به نام خدا

PHP به زبان ساده

مولف : یونس ابراهیمی

Younes.ebrahimi.1391@gmail.com

0938 21 22 200

1393/10/17

مرجع : www.w3-farsi.ir

PHPچیست و چرا به آن نیاز داریم؟

PHP شاید عمومی ترین زبان اسکریپتی تحت وب باشد و استفاده از آن روز به روز بیشتر می شود. با استفاده از PHP می توان صفحات ورودی ایجاد کرده، جزییات ورود اطلاعات از طریق فرم ها را بررسی نمایید، فروم ،گالری تصاویر و بسیاری از چیزهای دیگر را ایجاد کنید .

PHP به عنوان یک زبان برنامه نویسی سمت سرور مشهور است. چون نمی تواند در داخل کامپیوتر شما اجرا شود. دیگر زبان های برنامه نویسی که احتمالا اسم آنها را شنیده اید عبارتند ازASP: ، Python، .Perl در باره این زبان ها لازم نیست بیشتر از اینها بدانید چون در این آموزش فرض بر این است که شما برنامه نویس حرفه ای نیستید! عمومی ترین تعریف PHP این است که PHP مخفف کلمات Hypertext Pre-processor می باشد. شاید برایتان این سوال پیش بیاید که مخفف کلمات فوق HPP است. درست است، اما در نسخه های قبلی برنامه PHP را به عنوان مخفف کلمات Personal Home Page تعریف کرده اند. که مخفف آنها PHP می شود. در این آموزش به شما نحوه اجرای PHP را آموزش می دهیم و متوجه خواهید شد که یادگیری آن بسیار آسان است .

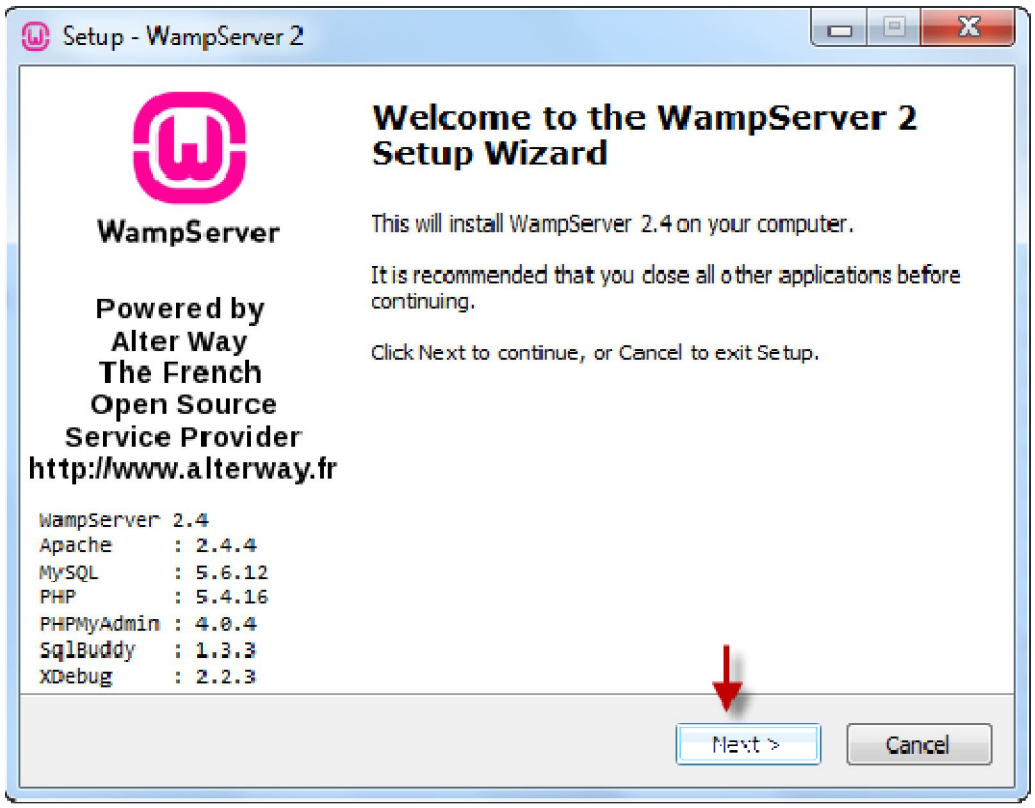
برای شروع کار با PHP چه چیزهایی لازم دارید؟

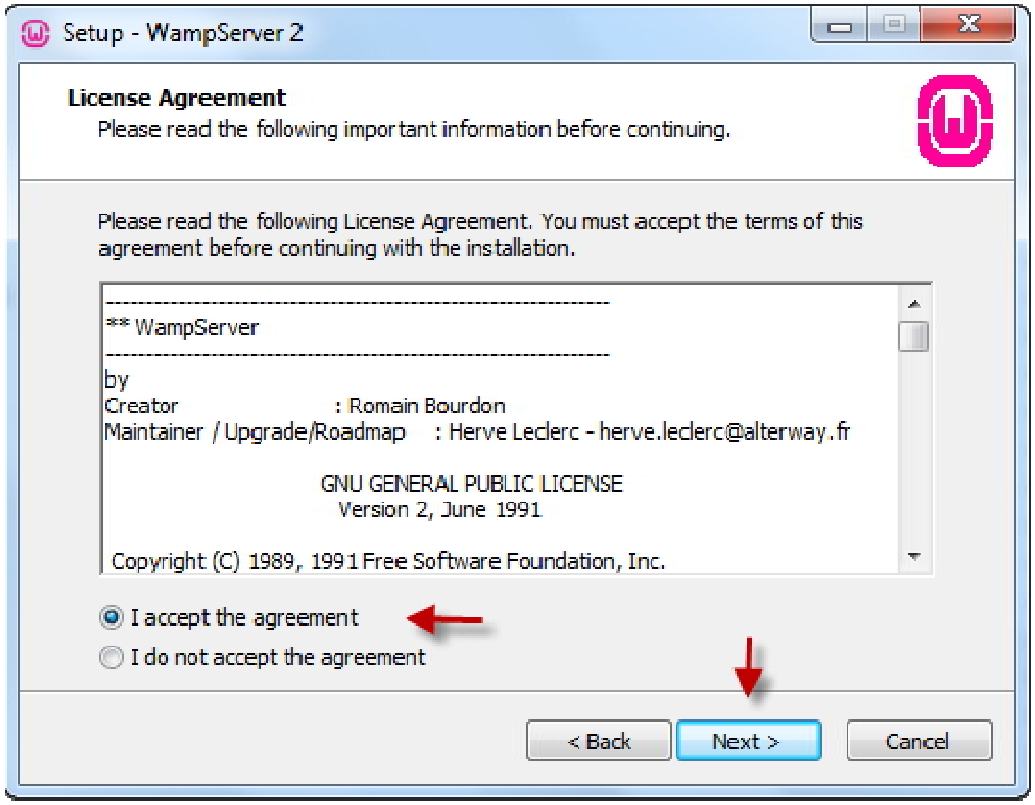
برای نوشتن و تست کدهای PHP ابتدا به یک سرور نیاز دارید. خوشبختانه، لازم نیست که یک سرور خریداری کنید و نیاز به خرج پول نیست. درست است که PHPیک زبان برنامه نویسی عمومی است اما چون یک زبان اسکریپتی سمت سرور است باید یک هاست (مقداری از فضای وب) که PHP را پشتیبانی کند خریداری کنید و یا کاری کنید که کامپیوترتان به عنوان یک سرور عمل کند. چون PHP نمی تواند بر روی کامپیوتر اجرا شود. این زبان بر روی سرور (server)اجرا شده و نتایج را به کامپیوتر سرویس گیرنده (client) برگشت می دهد. نگران راه اندازی و تست کدها بر روی کامپیوترتان نباشید. یک راه ساده برای اجرای کدها بر روی کامپیوتر استفاده از نرم افزاری به نام Wampserver می باشد.این نرم افزار تمام موارد لازم برای اجرای کدها را نصب می کند. نحوه نصب و استفاده از این نرم افزار را برای شما توضیح می دهیم. برای دانلود این نرم افزار بر روی لینک زیر کلیک کنید :

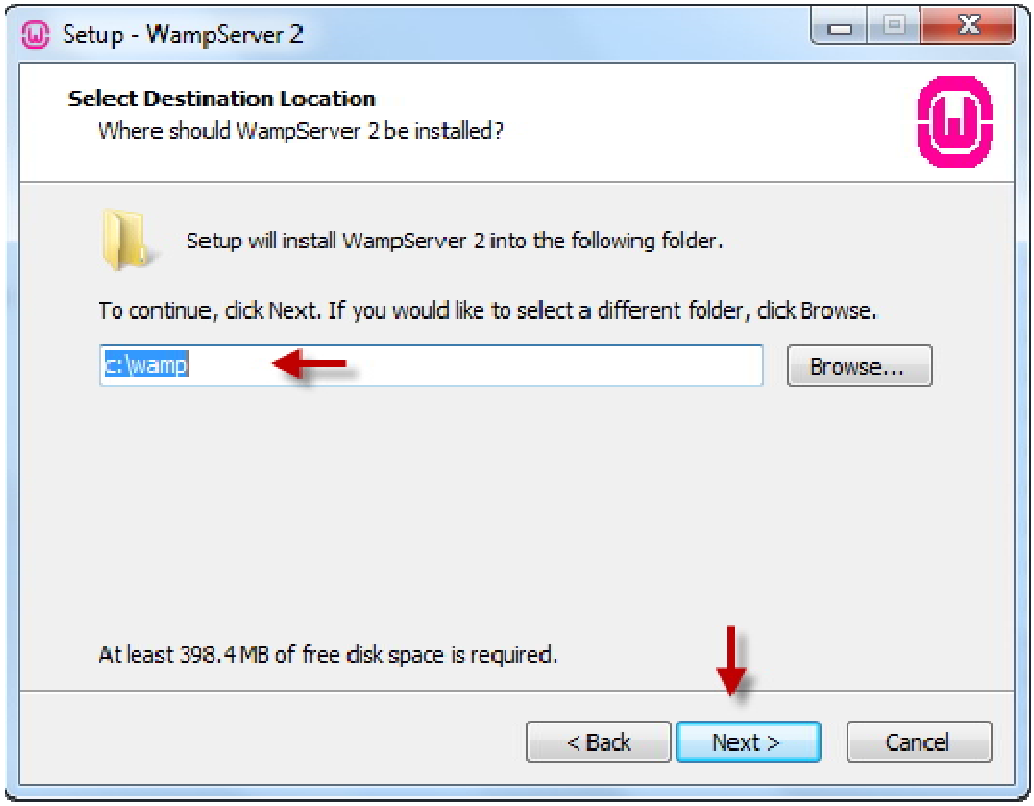
Download Wampserver

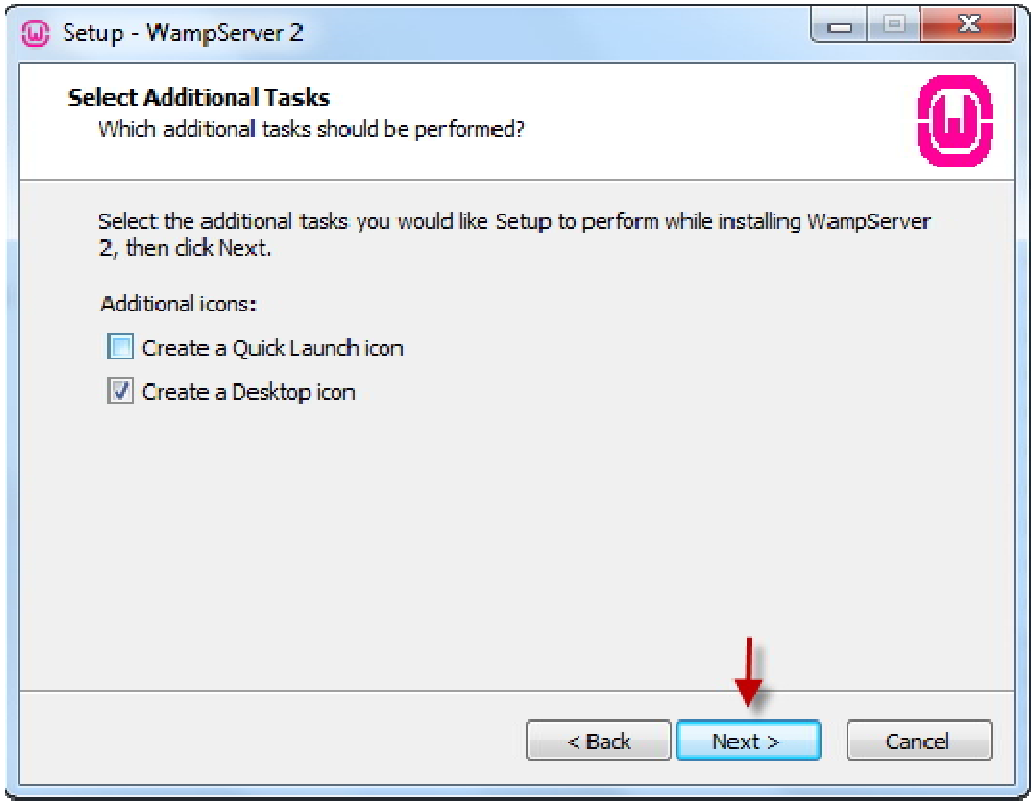
Wampserver نصب و تست

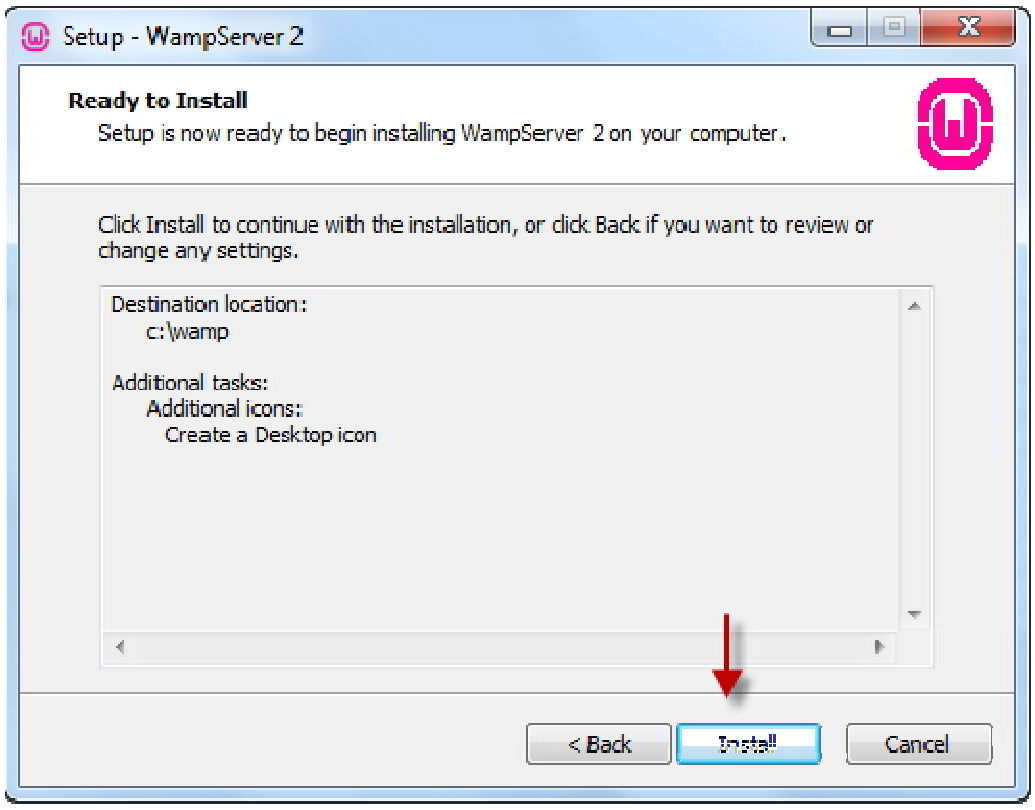
بعد از دانلود Wampserver مراحل زیر را برای نصب آن طی کنید:

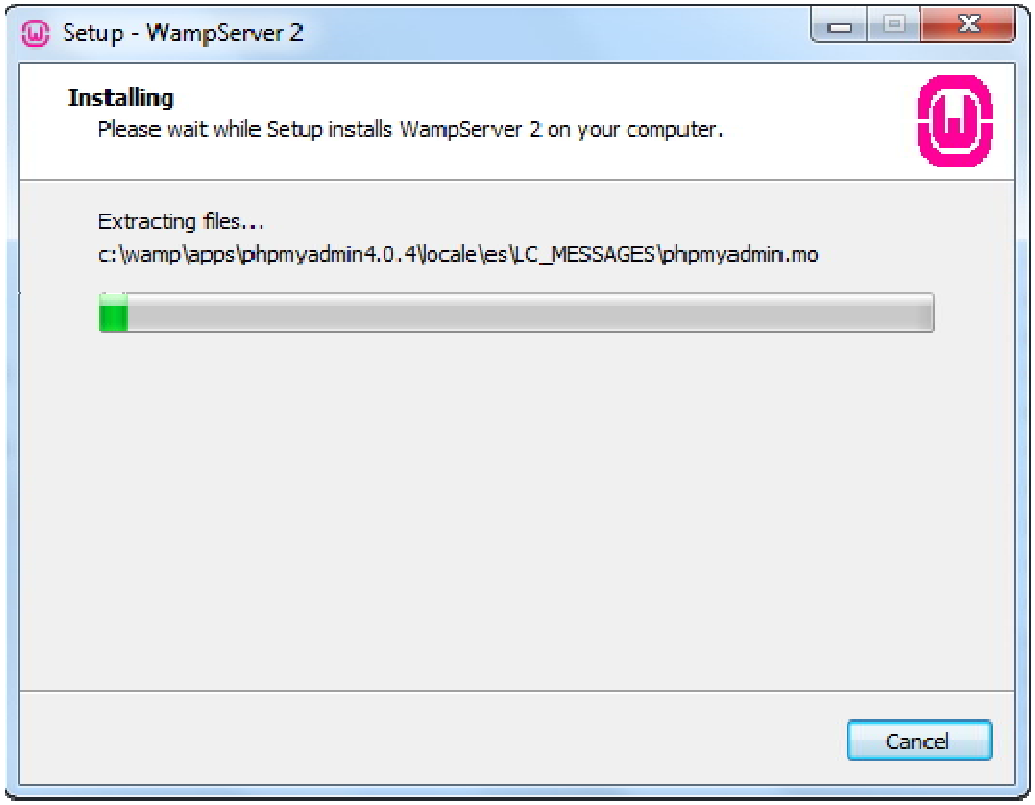

Setup - WampServer 2
WampServer
Powered by
Alter Way
The French
Open Source
Service Provider
http://www.alterway.fr
WampServer 2.4
Apache : 2.4.4
MySQL : 5.6.12
PHP : 5.4.16
PHPMyAdmin : 4.0.4
SqlBuddy : 1.3.3
XDebug : 2.2.3
Welcome to the WampServer 2 Setup Wizard
This will install WampServer 2.4 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel

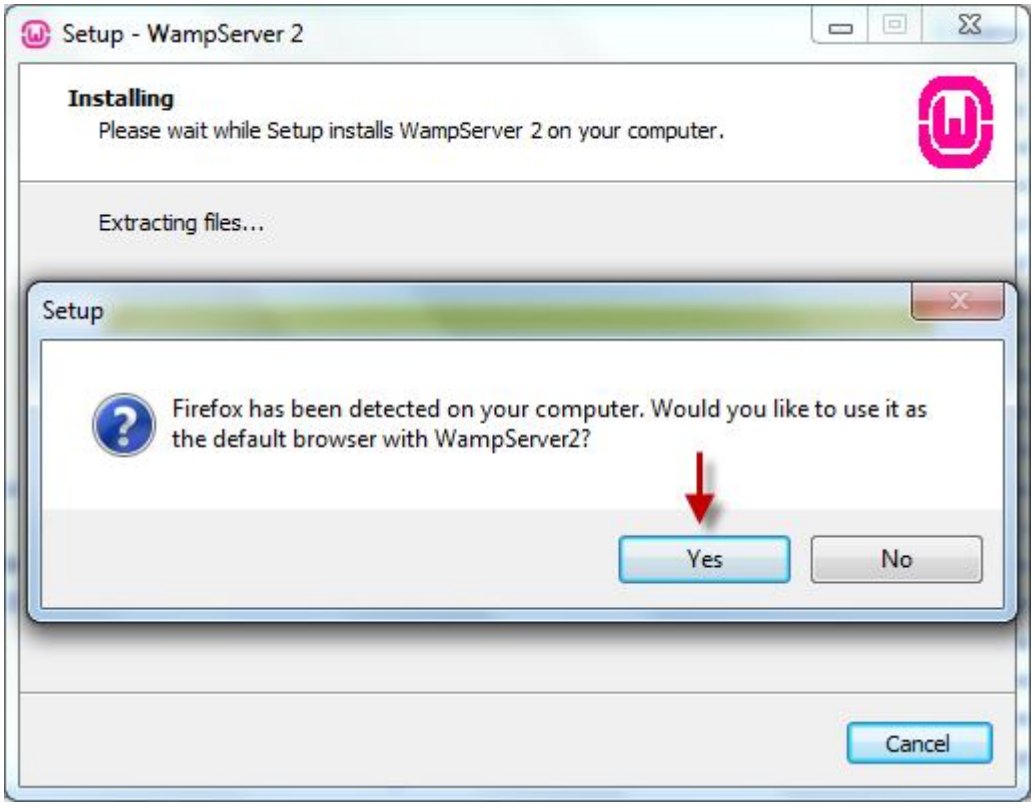

Setup - WampServer 2
License Agreement
Please read the following important information before continuing.
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
** WampServer
by
Creator : Romain Bourdon
Maintainer / Upgrade/Roadmap : Herve Leclerc - herve.leclerc@alterway.fr
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
I accept the agreement
I do not accept the agreement
< Back
Next >
Cancel

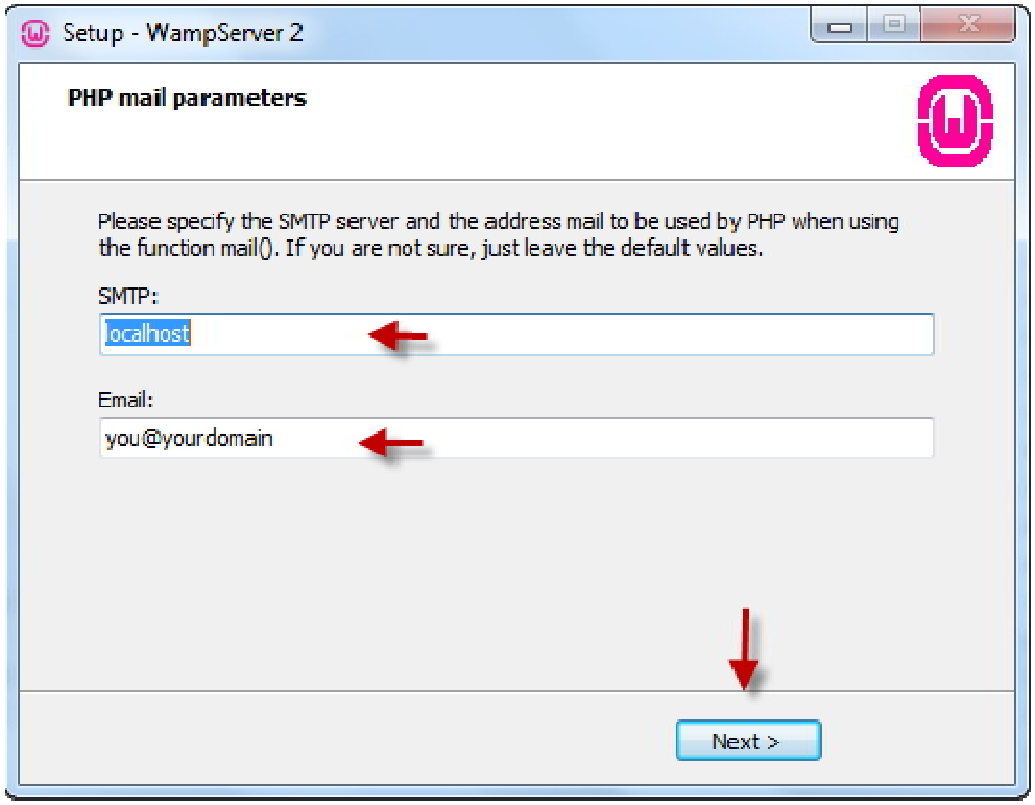

Setup - WampServer 2
Select Destination Location
Where should WampServer 2 be installed?
Setup will install WampServer 2 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
c:\wamp
Browse...
At least 398.4 MB of free disk space is required.
< Back
Next >
Cancel

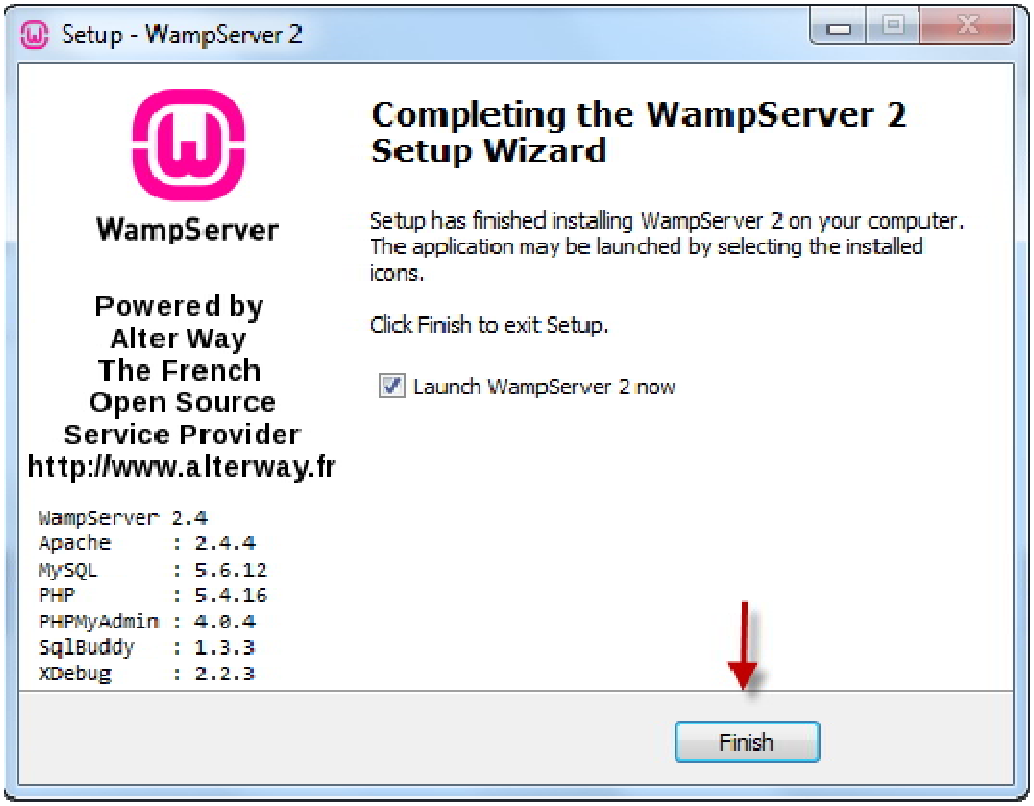

Setup - WampServer 2
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing WampServer 2, then click Next.
Additional icons:
Create a Quick Launch icon
Create a Desktop icon
< Back
Next >
Cancel


Setup - WampServer 2
Ready to Install
Setup is now ready to begin installing WampServer 2 on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
c:\wamp
Additional tasks:
Additional icons:
Create a Desktop icon
< Back
Install
Cancel


Setup - WampServer 2
Installing
Please wait while Setup installs WampServer 2 on your computer.
Extracting files...
c:\wamp\apps\phpmyadmin4.0.4\locale\es\LC_MESSAGES\phpmyadmin.mo
Cancel

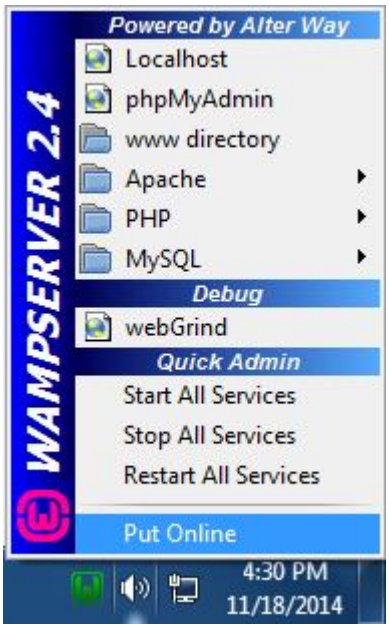

Setup - WampServer 2
Installing
Please wait while Setup installs WampServer 2 on your computer.
Extracting files...
Setup
Firefox has been detected on your computer. Would you like to use it as the default browser with WampServer2?
Yes
No
Cancel


Setup - WampServer 2
PHP mail parameters
Please specify the SMTP server and the address mail to be used by PHP when using the function mail(). If you are not sure, just leave the default values.
SMTP:
localhost
Email:
you@yourdomain
Next >

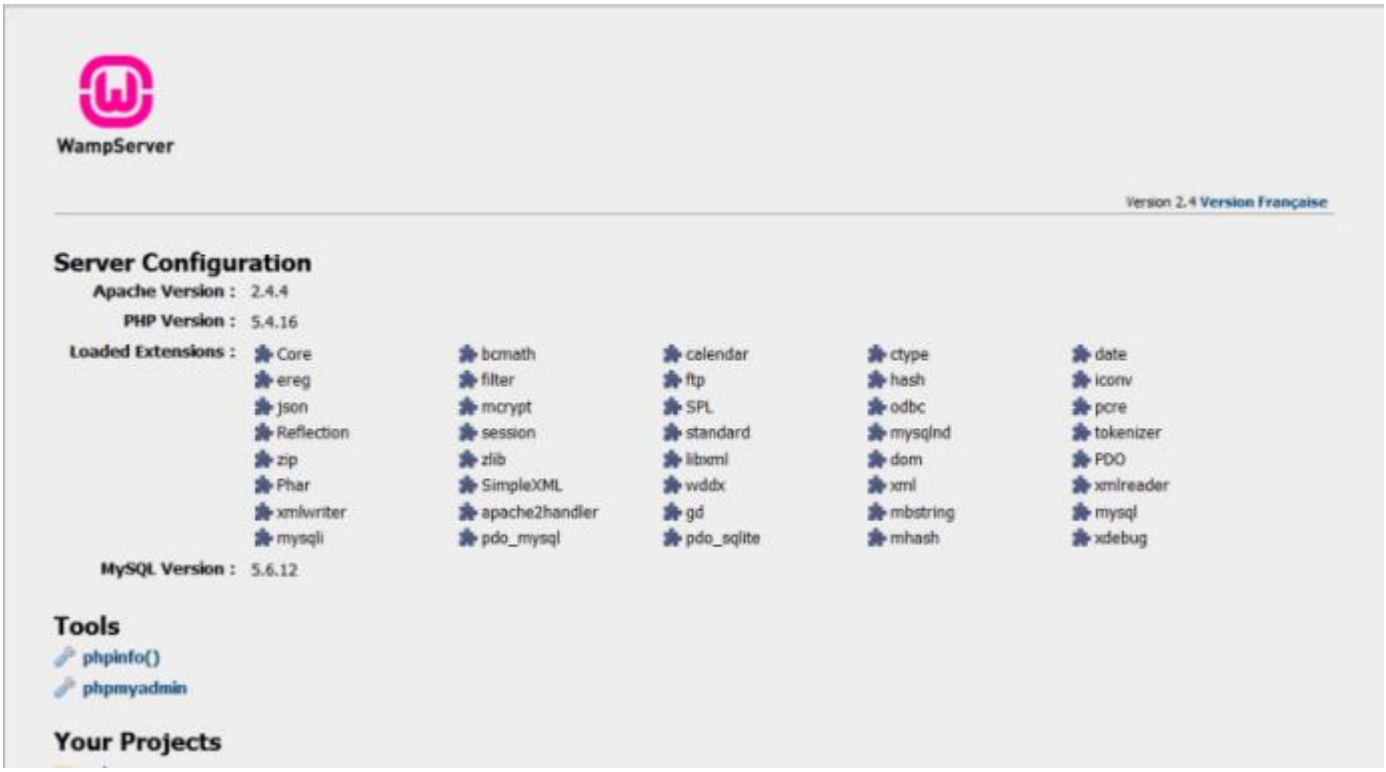


بعد از نصب آن یک آیکون جدید مانند شکل زیر در سمت راست و پایین صفحه مانیتور و درست کنار آیکون ساعت مشاهده خواهید کرد:

بر روی آیکونی که می بینید کلیک کنید تا یک منو باز شود.در منوی ظاهر شده می توانید سرور را متوقف کرده، از آن خارج شوید و صفحات پیکربندی را مشاهده نمایید. بر روی localhost کلیک کنید مشاهده می کنید که یک صفحه ظاهر می شود (localhost) : به سروری که بر روی کامپیوتر شما نصب شده است رجوع می کند،راه دیگر برای رجوع به سرور استفاده از IP ، 127.0.0.1می باشد(.



در قسمت **Tools** بر روی **phpinfo()** کلیک کنید.

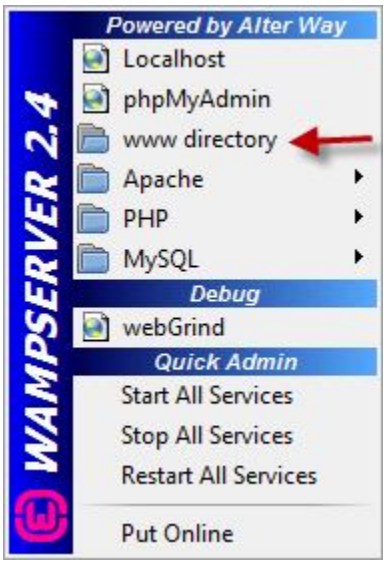


اگر همه چیز به خوبی پیش برود صفحه ای به شکل زیر مشاهده خواهید کرد که نشان دهنده نصب صحیح سرور بوده و شما می توانید شروع به کدنویسی کنید.

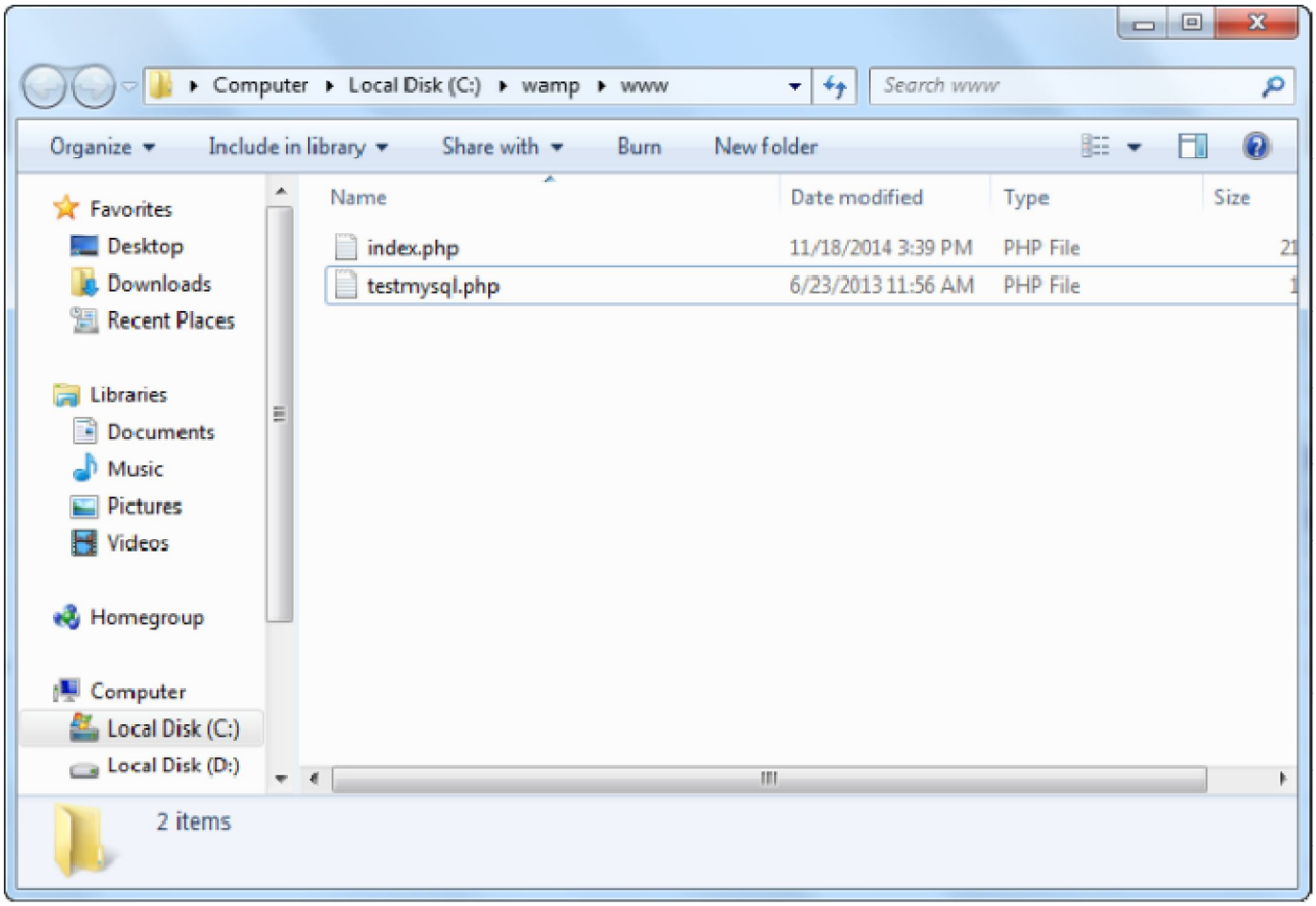
System	Windows NT ARTA-PC 6.1 build 7600 (Windows 7 Ultimate Edition) i586
Build Date	Jun 5 2013 20:58:05
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\wamp\bin\apache\apache2.4.4\bin\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini	(none)

ذخیره فایل های PHP

وقتی که یک صفحه **PHP** ایجاد کردید لازم است که آن را در پوشه **WWW** برنامه **Wampserver** ذخیره کنید. این پوشه با کلیک بر روی آیکون برنامه قابل مشاهده است. به شکل زیر توجه کنید:



وقتی بر روی **WWW** کلیک می کنید پنجره ای به شکل زیر ظاهر می شود. در داخل این پنجره ممکن است فقط دو فایل **index** و **testmysql** وجود داشته باشند.



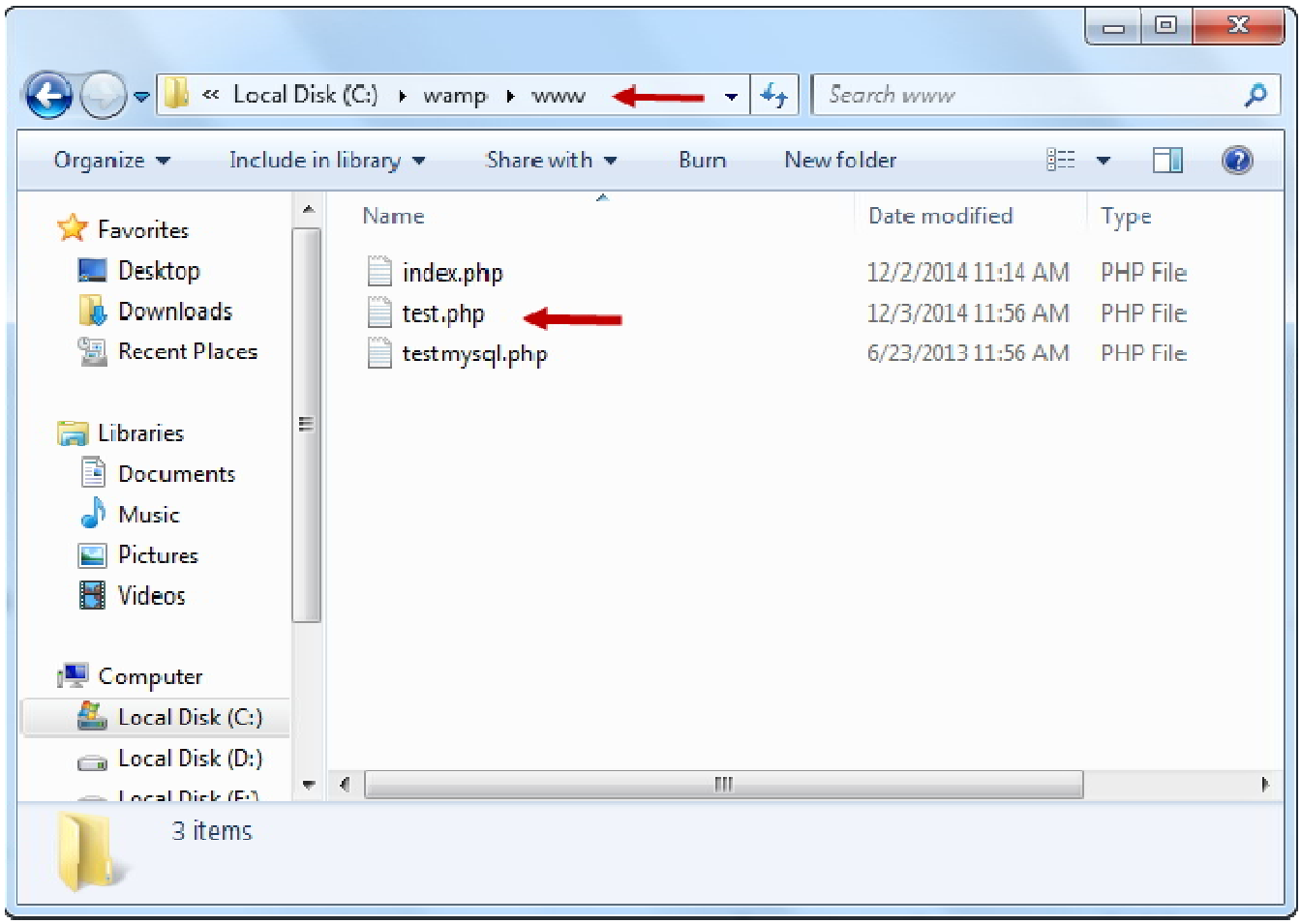
پوشه **WWW** معمولا در مسیر زیر قرار دارد :

```
c:/wamp/www/
```

پس برای اجرای فایل های **PHP** آنها را در این قسمت ذخیره کنید .

استفاده از PHP

برای شروع کار با **PHP** ابتدا یک فایل با پسوند **PHP** به نام مثلا **test.php** در پوشه **www** ایجاد کنید .



یک فایل **php** می تواند شامل کدها یا تگ های **HTML** هم باشد. برای این منظور فایل ایجاد شده را با یکی از برنامه های ویرایشگر متن مانند **Notepad** ویندوز، **Dreamweaver**و یا برنامه (**Notepad++** پیشنهاد ما) باز کرده و کدهای **HTML** زیر را در داخل آن بنویسید:

```
<html>
<head><title>PHP Test</title></head>
<body></body>
</html>
```

اضافه کردن کدهای PHP

برای اضافه کردن کدهای **PHP** چهار روش وجود دارد:

در حالت اول که استانداردترین حالت است از **php?>**و **<?** ااستفاده می شود. یعنی شما دستورات **PHP** را در داخل این دو علامت قرار می دهید:

```
<?php … ?>
```

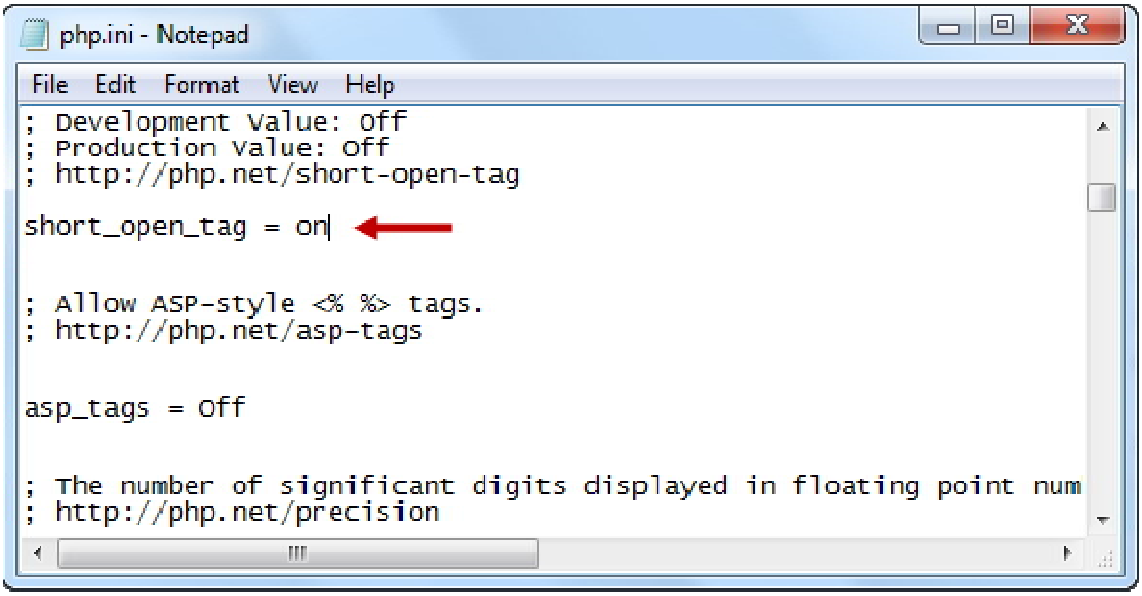
در حالت دوم که خلاصه شده حالت بالاست از **?>**و **<?** استفاده می شود:

```
<? … ?>
```

برای استفاده از روش بالا باید تغییراتی را در فایل **php.ini** اعمال کنید و آن را فعال کنید، بدین منظور به مسیر زیر بروید:

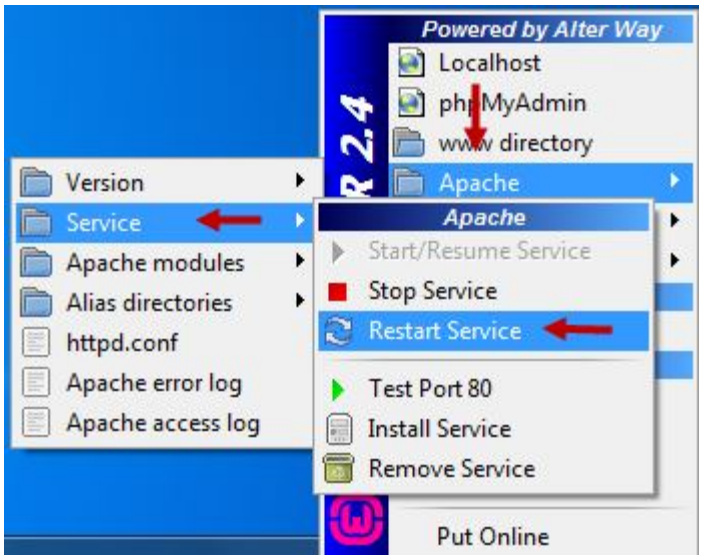
```
C:\wamp\bin\apache\Apache2.4.4\bin
```

و فایل **php.ini** را باز کرده و مقدار **short_open_tag** را به صورت زیر تغییر دهید:



بعد از اعمال تغییرات بالا، **apache**را ریستارت کنید

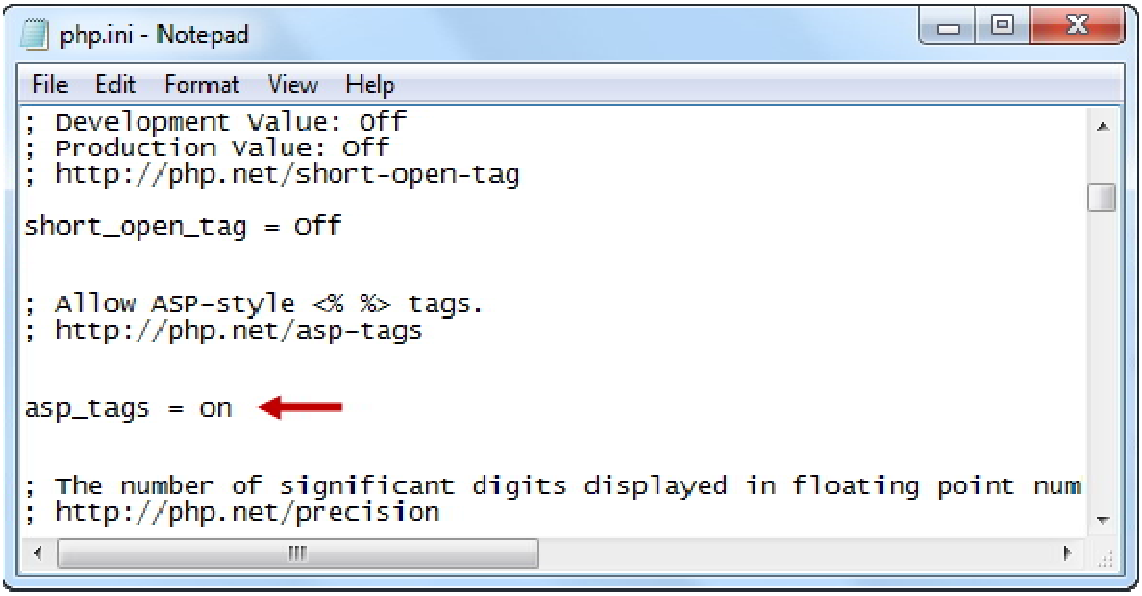
:



در حالت سوم از **%>**و **<%** استفاده می شود:

```
<% ... %>
```

برای استفاده از روش بالا باید تغییراتی را در فایل **php.ini** اعمال کنید و آن را فعال کنید، بدین منظور باید مقدار **asp_tags** را به صورت زیر تغییر دهید :



و در حالت چهارم نیز به صورت زیر عمل می شود :

```
<script language="php">...</script>
```

خروجی متن در PHP

برای چاپ مقادیر در **PHP** و نمایش آنها در مرورگر از دو دستور **echo** و **print** استفاده می شود. در پایان هر یک از این دستورات باید از علامت سمیکالن (;) استفاده شود. البته استفاده از سمیکالن در آخرین دستور اختیاری است.

```
<?php

   echo "Hello World";
   print "Hello World"

?>
```

برای تولید خروجی می توان از علامت **<?=**هم استفاده کرد. این ساختار در **PHP 5.4** ارائه شده است :

```
<?= "Hello World" ?>
```

به یاد داشته باشید که برای نمایش خروجی کد را در داخل تگ body به صورت زیر بنویسید :

```
<html>

<head><title>PHP Test</title></head>
<body>

<?php echo "Hello World"; ?>

</body>
</html>
```

یک برنامه ساده باPHP

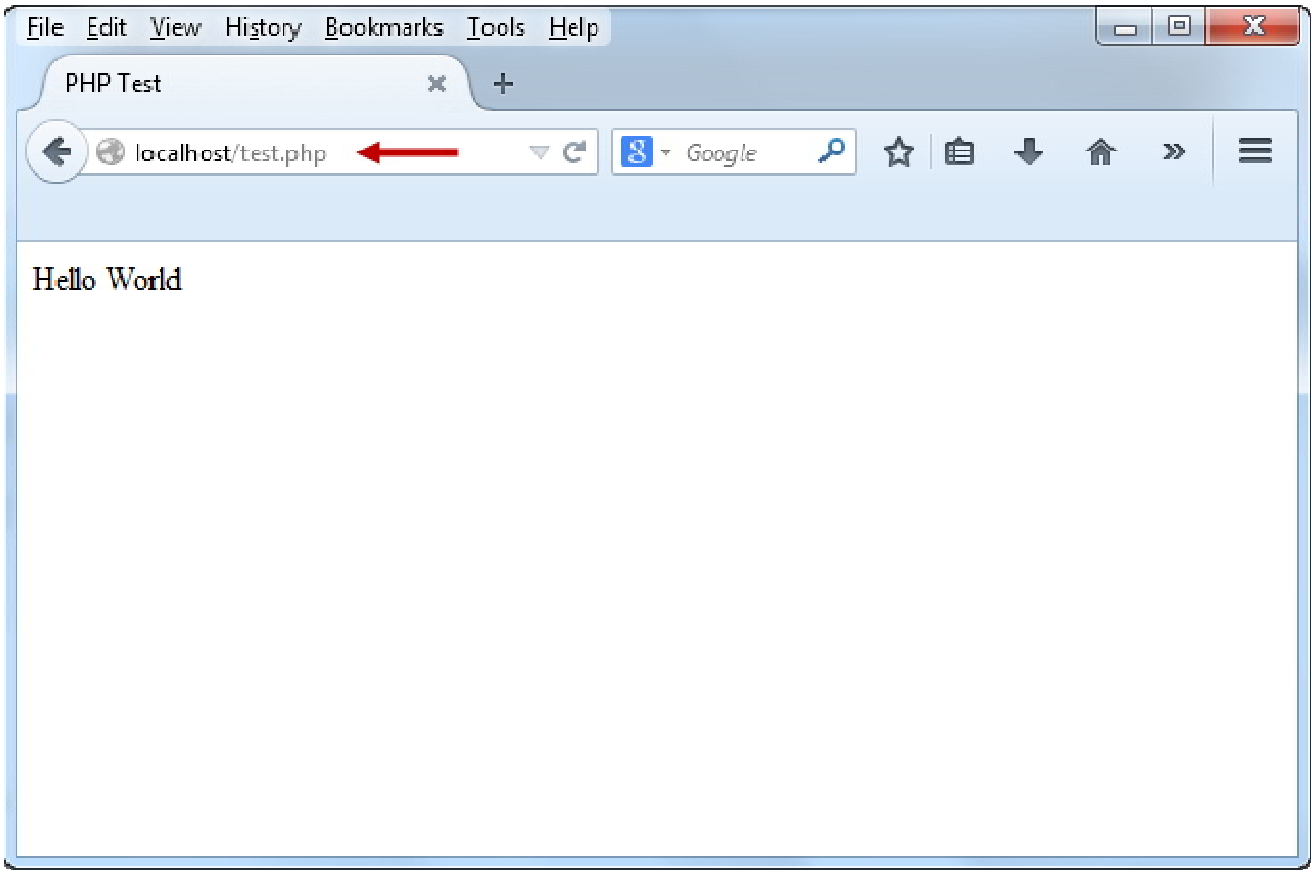
برای اینکه با عملکرد PHP بیشتر آشنا شوید ابتدا فایل test.php را که در پوشه www ایجاد کرده اید را با ویرایشگر متن باز کرده و کدهای زیر را داخل آن بنویسید :

```
<head><title>PHP Test</title></head>
<body>

<?php echo "Hello World"; ?>

</body>
</html>
```

حال کد را ذخیره کرده و مرورگرتان را باز و دستور زیر را در نوار آدرس آن بنویسید :

```
http://localhost/test.php
```

با زدن دکمه Enter کدهای PHP در سرور پردازش و پس از تبدیل به کدهایHTML ، به مرورگر ارسال می شوند. مرورگر هم با مشاهده و خواندن کدهای HTMLخروجی مطلوب را به ما ارائه می دهد :



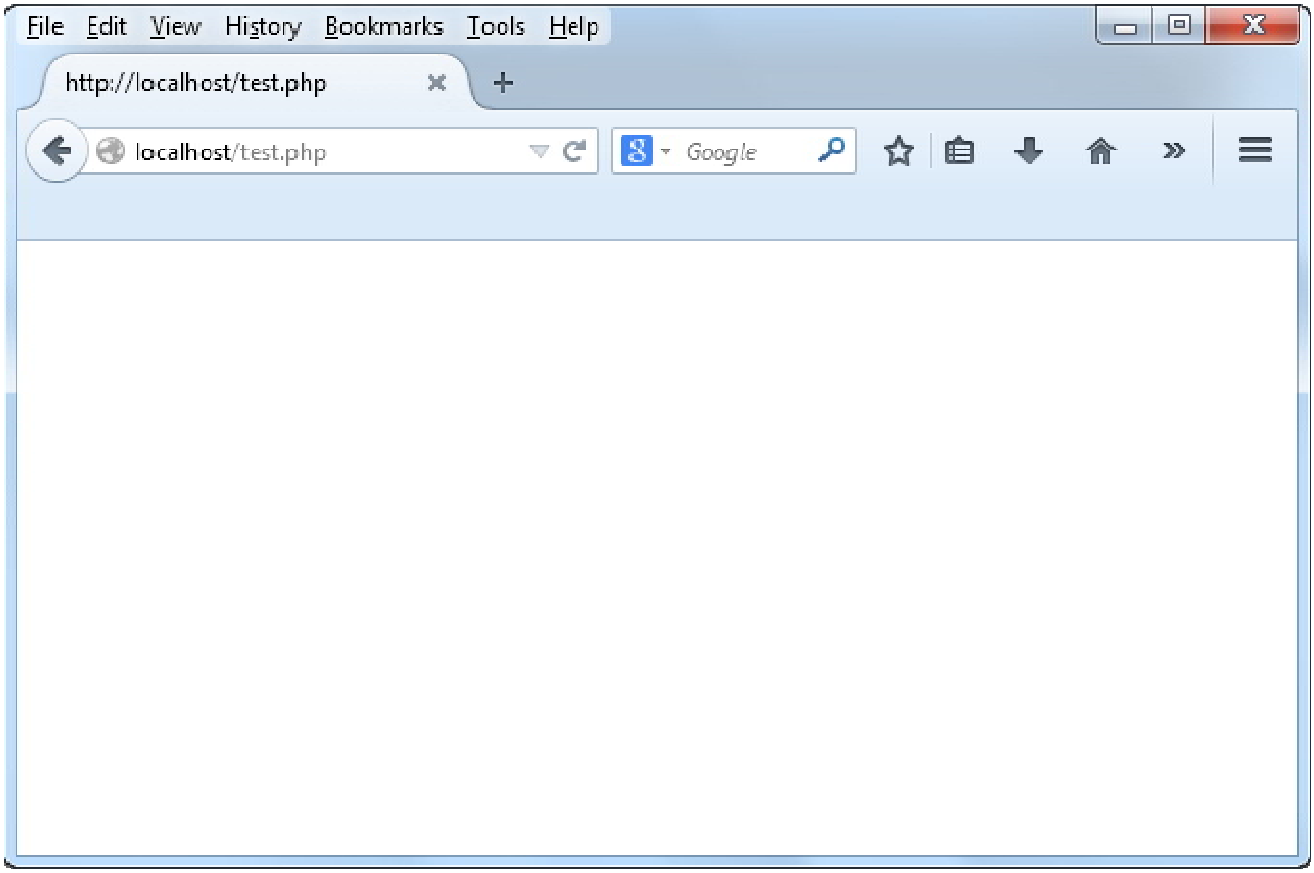
توضیحات

توضیحات در زبان های برنامه نویسی بسیار مفید هستند. آنها در به یاد آوری وظایف کدها به شما کمک می کنند. توضیحات در **PHP** به سه صورت اعمال می شوند.

```
<?php

   // single-line comment

   # single-line comment

   /* multi-line
   comment */

?>
```

استفاده از توضیحات در برنامه می تواند به شما و دیگران در فهم کدهایتان کمک کند. بدین صورت که در کارهای تیمی کسی که کدهای شما را می بیند با استفاده از توضیحاتی که در مورد کدها داده اید می فهمد که هر کد چه وظیفه ای دارد.

دستورات بالا را در داخل فایل **test.php** نوشته و ذخیره کنید. مشاهده می کنید که با اجرای آن دستورات بالا در مرورگر نمایش داده نمی شوند :



ادغام کدهای HTML وPHP

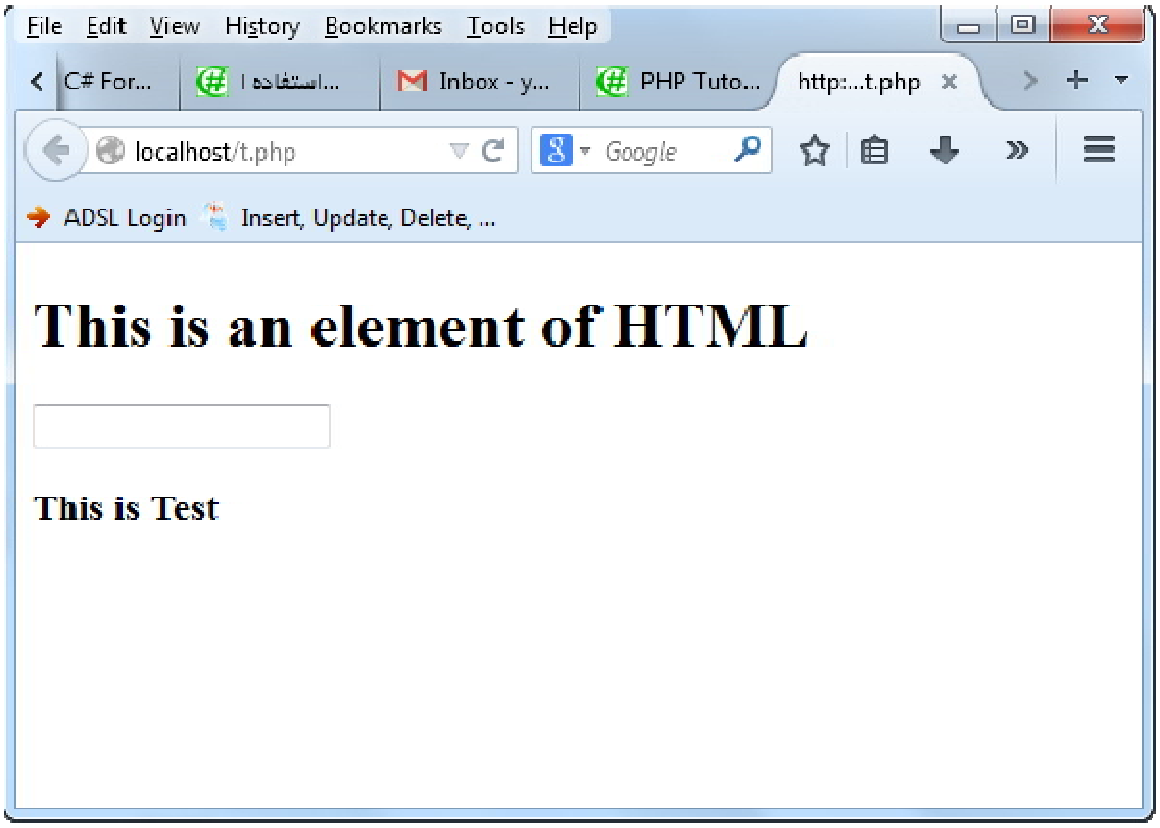
در یک فایل با پسوند **.php** می توان کدهای **HTML** و **PHP** را با هم ادغام کرد. یک مثال می زنیم. فرض کنید که می خواهید یک جعبه متن **HTML** را به وسیله دستور **PHP** ایجاد کنید برای این کار از دستور **echo** به صورت زیر استفاده می شود :

```
1:<?php
2:    echo '<h1> This is an element of HTML </H1>';
3:
4:    echo ' <input type="text"/> ';
5:
6:    echo '<H3>This is Test</H3>';
7:?>
```

روش دیگر این است که قبل از ایجاد کد **HTML** دستور **PHP** را تمام کنیم :

```php
<?php
    echo'<h1> This is an element of HTML </H1>';
?>


<input type="text"/>


<?php
    echo '<H3>This is Test</H3>';
?>
```

به خط 4 کد اول و به خط 6 کد دوم توجه کنید. درکد اول جعبه متن با استفاده از دستور echo ایجاد شده است، اما در کد دوم قبل از جعبه متن کد PHP را بسته و جعبه متن را ایجاد کرده ایم.خروجی هر دو کد بالا به صورت زیر است:



از کدهای PHP در داخل تگ های HTML هم می توان استفاده کرد. فرض کنید متن "Hello World" را یک بار با و بار دیگر بدون استفاده از PHP می خواهیم که در داخل جعبه متن بنویسیم:

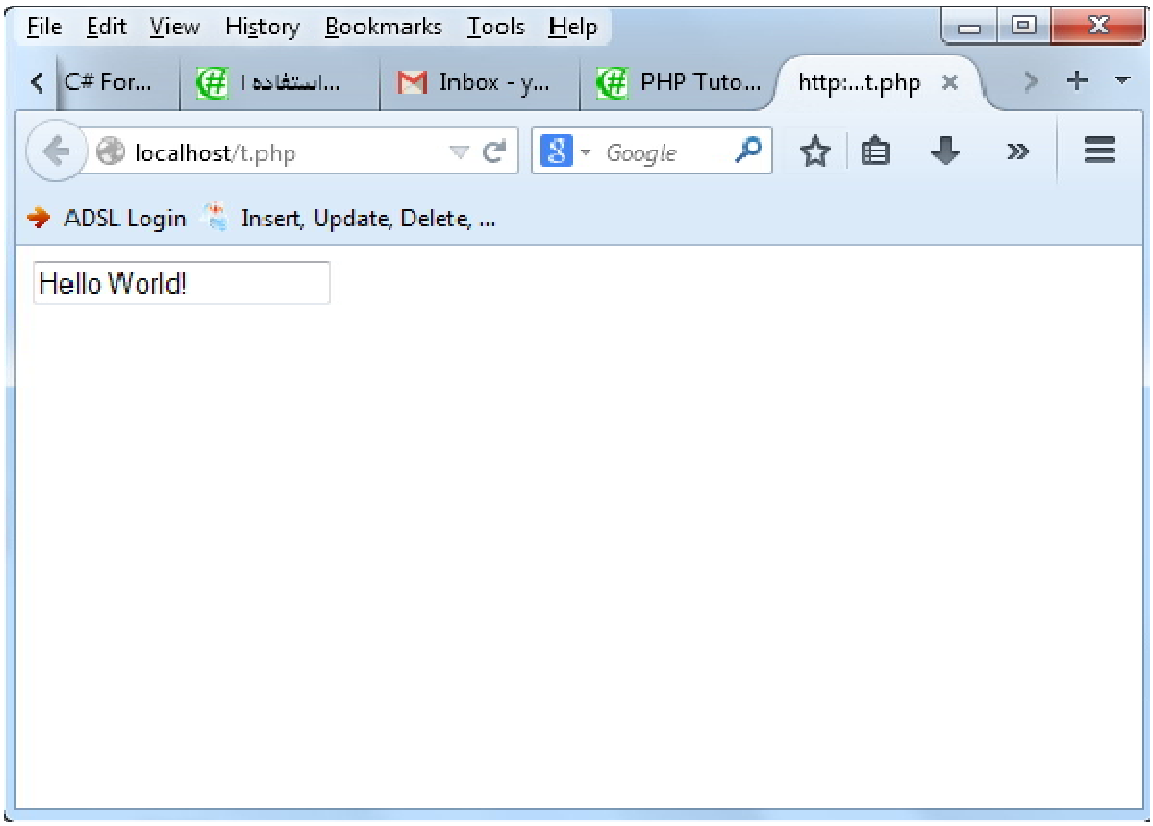
بدون استفاده از دستور PHP

```
<input type="text" value="Hello World!"/>
```

با استفاده از دستور PHP

```
<input type="text" value="<?php echo 'Hello World!'; ?>"/>
```

خروجی هر دو کد بالا به صورت زیر است :



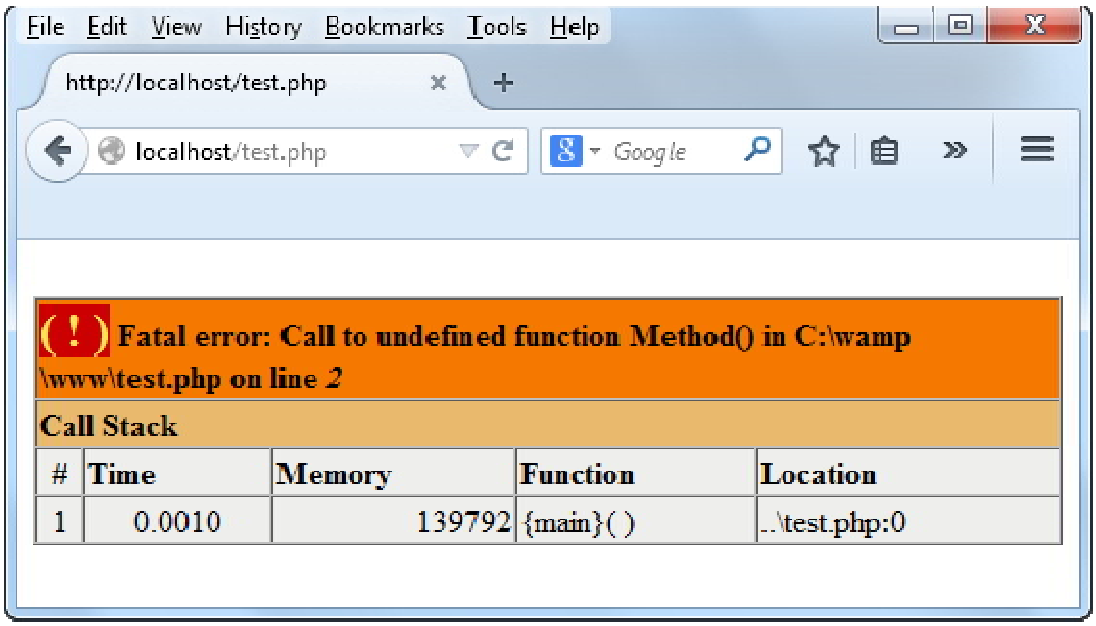
انواع خطاها درPHP

هنگام کدنویسی در PHP ممکن است با خطاهایی مواجه شویم. خطاها ممکن است بر اثر اشتباه تایپی و یا اشتباه در منطق برنامه به وجود بیایند. در جدول زیر لیست خطاهایی که ممکن است در هنگام برنامه نویسی PHP به وجود بیایند آمده است:

خطا	توضیح
Fatal error	این نوع از خطاها که به خطاهای بحرانی هم معروف هستند باعث می شوند که ادامه کار برنامه با مشکل مواجه شده و برنامه اجرا نشود.
Parse error	این نوع خطاها فقط در زمان اجرای برنامه تولید می شوند و اسم دیگر این نوع خطاها Syntax Error می باشد. فراموش کردن یک سمیکالن و یا خطای تایپی باعث به وجود آمدن این خطا ها می شود. این خطاها هم از اجرای برنامه بقیه برنامه جلوگیری می کنند.
Warning	این نوع خطاها توسط PHP به کاربر نمایش داده می شوند، اما مانع از اجرای بقیه برنامه نمی شوند.مثلا وقتی یک عدد رو بر صفر تقسیم می کنیم یک Warning دریافت می کنیم.
Notices	این نوع هم مثل انواع خطاهای قبلی می تواند خودکار توسط خود PHP و یا با استفاده از تابع trigger_error که توسط کاربر ایجاد شده است درست شوند. این نوع خطا بیشتر هشداری است.

مثالی از **Fatal Error**

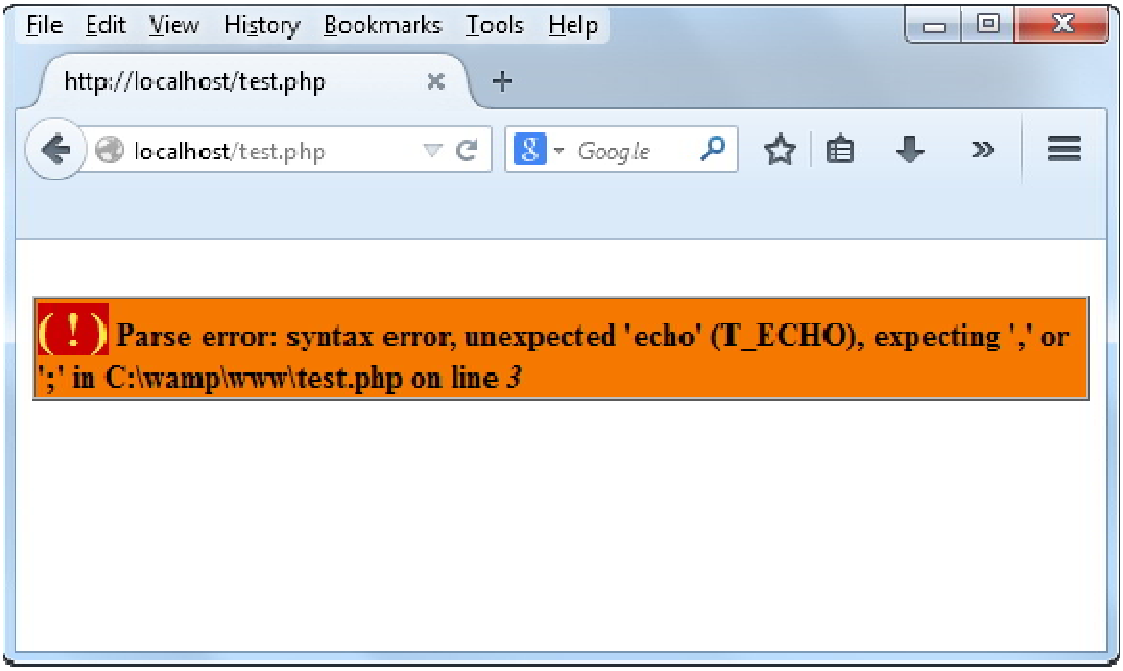
```php
<?php
    Method();
    echo "Save Successfully!"
?>
```



پیغام خطای بالا به این دلیل به وجود آمده است، که **PHP** نتوانسته است تابع **Method()** را پیدا کند، چون تابع در جایی تعریف نشده است.

مثالی از **Parse error**

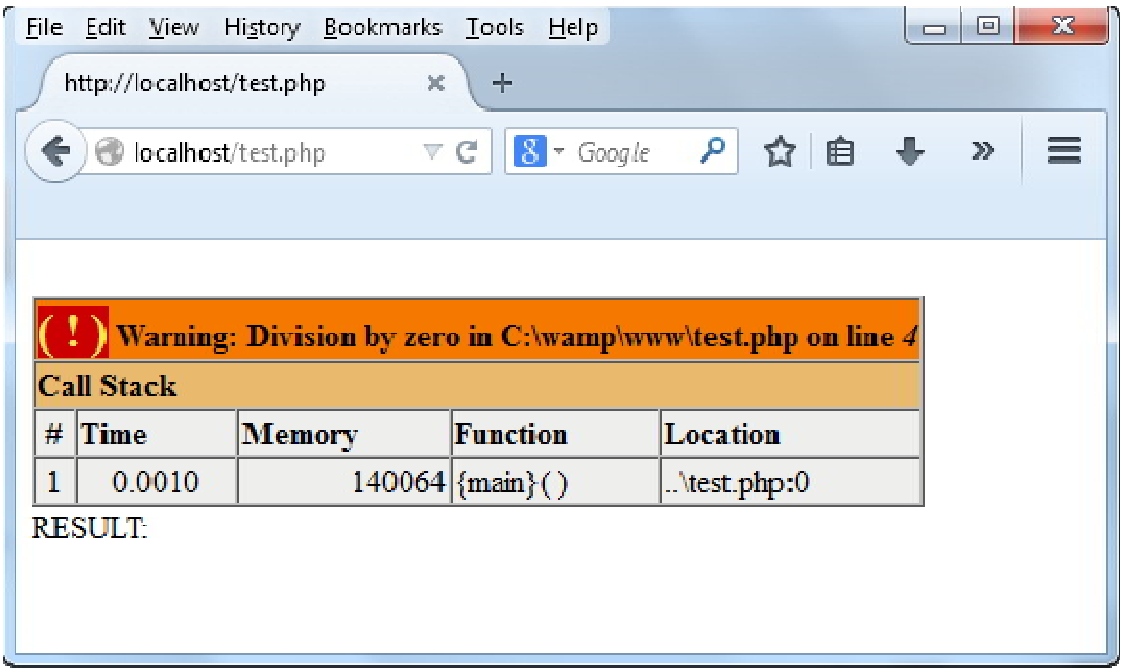
```php
<?php
    echo "Save Successfully!"
    echo "PHP Learning";
?>
```



پیغام خطای بالا به این دلیل به وجود آمده است، که در آخر کد اول یا دستور اول علامت سمیکالن (;) قرار داده نشده است .

مثالی از Warning

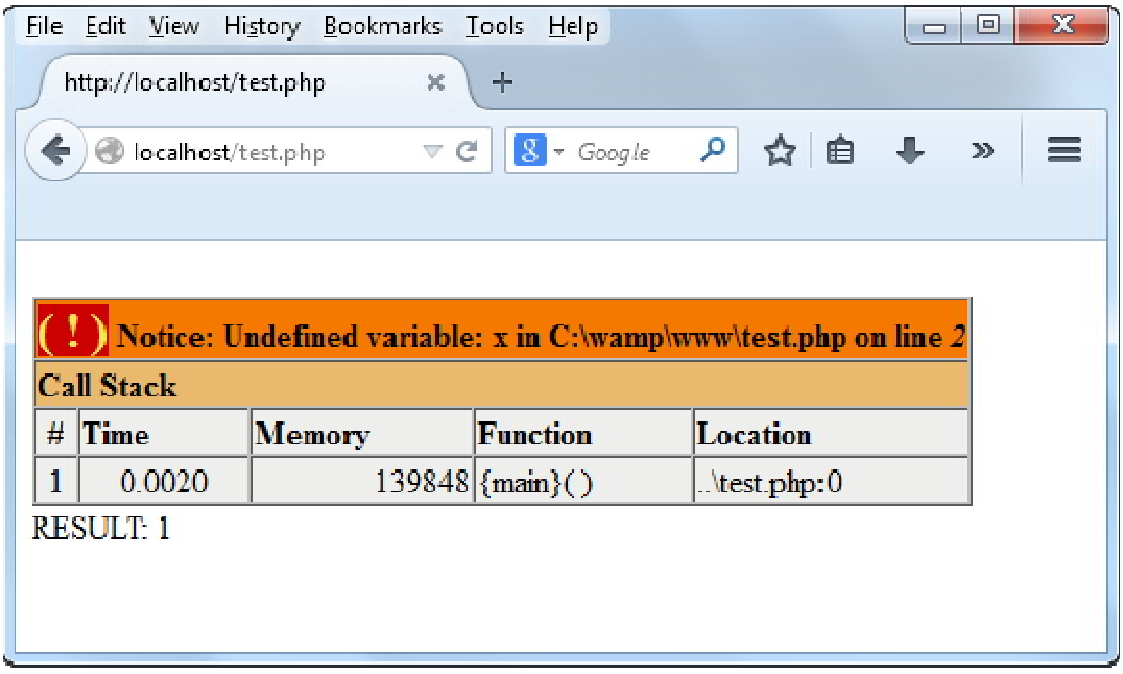
```php
<?php
    $x = 200;
    $y = 0;
    $z = $x/$y;
    echo "RESULT: ". $z;
?>
```



همانطور که در شکل بالا مشاهده می کنید، پیغام هشدار نمایش داده شده و بقیه کد هم اجرا شده است.

مثالی از **Notices**

```php
<?php
    $x += 1;
    echo "RESULT: ". $x;
?>
```



همانطور که مشاهده می کنید، برنامه اجرا و یک واحد به متغیر اضافه شده است. در پایان یاد آور می شویم که اگر کدهای بالا برای شما نا مفهوم است نگران نباشید و این بخش صرفا برای آشنایی شما با پیغام خطاهای متداول **PHP** بود.

متغیرها و انواع داده ها

متغیر یک فضای ذخیره سازی است. می توانید چیزهایی در این فضا (متغیرها) قرار دهید و در برنامه هایتان هر وقت که لازم شد از آنها استفاده کرده و یا آنها را دستکاری کنید. شما می توانید اعداد و متون را در این متغیرها ذخیره نمایید.

تعریف متغیرها

برای تعریف یک متغیر ابتدا علامت ($) و سپس نام را می نویسیم. یکی از روش های نامگذاری متغیرها این است که اگر نام متغیر چن بخش یا چند کلمه ای باشد، بجز کلمه اول، حرف اول سایر کلمات با حروف بزرگ شروع می شود:

```
$myVar;
```

مرحله بعد از نامگذاری، مقداردهی به متغیر است، برای اختصاص مقدار به یک متغیر از علامت مساوی استفاده می شود:

```
$myVar = 10;
```

وقتی که متغیر تعریف و مقدار دهی شد، می توان با استفاده از نام آن به مقدار آن دسترسی یافت. مثلا با استفاده از دستور **echo** و نام متغیر بالا می توان مقدار آن را چاپ کرد:

```
echo $myVar; // "10"
```

نکاتی در باره نامگذاری متغیرها وجود دارد که در زیر به آنها اشاره شده است:

- نام متغیرها به حروف بزرگ و کوچک حساس است. یعنی متغیری با نام **Myvar** با متغیر **myVar** فرق دارد.
- نام متغیر می تواند شامل علامت زیر خط و عدد باشد ولی نمی تواند با عدد شروع شود.
- نام متغیر نمی تواند دارای فضای خالی و یا کلماتی باشد مه برای **PHP** دارای معنی خاصی هستند.

انواع داده در PHP

PHPیک زبان با نوع داده ای ضعیف است. بدین معنی که برای تعریف متغیر لازم نیست که نوع مقداری که باید قبول کند را تعیین کنید. بلکه متغیر به طور خودکار به نوعی تبدیل می شود که به آن اختصاص داده شده است. به مثال زیر توجه کنید:

```
$myVar = 1; // int type
$myVar = 1.5; // float type
```

در کد بالا متغیر **$myVar** در خط اول به نوع **int** و در خط دوم به نوع **float** تبدیل شده است. چون مقادیری که به ان اختصاص داده شده است از نوع اعداد صحیح و اعداد اعشاری می باشند. پس نوع متغیر از همان نوعی است که به آن اختصاص داده شده است. در جدول زیر نه نوع داده ای که در **PHP** وجود دارند، ذکر شده است:

نوع	توضیح
int	اعداد بدون ممیز، صحیح هستند .
float	اعداد با ممیز شناور، اعدادی هستند که شامل یک ممیز هستند یا اعدادی که در قالب نماد ریاضی نشان داده می شوند .
bool	مقدار داده های Boolean می تواند TRUE یا FALSE باشد .
string	رشته، یک توالی از کاراکترهاست
array	آرایه ها انواع خاصی از متغیرها به حساب می آیند که می توانند چندین داده را در قالب یک نام ذخیره کنند .
object	یک شیء نوع داده ای است که هم داده ها و هم اطلاعات مربوط به نحوه پردازش آنها را ذخیره می کند .
resource	مرجعی به یک منبع خارجی مثل DB می باشد .
callable	متدها و توابع
null	با استفاده از مقدارNULL ، می توان نشان داد که یک متغییر مقدار ندارد

نوع صحیح(Integer type)

عدد صحیح یک عدد کامل است. عدد صحیح می تواند در مبنای 10 ، در مبنای 16، در مبنای 8 و یا در مبنای 2 ، نمایش داده شود. اعداد در مبنای 16 با "0x"، اعداد در مبنای 8 با "0" و اعداد در مبنای 2 با "0b" شروع می شوند .

```
$myInt = 1234; // decimal number
$myInt = 0b10; // binary number (2 decimal)
$myInt = 0123; // octal number (83 decimal)
$myInt = 0x1A; // hexadecimal number (26 decimal)
```

اعداد صحیح در PHP شامل اعداد منفی و مثبت می باشد .

نوع با ممیز اعشاری(Floating-point type)

در php اعداد اعشاری را میتوان به روشهای متفاوتی نوشت. اندازه ی این نوع داده نیز به نوع سیستم عامل و پردازنده بستگی دارد ولی حداکثر عدد 1.8e308 با 14 رقم اعشار را می توان در نظر گرفت .

```
$myFloat = 1.234;
$myFloat = 3e2; // 3*10^2 = 300
```

نوع بولی(Bool type)

ساده ترین نوع داده درPHP ، داده های Boolean می باشند که تنها دو مقدار True و False را دریافت می کنند.

```
$myBool = true;
```

نوع تهی(Null type)

این نوع داده که از نوع مخصوص می باشد تنها یک مقدار را دریافت می کند و آن هم مقدار NULL است .مشخص میکند که یک متغیر دارای مقدار نمی باشد.

```
$myNull = null; // variable is set to null
```

نکاتی در مورد کار با انواع داده درPHP

PHP نوع داده را بصورت اتوماتیک معین می کند. وقتی اسکریپت های PHP را می نویسید احتیاجی به معین کردن نوع داده ای که ذخیره می کنید، ندارید. مثال های زیر دو نوع داده مختلف درست می کنند.

```
$myVar1 = 123;
$myVar2 = "123";
```

مقدار برای $myVar به عنوان integer ذخیره می شود و مقدار متغیر دوم به عنوان string و رشته ای، برای اینکه مقدار این متغیر در داخل کوتیشن قرار گرفته است.
PHP وقتی احتیاج باشد، انواه داده ها را بصورت اتوماتیک تبدیل می کند. برای مثال اگر دو متغیر را جمع کنید و متغیر اول شامل عدد صحیح و دیگری اعشاری باشد، PHPبصورت خودکار integer را به float تبدیل می کند و بدین صورت قابل جمع با متغیر دوم می شود.

می توانید نوع داده را مشخص کنید. گه گاه ممکن است بخواهید مقدار متغیر را خارج از نوع داده ای که PHP به صورت اتوماتیک مشخص می کند، ذخیره کنید. می توانید انواع داده در PHP را بصورت زیر تعیین کنید:

```
$myVar1 = "111";
$myVar2 = (int) $var1;
```

این دستور، متغیر $myVar2 را مساوی $myVar1 می کند و مقدار رشته ای آن را به عدد صحیح تبدیل می کند. همچنین می توانید این تغییر انواع داده را با (float)یا (string) نیز انجام دهید.

می توانید نوع داده را جستجو کنید. توسط دستور و تابع `var_dump()` می توانید بفهمید چه نوع داده ای در یک متغیر ذخیره شده است. برای مثال می توانید یک متغیر را مانند زیر نمایش بدهید :

```
var_dump($myVar);
```

خروجی این متغیر چنین می شود:

```
int(111)
```

درباره سایر انواع داده ای در درس های آینده توضیح می دهیم .

ثابت ها

ثابت ها انواعی از متغیرها هستند که مقدار آنها در طول برنامه تغییر نمی کند. ثابت ها حتما باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می آید. بعد از این که به ثابت ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی توان آن را تغییر داد .

برای تعریف ثابت ها باید از کلمه کلیدی `const` و تابع `define` استفاده کرد.معمولا نام ثابت ها را طبق قرارداد با حروف بزرگ می نویسند تا تشخیص آنها در برنامه راحت باشد. نحوه تعریف ثابت به دو روش بالا در زیر آمده است :

```
const CONSTNAME= initial_value;
define('CONSTNAME', initial_value);
```

مثال:

```
<?php

    const NUMBER = 10;

    define('PI', 3.14);

    echo NUMBER;
    echo '<br/>';
    echo PI;

?>
10
3.14
```

همانطور که در مثال بالا مشاهده می کنید قبل از نام ثابت ها نباید از علامت $ و برای فراخوانی آنها در دستور `echo` نمی بایست از علامت " " استفاده کنیم. مقدار دادن به یک ثابت ،که قبلا مقدار دهی شده برنامه را با خطا مواجه می کند :

```php
<?php

    const NUMBER = 10;

    NUMBER = 12;

    echo NUMBER;

?>
```

```
(!) Parse error: syntax error, unexpected '='
```

نکته ی دیگری که نباید فراموش شود این است که نباید مقدار ثابت را با مقدار دیگر متغیرهای تعریف شده در برنامه برابر قرار داد . مثال :

```php
<?php

   $variable=12;
   const NUMBER = 10;

   NUMBER = $variable;

   echo NUMBER;

?>
```

```
(!) Parse error: syntax error, unexpected '='
```

ممکن است این سوال برایتان پیش آمده باشد که دلیل استفاده از ثابت ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی کنند بهتر است که آنها را به صورت ثابت تعریف کنید.این کار هر چند کوچک کیفیت برنامه شما را بالا می برد .

عملگرها

عملگر یک نماد است که یک عمل خاصی را در یک عبارت انجام می دهد. در PHP پنج نوع عملگر وجود دارد که عبارتند از :

- عملگرهای ریاضی
- عملگرهای تخصیصی
- عملگرهای مقایسه ای
- عملگرهای منطقی
- عملگرهای بیتی

عملگرهای ریاضی

عملگرهای ریاضی در PHP شامل عملیات جمع، تفریق، تقسیم، ضرب و قدر مطلق باقی‌مانده تقسیم است. در جدول زیر با روش نمایش این عملگرها در PHP آشنا می‌شویم :

عملگر	نام	توضیحات
x + y	جمع	جمع 2 , 2
x - y	تفریق	کم کردن دو مقدار از هم
x * y	ضرب	ضرب دو مقدار در هم
x / y	تقسیم	بدست اوردن خارج قسمت تقسیم دو مقدار
x % y	باقیمانده	باقیمانده تقسیم صحیح

مثال

```php
<?php

$number1=10;
$number2=6;

echo ($number1 + $number2);
echo "<br>";
echo ($number1 - $number2);
echo "<br>";
echo ($number1 * $number2);
echo "<br>";
echo ($number1 / $number2);
echo "<br>";
echo ($number1 % $number2);

?>
```

```
16
4
60
1.6666666666667
4
```

عملگرهای تخصیصی

برای اختصاص دادن یک مقدار به یک متغیر از عملگر Assignment یا انتسابی استفاده می‌شود. پایه این عملیات در پی اچ پی، علامت " = " است. در جدول زیر با این نماد این عملگرها در PHP و نحوه انجام دستور، آشنا شوید.

عملگر	مثال	توضیحات
x = y	x = y	مقدار سمت راست در متغیر سمت چپ قرار می گیرد
x += y	x = x + y	جمع
x -= y	x = x - y	تفریق
x *= y	x = x * y	ضرب
x /= y	x = x / y	تقسیم
x %= y	x = x % y	باقیمانده تقسیم صحیح

مثال

```php
<?php
$number1=10;
echo $number1;
echo "<br>";

$number2=20;
$number2 += 100;
echo  $number2;
echo  "<br>";

$number3=50;
$number3 -= 25;
echo  $number3;
echo  "<br>";

$number4=5;
$number4 *= 6;
echo  $number4;
echo  "<br>";

$number5=10;
$number5 /= 5;
```

```php
echo  $number5;
echo  "<br>";

$number6=15;
$number6 %= 4;
echo  $number6;
?>
```

```
10
120
25
30
2
3
```

عملگرهای افزایشی / کاهشی

این عملگرها در PHP با افزایش یا کاهش یک واحد، در مقدار متغیرها، تغییر ایجاد می‌کنند. گاه به این صورت که مقدار متغیر را افزایش یا کاهش دهد و سپس نمایش دهد و یا همان مقدار را نمایش می دهد و در صورتی که متغیر را دوباره بخوانیم، افزایش یا کاهش اعمال شده را ارسال می‌کند. در جدول زیر با نماد این عملگرها و نوع کار آن‌ها بیشتر آشنا می‌شویم :

عملگر	نام	توضیحات
++ x	افزایش تک واحدی	ابتدا مقدار x یک واحد اضافه می شود سپس مقدار جدید x در نظر گرفته می شود
x ++	افزایش تک واحدی	ابتدا مقدار x در نظر گرفته می شود سپس به ان یک واحد اضافه می شود
- x	کاهش تک واحدی	ابتدا مقدار x یک واحد می شود سپس مقدار جدید x در نظر گرفته می شود
x -	کاهش تک واحدی	ابتدا مقدار x در نظر گرفته می شود سپس از ان یک واحد کم می شود

مثال

```php
<?php
$number1=10;
echo ++$number1;
echo "<br>";

$number2=10;
echo $number2++;
echo "<br>";

$number3=5;
echo --$number3;
echo "<br>";
```

```
$number4=5;
echo $number4--;
?>
11
10
4
5
```

عملگرهای مقایسه ای

عملگرهای مقایسه‌ای در پی اچ پی، برای مقایسه ی دومقدار استفاده می‌شوند. این مقدارها می‌توانند از جنس عدد و متن (رشته ای) باشند. در جدول زیر نماد این عملگرها در PHP و کاربرد آن‌ها آشنا می‌شوید .

عملگر	نام	توضیحات
x == y	تساوی	trueاست اگر مقدار x , y باهم برابر باشند و نوع انها مهم نیست که مثل هم باشد مثلا 2 با "2" برابر ند
x === y	شناسایی	trueاست اگر مقدار x , y باهم برابر باشند و نوع انها هم مثل هم باشد ک مثلا 2 با "2" برابر نیستند
x != y	مساوی نیست با	trueاست اگر مقدار x , y با هم برابر نباشند نوع انها را در نظر نمی گیرد
x <> y	مساوی نیست با	trueاست اگر مقدار x , y با هم برابر نباشند نوع انها را در نظر نمی گیرد
x !== y	نه مقدار نه نوع انها مثل هم نباشند	trueمی شود اگر نه مقدار نه نوع داده متغیرهای x , y مثل هم نباشند
x > y	بیشتر از	trueمی شود اگر مقدار x بزرگتر از y باشد
x < y	کمتر از	trueمی شود اگر مقدار x کمتر از مقدار y باشد .
x >= y	بیشتر مساوی	trueمی شود اگر مقدار x بزرگتر مساوی مقدار y باشد .
x <= y	کمتر مساوی	trueمی شود اگر مقدار x کوچکتر مساوی مقدار y باشد .

مثال

```php
<?php
$number1=100;
$number2="100";

var_dump($number1 == $number2);
echo "<br>";
var_dump($number1 === $number2);
echo "<br>";
var_dump($number1 != $number2);
echo "<br>";
var_dump($number1 !== $number2);
echo "<br>";



$number3=50;
$number4=90;

var_dump($number3 > $number4);
echo "<br>";
var_dump($number3 <  $number4);
?>
```

```
boolean true

boolean false

boolean false

boolean true

boolean false

boolean true
```

عملگرهای تساوی

این عملگرها برای مقایسه داده ها از جنس آرایه استفاده می‌شوند. در جدول زیر با نماد و نوع عملکرد این عملگرها در PHP آشنا می‌شوید.

عملگر	نام	توضیحات
x + y	Union	اتحاد x , y
x == y	Equality	True if x and y have the same key/value pairs
x === y	Identity	True if x and y have the same key/value pairs in the same order and are of the same type

عملگر	نام	توضیحات
x != y	Inequality	True if x is not equal to y
x <> y	Inequality	True if x is not equal to y
x !== y	Non-identity	True if x is not identical to y

مثال

```php
<?php
$array1 = array("a" => "red", "b" => "green");
$array2 = array("c" => "blue", "d" => "yellow");
$array3 = $array1 + $array2; // union of $array1 and $array2
var_dump ($array3);
echo "<br>";
var_dump ($array1 == $array2);
echo "<br>";
var_dump ($array1 === $array2);
echo "<br>";
var_dump ($array1 != $array2);
echo "<br>";
var_dump ($array1 <> $array2);
echo "<br>";
var_dump ($array1 !== $array2);
?>
```

```
array (size=4)
  'a' => string 'red' (length=3)
  'b' => string 'green' (length=5)
  'c' => string 'blue' (length=4)
  'd' => string 'yellow' (length=6)


boolean false

boolean false

boolean true

boolean true

boolean true
```

عملگرهای منطقی

عملگرهای منطقی**and** ، **or**، **xor**، **&&**، **||**، **!**، در **PHP** بین دو متغیر قرارمی‌گیرند و شرایط درست یا نادرست بودن آن متغیرها را در **PHP** بیان می‌کند. لازم به ذکر است که عملگرهای **&&**و **||** در اولویت بالاتری هستند. جدول زیر خروجی این متغیرها را در صورت قرار گرفتن عملگرهای **PHP** آن‌ها بیان می‌کند:

عملگر	نام	توضیحات
x and y	And	true می شود اگر هر دو طرف ان true باشند
x or y	Or	true می شود در صورتیکه یکی از دو طرف true باشد
x xor y	Xor	در صورتیکه فقط یکی از دو طرف true باشد جواب برابر true می شود
x && y	And	true می شود اگر فقط هر دو طرف true شوند
x || y	Or	true می شود اگر یکی از دو طرف true باشد
! x	Not	true می شود اگر x برابر false باشد و false می شود اگر x برابر true باشد

مثال

```
<?php
$number1=6;
$number2=3;

var_dump($number1 < 10  &&  $number2 > 1);

var_dump($number1==5    ||  $number2==5);

var_dump(!($number1     ==  $number1));
?>
```

```
boolean true

boolean false

boolean false
```

عملگرهای بیتی

این عملگرها بر روی بیت های یک متغییر عملی را انجام می دهند و بیتها را به نسبت عملگر برمی گردانند. اگر متغییرها رشته هستند بر روی کدهای ASCII آنها عمل می کند .

عملگر	توضیحات	نتیجه
~	not	بیت هایی را برمی گرداند که در $x نیستند .

عملگر	توضیحات	نتیجه
&	and	بیت هایی را برمی گرداند که هم در $x و هم در $y هستند.
|	or	بیت هایی را که در $x یا در $y هستند برمی گرداند.
^	xor	بیت هایی را برمی گرداند که در $x یا در $y هستند اما در هر دو نیستند.
>>	شیفت به چپ	بیت های $x را به اندازه $y تا به سمت چپ انتقال می دهد.
<<	شیفت به راست	بیت های $x را به اندازه $y تا به سمت راست انتقال می دهد.

عملگر الحاق رشته ها

در رشته ها تنها عملگری که استفاده می شود نقطه "." می باشد که دو رشته را به همدیگه متصل می کند.

عملگر	نام	توضیحات
a . b	اتصال رشته	دو رشته را به هم وصل می کند
a .= b	a = a . b	اتصال رشته ها به هم

مثال

```
<?php
    $string1 = "Hello";
    $string2 = $string1 . " world!";
    echo $string2;

    echo "<br>";

    $string1="Hello";
    $string1 .= " world!";
    echo $string1;
?>
Hello world!
Hello world!
```

رشته ها

رشته ها در PHP مجموعه ای از کارکترهای متوالی هستند که بین دو گیومه (کوتِیشِن) تک ' ' یا جفت " " قرار می گیرند. مثال :

```
$string = 'This is a text.';
$string = "This is also a text.";
```

درصورتی که بخواهید از خود کارکتر گیومه (تک یا جفت) در وسط رشته ای که ابتدا و انتهای آن توسط همان نوع گیومه مشخص شده است، استفاده کنید، باید قبل از گیومه میان رشته، کارکتر \ استفاده کنید .

```
echo 'It\'s mine. My name is "Mohammad"';
echo "My friend's name is \"Ali\".";
```

به کارکتر \ اصطلاحاً (Escape فرار) می گویند؛ زیرا برای فرار از معنای اصلی کاراکترهای بعد از آن و ایجاد معنای دیگری برای آنها بکار می رود. بدیهی است که کاراکتر گیومه تک در داخل رشته محصور به گیومه جفت و همچنین کارکتر گیومه جفت در داخل رشته محصور به گیومه تک نیاز به Escape ندارد. در جدول زیر، تعدادی از کدهای Escape متداول را در PHP مشاهده می کنید :

Escape	نتیجه
\n	حرکت به سطر بعد
\r	حرکت به ابتدای سطر جاری
\t	کارکتر (Tab معادل 8 کارکتر Space)
\\	کارکتر \
\'	کارکتر ' (در رشته های محصور به گیومه تک)
\"	کارکتر " (در رشته های محصور به گیومه جفت)
\$	کارکتر $
\[0-7]	کارکتری که کد ASCII آن در مبنای 8 در جلوی \ نوشته شده است
\x[0-F]	کارکتری که کد ASCII آن در مبنای 16 در جلوی \x نوشته شده است

ادغام رشته ها

برای ادغام رشته ها در PHP از کارکتر نقطه (.) استفاده می شود :

```
$string1 = 'PHP';
$string2 = 'Programming';
echo $string1 . ' ' . $string2 . ' is full of enjoy.<br />';
PHP Programming is full of enjoy.<br />
```

در PHP میتوانید رشته ها را با اعداد نیز ترکیب کنید :

```
$number = 5;
echo 'Test' . $number;
Test5
```

تفاوت رشته های محصور در گیومه جفت و تک

دستورات زیر را درنظر بگیرید :

```
$string = 5;
$text1 = "String is $string";
$text2 = 'String is $string';

echo $text1 . '<br/>';
echo $text2;
String is 5
String is $string
```

با اجرای دستورات فوق، عبارت String is 5 در متغیر $text1 و عبارت String is $string در متغیر $text2 ذخیره خواهد شد. علت این تفاوت در عملکرد، آن است که اسامی متغیرها در رشته هایی که بین دو گیومه جفت قرار دارند، پردازش شده و بجای نام آنها، مقدارشان در عبارت مورد استفاده قرار میگیرد. درمقابل رشته های محصور به گیومه تک، مورد هیچ پردازشی قرار نمیگیرند و به همان شکل که نوشته میشوند، مورد استفاده قرار خواهند گرفت. بنابراین اگر در رشته موردنظرتان، متغیری وجود ندارد، بهتر است آنرا در گیومه تک بگذارید تا سرعت پردازش کد شما افزایش یابد. همچنین باید دقت کنید که پردازش رشته های محصور در گیومه جفت نیز تنها محدود به اسامی متغیرهاست و هیچگونه فراخوانی توابع یا عمل محاسباتی ریاضی و... مورد پردازش قرار نخواهد گرفت. برای مثال، حاصل دستورات زیر
:

```
$string = 5;
$text = "String = ($string + 5)";

echo $text ;
String = (5 + 5)
```

ذخیره شدن عبارت String = (5 + 5) در متغیر $text است. درواقع برای محاسبه صحیح مجموع و درج آن در رشته، باید بصورت زیر عمل کنید :

```php
$string = 5;
$text = 'String = ' . ($string + 5);

echo $text ;
String = 10
```

که به موجب آن، ابتدا نتیجه محاسبه **($string + 5)** با حاصل **10** و سپس استفاده از نتیجه این محاسبه در عبارت و ادغام با رشته **'String = '** و درنتیجه ذخیره عبارت نهایی **String = 10** در متغیر **$text** است.

استفاده از رشته بعنوان عدد

همانطور که در مثال قبل ملاحظه کردید، تبدیل عدد به رشته در زمان نیاز بطور خودکار انجام میشود. عکس این موضوع نیز صحیح است و رشته ها درصورت استفاده در عبارات محاسباتی، بطور خودکار به عدد تبدیل خواهند شد؛ بدین ترتیب که از ابتدای رشته، تا زمان رسیدن به اولین کارکتر غیر عددی، جدا شده و بصورت عدد تعبیر میشود. البته اگر رشته موردنظر با عدد شروع نشده باشد، یک ثابت خاص به نام **NaN** مخفف **(Not a Number** بازگردانده خواهد شد. مثال:

```php
$text = '52Ali';
$sum = 7 + $text;

echo $sum;
59
```

آرایه ها

آرایه نوعی متغیر است که لیستی از آدرسهای مجموعه ای از داده های هم نوع را در خود ذخیره می کند، البته در **PHP** داده ها هم نوع نباشند مهم نیست.تعریف چندین متغیر از یک نوع برای هدفی یکسان بسیار خسته کننده است.مثلا اگر بخواهید صد متغیر از نوع اعداد صحیح تعریف کرده و از آنها استفاده کنید. مطمئنا تعریف این همه متغیر بسیار کسالت آور و خسته کننده است.اما با استفاده از آرایه می توان همه آنها را در یک خط تعریف کرد. در زیر راهی ساده برای تعریف یک آرایه نشان داده شده است:

```php
$numbers = array() ;
```

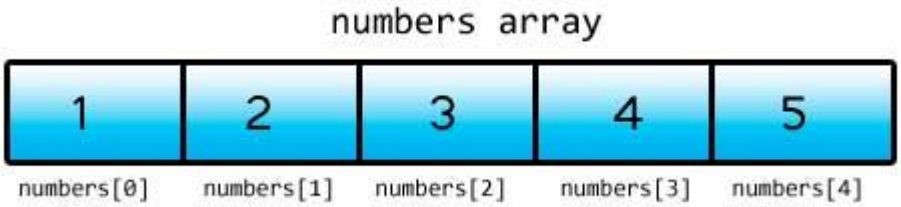
numbersنام آرایه را نشان می دهد.هنگام نامگذاری آرایه بهتر است که نام آرایه نشان دهنده نوع آرایه باشد.به عنوان مثال برای نامگذاری آرایه ای که اعداد را در خود ذخیره می کند از کلمه number استفاده کنید. با استفاده از دستور array یک آرایه ایجاد کرده ایم . حتی می توانیم به هنگام ایجاد آرایه مقادیر خانه های آن را نیز مشخص کنیم :

```
$numbers = array(1, 2, 3, 4, 5) ;
```

در این مثال 5 آدرس از فضای حافظه کامپیوتر شما برای ذخیره 5 مقدار رزرو می شود. حال چطور مقادیرمان را در هر یک از این آدرسها ذخیره کنیم؟ برای دسترسی و اصلاح مقادیر آرایه از اندیس یا مکان آنها استفاده می شود .

```
numbers[0] = 1;
numbers[1] = 2;
numbers[2] = 3;
numbers[3] = 4;
numbers[4] = 5;
```

اندیس یک آرایه از صفر شروع شده و به یک واحد کمتر از طول آرایه ختم می شود.به عنوان مثال شما یک آرایه 5 عضوی دارید،اندیس آرایه از 0 تا 4 می باشد چون طول آرایه 5 است پس 5-1 برابر است با 4.این بدان معناست که اندیس 0 نشان دهنده اولین عضو آرایه است و اندیس 1 نشان دهنده دومین عضو و الی آخر. برای درک بهتر مثال بالا به شکل زیر توجه کنید :



به هر یک از اجزاء آرایه و اندیسهای داخل کروشه توجه کنید.کسانی که تازه شروع به برنامه نویسی کرده اند معمولا در گذاشتن اندیس دچار اشتباه می شوند و مثلا ممکن است در مثال بالا اندیسها را از 1 شروع کنند .

انواع آرایه ها درPHP

در زبان PHP سه نوع آرایه وجود دارد :

- آرایه های عددی(Numeric Array)
- آرایه های انجمنی(Associative Array)
- آرایه های چند بعدی(Multidimensional Array)

آرایه های عددی(Numeric Array)

آرایه عددی یا **Numeric Array** آرایه ایست که در اکثر زبان های برنامه نویسی وجود دارد . در این آرایه ، به کل آرایه یک نام می دهیم بدین ترتیب هر یک از خانه های آرایه یک اندیس می گیرد و با آن اندیس می توانیم به یک خانه خاص آرایه دست بیابیم. به مثال زیر توجه کنید :

```
<?php

$name = array ('Jack' ,'Estephan' , 'Jordan');
echo $name[2];

?>
Jordan
```

آرایه های انجمنی(Associative Array)

آرایه انجمنی آرایه ای است که در آن به ازاء هر مقدار (هر خانه) یک فیلد بنام کلید نیز در نظر گرفته می شود که این کلید باید در بین کلید های خانه های دیگر منحصر به فرد و یکتا باشد . برای دستیابی به خانه های یک آرایه انجمنی ، دیگر از اندیس عددی استفاده نمی شود بلکه از فیلد کلید به عنوان رشته کلیدی دستیابی استفاده می شود . به مثال زیر توجه کنید :

```
<?php

$ages = array('Jack' => 21 , 'Estephan' => 24 , 'Jordan' => 19);
echo $ages["Jack"];

?>
21
```

این کد یک آرایه انجمنی با سه عضو ایجاد می کند و بعد به هنگام استفاده از کلید هایی که به هنگام تعریف داده شده برای دستیابی به عضو مربوطه استفاده می شود در اینجا باید حواستان باشد که این کلید مانند سایر موارد دیگر در **PHP** به حروف بزرگ و کوچک حساس است.**(Case Sensitive)**

آرایه های چند بعدی(Multidimensional Array)

این نوع آرایه ، آرایه ای است که هر عضو آن می تواند خود آرایه باشد . در حقیقت این نوع آرایه ، آرایه ای از آرایه ها می باشد . حال این آرایه می تواند عددی باشد یا انجمنی. به مثال زیر توجه کنید :

```
<?php

$Families = array(
                    "Griffin" => array("Peter", "Lois", "Megan") ,
                    "Quagmire"=> array("Glenn") ,
                    "Brown"   => array("Cleveland","Loretta","Junior")
```

```
                );
echo $Families['Griffin'][2];

?>
Megan
```

همانطور که از مثال فوق نیز پیداست یکی از تفاوت های عمده ای که آرایه در PHP با آرایه در سایر زبانها دارد اینست که در آرایه های چند بعدی PHP لازم نیست تعداد اعضای بعد های دوم به بعد با هم برابر باشند. و در آخر یاد آور می شویم که برای چاپ یک مقدار مورد نظر در آرایه های چند بعدی ابتدا نام آرایه اصلی سپس نام آرایه مورد نظر و بعد اندیس مقداری که قرار است چاپ شود را می نویسیم (مانند خط آخر مثال بالا)

دستورات شرطی

تقریبا همه زبانهای برنامه نویسی به شما اجازه اجرای کد را در شرایط مطمئن می دهند.حال تصور کنید که یک برنامه دارای ساختار تصمیم گیری نباشد و همه کدها را اجرا کند. این حالت شاید فقط برای چاپ یک پیغام در صفحه مناسب باشد ولی فرض کنید که شما بخواهید اگر مقدار یک متغیر با یک عدد برابر باشد سپس یک پیغام چاپ شود ان وقت با مشکل مواجه خواهید شد PHP . راه های مختلفی برای رفع این نوع مشکلات ارائه می دهد .

در این بخش با مطالب زیر آشنا خواهید شد :

- دستور**if**
- دستور**if…else**
- عملگر سه تایی
- دستور**switch**

دستورif

می توان با استفاده از دستور if و یک شرط خاص که باعث ایجاد یک کد می شود یک منطق به برنامه خود اضافه کنید.دتور if ساده ترین دستور شرطی است که برنامه می گوید اگر شرطی برقرار است کد معینی را انجام بده . ساختار دستور if به صورت زیر است :

```
if (condition)
   code to execute;
```

قبل از اجرای دستور if ابتدا شرط بررسی می شود.اگر شرط برقرار باشد یعنی درست باشد سپس کد اجرا می شود.شرط یک عبارت مقایسه ای است . می توان از عملگرهای مقایسه ای برای تست درست یا اشتباه بودن شرط استفاده کرد.اجازه بدهید که نگاهی به نحوه استفاده از دستور if در داخل برنامه بیندازیم .

برنامه زیر پیغام Hello World را اگر مقدار number کمتر از 10 و Goodbye World را اگر مقدار number از 10 بزرگتر باشد در صفحه نمایش می دهد:

```
<?php

    $number = 5;
    if ($number < 10)
       echo ("Hello World.");

    echo '<br/>';

    $number = 15;
    if ($number > 10)
       echo("Goodbye World.");

?>
```

```
Hello World.
Goodbye World.
```

در خط 3 یک متغیر با نام number تعریف و مقدار 5 به آن اختصاص داده شده است. وقتی به اولین دستور if در خط 4 می رسیم برنامه تشخیص می دهد که مقدار number از 10 کمتر است یعنی 5 کوچکتر از 10 است . منطقی است که نتیجه مقایسه درست می باشد بنابراین دستور if دستور را اجرا می کند(خط 5) و پیغام Hello World چاپ می شود. حال مقدار number را به 15 تغییر می دهیم (خط 9).وقتی به دومین دستور if در خط 10 می رسیم برنامه مقدار number را با 10 مقایسه می کند و چون مقدار number یعنی 15 از 10 بزرگتر است برنامه پیغام Goodbye World را چاپ می کند(خط 11). به این نکته توجه کنید که دستور if را می توان در یک خط نوشت:

```
if ($number > 10) echo("Goodbye World.");
```

شما می توانید چندین دستور را در داخل دستور if بنویسید. کافیست که از یک آکولاد برای نشان دادن ابتدا و انتهای دستورات استفاده کنید. همه دستورات داخل بین آکولاد جز بدنه دستور if هستند. نحوه تعریف چند دستور در داخل بدنه if به صورت زیر است:

```
if (condition)
{
    statement1;
    statement2;
    .
    .
    .
    statementN;
}
```

این هم یک مثال ساده:

```
<?php

    $number = 15;
    if ($number > 10)
    {
        echo ("x is greater than 10.").'<br/>';
        echo ("This is still part of the if statement.");
```

```
    }

?>
x is greater than 10.
This is still part of the if statement.
```

در مثال بالا اگر مقدار x از 10 بزرگتر باشد دو پیغام چاپ می شود. حال اگر به عنوان مثال آکولاد را حذف کنیم و مقدار x از 10 بزرگتر نباشد مانند کد زیر:

```
<?php

    $number = 5;
    if ($number > 10)
       echo ("x is greater than 10.").'<br/>';
       echo ("This is still part of the if statement.");

?>
```

کد بالا در صورتی بهتر خوانده می شود که بین دستورات فاصله بگذاریم.

```
    1:<?php
    2:
    3:    $number = 5;
    4:    if ($number > 10)
    5:       echo ("x is greater than 10.").'<br/>';
    6:
    7:
    8:       echo ("This is still part of the if statement.");
    9:
   10:?>
This is still part of the if statement.
```

می بینِد که دستور دوم(خط 8) در مثال بالا جز دستور if نیست.اینجاست که چون ما فرض را بر این گذاشته ایم که مقدار x از 10 کوچکتر است پس خط This is still part of the if statement. (Really?)چاپ می شود. در نتیجه اهمیت وجود آکولاد مشخص می شود.به عنوان تمرین همیشه حتی اگر فقط یک دستور در بدنه if داشتید برای آن یک آکولاد بگذارید.فراموش نکنید که از قلم انداختن یک آکولاد باعث به وجود آمدن خطا شده و یافتن آن را سخت می کند. یکی از خطاهای معمول کسانی که برنامه نویسی را تازه شروع کرده اند قرار دادن سیمیکولن در سمت راست پرانتز if است.
به عنوان مثال:

```
<?php
     if ($number > 10);
     echo ("x is greater than 10.").'<br/>';
?>
```

به یاد داشته باشید که if یک مقایسه را انجام می دهد و دستور اجرایی نیست.بنابراین برنامه شما با یک خطای منطقی مواجه می شود.

دستورif...else

فرق این دستور با دستور if در این است که شما حالت دومی را نیز برای شرط در نظر دارید، یعنی اگر شرط اول برقرار بود، دستورالعمل مناسب اجرا میشود ولی اگر شرط برقرار نبود دستورالعمل جایگزین اجرا میشود (بر خلاف دستور if که اگر شرط اول برقرار نباشد هیچ دستورالعملی اجرا نمی شود). نحوه ی استفاده از این دستور نیز مانند دستور if است با این تفاوت که بعد از بسته شدن آکولاد دستورif ، دستور else اجرا می شود:

```
if (condition)
{
    code to execute;
}
else
{
    another code to execute;
}
```

مثال درس قبلی را کاملتر می کنیم، می خواهیم اگر مقدار number از 10 کمتر باشد پیغام Hello World و در غیر اینصورت پیام Goodbye World چاپ شود:

```
<?php

    $number = 15;
    if ($number < 10)
    {
        echo ("Hello World.");
    }
    else
    {
        echo("Goodbye World.");
    }

?>
Goodbye World.
```

همانطور که مشاهده می کنید قسمت else اجرا می شود چون مقدار متغیر number از عدد 10 بیشتر است.

دستورif...else if

این دستور این امکان را فراهم می کند تا در صورت عدم برقراری شرط دستورif ، شرطهای دیگری را نیز بررسی کنیم. ساختار کلی استفاده از این دستور به شرح زیر است:

```
if (condition)
{
    code to execute;
}
else if (another condition)
```

```
{
    another code to execute;
}
```

به مثال درس قبل بر می گردیم. ابتدا مقدار متغیر را چک می کنیم که آیا از مقدار 10 کمتر است، سپس چک می کنیم که آیا از مقدار 10 بزرگتر است و در نهایت اگر هیچکدام از این دو حالت برقرار نبود پیغام مناسب به کاربر نمایش داده شود:

```
<?php

    $number = 15;
    if ($number < 10)
    {
        echo ("Hello World.");
    }
    else if ($number > 10)
    {
        echo("Goodbye World.");
    }
    else
    {
        echo("Number is Equal 10");
    }

?>
Goodbye World.
```

در اینجا مثال های بسیار ساده ای زده شد ولی برخی اوقات شما نیاز دارید تا موارد پیچیده تری را محاسبه کنید و از عملگرها بهره بگیرید.

عملگر سه تایی

عملگر شرطی (:?) در PHP مانند دستور شرطی if…else عمل می کند. در زیر نحوه استفاده از این عملگر آمده است:

```
<condition> ? <result if true> : <result if false>
```

عملگر شرطی در PHP نیاز به سه عملوند دارد:

شرط.

یک مقدار زمانی که شرط درست باشد.

یک مقدار زمانی که شرط نادرست باشد.

اجازه بدهید که نحوه استفاده این عملگر را در داخل برنامه مورد بررسی قرار دهیم.

```
<?php
```

```
    $number = 10;
    $Result = ($number == 10) ? "This is Ten Number" : "Error";
    echo $Result;
?>
```

برنامه بالا نحوه استفاده از این عملگر شرطی را نشان می دهد. خطی که به صورت پررنگ نمایش داده شده است به این صورت ترجمه می شود که : اگر مقدار متغیر number با عدد 10 برابر بود پیغام This is Ten Number و در غیر اینصورت پیغام Error چاپ شود.

دستورSwitch

در PHP ساختاری به نام switch وجود دارد که به شما اجازه می دهد که با توجه به مقدار ثابت یک متغیر چندین انتخاب داشته باشید. دستورswitch معادل دستور if تو در تو است با این تفاوت که در دستور switch متغیر فقط مقادیر ثابتی از اعداد ، رشته ها و یا کاراکترها را قبول می کند. مقادیر ثابت مقادیری هستند که قابل تغیر نیستند. در زیر نحوه استفاده از دستور switch آمده است:

```
switch (testVar)
{
   case compareVal1:
       code to execute if testVar == compareVa11;
       break;
   case compareVa12:
       code to execute if testVar == compareVa12;
       break;
   .
   .
   .
   case compareVa1N:
       code to execute if testVer == compareVa1N;
       break;
   default:
       code to execute if none of the values above match the testVar;
       break;
}
```

ابتدا یک مقدار در متغیر switch که در مثال بالا testVar است قرار می دهید.این مقدار با هر یک از عبارتهای case داخل بلوک switch مقایسه می شود. اگر مقدار متغیر با هر یک از مقادیر موجود در دستورات case برابر بود کد مربوط به آن case اجرا خواهد شد. به این نکته توجه کنید که حتی اگر تعداد خط کدهای داخل دستور case از یکی بیشتر باشد نباید از آکولاد استفاده کنیم. آخر هر دستور case با کلمه کلیدی break تشخیص داده می شود که باعث می شود برنامه از دستور switch خارج شده ودستورات بعد از آن اجرا شوند. اگر این کلمه کلیدی از قلم بیوفتد برنامه با خطا مواجه می شود. دستور switch یک بخش default دارد. این دستور در صورتی اجرا می شود که مقدار متغیر با هیچ یک از مقادیر دستورات case برابر نباشد.

دستور default اختیاری است و اگر از بدنه switch حذف شود هیچ اتفاقی نمی افتد. مکان این دستور هم مهم نیست اما بر طبق تعریف آن را در پایان دستورات می نویسند. به مثالی در مورد دستور switch توجه کنید:

```php
<?php

    $number = 2;
    switch ($number)
    {
        case 1:
            echo 'One';
            break;
        case 2:
            echo 'TWo';
            break;
        case 3:
            echo 'Tree';
            break;
        default:
            break;
    }

?>
```

```
TWo
```

همانطور که در کد بالا مشاهده می کنید مقداری که به متغیر number اختصاص داده اید با مقادیر case مقایسه می شود و با هر کدام از آن مقادیر که برابر بود پیغام مناسب نمایش داده خواهد شد. اگر هم با هیچ کدام از مقادیر case ها برابر نبود دستور default اجرا می شود. یکی دیگر از ویژگیهای دستور switchاین است که شما می توانید از دو یا چند case برای نشان داده یک مجموعه کد استفاده کنید. در مثال زیر اگر مقدار number ، 1، 2 یا 3 باشد یک کد اجرا می شود. توجه کنید که case ها باید پشت سر هم نوشته شوند.

```php
<?php

$number = 2;
switch($number)
{
    case 1:
    case 2:
    case 3:
       echo "This code is shared by three values." ;
       break;
}

?>
```

```
This code is shared by three values.
```

دستورات تکرار

ساختارهای تکرار به شما اجازه می دهند که یک یا چند دستور کد را تا زمانی که یک شرط برقرار است تکرار کنید.بدون ساختارهای تکرار شما مجبورید همان تعداد کدها را بنویسید که بسیار خسته کننده است.مثلا شما مجبورید 10 بار جمله "Hello World." را تایپ کنید مانند مثال زیر :

```
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World." . '<br/>';
echo "Hello World.";
```

```
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
Hello World.
```

البته شما می توانید با کپی کردن، این تعداد کد را راحت بنویسید ولی این کار در کل کیفیت کدنویسی را پایین می آورد.راه بهتر برای نوشتن کدهای بالا استفاده از حلقه ها است.

ساختارهای تکرار در PHP عبارتند از :

- while
- do while
- for
- foreach

دستورWhile

ابتدایی ترین ساختار تکرار در PHP حلقه While است.ابتدا یک شرط را مورد بررسی قرار می دهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا می شوند. ساختار حلقه While به صورت زیر است :

```
while(condition)
```

```
{
    code to loop;
}
```

می بینید که ساختار While مانند ساختار if بسیار ساده است. ابتدا یک شرط را که نتیجه آن یک مقدار بولی است مینویسیم اگر نتیجه درست یا trueباشد سپس کدهای داخل بلوک While اجرا می شوند. اگر شرط غلط یا false باشد وقتی که برنامه به حلقه While برسد هیچکدام از کدها را اجرا نمی کند. برای متوقف شدن حلقه باید مقادیر داخل حلقه While اصلاح شوند.

به یک متغیر شمارنده در داخل بدنه حلقه نیاز داریم. این شمارنده برای آزمایش شرط مورد استفاده قرار می گیرد و ادامه یا توقف حلقه به نوعی به آن وابسته است. این شمارنده را در داخل بدنه باید کاهش یا افزایش دهیم. در برنامه زیر نحوه استفاده از حلقه While آمده است:

```
 1:<?php
 2:
 3:    $number = 1;
 4:    while ($number <= 10)
 5:    {
 6:        echo 'Hello World!'.'<br/>';
 7:        $number++ ;
 8:    }
 9:
10:?>
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

برنامه بالا 10 بار پیغام Hello World! را چاپ می کند. اگر از حلقه در مثال بالا استفاده نمی کردیم مجبور بودیم تمام 10 خط را تایپ کنیم. اجازه دهید که نگاهی به کدهای برنامه فوق بیندازیم. ابتدا در خط 3 یک متغیر تعریف و از آن به عنوان شمارنده حلقه استفاده شده است.سپس به آن مقدار 1 را اختصاص می دهیم چون اگر مقدار نداشته باشد نمی توان در شرط از آن استفاده کرد.

در خط 4 حلقه While را وارد می کنیم. در حلقه While ابتدا مقدار اولیه شمارنده با 10 مقایسه می شود که آیا از 10 کمتر است یا با آن برابر است. نتیجه هر بار مقایسه ورود به بدنه حلقه While و چاپ پیغام است. همانطور که مشاهده می کنید بعد از هر بار مقایسه مقدار شمارنده یک واحد اضافه می شود (خط 7). حلقه تا زمانی تکرار می شود که مقدار شمارنده از 10 کمتر باشد.

اگر مقدار شمارنده یک بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز false نشود یک حلقه بینهایت به وجود می آید. به این نکته توجه کنید که در شرط بالا به جای علامت <از => استفاده شده است. اگر از علامت <استفاده می کردیم کد ما 9 بار تکرار می شد چون مقدار اولیه 1 است و هنگامی که شرط به 10 برسد false می شود چون 10 10 <نیست. اگر می خواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (true) باشد.

```
while(true)
{
   //code to loop
}
```

این تکنیک در برخی موارد کارایی دارد و آن زمانی است که شما بخواهید با استفاده از دستورات **break** و **return** که در آینده توضیح خواهیم داد از حلقه خارج شوید.

حلقه do while

حلقه **do while** یکی دیگر از ساختارهای تکرار است. این حلقه بسیار شبیه حلقه **while** است با این تفاوت که در این حلقه ابتدا کد اجرا می شودو سپس شرط مورد بررسی قرار می گیرد. ساختار حلقه **do while** به صورت زیر است :

```
do
{
   code to repeat;

} while (condition);
```

همانطور که مشاهده می کنید شرط در آخر ساختار قرار دارد. این بدین معنی است که کدهای داخل بدنه حداقل یکبار اجرا می شوند.برخلاف حلقه**while** که اگر شرط نادرست باشد دستورات داخل بدنه اجرا نمی شوند. به مثال زیر توجه کنید :

```
 1:<?php
 2:
 3:    $number = 1;
 4:    do
 5:    {
 6:        echo 'Hello World!' . '<br/>';
 7:        $number ++;
 8:    }
 9:    while ($number > 10);
10:
11:?>
Hello World!
```

همانطور که در کد بالا مشاهده می کنید ابتدا در خط **6** یک بار رشته **Hello World** چاپ می شود و سپس یک واحد به متغیر **number** در خط **7** اضافه می شود. در این خط مقدار متغیر **number** عدد **2** است و سپس عدد **2** در خط **9** با عدد **10** مقایسه می شود و چون از عدد **10** بزرگتر نیست شرط نادرست و حلقه متوقف می شود.در نتیجه حداقل یکبار حلقه اجرا می شود. اما همین مثال در صورتیکه شرط درست باشد نتیجه ای به صورت زیر خواهد داشت :

```
 1:<?php
 2:
 3:    $number = 1;
 4:    do
 5:    {
```

```
    6:         echo 'Hello World!' . '<br/>';
    7:         $number ++;
    8:     }
    9:     while ($number < 10);
   10:
   11:?>
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

حلقهfor

یکی دیگر از ساختارهای تکرار حلقه **for** است.این حلقه عملی شبیه به حلقه **while** انجام می دهد و فقط دارای چند خصوصیت اضافی است. ساختار حلقه **for** به صورت زیر است :

```
for(initialization; condition; operation)
{
   code to repeat;
}
```

مقدار دهی اولیه **(initialization)** اولین مقداری است که به شمارنده حلقه می دهیم.شمارنده فقط در داخل حلقه **for** قابل دسترسی است.

شرط **(condition)** در اینجا مقدار شمارنده را با یک مقدار دیگر مقایسه می کند و تعیین می کند که حلقه ادامه یابد یا نه.

عملگر **(operation)** که مقدار اولیه متغیر را کاهش یا افزایش می دهد.

در زیر یک مثال از حلقه **for** آمده است:

```
<?php

    for($i = 1; $i <= 10; $i++)
    {
        echo 'Hello World!' . '<br/>';
    }

?>
Hello World!
Hello World!
Hello World!
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

در برنامه بالا، ابتدا یک متغیر به عنوان شمارنده تعریف می کنیم و آن را با مقدار 1 مقدار دهی اولیه می کنیم. سپس با استفاده از شرط آن را با مقدار 10 مقایسه می کنیم که آیا کمتر است یا مساوی؟

توجه کنید که قسمت سوم حلقه (i++) فورا اجرا نمی شود. کد اجرا می شود و ابتدا رشته Hello World! را چاپ می کند. آنگاه یک واحد به مقدار i اضافه شده و مقدار i برابر 2 می شود و بار دیگر i با عدد 10 مقایسه می شود و این حلقه تا زمانی که مقدار شرط true شود ادامه می یابد. اگر بخواهید معکوس بازه ای از اعداد را پیاده سازی کنید یعنی اعداد از بزرگ به کوچک چاپ شوند باید به صورت زیر عمل کنید :

```php
for ($i = 10; $i > 0; $i--)
{
   echo $i . '<br/>';
}
```
```
10
9
8
7
6
5
4
3
2
1
```

کد بالا اعداد را از 10 به 1 چاپ می کند (از بزرگ به کوچک). مقدار اولیه شمارنده را 10 می دهیم و با استفاده از عملگر کاهش (--) برنامه ای که شمارش معکوس را انجام می دهد ایجاد می کنیم.

حلقه foreach

حلقهforeach یکی دیگر از ساختارهای تکرار در PHP می باشد که مخصوصا برای آرایه ها و مجموعه ها طراحی شده است.حلقه foreach با هر بار گردش در بین اجزاء ، مقادیر هر یک از آنها را در داخل یک متغیر موقتی قرار می دهد و شما می توانید بواسطه این متغیر به مقادیر دسترسی پیدا کنید. در زیر نحوه استفاده از حلقه foreach آمده است:

```php
foreach (array as temporaryVar)
{
```

```
    code to execute;
}
```

arrayنام آرایه است.سپس کلمه کلیدی as و بعد از آن(temporaryVar این نام کاملا اختیاری است) متغیری که مقادیر اجزای آرایه را در خود نگهداری می کند. در زیر نحوه استفاده از حلقه foreach امده است:

```
<?php

$number = array(1, 2, 3, 4, 5);
foreach ($number as $tempNumber)
{
echo $tempNumber . '<br/>';
}

?>
1
2
3
4
5
```

در برنامه آرایه ای با 5 جزء تعریف شده و مقادیر 1 تا 5 در آنها قرار داده شده است (خط 3). در خط 9 حلقه foreach شروع می شود.ما یک متغیر موقتی تعریف کرده ایم که اعداد آرایه را در خود ذخیره می کند. در هر بار تکرار از حلقه foreach متغیر موقتی tempNumber ، مقادیر عددی را از آرایه استخراج می کند. حلقه foreach مقادیر اولین تا آخرین جزء آرایه را در اختیار ما قرار می دهد.

حلقه foreach برای دریافت هر یک از مقادیر آرایه کاربرد دارد.بعد از گرفتن مقدار یکی از اجزای آرایه ،مقدار متغیر موقتی را چاپ می کنیم. روش دیگر استفاده از حلقه foreach که معمولا برای آرایه های انجمنی و عددی به کار می رود به صورت زیر می باشد:

```
foreach (array as $key => $value)
{
   statement
}
```

برای درک بهتر روش بالا به مثال زیر توجه کنید:

```
<?php

   $number = array('one' => 'a', 'two' => 'b', 'three' => 'c');
    foreach ($number as $key => $value)
    {
        echo $key. ' => '. $value . '<br/>';
    }

?>
one => a
two => b
three => c
```

همانطور که در کد بالا مشاهده می کنید یک آرایه در خط 3 ایجاد کرده ایم که شامل کلید و مقدار است. برای به دست آوردن کلیدها از حالت دوم حلقه foreach استفاده کرده ایم. البته یاد آور می شویم که این روش زمانی مفید است که ما به اندیس ها و یا کلیدها در آرایه نیاز داشته باشیم مثلا در مورد آرایه مثال اول هم می توانیم اندیس اعداد را به این روش به دست بیاوریم:

```
 1:<?php
 2:
 $number = array(1, 2, 3, 4, 5);      3:
 4:    foreach ($number as $key => $value)
 5:    {
 6:        echo $key. ' => '. $value . '<br/>';
 7:    }
 8:
 9:?>
0 => 1
1 => 2
2 => 3
3 => 4
4 => 5
```

خارج شدن از حلقه با استفاده از break وcontinue

گاهی اوقات با وجود درست بودن شرط می خواهیم حلقه متوقف شود. سوال اینجاست که چطور این کار را انجام دهید؟ با استفاده از کلمه کلیدی break حلقه را متوقف کرده و با استفاده از کلمه کلیدی continue می توان بخشی از حلقه را رد کرد و به مرحله بعد رفت. برنامه زیر نحوه استفاده از continue و breakرا نشان می دهد:

```
 1:<?php
 2:
 3:    echo 'Demonstrating the use of break.'.'<br/>';
 4:
 5:    for ($x = 1; $x < 10; $x++)
 6:    {
 7:       if ($x == 5)
 8:              break;
 9:
10:       echo $x . '<br/>';
11:    }
12:
13:    echo '<br/>'. 'Demonstrating the use of continue.'.'<br/>';
14:
15:    for ($x = 1; $x < 10; $x++)
16:    {
17:       if ($x == 5)
18:              continue;
19:
20:       echo $x . '<br/>';
```

```
 21:     }
 22:
 23:?>
Demonstrating the use of break.
1
2
3
4

Demonstrating the use of continue.
1
2
3
4
6
7
8
9
```

در این برنامه از حلقه **for** برای نشان دادن کاربرد دو کلمه کلیدی فوق استفاده شده است اگر به جای **for** از حلقه های **while** و **do…while** استفاده می شد نتیجه یکسانی به دست می امد. همانطور که در شرط برنامه (خط 7) آمده است وقتی که مقدار **x** به عدد **5** رسید سپس دستور **break** اجرا شود(خط 8). حلقه بلافاصله متوقف می شود حتی اگر شرط **x < 10** برقرار باشد.از طرف دیگر در خط **17** حلقه **for** فقط برای یک تکرار خاص متوقف شده و سپس ادامه می یابد.(وقتی مقدار **x** برابر **5** شود حلقه از **5** رد شده و مقدار **5** را چاپ نمی کند و بقیه مقادیر چاپ می شوند .

تابع

توابع به شما اجازه می دهند که یک رفتار یا وظیفه را تعریف کنید و مجموعه ای از کدها هستند که در هر جای برنامه می توان از آنها استفاده کرد. توابع در **PHP** و اکثر زبانهای برنامه نویسی بر دو نوعند :

- توابع از پیش تعریف شده
- توابعی که توسط کاربر تعریف می شوند .

ساده ترین ساختار یک تابع به صورت زیر است :

```
function MethodName()
{
 code to execute;
}
```

به برنامه ساده زیر توجه کنید.در این برنامه از یک تابع برای چاپ یک پیغام در صفحه نمایش استفاده شده است:

```
 1:<?php
 2:
 3:    function PrintMessage()
 4:    {
 5:       echo 'Hello World!';
 6:    }
 7:
 8:    PrintMessage ();
 9:
10:?>
```

در خطوط 3-6 یک تابع تعریف کرده ایم. همانطور که مشاهده می کنید در خط 3 و برای تعریف تابع از کلمه کلیدی function سپس نام تابع و بعد از آن پرانتز باز و بسته استفاده کرده ایم. نام تابع ما PrintMessage() است. به این نکته توجه کنید که در نامگذاری تابع از روش پاسکال (حرف اول هر کلمه بزرگ نوشته می شود) استفاده کرده ایم. این روش نامگذاری قراردادی است و می توان از این روش استفاده نکرد، اما پیشنهاد می شود که از این روش برای تشخیص توابع استفاده کنید. بهتر است در نامگذاری توابع از کلماتی استفاده شود که کار ان تابع را مشخص می کند مثلا نام هایی مانند GoToBed یا OpenDoor .

همچنین به عنوان مثال اگر مقدار برگشتی (در درس های آینده توضیح می دهیم) تابع یک مقدار بولی باشد می توانید اسم تابع خود را به صورت یک کلمه سوالی انتخاب کنید مانند IsLeapyear یا IsTeenager... ولی از گذاشتن علامت سوال در آخر اسم تابع خودداری کنید. دو پرانتزی که بعد از نام می آید نشان دهنده آن است که نام متعلق به یک تابع است. بعد از پرانتزها دو آکولاد قرار می دهیم که بدنه تابع را تشکیل می دهد و کدهایی را که می خواهیم اجرا شوند را در داخل این آکولاد ها می نویسیم. در خط 8 تابع را صدا می زنیم. برای صدا زدن یک تابع کافیست نام آن را نوشته و بعد از نام پرانتزها را قرار دهیم. برای اجرای تابع PrintMessage() برنامه از خط به محل تعریف تابع PrintMessage() می رود. مثلا وقتی ما تابع PrintMessage() را در خط 8 صدا می زنیم برنامه از خط 8 به خط 3 ، یعنی جایی که تابع تعریف شده می رود و کدهای آن را اجرا می کند.

مقدار برگشتی از یک تابع

توابع می توانند مقدار برگشتی از هر نوع داده ای داشته باشند. این مقادیر می توانند در محاسبات یا به دست آوردن یک داده مورد استفاده قرار بگیرند. در زندگی روزمره فرض کنید که کارمند شما یک تابع است و شما او را صدا می زنید و از او می خواهید که کار یک سند را به پایان برساند. سپس از او می خواهید که بعد از اتمام کارش سند را به شما تحویل دهد. سند همان مقدار برگشتی تابع است. نکته مهم در مورد یک تابع، مقدار برگشتی و نحوه استفاده شما از آن است. برگشت یک مقدار از یک تابع آسان است. کافیست در تعریف تابع به روش زیر عمل کنید:

```
function MethodName()
{
    return value;
}
```

همانطور که در خط مشاهده می کنید مقدار بازگشتی از تابع را جلوی دستور `return` می نویسیم. مثال زیر یک تابع که دارای مقدار برگشتی است را نشان می دهد:

```php
<?php

    function CalculateSum()
    {
        $firstNumber = 10;
        $secondNumber = 5 ;
        $sum = $firstNumber + $secondNumber ;

        return $sum;
    }

    $result = CalculateSum();
    echo $result;

?>
```

```
15
```

در خطوط **5** و **6** مثال فوق، دو متغیر تعریف و مقدار دهی شده اند. توجه کنید که این متغیرها، متغیرهای محلی هستند. و این بدان معنی است که این متغیرها در سایر تابعها قابل دسترسی نیستند و فقط در تابعی که در آن تعریف شده اند قابل استفاده هستند. در خط **7** جمع دو متغیر در متغیر `sum` قرار می گیرد. در خط **9** مقدار برگشتی `sum` توسط دستور `return` فراخوانی می شود. در خط **12** یک متغیر به نام `result` تعریف می کنیم و تابع `CalculateSum()`را فراخوانی می کنیم.

تابع `CalculateSum()` مقدار **15** را بر می گرداند که در داخل متغیر `result` ذخیره می شود. در خط **13** مقدار ذخیره شده در متغیر `result` چاپ می شود. تابعی که در این مثال ذکر شد تابع کاربردی و مفیدی نیست،با وجودیکه کدهای زیادی در تابع بالا نوشته شده ولی همیشه مقدار برگشتی **15** است، در حالیکه می توانستیم به راحتی یک متغیر تعریف کرده و مقدار **15** را به آن اختصاص دهیم. این تابع در صورتی کارامد است که پارامترهایی به آن اضافه شود که در درسهای آینده توضیح خواهیم داد. هنگامی که می خواهیم در داخل یک تابع از دستور `if` یا `switch` استفاده کنیم باید تمام کدها دارای مقدار برگشتی باشند. برای درک بهتر این مطلب به مثال زیر توجه کنید:

```php
<?php

    function GetNumber()
    {
        $number = 11 ;

        if ($number > 10)
        {
            return $number;
        }
        else
        {
            return 0;
        }
    }

    $result = GetNumber();
    echo $result;

?>
```

```
11
```

در خطوط 3-16 یک تابع با نام GetNumber() تعریف شده است. در خط 5 متغیری با مقدار 11 مقداردهی شده است که در خط خط 7 با مقدار 10 مقایسه می شود و چون مقدار این متغیر از 10 بیشتر است پس دستور return اول مقدار 11 را برمی گرداند. حال اگر مقدار این متغیر از 10 کمتر باشد دستور returnمربوط به قسمت else اجرا و مقدار صفر چاپ می شود. که از کاربر یک عدد بزرگتر از 10 را می خواهد. اگر قسمت else دستور if و یا دستور returnرا از آن حذف کنیم در هنگام اجرای برنامه نتیجه چاپ نمی شود. چون اگر شرط دستور if نادرست باشد برنامه به قسمت else می رود تا مقدار صفر را بر گرداند و چون قسمت else حذف شده است برنامه هیچ مقداری را چاپ نمی کند و همچنین اگر دستور return حذف شود چون برنامه نیاز به مقدار برگشتی دارد برنامه هیچ مقداری را چاپ نمی کند.و آخرین مطلبی که در این درس می خواهیم به شما آموزش دهیم این است که شما می توانید از یک تابع که مقدار برگشتی ندارد خارج شوید. استفاده از return باعث خروج از بدنه تابع و اجرای کدهای بعد از آن می شود.

```php
<?php

    function TestReturnExit()
    {
        echo 'Line 1 inside the method TestReturnExit()';

        return;

        echo 'Line 2 inside the method TestReturnExit()';
    }

    TestReturnExit();

?>
```

```
Line 1 inside the method TestReturnExit()
```

در برنامه بالا نحوه خروج از تابع با استفاده از کلمه کلیدی return و نادیده گرفتن همه کدهای بعد از این کلمه کلیدی نشان داده شده است. در پایان برنامه تابع تعریف شده (TestReturnExit()) فراخوانی و اجرا می شود.

پارامترها و آرگومان ها

پارامترها داده های خامی هستند که متد آنها را پردازش می کند و سپس اطلاعاتی را که به دنبال آن هستید در اختیار شما قرار می دهد. فرض کنید پارامترها مانند اطلاعاتی هستند که شما به یک کارمند می دهید که بر طبق آنها کارش را به پایان برساند. یک متد می تواند هر تعداد پارامتر داشته باشد. هر پارامتر می تواند از انواع مختلف داده باشد. در زیر یک متد با N پارامتر نشان داده شده است:

```php
function MethodName(param1,param2, ...paramN)
{
  code to execute;
}
```

پارامترها بعد از نام متد و بین پرانتزها قرار می گیرند. بر اساس کاری که متد انجام می دهد می توان تعداد پارامترهای زیادی به متد اضافه کرد. بعد از فراخوانی یک متد باید آرگومانهای آن را نیز تامین کنید. آرگومانها مقادیری هستند که به پارامترها اختصاص داده می شوند. ترتیب ارسال آرگومانها به پارامترها مهم است. عدم رعایت ترتیب در ارسال آرگومانها باعث به وجود آمدن خطای منطقی و خطای زمان اجرا می شود. اجازه بدهید که یک مثال بزنیم :

```php
<?php

    function CalculateSum($number1,$number2)
    {
        return $number1 + $number2;
    }

    $result = CalculateSum(10,5);
    echo $result;

?>
```

```
15
```

در برنامه بالا یک متد به نام(`CalculateSum()` خطوط 3-6) تعریف شده است که وظیفه آن جمع مقدار دو عدد است. متد دارای دو پارامتر است که اعداد را به آنها ارسال می کنیم. در بدنه متد دستور`return` نتیجه جمع دو عدد را بر می گرداند. در خط 8 دو عدد 5 و 10 را به عنوان آرگومان به متد ارسال می کنیم. بعد از ارسال مقادیر 5 و 10 به پارامترها ، پارامترها آنها را دریافت می کنند. به این نکته نیز توجه کنید که نام پارامترها طبق قرارداد به شیوه کوهان شتری یا `camelCasing` (حرف اول دومین کلمه بزرگ نوشته می شود) نوشته می شود. در داخل بدنه متد (خط 5) دو مقدار با هم جمع می شوند و در خط 9 نتیجه چاپ می شود .

دانستن مبانی مقادیر برگشتی و ارسال آرگومانها باعث می شود که شما متدهای کارامد تری تعریف کنید. تکه کد زیر نشان می دهد که شما حتی می توانید مقدار برگشتی از یک متد را به عنوان آرگومان به متد دیگر ارسال کنید .

```php
<?php

    function MyMethod()
    {
        return 5;
    }

    function AnotherMethod($number)
    {
        echo $number;
    }

    AnotherMethod(MyMethod());

?>
```

```
5
```

چون مقدار برگشتی متد `MyMethod()` عدد 5 است و به عنوان آرگومان به متد `AnotherMethod()` ارسال می شود خروجی کد بالا هم عدد 5 است .

پارامترهای اختیاری

پارامترهای اختیاری همانگونه که از اسمشان پیداست اختیاری هستند و می توان به آنها آرگومان ارسال کرد یا نه. این پارامترها دارای مقادیر پیشفرضی هستند.اگر به اینگونه پارامترها آرگومانی ارسال نشود از مقادیر پیشفرض استفاده می کنند. به مثال زیر توجه کنید :

```
 1:<?php
 2:
 3:    function PrintMessage($String = "Welcome to PHP Tutorials!")
 4:    {
 5:        echo $String . '<br/>';
 6:    }
 7:
 8:    PrintMessage();
 9:    PrintMessage("Learn PHP Today!");
10:
11:?>
Welcome to PHP Tutorials!
Learn PHP Today!
```

متد PrintMessage() خطوط 3-6 یک پارامتر اختیاری دارد. برای تعریف یک پارامتر اختیاری می توان به آسانی و با استفاده از علامت = یک مقدار را به یک پارامتر اختصاص داد (مثال بالا خط 3). دو بار متد را فراخوانی می کنیم. در اولین فراخوانی (خط 8) ما آرگومانی به متد ارسال نمی کنیم بنابراین متد از مقدار پیشفرض (Welcome to PHP Tutorials!) استفاده می کند. در دومین فراخوانی(خط 9) یک پیغام (آرگومان) به متد ارسال می کنیم که جایگزین مقدار پیشفرض پارامتر می شود.

محدوده متغیر

متغیرها در PHP دارای محدوده هستند. محدوده یک متغیر به شما می گوید که در کجای برنامه می توان از متغیر استفاده کرد و یا متغیر قابل دسترسی است. به عنوان مثال متغیری که در داخل یک متد تعریف می شود فقط در داخل بدنه متد قابل دسترسی است. می توان دو متغیر با نام یکسان در دو متد مختلف تعریف کرد. برنامه زیر این ادعا را اثبات می کند :

```
 1:<?php
 2:
 3:    function firstLocalVariable()
 4:    {
 5:        $number = 10;
 6:        echo $number;
 7:    }
 8:
 9:    function secondLocalVariable()
10:    {
11:        $number = 5;
12:        echo $number;
13:    }
14:
```

```
 15:     firstLocalVariable ();
 16:     echo '<br/>';
 17:     secondLocalVariable ();
 18:
 19:?>
10
5
```

مشاهده می کنید که حتی اگر ما دو متغیر با نام یکسان تعریف کنیم (خطوط **5** و **11**) که دارای محدوده های متفاوتی هستند، می توان به هر کدام از آنها مقادیر مختلفی اختصاص داد. متغیر تعریف شده در داخل متد **firstLocalVariable()** هیچ ارتباطی به متغیر داخل متد **secondLocalVariable()** ندارد. وقتی به مبحث کلاسها رسیدیم در این باره بیشتر توضیح خواهیم داد. **php** دارای چهار محدوده است:

- متغیرهای محلی **(Local)**
- متغیرهای سراسری **(Global)**
- متغیرهای ایستا **(Static)**
- پارامتر **(Parameter)**

متغیرهای محلی

متغیرهایی که داخل توابع تعریف می شوند محلی هستند و فقط داخل همان تابع قابل استفاده اند. به مثال زیر توجه کنید:

```
  1:<?php
  2:
  3:     function LocalVariable()
  4:     {
  5:         $number = 10;
  6:         echo $number;
  7:     }
  8:
  9:     LocalVariable ();
 10:     echo $number;
 11:
 12:?>
10
Notice: Undefined variable: number in C:\wamp\www\test.php on line 10
```

همانطور که مشاهده می کنید با فراخوانی متد در خط **9** مقدار متغیر **number** چاپ می شود ولی در خط **10** که سعی در چاپ مقدار این متغیر داریم با پیغام خطا مواجه می شویم چون طول عمر این متغیر تا زمانی است که تابع به پایان نرسیده است. با پایان تابع متغیر و مقدار آن هم از بین می رود در نتیجه در خارج از تابع نمی توان مقدار آن را چاپ کرد.

متغیرهای سراسری

متغیرهایی که در بیرون تابع تعریف می شوند از نوع سراسری هستند. به مثال زیر توجه کنید:

```php
<?php

    $firstNumber  = 10;
    $secondNumber = 5;
    $Sum;

    function GlobalVariable()
    {
        global $firstNumber,$secondNumber,$Sum;
        $Sum = $firstNumber + $secondNumber;
    }

    GlobalVariable ();
    echo $Sum;

?>
```

```
15
```

متغیرهای **firstNumber** و **secondNember** و **Sum** در بیرون تابع تعریف شده اند و از نوع سراسری هستند، در داخل تابع اگر بخواهیم به مقدار آنها دسترسی پیدا کنیم باید ابتدا با کلمه کلیدی **global** در تابع تعریف کنیم سپس از آن استفاده نماییم.به این نکته نیز توجه کنید که در **php** متغیرهای سراسری در آرایه ای با نام **GLOBALS** ذخیره می شوند که به این صورت نیز می توانیم به مقادیر آنها دسترسی داشته باشیم:

```php
<?php

    $firstNumber  = 10;
    $secondNember = 5;
    $Sum;

    function GlobalVariable()
    {
        $GLOBALS['Sum'] = $GLOBALS['firstNumber'] + $GLOBALS['secondNember'];
    }

    GlobalVariable ();
    echo $Sum;

?>
```

```
15
```

متغیرهای ایستا

با اتمام اجرای تابع تمام متغیرها و آن تابع از بین می شوند، به غیر از متغیرهایی که بصورت **static** تعریف شده باشند، به مثال زیر توجه کنید:

```php
<?php

    function StaticVariable()
```

```
    {
        static $firstNumber  = 10;
        echo $firstNumber . '<br/>';
        $firstNumber ++;
    }

    StaticVariable();
    StaticVariable();
    StaticVariable();

?>
```

```
10
11
12
```

همانطور که در کد بالا مشاهده می کنید در خطوط 3-8 یک متد و در داخل آن یک متغیر از نوع ایستا (static) تعریف شده است (خط 5). در خطوط 10-12 سه بار متد را فراخوانی کرده ایم. در فراخوانی اول مقدار 10 چاپ می شود. در خط 7 یک واحد به این متغیر اضافه می شود و این مقدار در فراخوانی دوم چاپ می شود (مقدار 11). در فراخوانی سوم هم یک واحد به مقدار قبلی اضافه شده (11+1) و این مقدار یعنی 12 چاپ می شود.

پارامترها

پارامترها متغیرهایی هستند که مقادیر ارسالی به تابع را موقع فراخوانی مشخص میکنند و مقادیر ارسالی داخل آنها قرار میگیرد، با انتهای استفاده از تابع نیز مقادیر آنها از بین خواهند رفت.

```
<?php

    function ParameterVariable($firstNumber)
    {
        echo $firstNumber . '<br/>';
    }

    ParameterVariable(5);

    echo $firstNumber;

?>
```

```
5
```

در رابطه با کد بالا همانطور که مشاهده می کنید خروجی اجرای آن عدد 5 است و این خروجی مربوط به فراخوانی متد در خط 8 است نه چاپ متغیر number در خط 10. همانطور که مشاهده می کنید نمی توان مقدار متغیر number را در خارج از تابع چاپ کرد، چون با اتمام تابع مقدار آن از بین می رود.

معرفی چند تابع مفید

در php سه تابع (در مورد توابع در درس های آینده توضیح می دهیم) از پیش تعریف شده برای به دست آوردن اطلاعاتی درباره انواع داده ها وجود دارد. این توابع عبارتند از :

- print_r برای مشاهده مقدار یک داده به کار می رود .
- var_dump علاوه بر مقدار، نوع و اندازه داده را نیز نمایش می دهد .
- var_export مانند تابع print_r عمل می کن، با این تفاوت که میتوان از خروجی ایجاد شده توسط آن در کدهای php استفاده کرد .

به مثال زیر توجه کنید :

```php
<?php

    $number = array(20,'Jack');

    print_r($number);

    var_dump($number);

    var_export($number);

?>
```

```
Array ( [0] => 20 [1] => Jack )

array (size=2)
0 => int 20
1 => string 'Jack' (length=4)

array ( 0 => 20, 1 => 'Jack', )
```

توصیه می شود که از این توابع برای خطایابی برنامه استفاده کنید .

برنامه نویسی شیء گرا

برنامه نویسی شی گرا (OOP) شامل تعریف کلاسها و ساخت اشیاء مانند ساخت اشیاء در دنیای واقعی است.برای مثال یک ماشین را در نظر بگیرید.این ماشین دارای خواصی مانند رنگ ، سرعت ، مدل ، سازنده و برخی خواص دیگر است. همچنین دارای رفتارها و حرکاتی مانند شتاب و پیچش به چپ و راست و ترمز است.اشیاء در PHP تقلیدی از یک شی مانند ماشین در دنیای واقعی هستند.برنامه نویسی شی گرا با استفاده از کدهای دسته بندی شده کلاسها و اشیاء را بیشتر قابل کنترل می کند . در ابتدا ما نیاز به تعریف یک کلاس برای ایجاد اشیاء مان داریم.شیء در برنامه نویسی شی گراء از روی کلاسی که شما تعریف کرده اید ایجاد می شود.برای مثال نقشه ساختمان شما یک کلاس است که ساختمان از روی آن ساخته شده است.کلاس شامل خواص یک ساختمان مانند مساحت،بلندی و مواد مورد استفاده در ساخت خانه می باشد. در دنیای واقعی ساختمان ها نیز بر اساس یک نقشه (کلاس) پایه گذاری(تعریف)شده اند . برنامه نویسی شیء گرا یک روش جدید در برنامه نویسی است که بوسیله برنامه نویسان مورد استفاده قرار می گیرد و به آنها کمک می کند که برنامه هایی با قابلیت استفاده مجدد، خوانا و راحت طراحی کنند PHP.نیز یک برنامه شی گراست. در درس بعد به شما نحوه تعریف کلاس و استفاده از اشیاء آموزش داده خواهد شد.همچنین شما با مفهوم وراثت که از مباحث مهم در برنامه نویسی شیء گرا است در آینده آشنا می شوید .

کلاس

کلاس به شما اجازه می دهد یک نوع داده ای که توسط کاربر تعریف می شود و شامل متغیرها و خواص (properties) و متدها است را ایجاد کنید. کلاس در حکم یک نقشه برای یک شی می باشد. شی یک چیز واقعی است که از ساختار، خواص و یا رفتارهای کلاس پیروی می کند. وقتی یک شی می سازید یعنی اینکه یک نمونه از کلاس ساخته اید (در درس ممکن است از کلمات شی و نمونه به جای هم استفاده شود).برای تعریف یک کلاس از کلمه کلیدی class استفاده شود :

```
class ClassName
{
    Variable1;
    Variable2;
    ...
    VariableN;

    method1;
    method2;
    ...
    methodN;
}
```

این کلمه کلیدی را قبل از نامی که برای کلاسمان انتخاب می کنیم می نویسیم. در نامگذاری کلاسها هم از روش نامگذاری Pascal استفاده می کنیم. در بدنه کلاس متغیرها و متدهای آن قرار داده می شوند. متغیرها اعضای داده ای خصوصی هستند که کلاس از آنها برای رفتارها و ذخیره مقادیر خاصیت هایش (property)استفاده می کند. متدها رفتارها یا کارهایی هستند که یک کلاس می تواند انجام دهد. در زیر نحوه تعریف و استفاده از یک کلاس ساده به نام personنشان داده شده است :

```
1:<?php
2:
3:    class Person
```

```php
    {
        public $name;
        public $age;
        public $height;

        public function TellInformation()
        {
           echo 'Name: '  . $this -> name .'<br/>';
           echo 'Age:  '  . $this -> age .'<br/>';
           echo 'Height: '. $this -> height;
        }
    }


    $person1 = new Person();
    $person2 = new Person();

    $person1 -> name = 'Jack';
    $person1 -> age = 21;
    $person1 -> height = 180;
    $person1 -> TellInformation ();

    echo "<br/><br/>";//Separator

    $person2 -> name = 'Mike';
    $person2 -> age = 23;
    $person2 -> height = 158;
    $person2 -> TellInformation ();

?>
```

```
Name: Jack
Age: 21
Height: 160

Name: Mike
Age: 23
Height: 158
```

همانطور که در کد بالا مشاهده می کنید، در خطوط **3-15** کلاسی به نام **Person** تعریف شده است. در خط **3** یک نام به کلاس اختصاص داده ایم تا به وسیله آن قابل دسترسی باشد. در داخل بدنه کلاس متغیرهای آن تعریف شده اند (خطوط **5-7**).همچنین سطح دسترسی آنها را **public** تعریف کرده ایم تا در دیگر کلاسها قابل شناسایی باشند. درباره سطوح دسترسی در یک درس جداگانه بحث خواهیم کرد.

این سه متغیر تعریف شده خصوصیات واقعی یک فرد در دنیای واقعی را در خود ذخیره می کنند. یک فرد در دنیای واقعی دارای نام، سن، و قد می باشد. در خطوط **9-14** یک متد هم در داخل کلاس به نام **TellInformation()** تعریف شده است که رفتار کلاسمان است و مثلا اگر از فرد سوالی بپرسیم در مورد خودش چیزهایی می گوید. در داخل متد کدهایی برای نشان دادن مقادیر موجود در متغیرها نوشته شده است. نکته ای درباره متغیرها وجود دارد و این است که چون متغیرها در داخل کلاس تعریف و به عنوان اعضای کلاس در نظر گرفته شده اند، محدوده آنها یک کلاس است.

این بدین معناست که متغیرها فقط می توانند در داخل کلاس یعنی جایی که به آن تعلق دارند و یا به وسیله نمونه ایجاد شده از کلاس مورد استفاده قرار بگیرند.در خطوط **18** و **19** دو نمونه یا دو شی از کلاس **Person** ایجاد می کنیم. برای ایجاد یک نمونه از یک کلاس باید از کلمه کلیدی **new** و به دنبال آن نام کلاس و یک جفت پرانتز قرار دهیم:

```
18:$person1 = new Person();
19:$person2 = new Person();
```

در خطوط **21-23** مقادیری به متغیرهای اولین شی ایجاد شده از کلاس **person1** اختصاص داده شده است. برای دسترسی به متغیرها یا متدهای یک شی از علامت **->**استفاده می شود. به عنوان مثال کد **$person1 -> name** نشان دهنده متغیر **name** از شی **person1** می باشد. برای چاپ مقادیر متغیرها باید متد **()TellInformation** شی **person1** را فراخوانی می کنیم(خط **24**). به کلمه کلیدی **this** در خطوط **11-13** توجه کنید. این کلمه کلیدی اشاره به شی جاری دارد. یعنی وقتی مقادیر از طریق شی **person1** ارسال می شوند منظور از **this** شی **person1** و وقتی از طریق شی **person2** ارسال می شوند منظور از **this** شی **person2** می باشد. همانطور که در خطوط **11-13**، **21-23** و **28-30** مشاهده می کنید برای دسترسی به متغیرها، قبل از نام آنها علامت $ را به کار نمی بریم.

در خطوط **28-30** نیز مقادیری به شی دومی که قبلا از کلاس ایجاد شده تخصیص می دهیم و سپس متد **()TellInformation** را فراخوانی می کنیم. به این نکته توجه کنید که **person1** و **person2** نسخه های متفاوتی از هر متغیر دارند بنابراین تعیین یک نام برای **person2** هیچ تاثیری بر نام **person1**ندارد. در مورد اعضای کلاس در درسهای آینده توضیح خواهیم داد.

سازنده

سازنده ها متدهای خاصی هستند که وجود آنها برای ساخت اشیا لازم است. آنها به شما اجازه می دهند که متغیرهای کلاس را مقداردهی اولیه کنید و کدهایی که را که می خواهید هنگام ایجاد یک شی اجرا شوند را به برنامه اضافه کنید. اگر از هیچ سازنده ای در کلاس تان استفاده نکنید، **PHP**از سازنده پیشفرض که یک سازنده بدون پارامتر است استفاده می کند:

در مثال زیر یک کلاس که شامل سازنده پیشفرض (خطوط **9-12**) است را مشاهده می کنید:

```
1 <?php
2
3    class Person
4    {
5        public $name;
6        public $age;
7        public $height;
8
9        public function Person()
10       {
11
12       }
13
14       public function TellInformation()
15       {
16           echo 'Name: '   . $this -> name .'<br/>';
17           echo 'Age: '    . $this -> age .'<br/>';
18           echo 'Height: '. $this -> height;
```

```php
        }
    }

    $person1 = new Person();
    var_dump ($person1);

?>
```

```
object(Person)[1]
  public 'name' => null
  public 'age' => null
  public 'height' => null
```

می توانیم این سازنده را هم تعریف نکنیم چون **PHP** به طور خودکار آن را ایجاد می کند.همانطور که در خط **22** و **23** مشاهده می کنید ما یک شی یا یک نمونه از کلاس ایجاد کرده ایم (در درس بعد بیشتر توضیح می دهیم) و با استفاده از تابع **var_dump** مقادیر موجود در این شی را چاپ کرده ایم. در خروجی مشاهده می کنید که سازنده پیشفرض به هر سه متغیر مقدار **null** را اختصاص داده است.مثلا بهتر است که با استفاده از سازنده مقدار پیشفرض به متغیرها اختصاص دهیم. مثلا فردی که به دنیا می آید نام **(name)** ندارد ولی سن **(age)** و قد **(height)** دارد. پس می توانیم به صورت زیر این مقادیر را به متغیرها با استفاده از سازنده پیشفرض اختصاص دهیم:

```php
<?php

    class Person
    {
        public $name;
        public $age;
        public $height;

        public function Person()
        {
            $this -> name   ='';
            $this -> age    = 9;
            $this -> height = 30;
        }

        public function TellInformation()
        {
            echo 'Name: '  . $this -> name   .'<br/>';
            echo 'Age: '   . $this -> age    .' Month'.'<br/>';
            echo 'Height: '. $this -> height .' cm';
        }
    }

    $person1 = new Person();
    $person1 -> TellInformation ();

?>
```

```
Name:
Age: 9 Month
Height: 30 cm
```

به این نکته توجه کنید که سازنده درست شبیه به یک متد است با این تفاوت که

- مقدار برگشتی ندارد.
- نام سازنده باید دقیقا شبیه نام کلاس باشد.

البته در PHP5 لازم نیست که نام سازنده دقیقا شبیه نام کلاس باشد و کافیست به جای نام سازنده از کلمه construct و دو علامت زیر خط به صورت زیر استفاده کنیم:

```
public function __construct()
{
   ....
}
```

حال فرض کنید می خواهیم سازنده ای ایجاد کنیم که بعد از ایجاد یک شی از کلاس، متغیرهای شی ایجاد شده را خودمان و با استفاده از سازنده ای که تعریف کرده ایم مقداردهی کنیم. به کد زیر توجه کنید:

```
<?php

  class Person
  {
      public $name;
      public $age;
      public $height;

      public function __construct($n, $a, $h)
      {
             $this -> name   = $n;
             $this -> age    = $a;
             $this -> height = $h;
      }

      public function TellInformation()
      {
             echo 'Name: '   . $this -> name .'<br/>';
             echo 'Age: '    . $this -> age  .'<br/>';
             echo 'Height: '. $this -> height;
      }
  }

  $person1 = new Person("Jack", 21, 160);
  $person1 -> TellInformation();

  echo '<br/><br/>';

  $person2 = new Person("Mike", 32, 158);
  $person2 -> TellInformation();

?>
Name: Jack
Age: 21
Height: 160

Name: Mike
```

```
Age: 32
Height: 158
```

همانطور که مشاهده می کنید در مثال بالا سازنده ای را سه آرگومان قبول می کند به کلاس **Person** اضافه کرده ایم (خطوط **14-9**). در خطوط **24** و **29** بعد از ایجاد شی و در داخل پرانتزها سه مقدار را به سازنده (خط **9**) ارسال می کنیم و سازنده این مقادیر را به متغیرها (خطوط **7-5**) اختصاص می دهد.

مخرب

مخرب ها نقطه مقابل سازنده ها هستند. مخرب ها متدهای خاصی هستند که هنگام تخریب یک شی فراخوانی می شوند. اشیا از حافظه کامپیوتر استفاده می کنند و اگر پاک نشوند ممکن است با کمبود حافظه مواجه شوید. می توان از مخرب ها برای پاک کردن منابعی که در برنامه مورد استفاده قرار نمی گیرند استفاده کرد. معمولا **PHP** به صورت اتوماتیک از زباله روب **(garbage collection)** برای پاک کردن حافظه استفاده می کند و لازم نیست شما به صورت دستی اشیا را از حافظه پاک کنید. به عنوان مثال وقتی کاربر یک فایل متنی را برای خواندن باز می کند و آن را نمی بندد، می توان عمل بستن فایل را با استفاده از مخرب انجام داد.دستور نوشتن مخرب به صورت زیر است:

```
public function __destruct()
{
   code to execute;
}
```

برنامه زیر نحوه فراخوانی سازنده و مخرب را نشان می دهد:

```
<?php

   class Test
   {
       public function __construct()
       {
           echo "Constructor was called." . '<br/>';
       }

       public function __destruct()
       {
           echo "Destructor was called.";
       }
   }

   $test = new Test();

?>
Constructor was called.
Destructor was called.
```

در کلاس Test یک سازنده (خطوط 8-5) و یک مخرب (خطوط 13-10) تعریف شده است. سپس یک نمونه از کلاس ایجاد کرده ایم (خط 16). وقتی یک نمونه از کلاس ایجاد می کنیم (خط 16) سازنده و مخرب فراخوانی شده و پیغام مناسب نمایش داده می شود .

سطح دسترسی

سطح دسترسی مشخص می کند که متدها یک کلاس یا متغیرها در چه جای برنامه قابل دسترسی هستند.در PHP سه سطح دسترسی وجود دارد :

- Public (عمومی)
- Private (خصوصی)
- Protect (محافظت شده)

در این درس می خواهیم به سطح دسترسی private و publicنگاهی بیندازیم. سطح دسترسی public زمانی مورد استفاده قرار می گیرد که شما بخواهید به یک متد یا متغیر در خارج از کلاس و حتی پروژه دسترسی یابید. به عنوان مثال به کد زیر توجه کنید :

```
 1:<?php
 2:
 3:    class Test
 4:    {
 5:        public  $number1 = 10;
 6:        private $number2 = 20;
 7:    }
 8:
 9:    $x = new Test();
10:
11:    echo $x -> number1;
12:    echo $x -> number2;
13:
14:?>
10
Fatal error: Cannot access private property Test::$number2 in C:\wamp\www\test.php on line 13
```

در این مثال یک کلاس با نام Test تعریف کرده ایم. سپس دو متغیر، یکی به صورت public خط 5(و دیگری به صورت private در داخل کلاس Testتعریف می کنیم (خط 6). در خط 9 یک نمونه از کلاس ایجاد کرده و در خطوط 11 و 12 سعی می کنیم که مقدار متغیرهای آن را چاپ کنیم. همانطور که در خروجی مشاهده می کنید متغیر number1 که به صورت public تعریف شده است قابل دسترسی و متغیر number2 که به صورت private تعریف شده است غیر قابل دسترسی می باشد. به طور کلی متغیرها و متدهایی که به صورت public تعریف می شوند در داخل کلاس و نمونه های ایجاد شده از آن قابل دسترسی و متغیرها و متدهایی که به صورت private تعریف می شوند فقط در داخل کلاس قابل دسترسی هستند و برای دسترسی به آنها در خارج از کلاس باید از خاصیت ها استفاده کرد که در درس بعد توضیح می دهیم. سطح دسترسی protect را بعد از مبحث وراثت در درسهای آینده آموزش می دهیم .

کپسوله سازی

کپسوله کردن(تلفیق داده ها با یکدیگر) یا مخفی کردن اطلاعات فرایندی است که طی آن اطلاعات حساس یک موضوع از دید کاربر مخفی می شود و فقط اطلاعاتی که لازم باشد برای او نشان داده می شود .

وقتی که یک کلاس تعریف می کنیم معمولا تعدادی فیلد برای ذخیره مقادیر مربوط به شی نیز تعریف می کنیم. برخی از این فیلدها توسط خود کلاس برای عملکرد متدها و برخی دیگر از آنها به عنوان یک متغیر موقت به کار می روند. لازم نیست که کاربر به تمام فیلدها یا متدهای کلاس دسترسی داشته باشد. اینکه فیلدها را طوری تعریف کنیم که در خارج از کلاس قابل دسترسی باشند بسیار خطرناک است چون ممکن است کاربر رفتار و نتیجه یک متد را تغییر دهد. به برنامه ساده زیر توجه کنید :

```php
<?php

    class Test
    {
        public $five = 5;

        public function AddFive($number)
        {
            $this -> five += $number;
            return $this -> five;
        }
    }

    $test = new Test;

    $test -> five = 10;
    echo $test -> AddFive(100)

?>
```

```
110
```

متد داخل کلاس Test به نام AddFive خطوط 11-7) دارای هدف ساده ای است و آن اضافه کردن مقدار 5 به هر عدد می باشد (همانطور که از اسم متد پیداست . AddFive) : در خط 14 یک نمونه از کلاس Test ایجاد کرده ایم و مقدار فیلد آن را در خط 16 از 5 به 10 تغییر می دهیم (در اصل نباید تغییر کند چون ما از برنامه خواسته ایم هر عدد را با 5 جمع کند ولی کاربر به راحتی آن را به 10 تغییر می دهد). همچنین متد AddFive() را در خط 17 فراخوانی و مقدار 100 را به آن ارسال می کنیم. مشاهده می کنید که قابلیت متد AddFive() به خوبی تغییر می کند و شما نتیجه متفاوتی مشاهده می کنید. اینجاست که اهمیت کپسوله سازی مشخص می شود. اینکه ما در درسهای قبلی فیلدها را به صورت public تعریف کردیم و به کاربر اجازه دادیم که در خارج از کلاس به آنها دسترسی داشته باشد کار اشتباهی بود. فیلدها باید همیشه به صورت private تعریف شوند .

خواص

property (خصوصیت) استانداردی برای دسترسی به متغیرهایی با سطح دسترسی private در داخل یک کلاس می باشد. هر property دارای دو بخش می باشد، یک بخش جهت مقدار دهی (بلوک set) و یک بخش برای دسترسی به مقدار (بلوک get) یک داده private می باشد property .

در مثال زیر نحوه تعریف و استفاده از property آمده است:

```
 2:<?php
 2:
 3:    class Person
 4:    {
 5:        private $name;
 6:
 7:        public function set_name($my_name)
 8:        {
 9:            $this->name = $my_name;
10:        }
11:
12:        public function get_name()
13:        {
14:            return $this->name;
15:        }
16:    }
17:
18:    $person1 = new Person();
19:
20:    $person1 -> set_name('Jack');
21:    echo $person1 -> get_name();
22:
23:?>
Jack
```

در برنامه بالا نحوه استفاده از property آمده است.همانطور که مشاهده می کنید در این برنامه ما یک خصوصیت تعریف کرده ایم و یک متغیر با سطح دسترسی private.

```
private $name;
```

دسترسی به مقدار این متغیر فقط از طریق property های ارائه شده امکان پذیر است .

```
public function set_name($my_name)
{
    $this->name = $my_name;
}

public function get_name()
{
    return $this->name;
}
```

در نامگذاری property ها به صورت قراردادی ابتدا کلمه set یا get و سپس علامت زیر و بعد نام متغیر را بنویسید:

```
set_name()
```

```
{

}
get_name()
{

}
```

در داخل کلاس دو بخش می بینید ، یکی بخش set (7-10) و دیگری بخش get. (12-15) بخش get ، که با کلمه کلیدی get نشان داده شده است به شما اجازه می دهد که یک مقدار را از متغیرهای private استخراج کنید. بخش set ، که با کلمه کلیدی set نشان داده شده است برای مقدار دهی به متغیرهای private به کار می رود. برای دسترسی به یک خاصیت می توانید از علامت ->استفاده کنید.

```
$person1 -> set_name('Jack');
```

دستور بالا بخش set مربوط به property را فراخوانی کرده و مقادیری به متغیر اختصاص می دهد. استفاده از property ها کد نویسی را انعطاف پذیر می کند مخصوصا اگر بخواهید یک اعتبارسنجی برای اختصاص یک مقدار به متغیر یا استخراج یک مقدار از آن ایجاد کنید. مثلا در مثال زیر شما می توانید یک محدودیت ایجاد کنید که فقط اعداد مثبت به فیلد age سن) اختصاص داده شود. می توانید با تغییر بخش set خاصیت Age این کار را انجام دهید:

```
 1:<?php
 2:
 3:    class Person
 4:    {
 5:        private $age;
 6:
 7:        public function set_age($my_age)
 8:        {
 9:            if($my_age > 0)
10:            {
11:                $this->age = $my_age;
12:            }
13:            else
14:            {
15:                echo 'Age can not negative or zero!';
16:            }
17:        }
18:
19:        public function get_age()
20:        {
21:            return $this->age;
22:        }
23:    }
24:
25:    $person1 = new Person();
26:
27:    $person1 -> set_age(12);
28:    echo $person1 -> get_age();
29:
30:?>
```

وراثت

وراثت به یک کلاس اجازه می دهد که خصوصیات یا متدهایی را از کلاس دیگر به ارث برد. وراثت مانند رابطه پدر و پسری می ماند به طوریکه فرزند خصوصیاتی از قبیل قیافه و رفتار را از پدر خود به ارث برده باشد .

کلاس پایه یا کلاس والد کلاسی است که بقیه کلاسها از آن ارث می برند .

کلاس مشتق یا کلاس فرزند کلاسی است که از کلاس پایه ارث بری می کند .

همه متد و خصوصیات کلاس پایه می توانند در کلاس مشتق مورد استفاده قرار بگیرند به استثنای اعضا و متدهای با سطح دسترسی `private` . مفهوم اصلی وراثت در مثال زیر نشان داده شده است :

```php
<?php

    class classParent
    {
        private function privateMessage()
        {
            echo 'This is private Message From Parent Class!';
        }

        public function publicMessage()
        {
            echo 'This is public Message From Parent Class!';
        }
    }

    class classChild extends classParent
    {

    }

    $child = new classChild();

    $child -> publicMessage();
    $child -> privateMessage ();

?>
```

```
This is public Message From Parent Class!
 Fatal error: Call to private method classParent::privateMessage()
```

همانطور که مشاهده می کنید در کد بالا دو کلاس تعریف کرده ایم : یکی کلاس(classParent خطوط 3-14) که دارای دو متد یکی با سطح دسترسی privateو دیگری با سطح دسترسی public است و کلاس دیگر (خطوط 16-19) که در بدنه خود هیچ متد یا متغیری ندارد. نحوه ارث بری یک کلاس به صورت زیر است :

```
class Child extends Parent
```

که در خط 16 مشخص کرده ایم که کلاس classChild قرار است از کلاس classParent ارث بری کند :

```
class classChild extends classParent
```

در خط 21 یک نمونه از کلاس فرزند ایجاد می کنیم و در خطوط 23 و 24 دو متد کلاس پدر ر فراخوانی می کنیم. همانطور که در خروجی مشاهده می کنید متدی که دارای سطح دسترسی public است فراخوانی و اجرا شده ولی در فراخوانی متدی با سطح دسترسی private با خطا مواجه می شویم. در پایان یاد آور می شویم که کلاس فرزند (خطوط 16-19) هیچ متد یا متغیری در بدنه خود ندارد ولی چون از کلاس classParent ارث بری کرده است متد با سطح دسترسی publicآن را می تواند برای خود داشته باشد .

سطح دسترسیProtect

سطح دسترسی protect اجازه می دهد که اعضای کلاس، فقط در کلاسهای مشتق شده از کلاس پایه قابل دسترسی باشند. بدیهی است که خود کلاس پایه هم می تواند به این اعضا دسترسی داشته باشد. کلاسهایی که از کلاس پایه ارث بری نکرده اند نمی توانند به اعضای با سطح دسترسی protectیابند .
در مورد سطوح دسترسی public و private قبلا توضیح دادیم. در جدول زیر نحوه دسترسی به سه سطح ذکر شده نشان داده شده است :

قابل دسترسی در	public	private	protected
داخل کلاس	true	true	true
خارج از کلاس	true	false	false
کلاس مشتق	true	false	true

مشاهده می کنید که public بیشترین سطح دسترسی را داراست. صرف نظر از مکان، اعضای public در هر جا فراخوانی می شوند و قابل دسترسی هستند.اعضای private فقط در داخل کلاسی که به آن تعلق دارند قابل دسترسی هستند. کد زیر رفتار اعضای دارای این سه سطح دسترسی را نشان می دهد :

```
1:<?php
2:
```

```php
    class classParent
    {
        protected $protectedMember = 10;
        private   $privateMember   = 20;
        public    $publicMember    = 30;
    }

    class classChild extends classParent
    {
        public function __construct ()
        {
            echo $this -> publicMember    .'<br/>';
            echo $this -> protectedMember .'<br/>';
            echo $this -> privateMember;
        }
    }

    $child = new classChild();

?>
```

```
10
20

(!) Notice: Undefined property: classChild::$privateMember
```

کدهایی که با خط قرمز نشان داده شده اند نشان دهنده وجود خطا است. همانطور که در خط **16** مشاهده می کنید کلاس **classChild** سعی می کند که به عضو **private** کلاس **classParent** دسترسی یابد. از آنجاییکه اعضای **private** در خارج از کلاس قابل دسترسی نیستند، حتی کلاس مشتق در خط **16** نیز ایجاد خطا می کند. اگر شما به خط **14** توجه کنید کلاس **classChild** می تواند به عضو **protect** کلاس **classParent** دسترسی یابد چون کلاس **classChild** از کلاس **classParent** مشتق شده است. حال کد زیر را در نظر بگیرید:

```php
<?php

    class classParent
    {
        public    $publicMember    = 10;
        protected $protectedMember = 20;
        private   $privateMember   = 30;
    }

    $parent =new classParent();

    echo $parent -> publicMember;
    echo $parent -> protectedMember;
    echo $parent -> privateMember;

?>
```

```
10
(!) Fatal error: Cannot access protected property classParent::$protectedMember
```

همانطور که در کد بالا مشاهده می کنید از آنجاییکه اعضای `protected` و `private` در خارج از کلاس قابل دسترسی نیستند خطوط 13 و 14 ایجاد خطا می کنند.