` function accepts the following parameters.

Parameter	Description
array1	Optional. Specifies the first array to merge.
array2	Optional. Specifies the second array to merge.
...	Optional. Specifies more arrays to merge.

As of PHP version 7.4.0 the array_merge() function can now be called without any parameter. Earlier, at least one parameter has been required.

More Examples

Here're some more examples showing how `array_merge()` function basically works:

If the input arrays have the same string keys, then the later value for that key will overwrite the previous one, as you can see in the following example:

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>PHP Merging Two Arrays Having Same String Keys</title>
</head>
<body>
<pre>

<?php
// Sample arrays
$array1 = array("a"=>"red", "b"=>"green", "c"=>"blue");
$array2 = array("a"=>"apple", "m"=>"mango", "o"=>"orange");

// Merging two associative arrays
$result = array_merge($array1, $array2);
print_r($result);
?>

</pre>
</body>
</html>
```

```
Array
(
    [a] => apple
    [b] => green
    [c] => blue
    [m] => mango
    [o] => orange
)
```

However, if the arrays contain numeric keys, the later value will not overwrite the original value, but will be appended, as demonstrated in the following example:

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>PHP Merging Two Arrays Having Numeric Keys</title>
</head>
<body>
<pre>

<?php
// Sample arrays
$array1 = array(5, 10, 15, 20);
$array2 = array(10, 30, 50, 80);

// Merging two indexed arrays
$result = array_merge($array1, $array2);
print_r($result);
?>

</pre>
</body>
</html>
```

```
Array
(
    [0] => 5
    [1] => 10
    [2] => 15
    [3] => 20
    [4] => 10
    [5] => 30
    [6] => 50
    [7] => 80
)
```

៥.តំរៀបអារេPHP Sorting Arrays

៥.១ PHP Functions For Sorting Arrays

In the previous chapter you've learnt the essentials of PHP arrays i.e. what arrays are, how to create them, how to view their structure, how to access their elements etc. You can do even more things with arrays like sorting the elements in any order you like.

PHP comes with a number of built-in functions designed specifically for sorting array elements in different ways like alphabetically or numerically in ascending or descending order. Here we'll explore some of these functions most commonly used for sorting arrays.

- `sort()` and `rsort()` — For sorting indexed arrays
- `asort()` and `arsort()` — For sorting associative arrays by value
- `ksort()` and `krsort()` — For sorting associative arrays by key

៥.២ Sorting Indexed Arrays in Ascending Order

The `sort()` function is used for sorting the elements of the indexed array in ascending order (alphabetically for letters and numerically for numbers).

```
<?php
// Define array
$colors = array("Red", "Green", "Blue", "Yellow");

// Sorting and printing array
sort($colors);
print_r($colors);
?>
```

This `print_r()` statement gives the following output:

Array ([0] => Blue [1] => Green [2] => Red [3] => Yellow)

Similarly you can sort the numeric elements of the array in ascending order.

```
<?php
// Define array
$numbers = array(1, 2, 2.5, 4, 7, 10);

// Sorting and printing array
sort($numbers);
print_r($numbers);
?>
```

This `print_r()` statement gives the following output:

Array ([0] => 1 [1] => 2 [2] => 2.5 [3] => 4 [4] => 7 [5] => 10)

៥.៣ Sorting Indexed Arrays in Descending Order

The `rsort()` function is used for sorting the elements of the indexed array in descending order (alphabetically for letters and numerically for numbers).

```
<?php
// Define array
$colors = array("Red", "Green", "Blue", "Yellow");

// Sorting and printing array
rsort($colors);
print_r($colors);
?>
```

This `print_r()` statement gives the following output:

Array ([0] => Yellow [1] => Red [2] => Green [3] => Blue)

Similarly you can sort the numeric elements of the array in descending order.

```
<?php
// Define array
$numbers = array(1, 2, 2.5, 4, 7, 10);

// Sorting and printing array
rsort($numbers);
print_r($numbers);
?>
```

This `print_r()` statement gives the following output:
Array ([0] => 10 [1] => 7 [2] => 4 [3] => 2.5 [4] => 2 [5] => 1)

៥.៤ Sorting Associative Arrays in Ascending Order By Value

The `asort()` function sorts the elements of an associative array in ascending order according to the value. It works just like `sort()`, but it preserves the association between keys and its values while sorting.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by value and print
asort($age);
```

```
print_r($age);
?>
```

This `print_r()` statement gives the following output:

Array ([Harry] => 14 [Peter] => 20 [Clark] => 35 [John] => 45)

៥.៥ Sorting Associative Arrays in Descending Order By Value

The `arsort()` function sorts the elements of an associative array in descending order according to the value. It works just like `rsort()`, but it preserves the association between keys and its values while sorting.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by value and print
arsort($age);
print_r($age);
?>
```

This `print_r()` statement gives the following output:

Array ([John] => 45 [Clark] => 35 [Peter] => 20 [Harry] => 14)

៥.៦ Sorting Associative Arrays in Ascending Order By Key

The `ksort()` function sorts the elements of an associative array in ascending order by their keys. It preserves the association between keys and its values while sorting, same as `asort()` function.

```
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by key and print
ksort($age);
print_r($age);
?>
```

This `print_r()` statement gives the following output:

Array ([Clark] => 35 [Harry] => 14 [John] => 45 [Peter] => 20)

៥.៧ Sorting Associative Arrays in Descending Order By Key

The `krsort()` function sorts the elements of an associative array in descending order by their keys. It preserves the association between keys and its values while sorting, same as `arsort()` function.

```php
<?php
// Define array
$age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);

// Sorting array by key and print
krsort($age);
print_r($age);
?>
```

This `print_r()` statement gives the following output:

Array ([Peter] => 20 [John] => 45 [Harry] => 14 [Clark] => 35)

មេរៀន ទី១០ ឃ្លារបញ្ជារវិលជុំ

(Loop Statement)

នៅក្នុងការបង្រៀននេះអ្នកនឹងរៀនពីរបៀបធ្វើសេរីឡើងវិញនៃសកម្មភាពដោយប្រើរង្វិលជុំនៅក្នុង PHP ។

១.ប្រភេទខុសគ្នានៃ រង្វិលជុំ

រង្វិលជុំត្រូវបានប្រើដើម្បីប្រតិបត្តិប្លុកលេខកូដដដែលម្តងហើយម្តងទៀតដរាបណាលក្ខខណ្ឌជាក់លាក់ត្រូវបានបំពេញ។ គំនិតជាមូលដ្ឋាននៅពីក្រោយរង្វិលជុំគឺស្វ័យប្រវត្តិភារកិច្ចច្រំដែលនៅក្នុងកម្មវិធីដើម្បីសន្សំពេលវេលានិងការខំប្រឹងប្រែង។ កម្មវិធី PHPគាំទ្ររង្វិលជុំ ៤ប្រភេទផ្សេងៗគ្នា។

- **while** — loops through a block of code as long as the condition specified evaluates to true.
- **do...while** — the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.
- **for** — loops through a block of code until the counter reaches a specified number.
- **foreach** — loops through a block of code for each element in an array.

You will also learn how to loop through the values of array using `foreach()` loop at the end of this chapter. The `foreach()` loop work specifically with arrays.

២.ឃ្លារបញ្ជារឡូលជុំ while Loop

The `while` statement will loops through a block of code as long as the condition specified in the `while` statement evaluate to true.

```
while(condition){
    // Code to be executed
}
```

The example below define a loop that starts with `$i=1`. The loop will continue to run as long as `$i` is less than or equal to 3. The `$i` will increase by 1 each time the loop runs:

```php
<?php
$i = 1;
while($i <= 3){
    $i++;
    echo "The number is " . $i . "<br>";
}
?>
```

៣.ប្លុកបញ្ជារង្វិលដុំ do...while Loop

The `do-while` loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a `do-while` loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

```php
do{
    // Code to be executed
}
while(condition);
```

The following example define a loop that starts with `$i=1`. It will then increase `$i` with 1, and print the output. Then the condition is evaluated, and the loop will continue to run as long as `$i` is less than, or equal to 3.

```php
<?php
    $i = 1;
    do{
        $i++;
        echo "The number is " . $i . "<br>";
    }
    while($i <= 3);
?>
```

៤.តាងខុសគ្នានៃ while និង do...while loop

The `while` loop differs from the `do-while` loop in one important way — with a `while` loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a `do-while` loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

៥.ប្លុកបញ្ជារង្វិលជុំ for Loop

The `for` loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times.

```
for(initialization; condition; increment){
    // Code to be executed
}
```

The parameters of `for` loop have following meanings:

- `initialization` — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.
- `condition` — in the beginning of each iteration, condition is evaluated. If it evaluates to `true`, the loop continues and the nested statements are executed. If it evaluates to `false`, the execution of the loop ends.
- `increment` — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

The example below defines a loop that starts with `$i=1`. The loop will continued until `$i` is less than, or equal to 3. The variable `$i` will increase by 1 each time the loop runs:

```
<?php
    for($i=1; $i<=3; $i++){
        echo "The number is " . $i . "<br>";
    }
?>
```

៦.ប្លុកបញ្ជារង្វិលជុំ foreach Loop

The `foreach` loop is used to iterate over arrays.

```
foreach($array as $value){
    // Code to be executed
}
```

The following example demonstrates a loop that will print the values of the given array:

```php
<?php
$colors = array("Red", "Green", "Blue");

// Loop through colors array
foreach($colors as $value){
    echo $value . "<br>";
}
?>
```

There is one more syntax of `foreach` loop, which is extension of the first.

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

```php
<?php
$superhero = array(
    "name" => "Peter Parker",
    "email" => "peterparker@mail.com",
    "age" => 18
);

// Loop through superhero array
foreach($superhero as $key => $value){
    echo $key . " : " . $value . "<br>";
}
?>
```

មេរៀន ទី១១ អនុគមន៍ក្នុង PHP
(PHP Functions)

១. PHP Built-in Functions

A function is a self-contained block of code that performs a specific task.

PHP has a huge collection of internal or built-in functions that you can call directly within your PHP scripts to perform a specific task, like `gettype()`, `print_r()`, `var_dump`, etc.

Please check out PHP reference section for a complete list of useful PHP built-in functions.

២. PHP User-Defined Functions

In addition to the built-in functions, PHP also allows you to define your own functions. It is a way to create reusable code packages that perform specific tasks and can be kept and maintained separately form main program. Here are some advantages of using functions:

- **Functions reduces the repetition of code within a program** — Function allows you to extract commonly used block of code into a single component. Now you can perform the same task by calling this function wherever you want within your script without having to copy and paste the same block of code again and again.
- **Functions makes the code much easier to maintain** — Since a function created once can be used many times, so any changes made inside a function automatically implemented at all the places without touching the several files.
- **Functions makes it easier to eliminate the errors** — When the program is subdivided into functions, if any error occur you know exactly what function causing the error and where to find it. Therefore, fixing errors becomes much easier.
- **Functions can be reused in other application** — Because a function is separated from the rest of the script, it's easy to reuse the same function in other applications just by including the php file containing those functions.

The following section will show you how easily you can define your own function in PHP.

៣. Creating and Invoking Functions

The basic syntax of creating a custom function can be give with:

```
function functionName(){
    // Code to be executed
}
```

The declaration of a user-defined function start with the word function, followed by the name of the function you want to create followed by parentheses i.e. () and finally place your function's code between curly brackets {}.

This is a simple example of an user-defined function, that display today's date:

```
<?php
// Defining function
function whatIsToday(){
    echo "Today is " . date('l', mktime());
}
// Calling function
whatIsToday();
?>
```

A function name must start with a letter or underscore character not with a number, optionally followed by the more letters, numbers, or underscore characters. Function names are case-insensitive.

៤. Functions with Parameters

You can specify parameters when you define your function to accept input values at run time. The parameters work like placeholder variables within a function; they're replaced at run time by the values (known as argument) provided to the function at the time of invocation.

```
function myFunc($oneParameter, $anotherParameter){
    // Code to be executed
}
```

You can define as many parameters as you like. However for each parameter you specify, a corresponding argument needs to be passed to the function when it is called.

The getSum() function in following example takes two integer values as arguments, simply add them together and then display the result in the browser.

```
<?php
  // Defining function
  function getSum($num1, $num2){
    $sum = $num1 + $num2;
    echo "Sum of the two numbers $num1 and $num2 is :
  $sum";
  }

  // Calling function
  getSum(10, 20);
?>
```

The output of the above code will be:

Sum of the two numbers 10 and 20 is : 30

An argument is a value that you pass to a function, and a parameter is the variable within the function that receives the argument. However, in common usage these terms are interchangeable i.e. an argument is a parameter is an argument.

៥. Functions with Optional Parameters and Default Values

You can also create functions with optional parameters — just insert the parameter name, followed by an equals (=) sign, followed by a default value, like this.

```
<?php
// Defining function
function customFont($font, $size=1.5){
    echo "<p style=\"font-family: $font; font-size:
{$size}em;\">Hello, world!</p>";
}

// Calling function
customFont("Arial", 2);
customFont("Times", 3);
customFont("Courier");
```

```
?>
```

As you can see the third call to customFont() doesn't include the second argument. This causes PHP engine to use the default value for the $size parameter which is 1.5.

៦. Returning Values from a Function

A function can return a value back to the script that called the function using the return statement. The value may be of any type, including arrays and objects.

```
<?php
// Defining function
function getSum($num1, $num2){
    $total = $num1 + $num2;
    return $total;
}

// Printing returned value
echo getSum(5, 10); // Outputs: 15
?>
```

A function can not return multiple values. However, you can obtain similar results by returning an array, as demonstrated in the following example.

```
<?php
// Defining function
function divideNumbers($dividend, $divisor){
    $quotient = $dividend / $divisor;
    $array = array($dividend, $divisor, $quotient);
    return $array;
}

// Assign variables as if they were an array
list($dividend, $divisor, $quotient) = divideNumbers(10, 2);
echo $dividend;  // Outputs: 10
echo $divisor;   // Outputs: 2
echo $quotient;  // Outputs: 5
?>
```

៧. Passing Arguments to a Function by Reference

In PHP there are two ways you can pass arguments to a function: *by value* and *by reference*. By default, function arguments are passed by value so that if the value of the argument within

the function is changed, it does not get affected outside of the function. However, to allow a function to modify its arguments, they must be passed by reference.

Passing an argument by reference is done by prepending an ampersand (&) to the argument name in the function definition, as shown in the example below:

```
<?php
/* Defining a function that multiply a number
by itself and return the new value */
function selfMultiply(&$number){
    $number *= $number;
    return $number;
}

$mynum = 5;
echo $mynum; // Outputs: 5

selfMultiply($mynum);
echo $mynum; // Outputs: 25
?>
```

៨. Understanding the Variable Scope

However, you can declare the variables anywhere in a PHP script. But, the location of the declaration determines the extent of a variable's visibility within the PHP program i.e. where the variable can be used or accessed. This accessibility is known as *variable scope*.

By default, variables declared within a function are local and they cannot be viewed or manipulated from outside of that function, as demonstrated in the example below:

```
<?php
// Defining function
function test(){
    $greet = "Hello World!";
    echo $greet;
}

test(); // Outputs: Hello World!

echo $greet; // Generate undefined variable error
?>
```

Similarly, if you try to access or import an outside variable inside the function, you'll get an undefined variable error, as shown in the following example:

```php
<?php
$greet = "Hello World!";

// Defining function
function test(){
    echo $greet;
}

test();  // Generate undefined variable error

echo $greet; // Outputs: Hello World!
?>
```

As you can see in the above examples the variable declared inside the function is not accessible from outside, likewise the variable declared outside of the function is not accessible inside of the function. This separation reduces the chances of variables within a function getting affected by the variables in the main program.

It is possible to reuse the same name for a variable in different functions, since local variables are only recognized by the function in which they are declared.

៩. The global Keyword

There may be a situation when you need to import a variable from the main program into a function, or vice versa. In such cases, you can use the global keyword before the variables inside a function. This keyword turns the variable into a global variable, making it visible or accessible both inside and outside the function, as show in the example below:

```php
<?php
$greet = "Hello World!";

// Defining function
function test(){
    global $greet;
    echo $greet;
}

test(); // Outpus: Hello World!
echo $greet; // Outpus: Hello World!
```

```
// Assign a new value to variable
$greet = "Goodbye";

test(); // Outputs: Goodbye
echo $greet; // Outputs: Goodbye
?>
```

You will learn more about visibility and access control in PHP classes and objects chapter.

១0. Creating Recursive Functions

A recursive function is a function that calls itself again and again until a condition is satisfied.

Recursive functions are often used to solve complex mathematical calculations, or to process deeply nested structures e.g., printing all the elements of a deeply nested array.

The following example demonstrates how a recursive function works.

```
<?php
// Defining recursive function
function printValues($arr) {
    global $count;
    global $items;

    // Check input is an array
    if(!is_array($arr)){
        die("ERROR: Input is not an array");
    }

    /*
    Loop through array, if value is itself an array recursively call the
    function else add the value found to the output items array,
    and increment counter by 1 for each value found
    */
    foreach($arr as $a){
        if(is_array($a)){
            printValues($a);
        } else{
            $items[] = $a;
            $count++;
        }
    }

    // Return total count and values found in array
    return array('total' => $count, 'values' => $items);
}

// Define nested array
$species = array(
    "birds" => array(
        "Eagle",
        "Parrot",
        "Swan"
```

```
    ),
    "mammals" => array(
        "Human",
        "cat" => array(
            "Lion",
            "Tiger",
            "Jaguar"
        ),
        "Elephant",
        "Monkey"
    ),
    "reptiles" => array(
        "snake" => array(
            "Cobra" => array(
                "King Cobra",
                "Egyptian cobra"
            ),
            "Viper",
            "Anaconda"
        ),
        "Crocodile",
        "Dinosaur" => array(
            "T-rex",
            "Alamosaurus"
        )
    )
);

// Count and print values in nested array
$result = printValues($species);
echo $result['total'] . ' value(s) found: ';
echo implode(', ', $result['values']);
?>
```

Be careful while creating recursive functions, because if code is written improperly it may result in an infinite loop of function calling.

មេរៀន ទី១២ ប្រមាណវិធីគណិត

(PHP Math Operations)

នៅក្នុងឯកសារបង្រៀននេះអ្នកនឹងរៀនពីរបៀបអនុវត្តគណិតវិទ្យាក្នុង PHP ។

១.Performing Math Operations

PHP has several built-in functions that help you perform anything from simple additions or subtraction to advanced calculations. You've already seen how to perform basic mathematical operations in PHP operators chapter. Let's check out one more example:

```
<?php
      echo 7 + 3; // Outputs: 10
      echo 7 - 2; // Outputs: 5
      echo 7 * 2; // Outputs: 14
      echo 7 / 2; // Outputs: 3.5
      echo 7 % 2; // Outputs: 1
?>
```

Every math operation has a certain precedence level; generally multiplication and division are performed before addition and subtraction. However, parentheses can alter this precedence; expressions enclosed within parentheses are always evaluated first, regardless of the operation's precedence level, as demonstrated in the following example:

```
<?php
      echo 5 + 4 * 10;          // Outputs: 45
      echo (5 + 4) * 10;        // Outputs: 90
      echo 5 + 4 * 10 / 2;      // Outputs: 25
      echo 8 * 10 / 4 - 2;      // Outputs: 18
      echo 8 * 10 / (4 - 2);    // Outputs: 40
      echo 8 + 10 / 4 - 2;      // Outputs: 8.5
      echo (8 + 10) / (4 - 2); // Outputs: 9
?>
```

In the following section we're going to look at some built-in PHP functions that are most frequently used in performing mathematical operations.

២.Find the Absolute Value of a Number

The absolute value of an integer or a float can be found with the abs() function, as demonstrated in the following example:

```php
<?php
echo abs(5);     // Outputs: 5 (integer)
echo abs(-5);    // Outputs: 5 (integer)
echo abs(4.2);   // Outputs: 4.2 (double/float)
echo abs(-4.2); // Outputs: 4.2 (double/float)
?>
```

As you can see if the given number is negative, the valued returned is positive. But, if the number is positive, this function simply returns the number.

៣.Round a Fractional Value Up or Down

The `ceil()` function can be used to round a fractional value up to the next highest integer value, whereas the `floor()` function can be used to round a fractional value down to the next lowest integer value, as demonstrated in the following example:

```php
<?php
// Round fractions up
echo ceil(4.2);     // Outputs: 5
echo ceil(9.99);    // Outputs: 10
echo ceil(-5.18);   // Outputs: -5

// Round fractions down
echo floor(4.2);     // Outputs: 4
echo floor(9.99);    // Outputs: 9
echo floor(-5.18);   // Outputs: -6
?>
```

៤.Find the Square Root of a Number

You can use the `sqrt()` function to find the square root of a positive number. This function returns a special value `NAN` for negative numbers. Here's an example:

```php
<?php
echo sqrt(9);    // Outputs: 3
echo sqrt(25);   // Outputs: 5
echo sqrt(10);   // Outputs: 3.1622776601684
echo sqrt(-16); // Outputs: NAN
?>
```

៥.Generate a Random Number

The `rand()` function can be used to generate a random number. You can optionally specify a range by passing the min, max arguments, as shown in the following example:

```php
<?php
// Generate some random numbers
echo rand() . "<br>";
echo rand() . "<br>";

// Generate some random numbers between 1 and 10 (inclusive)
echo rand(1, 10) . "<br>";
echo rand(1, 10) . "<br>";
?>
```

If `rand()` function is called without the optional **`min`**, **`max`** arguments, it returns a pseudo-random number between `0` and `getrandmax()`. The `getrandmax()` function show the largest possible random value, which is only 32767 on Windows platform. So, if you require a range larger than 32767, you may simply specify the **`min`** and **`max`** arguments.

៦.Convert Decimal Numbers to Binary and Vice Versa

The `decbin()` function is used to convert a decimal number into binary number. Whereas its counterpart the `bindec()` function converts a number from binary to decimal.

```php
<?php
// Convert Decimal to Binary
echo decbin(2);    // Outputs: 10
echo decbin(12);   // Outputs: 1100
echo decbin(100);  // Outputs: 1100100

// Convert Binary to Decimal
echo bindec(10);       // Outputs: 2
echo bindec(1100);     // Outputs: 12
echo bindec(1100100);  // Outputs: 100
?>
```

៧.Convert Decimal Numbers to Hexadecimal and Vice Versa

The `dechex()` function is used to convert a decimal number into hexadecimal representation. Whereas, the `hexdec()` function is used to converts a hexadecimal string to a decimal number.

```php
<?php
// Convert decimal to hexadecimal
echo dechex(255);  // Outputs: ff
echo dechex(196);  // Outputs: c4
echo dechex(0);    // Outputs: 0

// Convert hexadecimal to decimal
echo hexdec('ff');  // Outputs: 255
echo hexdec('c4');  // Outputs: 196
echo hexdec(0);     // Outputs: 0
```

```
?>
```

៧.Convert Decimal Numbers to Octal and Vice Versa

The `decoct()` function is used to convert a decimal number into octal representation. Whereas, the `octdec()` function is used to converts a octal number to a decimal number.

```
<?php
// Convert decimal to octal
echo decoct(12);   // Outputs: 14
echo decoct(256);  // Outputs: 400
echo decoct(77);   // Outputs: 115

// Convert octal to decimal
echo octdec('14');   // Outputs: 12
echo octdec('400');  // Outputs: 256
echo octdec('115');  // Outputs: 77
?>
```

៨.Convert a Number from One Base System to Another

The `base_convert()` function can be used to convert a number from one base system to other. For example, you can convert decimal (*base 10*) to binary (*base 2*), hexadecimal (*base 16*) to octal (*base 8*), octal to hexadecimal, hexadecimal to decimal, and so on.

This function accepts three parameters: the number to convert, the base it's currently in, and the base it's to be converted to. The basic syntax is as follows:

```
base_convert(number, frombase, tobase);
```

Here, the number can be either an integer or a string representing an integer.
Both **`frombase`** and **`tobase`** have to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a-z, where a means 10, b means 11 and z means 35. Here's a simple example to show how this function works:

```
<?php
// Convert decimal to binary
echo base_convert('12', 10, 2);  // Outputs: 1100
// Convert binary to decimal
echo base_convert('1100', 2, 10);  // Outputs: 12

// Convert decimal to hexadecimal
echo base_convert('10889592', 10, 16);  // Outputs: a62978
// Convert hexadecimal to decimal
echo base_convert('a62978', 16, 10);  // Outputs: 10889592

// Convert decimal to octal
echo base_convert('82', 10, 8);  // Outputs: 122
// Convert octal to decimal
echo base_convert('122', 8, 10);  // Outputs: 82
```

```php
// Convert hexadecimal to octal
echo base_convert('c2c6a8', 16, 8);  // Outputs: 60543250
// Convert octal to hexadecimal
echo base_convert('60543250', 8, 16);  // Outputs: c2c6a8

// Convert octal to binary
echo base_convert('42', 8, 2);  // Outputs: 100010
// Convert binary to octal
echo base_convert('100010', 2, 8);  // Outputs: 42

// Convert hexadecimal to binary
echo base_convert('abc', 16, 2);  // Outputs: 101010111100
// Convert binary to hexadecimal
echo base_convert('101010111100', 2, 16);  // Outputs: abc
?>
```

The base_convert() function will always return a string value. If the returned value is in base 10 the resulting string can be used as a numeric string in calculations and PHP will convert it to a number when the calculation is performed.

មេរៀន ទី១៤ អនុគមន៍ $_GET( ) និង $_POST( )

(PHP GET and POST)

នៅក្នុងការណែនាំនេះអ្នកនឹងរៀនពីរបៀបផ្ញើព័ត៌មានទៅម៉ាស៊ីនមេដោយប្រើវិធីសាស្ត្រ HTTP GET និង POST ហើយទាញយកមកវិញដោយប្រើ PHP ។

ការបញ្ជូនព័ត៌មានពីអ្នកប្រើប្រាស់ទៅកាន់ Server មាន ២ របៀបដែលសំខាន់ ៖

- GET method : ព័ត៌មានត្រូវបានរក្សាទុក និងបង្ហាញនៅលើ URL ។
- POST method : ព័ត៌មានមិនអាចមើលឃើញធម្មតាៗដូច GET method ទេ ។

១.ទម្រង់នៃព័ត៌មាន

មានព័ត៌មានដូចជា color តម្លៃ blue, font-size តម្លៃ medium និង theme តម្លៃ dark ។ វាត្រូវបានកំណត់ក្រោមទម្រង់ **name**=*john*&**age**=*24*។

បើវាត្រូវបានកំណត់ក្រោមទម្រង់ GET method វានឹងត្រូវបានឃើញខាងស្តាំនៃសញ្ញា ? នៅក្នុង URL បែបនេះ `http://www.example.com/action.php?name=john&age=24` ។ បើសិនជាត្រូវបានកំណត់ក្រោមទម្រង់ POST method វាមិនត្រូវបានបង្ហាញឲ្យឃើញនោះទេ ។ ដូចនេះ ព័ត៌មានដែលសំខាន់ៗ ដូចជាទាក់ទងទៅនឹងអក្សរសម្ងាត់ គេតែងប្រើ POST method ព្រោះយើងមិនអាចឃើញវាធម្មតាៗបាន ។

២.ដើម្បីបញ្ជូនព័ត៌មាន`<form>`

ចំណុចសំខាន់ៗនៃទម្រង់បែបបទ

Attribute	បម្រើបម្រាស់
method	បង្ហាញអំពី method ដែលត្រូវប្រើ ។ តម្លៃដើម get ។

Attribute	បម្រើបម្រាស់
action	កំណត់ URL ដែលត្រូវបញ្ជូនព័ត៌មានក្នុង <form> បច្ចុប្បន្ន ទៅកាន់ ។ តម្លៃដើមគឺជាទំព័របច្ចុប្បន្ន ។
name	កំណត់ឈ្មោះទៅឲ្យព័ត៌មាននីមួយៗ ដែលត្រូវបញ្ជូន ។ បើមិនកំណត់ ព័ត៌មានមួយនោះនឹងមិនត្រូវបានបញ្ជូនទេ ។

៣.ចាប់ព័ត៌មាន

ដើម្បីចាប់យកព័ត៌មានទាំងឡាយចេញពី GET method យើងត្រូវប្រើ `$_GET` ។ រីង POST method យើងប្រើ `$_POST` ។

The GET Method

In GET method the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (`&`). In general, a URL with GET data will look like this:

```
http://www.example.com/action.php?name=john&age=24
```

The bold parts in the URL are the GET parameters and the italic parts are the value of those parameters. More than one `parameter=value` can be embedded in the URL by concatenating with ampersands (`&`). One can only send simple text data via GET method.

Advantages and Disadvantages of Using the GET Method

- Since the data sent by the GET method are displayed in the URL, it is possible to bookmark the page with specific query string values.
- The GET method is not suitable for passing sensitive information such as the username and password, because these are fully visible in the URL query string as well as potentially stored in the client browser's memory as a visited page.
- Because the GET method assigns data to a server environment variable, the length of the URL is limited. So, there is a limitation for the total data to be sent.

PHP provides the superglobal variable `$_GET` to access all the information sent either through the URL or submitted through an HTML form using the `method="get"`.

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <title>Example of PHP GET method</title>
</head>
<body>
<?php
if(isset($_GET["name"])){
    echo "<p>Hi, " . $_GET["name"] . "</p>";
}
?>
<form method="get" action="<?php echo $_SERVER["PHP_SELF"];?>">
    <label for="inputName">Name:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Submit">
</form>
</body>
```

The POST Method

In POST method the data is sent to the server as a package in a separate communication with the processing script. Data sent through POST method will not visible in the URL.

Advantages and Disadvantages of Using the POST Method

- It is more secure than GET because user-entered information is never visible in the URL query string or in the server logs.
- There is a much larger limit on the amount of data that can be passed and one can send text data as well as binary data (uploading a file) using POST.
- Since the data sent by the POST method is not visible in the URL, so it is not possible to bookmark the page with specific query.

Like `$_GET`, PHP provide another superglobal variable `$_POST` to access all the information sent via post method or submitted through an HTML form using the `method="post"`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example of PHP POST method</title>
</head>
<body>
<?php
if(isset($_POST["name"])){
    echo "<p>Hi, " . $_POST["name"] . "</p>";
}
?>
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
    <label for="inputName">Name:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Submit">
```

```
</form>
</body>
```

The $_REQUEST Variable

PHP provides another super global variable `$_REQUEST` that contains the values of both the `$_GET` and `$_POST` variables as well as the values of the `$_COOKIE` super global variable.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example of PHP $_REQUEST variable</title>
</head>
<body>
<?php
if(isset($_REQUEST["name"])){
    echo "<p>Hi, " . $_REQUEST["name"] . "</p>";
}
?>
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
    <label for="inputName">Name:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Submit">
</form>
</body>
```

You will learn more about PHP cookies and form handling in advanced section.

The super global variables $_GET, $_POST and $_REQUEST are built-in variables that are always available in all scopes throughout a script.

PHP Global Variables - Superglobals

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

The next chapters will explain some of the superglobals, and the rest will be explained in later chapters.

PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

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

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

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

In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!

PHP $_SERVER

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

```php
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

The following table lists the most important elements that can go inside $_SERVER:

Element/Code	Description
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)

$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the

	value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

PHP $_REQUEST

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

```
<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>
```

PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

```
<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>
```

PHP $_GET

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

The example below shows the code in "test_get.php":

```
<html>
<body>
```

```php
<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

មេរៀន ទី១៥ PHP Forms

PHP Form Handling

The PHP superglobals $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method:

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

PHP Form Validation

Think SECURITY when processing PHP forms!

These pages will show how to process PHP forms with security in mind. Proper validation of form data is important to protect your form from hackers and spammers!

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace

E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
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
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);
?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Big Note on PHP Form Security

The $_SERVER["PHP_SELF"] variable can be used by hackers!

If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

So far, so good.

However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script
>
```

This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.

Be aware of that **any JavaScript code can be added inside the <script> tag!** A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

How To Avoid $_SERVER["PHP_SELF"] Exploits?

$_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"
]);?>">
```

The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post" action="test_form.php/&quot;&gt;&lt;script&gt;alert('ha
cked')&lt;/script&gt;">
```

The exploit attempt fails, and no harm is done!

Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Notice that at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

However, in the example above, all input fields are optional. The script works fine even if the user does not enter any data.

The next step is to make input fields required and create error messages if needed.

PHP - Required Fields

From the validation rules table on the previous page, we see that the "Name", "E-mail", and "Gender" fields are required. These fields cannot be empty and must be filled out in the HTML form.

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

In the previous chapter, all input fields were optional.

In the following code we have added some new variables: $nameErr, $emailErr, $genderErr, and $websiteErr. These error variables will hold error messages for the required fields. We have also added an `if else` statement for each $_POST variable. This checks if the $_POST variable is empty (with the PHP `empty()` function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the `test_input()` function:

```
<?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"]);
  }
}
?>
```

PHP - Display The Error Messages

Then in the HTML form, we add a little script after each required field, which generates the correct error message if needed (that is if the user tries to submit the form without filling out the required fields):

```
<!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>
```

The next step is to validate the input data, that is "Does the Name field contain only letters and whitespace?", and "Does the E-mail field contain a valid e-mail address syntax?", and if filled out, "Does the Website field contain a valid URL?".

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-
a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

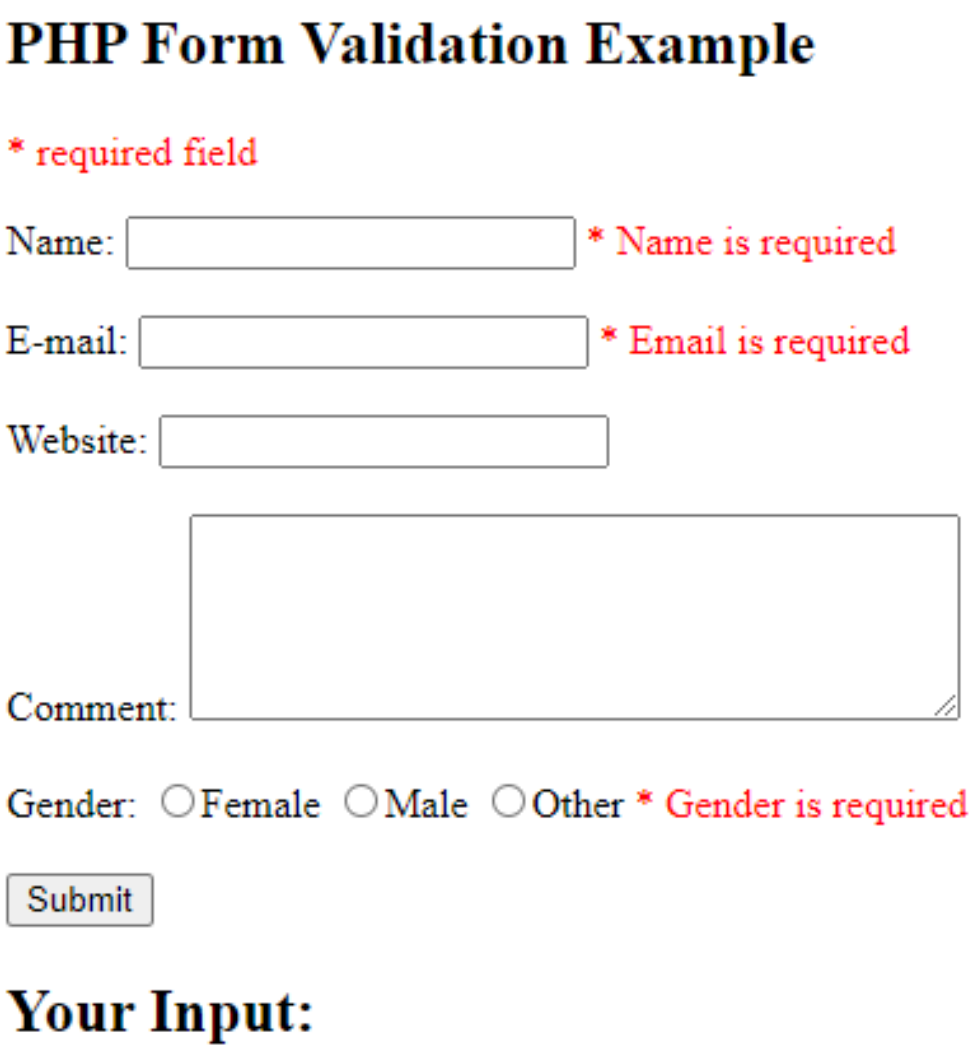
PHP - Validate Name, E-mail, and URL

Now, the script looks like this:

```
<!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"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  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>
```



PHP - Keep The Values in The Form

To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the $name, $email, $website, and $comment variables.

Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):

```
Name: <input type="text" name="name" value="<?php echo $name;?>">
```

```
E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></t
extarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="other") echo "checked";?>
value="other">Other
```

មេរៀន ទី១៥ Class and Object

What is Object Oriented Programming

Object-Oriented Programming (OOP) is a programming model that is based on the concept of classes and objects. As opposed to procedural programming where the focus is on writing procedures or functions that perform operations on the data, in object-oriented programming the focus is on the creations of objects which contain both data and functions together.

Object-oriented programming has several advantages over conventional or procedural style of programming. The most important ones are listed below:

- It provides a clear modular structure for the programs.
- It helps you adhere to the "don't repeat yourself" (DRY) principle, and thus make your code much easier to maintain, modify and debug.
- It makes it possible to create more complicated behavior with less code and shorter development time and high degree of reusability.

The following sections will describe how classes and objects work in PHP.

Tip: The idea behind Don't Repeat Yourself (DRY) principle is reducing the repetition of code by abstracting out the code that are common for the application and placing them at a single place and reuse them instead of repeating it.

Understanding Classes and Objects

Classes and objects are the two main aspects of object-oriented programming. A class is a self-contained, independent collection of variables and functions which work together to perform one or more specific tasks, while objects are individual instances of a class.

A class acts as a template or blueprint from which lots of individual objects can be created. When individual objects are created, they inherit the same generic properties and behaviors, although each object may have different values for certain properties.

For example, think of a class as a blueprint for a house. The blueprint itself is not a house, but is a detailed plan of the house. While, an object is like an actual house built

according to that blueprint. We can build several identical houses from the same blueprint, but each house may have different paints, interiors and families inside, as shown in the illustration below.

Blueprint *Houses built according to the blueprint*

A class can be declared using the `class` keyword, followed by the name of the class and a pair of curly braces (`{}`), as shown in the following example.

Let's create a PHP file named Rectangle.php and put the following example code inside it so that our class code should be separated from rest of the program. We can then use it wherever it's needed by simply including the Rectangle.php file.

```php
<?php
class Rectangle
{
    // Declare  properties
    public $length = 0;
    public $width = 0;

    // Method to get the perimeter
    public function getPerimeter(){
        return (2 * ($this->length + $this->width));
    }

    // Method to get the area
    public function getArea(){
        return ($this->length * $this->width);
    }
}
?>
```

The `public` keyword before the properties and methods in the example above, is an access modifier, which indicates that this property or method is accessible from anywhere. We will learn more about this a little later in this chapter.

Note: Syntactically, variables within a class are called *properties*, whereas functions are called *methods*. Also class names conventionally are written in PascalCase i.e. each concatenated word starts with an uppercase letter (e.g. MyClass).

Once a class has been defined, objects can be created from the class with the `new` keyword. Class methods and properties can directly be accessed through this object instance.

Create another PHP file name test.php and put the following code inside it.

```php
<?php
// Include class definition
require "Rectangle.php";

// Create a new object from Rectangle class
$obj = new Rectangle;

// Get the object properties values
echo $obj->length; // Output: 0
echo $obj->width; // Output: 0

// Set object properties values
$obj->length = 30;
$obj->width = 20;

// Read the object properties values again to show the change
echo $obj->length; // Output: 30
echo $obj->width; // Output: 20


// Call the object methods
echo $obj->getPerimeter(); // Output: 100
echo $obj->getArea(); // Output: 600
?>
```

The arrow symbol (`->`) is an OOP construct that is used to access contained properties and methods of a given object. Whereas, the pseudo-variable `$this` provides a reference to the calling object i.e. the object to which the method belongs.

The real power of object oriented programming becomes evident when using multiple instances of the same class, as shown in the following example:

```php
<?php
// Include class definition
require "Rectangle.php";

// Create multiple objects from the Rectangle class
$obj1 = new Rectangle;
$obj2 = new Rectangle;
```

```php
// Call the methods of both the objects
echo $obj1->getArea(); // Output: 0
echo $obj2->getArea(); // Output: 0

// Set $obj1 properties values
$obj1->length = 30;
$obj1->width = 20;

// Set $obj2 properties values
$obj2->length = 35;
$obj2->width = 50;

// Call the methods of both the objects again
echo $obj1->getArea(); // Output: 600
echo $obj2->getArea(); // Output: 1750
?>
```

As you can see in the above example, calling the `getArea()` method on different objects causes that method to operate on a different set of data. Each object instance is completely independent, with its own properties and methods, and thus can be manipulated independently, even if they're of the same class.

Using Constructors and Destructors

To make the object-oriented programming easier, PHP provides some magic methods that are executed automatically when certain actions occur within an object.

For example, the magic method `__construct()` (known as *constructor*) is executed automatically whenever a new object is created. Similarly, the magic method `__destruct()` (known as *destructor*) is executed automatically when the object is destroyed. A destructor function cleans up any resources allocated to an object once the object is destroyed.

```php
<?php
class MyClass
{
    // Constructor
    public function __construct(){
        echo 'The class "' . __CLASS__ . '" was initiated!<br>';
    }

    // Destructor
    public function __destruct(){
        echo 'The class "' . __CLASS__ . '" was destroyed.<br>';
    }
}
```

```php
// Create a new object
$obj = new MyClass;

// Output a message at the end of the file
echo "The end of the file is reached.";
?>
```

The PHP code in the above example will produce the following output:

The class "MyClass" was initiated!
The end of the file is reached.
The class "MyClass" was destroyed.

A destructor is called automatically when a scripts ends. However, to explicitly trigger the destructor, you can destroy the object using the PHP `unset()` function, as follow:

```php
<?php
class MyClass
{
    // Constructor
    public function __construct(){
        echo 'The class "' . __CLASS__ . '" was initiated!<br>';
    }

    // Destructor
    public function __destruct(){
    echo 'The class "' . __CLASS__ . '" was destroyed.<br>';
    }
}

// Create a new object
$obj = new MyClass;

// Destroy the object
unset($obj);

// Output a message at the end of the file
echo "The end of the file is reached.";
?>
```

Now, the PHP code in the above example will produce the following output:

The class "MyClass" was initiated!
The class "MyClass" was destroyed.
The end of the file is reached.

Tip: PHP automatically clean up all resources allocated during execution when the script is finished, e.g. closing database connections, destroying objects, etc.

Note: The __CLASS__ is a magic constant which contains the name of the class in which it is occur. It is empty, if it occurs outside of the class.

Extending Classes through Inheritance

Classes can inherit the properties and methods of another class using the `extends` keyword. This process of extensibility is called inheritance. It is probably the most powerful reason behind using the object-oriented programming model.

```php
<?php
// Include class definition
require "Rectangle.php";

// Define a new class based on an existing class
class Square extends Rectangle
{
    // Method to test if the rectangle is also a square
    public function isSquare(){
        if($this->length == $this->width){
            return true; // Square
        } else{
            return false; // Not a square
        }
    }
}

// Create a new object from Square class
$obj = new Square;

// Set object properties values
$obj->length = 20;
$obj->width = 20;

// Call the object methods
if($obj->isSquare()){
    echo "The area of the square is ";
} else{
    echo "The area of the rectangle is ";
};
echo $obj->getArea();
?>
```

The PHP code in the above example will produce the following output:

The area of the square is 400

As you can see in the above example, even though the class definition of Square doesn't explicitly contain `getArea()` method nor the `$length` and `$width` property, instances of the Square class can use them, as they inherited from the parent Rectangle class.

Tip: Since a child class is derived from a parent class, it is also referred to as a derived class, and its parent is called the base class.

Controlling the Visibility of Properties and Methods

When working with classes, you can even restrict access to its properties and methods using the *visibility keywords* for greater control. There are three visibility keywords (from most visible to least visible): `public`, `protected`, `private`, which determines how and from where properties and methods can be accessed and modified.

- **public** — A public property or method can be accessed anywhere, from within the class and outside. This is the default visibility for all class members in PHP.
- **protected** — A protected property or method can only be accessed from within the class itself or in child or inherited classes i.e. classes that extends that class.
- **private** — A private property or method is accessible only from within the class that defines it. Even child or inherited classes cannot access private properties or methods.

The following example will show you how this visibility actually works:

```php
<?php
// Class definition
class Automobile
{
    // Declare  properties
    public $fuel;
    protected $engine;
    private $transmission;
}
class Car extends Automobile
{
    // Constructor
    public function __construct(){
        echo 'The class "' . __CLASS__ . '" was initiated!<br>';
    }
}

// Create an object from Automobile class
$automobile = new Automobile;

// Attempt to set $automobile object properties
$automobile->fuel = 'Petrol'; // ok
$automobile->engine = '1500 cc'; // fatal error
$automobile->transmission = 'Manual'; // fatal error
```

```php
// Create an object from Car class
$car = new Car;

// Attempt to set $car object properties
$car->fuel = 'Diesel'; // ok
$car->engine = '2200 cc'; // fatal error
$car->transmission = 'Automatic'; // undefined
?>
```

Static Properties and Methods

In addition to the visibility, properties and methods can also be declared as `static`, which makes them accessible without needing an instantiation of the class. Static properties and methods can be accessed using the scope resolution operator (`::`), like this: `ClassName::$property` and `ClassName::method()`.

A property declared as static cannot be accessed via the object of that class though a static method can be, as demonstrated in the following example:

```php
<?php
// Class definition
class HelloClass
{
    // Declare a static property
    public static $greeting = "Hello World!";

    // Declare a static method
    public static function sayHello(){
        echo self::$greeting;
    }
}
// Attempt to access static property and method directly
echo HelloClass::$greeting; // Output: Hello World!
HelloClass::sayHello(); // Output: Hello World!

// Attempt to access static property and method via object
$hello = new HelloClass;
echo $hello->greeting; // Strict Warning
$hello->sayHello(); // Output: Hello World!
?>
```

The keyword `self` in the above example means "the current class". It is never preceded by a dollar sign (`$`) and always followed by the `::` operator (e.g. `self::$name`).

The `self` keyword is different from the `this` keyword which means "the current object" or "the current instance of a class". The `this` keyword is always preceded by a dollar sign (`$`) and followed by the `->` operator (e.g. `$this->name`).

Note: Since static methods can be called without an instance of a class (i.e. object), the pseudo-variable `$this` is not available inside the method declared as static.

We hope you've understood the basic concepts of object-oriented programming by now. You'll find more examples on OOP in PHP and MySQL database section.