PHP

به زبان ساده

مؤلف: یونس ابراهیمی

سر شناسه : ابراهیمی، یونس - ۱۳۶۰

عنوان و نام پدید آورنده : PHP به زبان ساده / مؤلف: یونس ابراهیمی

مشخصات نشر : نبض دانش، تهران – ۱۳۹۵.

مشخصات ظاهری : ۳۳۲ صفحه مصور.

شابک : ۹۷۸-۶۰۰-۷۷۰۳-۹۲-۲

وضعیت فهرست نویسی : فیپا

موضوع : پی.اچ.پی.(زبان برنامه نویسی کامپیوتر)

رده بندی دیویی : ۰۰۵/۲۷۶۲

رده بندی کنگره : ۱۳۹۵ ۲ الف ۸۶ پ / ۷۳ / ۷۶ QA

شماره کتاب شناسی ملی : ۴۲۹۵۶۸۰

عنوان : PHP به زبان ساده

مؤلف : یونس ابراهیمی

ناشر : نبض دانش

سال چاپ : ۱۳۹۵

نوبت چاپ : دوم

تیراژ : ۲۰۰

قیمت :

تقدیم به

پدر و مادر عزیزم

فصل اول : مبانی زبان PHP

فصل دوم : توابع از پیش تعریف شده

فصل سوم : کار با فرم های HTML

فصل چهارم : کار با تاریخ و زمان

فصل پنجم : کار با ایمیل

فصل ششم : Session و Cookie

فصل هفتم : عبارات با قاعده

فصل هشتم : کار با فایل های XML

فصل نهم : کار با بانک اطلاعاتی

مقدمه

با توجه به گستردگی صفحات وب و توسعه روز افزون آنها، نیاز به زبان های برنامه نویسی طراحی سایت بصورت پویا بیشتر شده است. در اینجا ما قصد داریم زبان پی اچ پی (PHP) را آموزش دهیم. قبل از شروع باید بدانید یادگیری یک زبان برنامه نوسی نیاز به گذاشتن زمان و تمرین (انجام پروژه) است.

PHP یک زبان برنامه نویسی برای توسعه وب است، از این زبان می توان به عنوان یک زبان عمومی نیز استفاده کرد. این زبان در سال ۱۹۹۵ توسط Rasmus Lerdorf بوجود آمد و همچنان در حال توسعه است. بدلیل اینکه PHP در ابتدا برای استفاده شخصی ارائه شد، PHP از سر نام Personal Home Page Tools (PHP Tools) version 1.0 گرفته شده بود. اما بعد از تغییراتی که بر روی آن رخ داد، بعد از PHP 3 آنرا به PHP: Hypertext Preprocessor تغییر نام دادند.

یادگیری PHP خیلی آسان است. اما قبل از شروع به یادگیری PHP شما باید با مفاهیم HTML، CSS، JavaScript آشنا باشید تا بتوانید در این زبان بخوبی برنامه نویسی کنید. دستور زبان PHP خیلی شبیه به سبک زبان C است. به این دلیل زبان PHP شبیه زبان های C++، C#، Java و Perl است.

شاید یادگیری PHP در نگاه اول کاری پیچیده به نظر برسد اما اصولا برای یادگیری زبان های برنامه نویسی کافیست ایده های ذهنی مان را خط به خط روی کاغذی خیالی بنویسیم و سپس با توجه به شیوه نوشتن (syntax) هر زبانی به دنبال راه حل تبدیل و فهماندن ایده به سیستم شویم، از طرفی هیچ کس نمی تواند مدعی شود که همه چیز را می داند ولی دانستن اصول اولیه و داشتن تجربه تا حد رفع نیاز، چیزی است که با کمی تلاش و دقت می توانیم به آن دست یابیم؛ ما در آموزش هایمان سعی خواهیم کرد که در عین مختصر گویی به حد کافی نیز شما را با دنیای جالب کدنویسی php آشنا کنیم.

بی شک این اثر، خالی از اشکال نیست و از شما خوانندگان عزیز می‌خواهم که با نظرات و پیشنهادات خود بنده را در تکمیل و رفع نواقص آن از طریق پست الکترونیکی younes.ebrahimi.1391@gmail.com یاری بفرمایید.

برای دریافت فایل‌ها و آپدیت‌های جدید این کتاب به سایت **www.w3-farsi.com** مراجعه فرمایید.

راه‌های ارتباط با نویسنده

وب سایت: www.w3-farsi.com

لینک تلگرام: https://telegram.me/ebrahimi_younes

ID تلگرام: @ebrahimi_younes

پست الکترونیکی: younes.ebrahimi.1391@gmail.com

فصل اول

مبانی زبان PHP

HTML چیست

HTML مخفف Hyper Text Markup Language به معنی زبان نشانه گذاری فوق متن است. HTML زبان استاندارد طراحی صفحات وب است و تنها زبان قابل فهم برای مرورگرها می‌باشد. به طوریکه کدهای نوشته شده با هر زبان برنامه نویسی تحت وب در نهایت به صورت HTML به مرورگر تحویل داده می‌شوند تا مرورگر بتواند آن را پردازش کرده و به کاربر نمایش دهد.

در یک صفحه HTML می‌توان انواع عناصر از قبیل متن، تیتر، عکس، جدول و ... را قرار داد. صفحات HTML فقط از کدها که به صورت متن هستند، تشکیل شده‌اند. HTML یک زبان نشانه گذاری است، به این معنی که بخش‌های مختلف توسط اجزایی به نام تگ (Tag) از هم جدا شده، که هر کدام دارای کاربرد و خواص مربوط خود هستند. این تگ‌ها به مرورگر اعلام می‌کنند که هر بخش از صفحه باید به چه صورت نمایش داده شود.

هر یک از تگ‌های HTML، معنا و مفهوم خاصی دارند و تأثیر مشخصی بر محتوا می‌گذارند. مثلاً تگ‌هایی برای تغییر شکل ظاهری متن، نظیر درشت و ضخیم کردن یک کلمه یا برقراری پیوند به صفحات دیگر در HTML تعریف شده‌اند. یک فایل HTML، ترکیبی از تگ‌ها می‌باشد. هر تگ در HTML یک بخش ابتدایی و یک بخش انتهایی دارد که هم نام بوده و به صورت استاندارد طبق شکل کلی زیر به کار می‌روند:

```
<Open> Content </Close>
```

در بین دو تگ باز و بسته می‌توان هم تگ‌های دیگر را قرار داد و هم عبارت و هر چیزی که مد نظرتان است. کلیه نوشته‌ها و تگ‌های دیگری که در بین تگ ابتدا و پایان نوشته می‌شوند، محتویات تگ را تشکیل می‌دهند.

انواع تگ در HTML

تگ‌ها در HTML به صورت کلی به دو دسته تقسیم می‌شوند:

۱. تگ‌های که تگ پایانی ندارند.
۲. تگ‌هایی که تگ پایانی دارند.

در جدول زیر لیست برخی از تگ‌های HTML و کاربرد آنها ذکر شده است:

ابتدا	کاربرد	انتها
<h1>	برای ایجاد یک تیتر بزرگ به کار می‌رود.	</h1>
	برای ایجاد یک پاراگراف به کار می‌رود.	
<a>	برای ایجاد یک لینک به کار می‌رود.	
<table>	برای ایجاد یک جدول به کار می‌رود.	</table>
<tr>	برای ایجاد یک سطر در جدول به کار می‌رود.	</tr>
<td>	برای ایجاد یک سلول در جدول به کار می‌رود.	</td>
	کلمه‌ای که بین این تگ قرار بگیرد ضخیم می‌شود.	

	کلمات یا هر چیزی که بین این تگ قرار بگیرد وسط چین می‌شود.	
	برای تقسیم بندی فضا در صفحات وب به کار می‌رود.	
<form>	برای ایجاد یک فرم به کار می‌رود.	</form>
<q>	برای ایجاد نقل قول به کار می‌رود.	</q>
<small>	برای نمایش متن به صورت کوچک به کار می‌رود.	</small>
	برای نمایش متن به صورت درشت به کار می‌رود.	
 	برای ایجاد یک خط جدید به کار می‌رود.	
<img/>	برای قرار دادن تصویر در صفحه به کار می‌رود.	
	برای ایجاد یک خط در عرض صفحه به کار می‌رود.	

تگ‌های HTML نسبت به بزرگ و کوچک بودن حروف حساس نیستند، یعنی تگ‌های <b> با <B> برابر هستند. HTML فضاهای خالی را نادیده می‌گیرد، بنابراین به جای نوشتن یک فایل در یک خط می‌توانید آن را در چند خط بنویسید تا خوانایی آن بالاتر رود. بین عناصر HTML ممکن است رابطه پدر-فرزندی وجود داشته باشد:

```
<table>
   <tr>
      <td></td>
      <td></td>
   </tr>
   <tr>
      <td></td>
      <td></td>
   </tr>
</table>
```

در کد بالا تگ table پدر همه تگ‌های tr و td و همچنین تگ tr پدر تگ‌های td است که در داخل آن قرار دارند. همین کد بالا یک جدول با دو سطر و دو ستون ایجاد می‌کند. لیست کامل تگ‌های HTML در لینک زیر آمده است:

```
https://www.w3schools.com/tags/default.asp
```

تگ‌های بلاکی و غیر بلاکی

تگ‌های HTML از لحاظ اینکه در صفحه چه رفتاری از خود نشان می‌دهند به دو نوع بلاکی و داخلی تقسیم می‌شوند. تگ‌های بلاکی از خط جدید شروع می‌شوند و تمام صفحه را در بر می‌گیرند ولی تگ‌های غیر بلاکی فقط به اندازه محتوایی که دارند، تغییر اندازه می‌دهند. در زیر لیست تگ‌های بلاکی آمده است:

```
<address>       <article>       <aside>       <blockquote>    <canvas>
<dd>            <div>           <dl>          <dt>            <fieldset>
<figcaption>    <figure>        <footer>      <form>          <h1>-<h6>
<header>        <hr>            <li>          <main>          <nav>
<noscript>      <ol>            <output>      <p>             <pre>
```

```
<section>          <table>          <tfoot>          <ul>          <video>
```

و تگ‌های غیر بلاکی هم عبارتند از:

```
<a>          <abbr>          <acronym>     <b>          <bdo>
<big>        <br>            <button>      <cite>       <code>
<dfn>        <em>            <i>           <img>        <input>
<kbd>        <label>         <map>         <object>     <q>
<samp>       <script>        <select>      <small>      <span>
<strong>     <sub>           <sup>         <textarea>   <time>
<tt>         <var>
```

خواص تگ‌های HTML

هر تگ دارای مجموعه از خواص است که ویژگی‌های مختلف آنها را تعیین می‌کند. هر یک از این خواص را می‌توان در درون تگ ابتدایی معرفی و مقدار دهی کرد. در هنگام تعریف باید بین خاصیت و مقدار آن علامت = قرار داده و همچنین مقدارها باید درون علامت " " قرار بگیرند:

```
<tag attribute = "value" > Content </tag>
```

برخی از خاصیت‌های بین اکثر تگ‌ها مشترک هستند و لیست آنها در جدول زیر آمده است:

نام خاصیت	شرح
class	مشخص کننده کلاس تگ در کد نویسی برنامه است. این کلاس در کدنویسی CSS و زبان‌های اسکریپتی کاربرد دارد.
id	مشخص کننده یک شناسه منحصر به فرد برای تگ در درون سند HTML است. id هر تگ در کدنویسی CSS و زبان‌های اسکریپتی کاربرد دارد. توسط id می‌توان به تگ مورد نظر دست یافت.
style	یک خاصیت چند مقداری است که خصوصیات قالب دهی و اعمال سبک‌ها (CSS) را برای تگ مورد نظر مشخص می‌کند.
title	متنی است که به صورت tooltip در یک کادر زرد رنگ، در هنگام قرار گرفتن موس بر روی عنصر نمایش داده می‌شود.
dir	جهت قرار گرفتن نوشته را تعیین می‌کند.
language	مشخص کننده زبان برنامه نویسی کد مربوط به تگ است.
accesskey	یک میانبر صفحه کلید برای دستیابی به عنصر است.
tabindex	شماره ترتیبی قرار کرفتن فوکوس صفحه بر روی عنصر مورد نظر را در هنگام فشردن کلید tab مشخص می‌کند.

برخی دیگر از خاصیت‌ها مختص به یک تگ خاص هستند. مثلاً تگ a دارای یک خاصیت به نام href است که مقدار آن یک لینک می‌باشد :

```
<a href = "http://www.google.com"> Google </a>
```

رویدادهای تگ‌های HTML

جنس خاصیت‌های تگ‌های HTML می‌تواند از نوع رویداد باشد. بدین معنی که تگ‌های HTML می‌توانند شامل رویدادهایی باشند. رویدادها مجموعه عمل‌هایی هستند که در صورت بروز یک اتفاق در صفحه (مثل کلیک کردن بر روی یک عنصر، دابل کلیک، فشردن دکمه خاص و ...) عکس العمل نشان داده و باعث اجرای دستور یا دستورات تعیین شده برای آن اتفاق خاص می‌شوند. رویدادها را هم مانند خاصیت‌ها در تگ آغازین می‌نویسند:

```
<tag event = "value" > Content </tag>
```

رویدادها به چند دسته زیر تقسیم می‌شوند:

- رویدادهای پنجره در HTML
- رویدادهای عناصر فرم در HTML
- رویدادهای ماوس در HTML
- رویدادهای صفحه کلید در HTML

در جدول زیر لیست برخی از رویدادهای پرکاربرد ذکر شده است:

رویداد	کاربرد
onclick	این رویداد در هنگام کلیک کردن بر روی عنصر مورد نظر فعال می‌شود.
ondblclick	این رویداد در هنگام دابل کلیک کردن بر روی عنصر مورد نظر فعال می‌شود.
onblur	در هنگام از دست دادن فوکوس کنترل فعال می‌شود.
onfocus	در هنگامی که کنترل فوکوس را به دست می‌آورد، فعال می‌شود.
onkeydown	رویدادی که در هنگام فشردن یک کلید بر روی عنصر مورد نظر فعال می‌شود.
onkeypress	رویدادی که در هنگام فشردن و رها کردن یک کلید بر روی عنصر مورد نظر فعال می‌شود.
onload	رویدادی است که در هنگام لود شدن (بار گذاری) صفحه اجرا می‌شود.
onresize	رویدادی است که در هنگام تغییر سایز پنجره اجرا می‌شود.

در لینک زیر خواص عمومی و خصوصی و همچنین رویدادهای تگ‌های HTML آمده است:

```
https://www.w3schools.com/tags/ref_attributes.asp
```

در همین حد به توضیح HTML بسنده می‌کنیم و در ادامه با یک توضیح ساده کاربرد موارد گفته شده را در عمل به شما آموزش می‌دهیم.

ایجاد یک فایل HTML

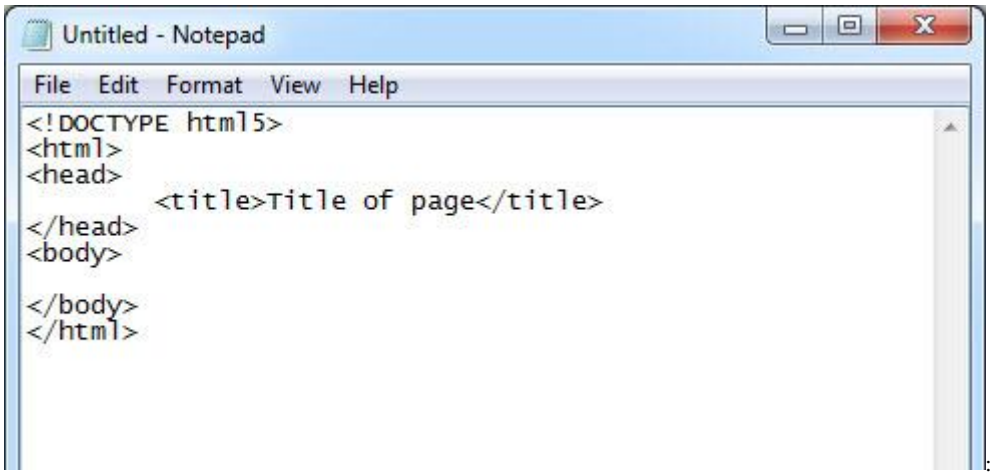
به کد زیر توجه کنید:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Title of page</title>
5  </head>
6  <body>
7
8  </body>
9  </html>
```

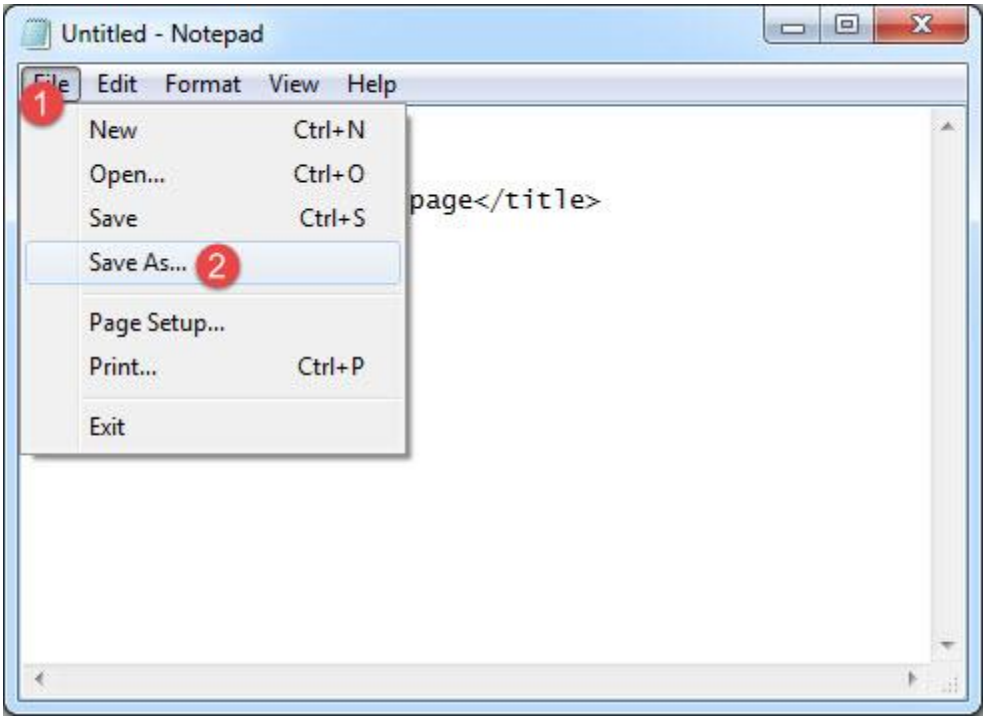
کد بالا ساختار استاندارد یک فایل HTML را نشان می‌دهد. در خط ۱، از DOCTYPE برای تعیین نسخه HTML استفاده شده است. DOCTYPE یک تگ نیست و فقط مسئول تعیین نسخه HTML است. در خط ۲ تگ html آمده و در خط ۹ بسته شده است. این تگ در واقع برای ایجاد یک سند HTML به کار می‌رود و همه تگ‌های دیگر فرزند آن محسوب می‌شوند .

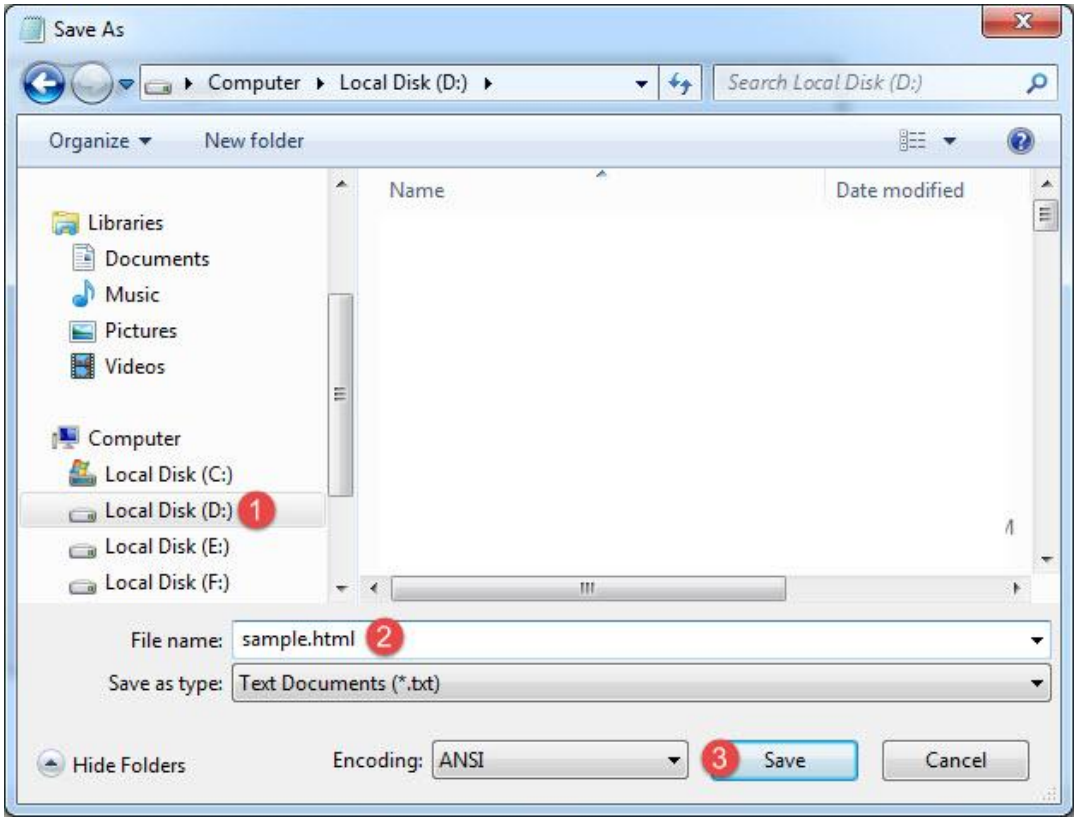
تگ head یک تگ بسیار مهم است. اگر چیزی در داخل این تگ بنویسید در صفحه نمایش داده نمی‌شود. در داخل این تگ، تگ‌های مهمی قرار می‌گیرند که از آنها برای عنوان صفحه و اطلاعاتی در مورد کد گذاری صفحه استفاده می‌شود. همانطور که در خط ۴ مشاهده می‌کنید تگ title در داخل تگ head قرار دارد و وظیفه آن اختصاص یک عنوان به سند HTML است.

در خط ۶ و ۸ تگ‌های باز و بسته body آمده‌اند. هر چیزی که بین تگ body قرار بگیرد در مرورگر مشاهده می‌شود. یعنی اگر می‌خواهید محتوایی را به کاربران نشان دهید باید آن را در داخل این تگ بنویسید. برنامه NotePad ویندوز را باز کرده و کدهای بالا را در داخل آن بنویسید

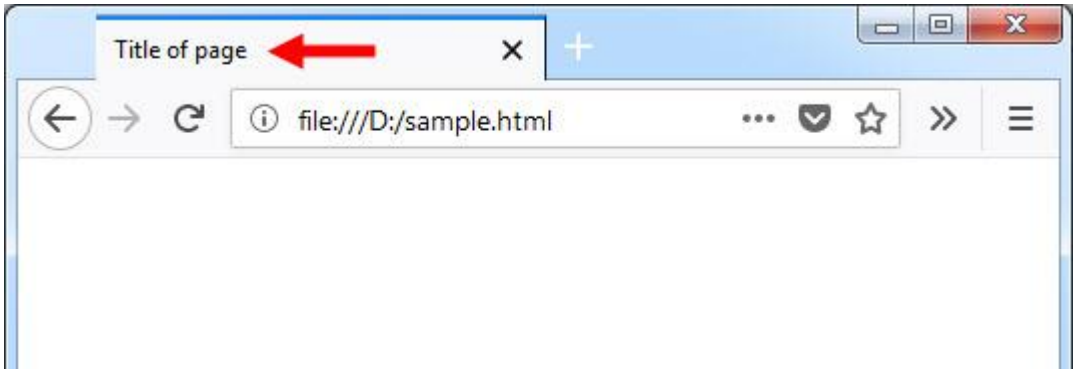


حل نوبت به ذخیره این فایل می‌رسد. فایل‌های HTML را باید با پسوند .html یا .htm ذخیره کنید. برای منظور فایل بالا را با نام sample.html در درایو D به صورت زیر ذخیره نمایید:





حال به درایو `D` رفته و بر روی فایل `sample.thml` دو بار کلیک کنید تا در مرورگر نمایش داده شود:

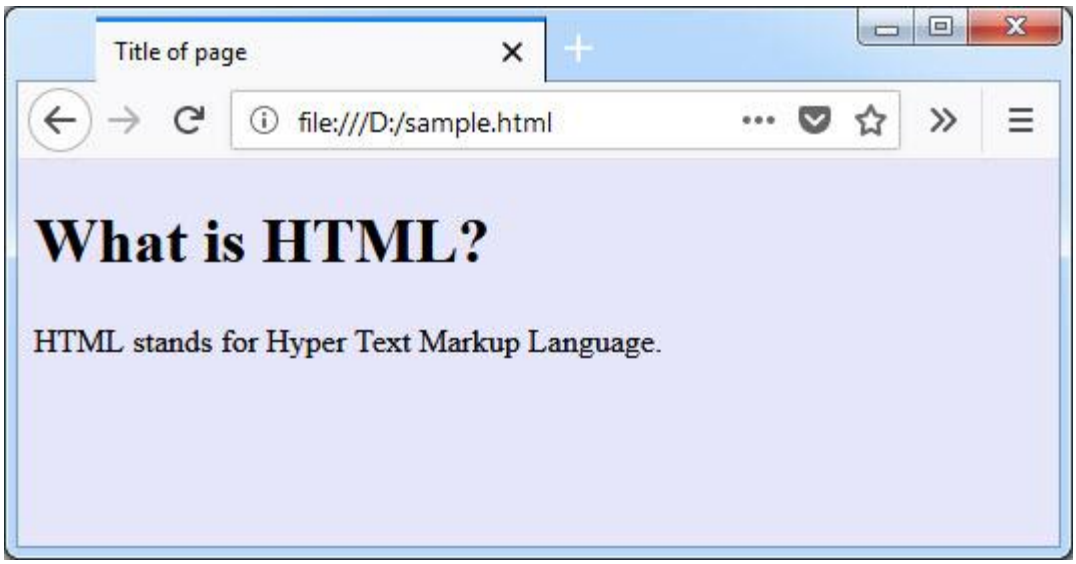


همانطور که در شکل بالا مشاهده می‌کنید، فقط عنوانی که در خط ۴ برای صفحه و در داخل تگ `title` مشخص کردیم نمایش داده شده است.

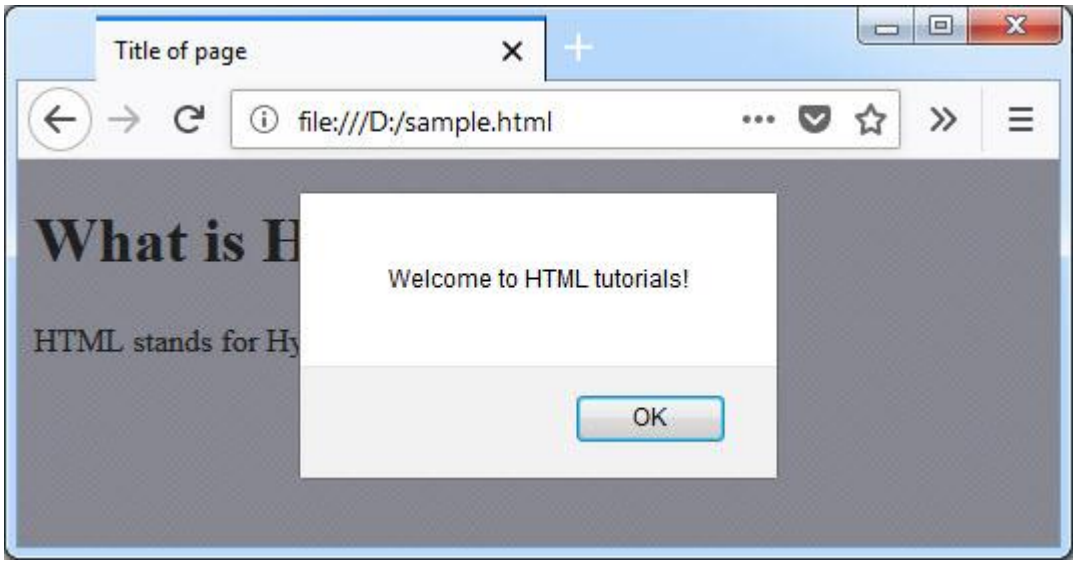
حال در کد بالا تغییرات زیر را اعمال کنید:

```
<!DOCTYPE html>
<html>
<head>
  <title>Title of page</title>
</head>
<body bgcolor="#E6E6FA">
<h1>What is HTML?</h1>
<p>HTML stands for <span onclick="alert('Welcome to HTML tutorials!');">Hyper Text Markup
Language</span>.</p>
</body>
</html>
```

در کد بالا ما برای تگ `body` یک خاصیت تعریف کرده‌ایم. در ضمن دو تگ بلاکی `h1` و `p` را هم در داخل این تگ قرار داده‌ایم. در داخل تگ P یک تگ span قرار دارد که برای آن یک رویداد تعریف کرده‌ایم. همانطور که قبلاً ذکر شد این تگ جز تگ‌های غیر بلاکی بوده و به اندازه محتوایی که در داخلش قرار می‌گیرد تغییر اندازه می‌دهد. حال برنامه را اجرا کنید:



همانطور که مشاهده می‌کنید دو تگ h1 و p بعد از اجرای برنامه از خط جدید شروع شده‌اند، چون جز تگ های بلاکی بوده و یک خط را کامل به خود اختصاص می دهند. خاصیت bgcolor باعث تغییر رنگ پس زمینه کل سند می‌شود. حال بر روی متن Hyper Text Markup Language کنید:



با کلیک بر روی این متن رویداد onclick به وقوع می‌پیوندد و باعث نمایش پیغام Welcome to HTML tutorials می‌شود. لازم نیست که بدانید alert چیست. این مثال را برای این ذکر کردیم که با موارد گفته شده از ابتدای درس آشنا شوید.

آشنایی با چند اصطلاح

در درس قبل در مورد HTML توضیح دادیم و گفتیم که این زبان نشانه گذاری، تنها زبانی است که برای مرورگر قابل فهم است. قبل از شروع یادگیری PHP بهتر است با چند اصطلاح پرکاربرد آشنا شوید که به شما در درک عمیقتر برنامه نویسی وب کمک می کنند. توصیه می کنیم که حتما این درس را با دقت بخوانید.

اینترنت

اینترنت (Internet) یک شبکه جهانی است که شامل میلیون‌ها کامپیوتر متصل به هم از سرتاسر دنیا بوده و اجازه انتقال انبوهی از داده‌ها و اطلاعات را میان کامپیوترهای موجود می‌دهد.

سرور و کلاینت

سرور (Server) یک کامپیوتر همیشه روشن و همیشه در دسترس است. سرورها وظیفه دارند تا اطلاعاتی که در هر زمان از شبانه روز، در اینترنت جستجو می‌کنید را برای ملاحظه شما، در اختیار قرار دهند. به سرور، کامپیوتر سرویس دهنده هم می گویند.

کلاینت (Client) در واقع همان کامپیوتر خودتان است. کامپیوتری که سرویس می‌گیرد. زمانی که شما به مرور وب و استفاده از انواع خدمات اینترنتی می‌پردازید، در واقع در نقش سرویس گیرنده عمل می‌کنید و کامپیوتری که از آنسوی شبکه به شما خدمات را ارائه می‌کند، سرویس دهنده و یا همان سرور نام دارد و اینترنت با ارتباط میلیون‌ها کامپیوتر سرویس دهنده و سرویس گیرنده شکل می‌گیرد.

هاست(Host)

به زبان ساده، وب سایت شما باید روی یک سرور یا سرویس دهنده میزبان قرار گیرد تا بر روی شبکه بین المللی (اینترنت) برای همه قابل دیدن باشد. همانطور که گفته شد، سرور یک کامپیوتر است. کامپیوتر هم دارای قطعه‌ای به نام هارد برای ذخیره سازی اطلاعات می‌باشد. برخی شرکت‌های ارائه دهنده میزبانی‌هاست، فضای Hard سرورهای خود را به فضاهای کوچک‌تری تقسیم کرده و شما می‌توانید با خرید این فضا، سایت خود را در آن قرار دهید. به این فضاهاست می گویند.

دامین(Domain)

در واقع نامی است که وب سایت از طریق آن قابل دستیابی خواهد بود. به طور کلی یک آدرس اینترنتی، از دو بخش نام دامنه و دامنه تشکیل شده‌است. برای مثال در یک آدرس اینترنتی مثلاً example.ir نام دامنه همان کلمه‌ی example است و دامنه نیز همان ir. یا دیگر موارد مشابه مانندcom. یاorg.

نام دامنه بخشی است که گویای نام برند یا فعالیت شما بوده و آن را به دلخواه خود انتخاب می‌نمایید. انتخاب نام پرمعنا برای سایت یکی از مهمترین مراحل داشتن وب سایت می‌باشد. ثبت نام دامنه خوب ظرایفی دارد که بی‌توجهی به آن‌ها می‌تواند به هویت آنلاین شما ضربه بزند. نام دامنه همانند لوگو و برند یک شرکت است که اگر درست انتخاب شود مردم آن را برای معرفی محصول خاص خواهند شناخت و در جذب مشتری آنلاین و بازاریابی موفق بسیار مؤثر خواهد بود. انتخاب نام دامنه مناسب، ساده، کاربرپسند، مختصر، به یاد ماندنی بسیار مهم بوده و همچنین در رتبه‌بندی (Ranking) نتایج جستجوی گوگل نیز تأثیر گذار می‌باشد.

وب سایت(Web Site)

یک وب سایت را می‌توان یک پوشه درنظر گرفت که این پوشه دارای یک نام بوده و در داخل آن فایل‌ها و پوشه‌های دیگری قرار دارند.

وب سرور(Web Server)

وب سرور در واقع یک نرم افزار یا برنامه کامپیوتری است که بر روی سرور نصب می‌شود و درخواست‌هایی را که از طرف کاربر و از طریق مرورگر به سرور ارسال می‌شوند را مدیریت می‌کند.

سیستم نام دامنه(DNS)

کلمه DNS، مخفف Domain Name System یا "سیستم نام دامنه" است. سیستم نام دامنه (DNS) یک سیستم پایگاه داده است که نام کامل دامنه یک کامپیوتر را به یک آدرس IP ترجمه می‌کند. کامپیوترهای موجود در یک شبکه برای اتصال به یکدیگر از آدرسهای IP استفاده می‌کنند، ولی به یاد داشتن آدرس‌های IP کامپیوترهای یک شبکه برای افرادی که قصد اتصال به آنان را دارند بسیار دشوار است.

مثلاً به خاطر سپردن نام دامنه google.com بسیار ساده‌تر از به خاطر سپردن آدرس IP نظیر آن (207.171.166.48) است. به همین علت اغلب ما برای اتصال به سایت‌ها، نام دامنه آن را وارد می‌کنیم. لذا DNS به شما امکان می‌دهد تا به جای استفاده از آدرس‌های عددی IP برای اتصال به یک کامپیوتر خاص در شبکه‌ای دیگر (یا برای دسترسی به یک سرویس راه دور)، با به کارگیری نام دامنه‌ای که به خاطر آوردن آن برای شما راحت‌تر است به آن کامپیوتر متصل شده یا از آن سرویس بهره بگیرید.

حال که با مفاهیم بالا آشنا شدید، بهتر است که با طرح یک سؤال ارتباط بین این اصطلاحات را برای شما توضیح دهیم و آن سؤال این است که: وقتی شما یک آدرس را در مرورگر تایپ می‌کنید دکمه Enter را می‌زنید چه اتفاقی می افتد؟ در اصل وقتی شما یک آدرس در مرورگر می‌نویسید یا آدرس یک سایت است یا آدرس یک صفحه از سایت. با نوشتن این آدرس، یک درخواست به سرور ارسال می‌شود که این درخواست شامل مراحل زیر است:

۱. کاربر آدرس یک سایت (مثلاً Example.com) را در مرورگر وارد می‌کند و دکمه Enter را می زند.
۲. مرورگر برای باز کردن سایت به یک عدد نیاز دارد که این عدد همان IP است. مثلاً IP سایت گوگل 216.58.214.238 می‌باشد. یعنی اگر این عدد را در مرورگر نوشته و دکمه Enter را بزنید سایت گوگل باز می‌شود. تبدیل نام سایت به IP توسط کامپیوترهایی انجام می‌شود که به آنها DNS می گویند. این کامپیوترها IP سایت‌ها را در خود ذخیره می‌کنند DNS،IP را در اختیار مرورگر می‌گذارد.
۳. مرورگر درخواستی را به این IP که در اصل همان سرور است، ارسال می‌کند.
٤. سرور که سایت مورد نظر ما بر روی آن قرار دارد، درخواست را از مرورگر دریافت می‌کند.
۵. بعد از دریافت درخواست توسط سرور، این درخواست توسط یک نرم افزار که بر روی سرور نصب است و به آن وب سرور (Web Server) گفته می‌شود، پردازش می‌شود. فرض کنیم که این درخواست آدرس یک صفحه است. وب سرور به داخل پوشه سایت مورد نظر ما رفته و فایل را پیدا می‌کند.
٦. وب سرور بعد از پیدا کردن فایل آن را به مفسر PHP داده تا محتویات آن را تفسیر کند.
۷. مفسر PHP بعد از خواندن محتویات فایل، تمام موارد گفته شده در فایل، اعم از اطلاعاتی که لازم است از بانک گرفته شود یا عکس و ... را جمع آوری کرده و در اختیار وب سرور قرار می‌دهد.
۸. وب سرور هم اطلاعات گرفته شده را در قالب یک فایل HTML به مرورگر ارسال می‌کند.
۹. مرورگر فایل HTML را دریافت و پردازش کرده و آن را به کاربر نمایش می‌دهد.

PHP چیست؟

PHP (پی‌اچ‌پی) یک زبان برنامه‌نویسیِ شیءگرا است که در سال ۱۹۹۵ میلادی توسط راسموس لردورف(Rasmus Lerdorf) ساخته شد. PHP شاید عمومی‌ترین زبان اسکریپتی تحت وب باشد، به طوریکه، تا ژانویه سال ۲۰۱۳ میلادی پی‌اچ‌پی بر روی ۲۴۴ میلیون وب سایت نصب شده و استفاده از آن روز به روز بیشتر می‌شود. PHP یک زبان برنامه نویسی سمت سرور (Server-Side) است

سرور به کامپیوتری متصل به اینترنت گفته می‌شود که حاوی یک یا چند وب سایت می‌باشد. کدهای PHP در سمت سرور پردازش و اجرا می‌شوند، نه در مرورگر. نتیجه این پردازش به صورت خروجی HTML برای مرورگر شما ارسال شده و شما نتیجه را در صفحه مرورگر خود مشاهده می‌کنید.

عمومی‌ترین تعریف PHP این است که PHP مخفف کلمات Hypertext Pre-processor می‌باشد. شاید برایتان این سؤال پیش بیاید که مخفف کلمات فوق HPP است. درست است، اما در نسخه‌های قبلی برنامه PHP را به عنوان مخفف کلمات Personal Home Page تعریف کرده‌اند. که مخفف آنها PHP می‌شود. در جدول زیر نسخه‌های مختلف این زبان ذکر شده‌اند:

نسخه	تاریخ پیدایش	پشتیبانی تا تاریخ
1.0	8 June 1995	
2.0	1 November 1997	
3.0	6 June 1998	20 October 2000
4.0	22 May 2000	23 June 2001
4.1	10 December 2001	12 March 2002
4.2	22 April 2002	6 September 2002
4.3	27 December 2002	31 March 2005
4.4	11 July 2005	7 August 2008
5.0	13 July 2004	5 September 2005
5.1	24 November 2005	24 August 2006
5.2	2 November 2006	6 January 2011
5.3	30 June 2009	14 August 2014
5.4	1 March 2012	3 September 2015
5.5	20 June 2013	21 July 2016
5.6	28 August 2014	31 December 2018
6.x	Not released	N/A
7.0	3 December 2015	3 December 2018
7.1	1 December 2016	1 December 2019
7.2	30 November 2017	30 November 2020

از این به بعد به شما کدنویسی PHP را آموزش می‌دهیم و متوجه خواهید شد که یادگیری آن بسیار آسان است.

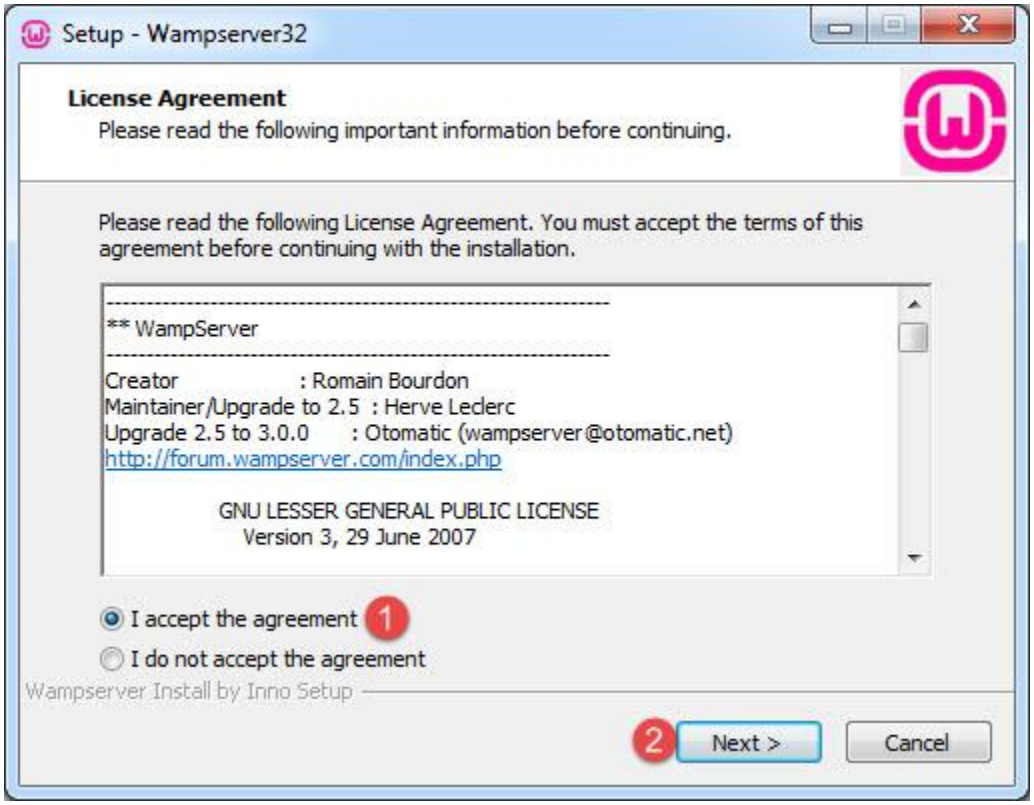
برای شروع کار با PHP چه چیزهایی لازم دارید؟

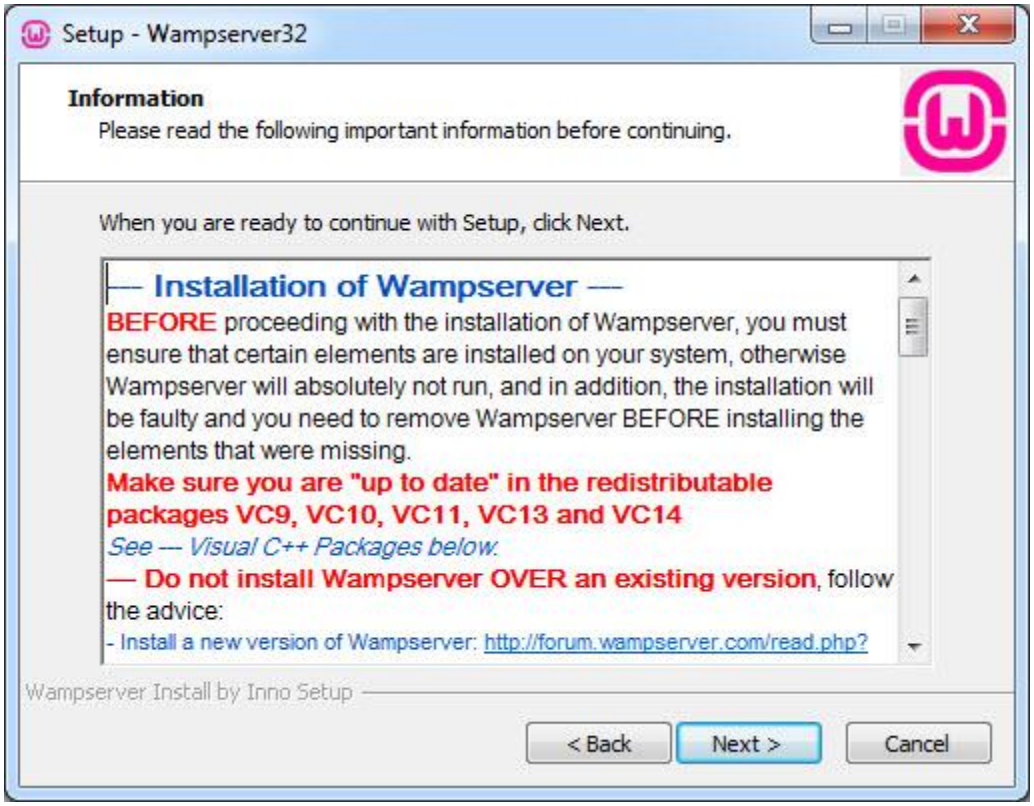
همانطور که در درس های ابتدایی گفته شد، برای نوشتن و تست کدهای PHP ابتدا به یک سرور نیاز دارید. خوشبختانه، لازم نیست که یک سرور خریداری کنید و نیاز به خرج پول نیست. درست است که PHP یک زبان برنامه نویسی عمومی است اما چون یک زبان اسکریپتی سمت سرور است باید یک هاست (مقداری از فضای وب) که PHP را پشتیبانی کند خریداری کنید و یا کاری کنید که کامپیوترتان به عنوان یک سرور عمل کند. چون PHP نمی‌تواند بر روی کامپیوتر اجرا شود. این زبان بر روی سرور (server) اجرا شده و نتایج را به کامپیوتر سرویس گیرنده (client) برگشت می‌دهد. 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 می‌باشد. این نرم افزار تمام موارد لازم برای اجرای کدها را نصب می‌کند. نحوه نصب و استفاده از این نرم افزار را برای شما توضیح می‌دهیم. برای دانلود این نرم افزار بر روی لینک زیر کلیک کنید:

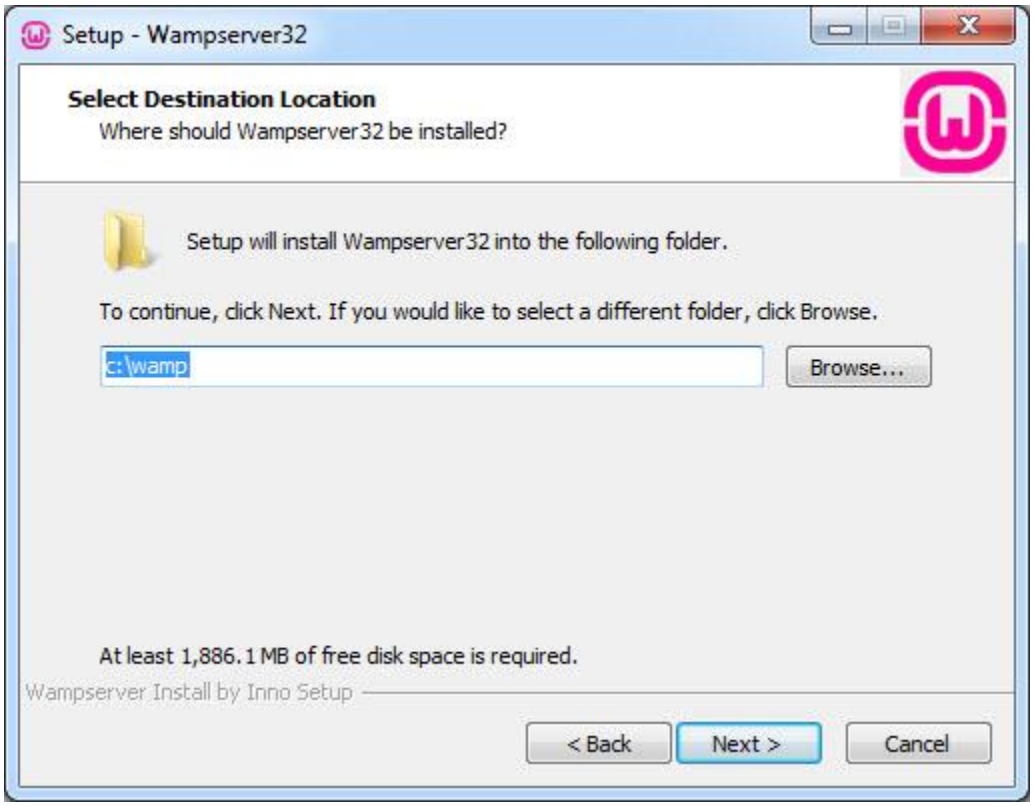
```
http://www.wampserver.com/en/
```

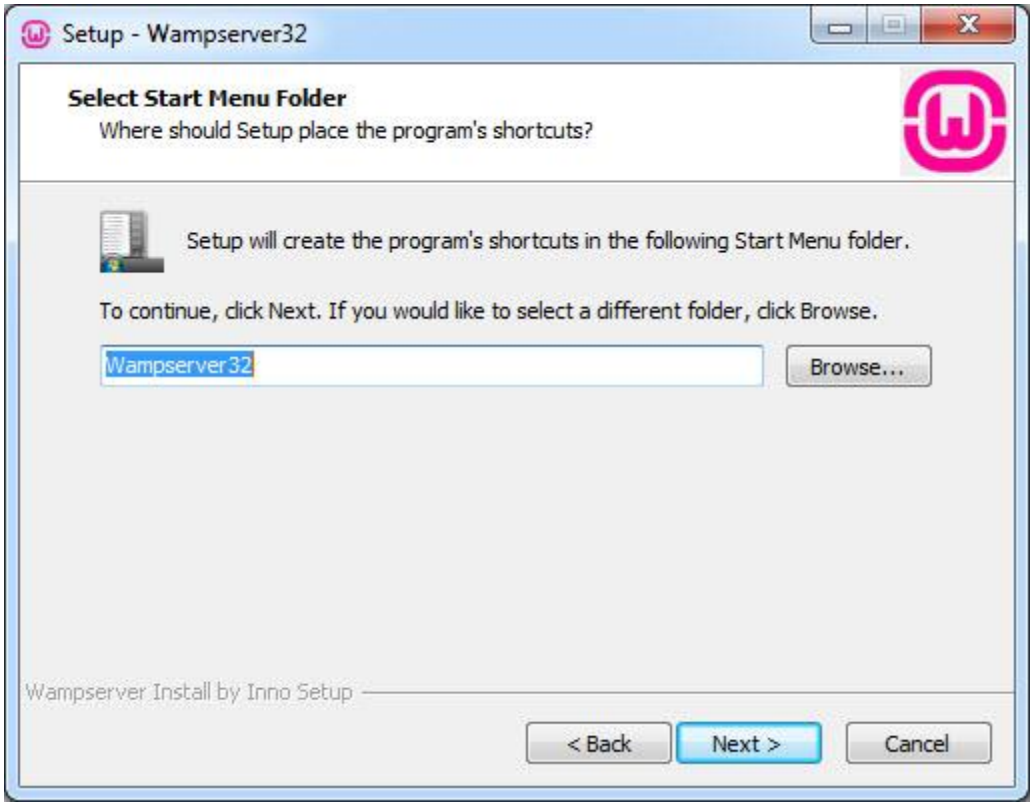
نصب و تست Wampserver

بعد از دانلود Wampserver مراحل زیر را برای نصب آن طی کنید. در صفحه زیر بر روی دکمه OK و در صفحات بعدی بر روی دکمه Next کلیک کنید:

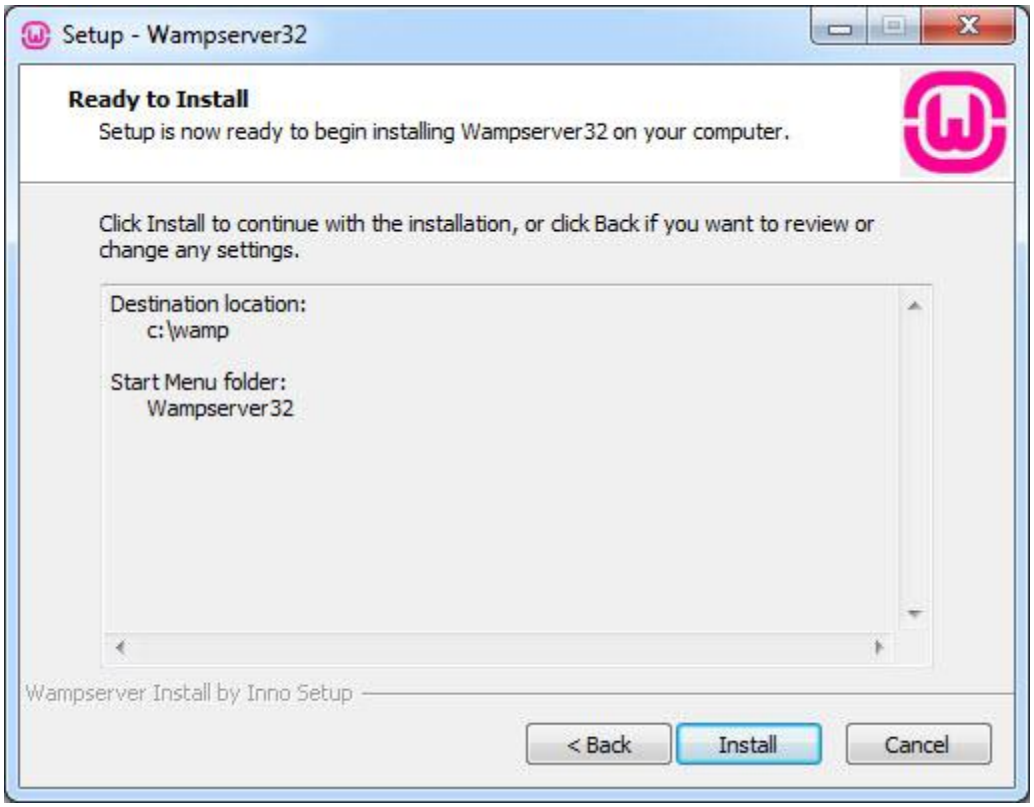


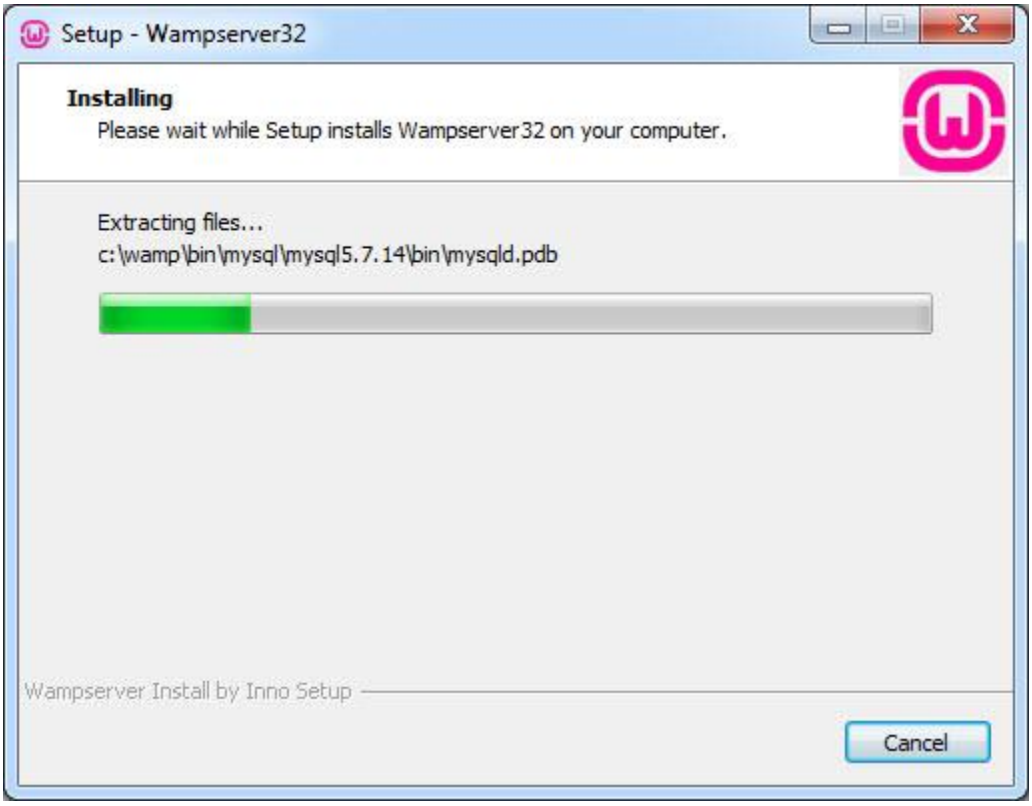

Setup - Wampserver32
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
— Installation of Wampserver —
BEFORE proceeding with the installation of Wampserver, you must ensure that certain elements are installed on your system, otherwise Wampserver will absolutely not run, and in addition, the installation will be faulty and you need to remove Wampserver BEFORE installing the elements that were missing.
Make sure you are "up to date" in the redistributable packages VC9, VC10, VC11, VC13 and VC14
See --- Visual C++ Packages below.
— Do not install Wampserver OVER an existing version, follow the advice:
- Install a new version of Wampserver: http://forum.wampserver.com/read.php?
Wampserver Install by Inno Setup
< Back
Next >
Cancel


Setup - Wampserver32
Select Destination Location
Where should Wampserver32 be installed?
Setup will install Wampserver32 into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
c:\wamp
Browse...
At least 1,886.1 MB of free disk space is required.
Wampserver Install by Inno Setup
< Back
Next >
Cancel

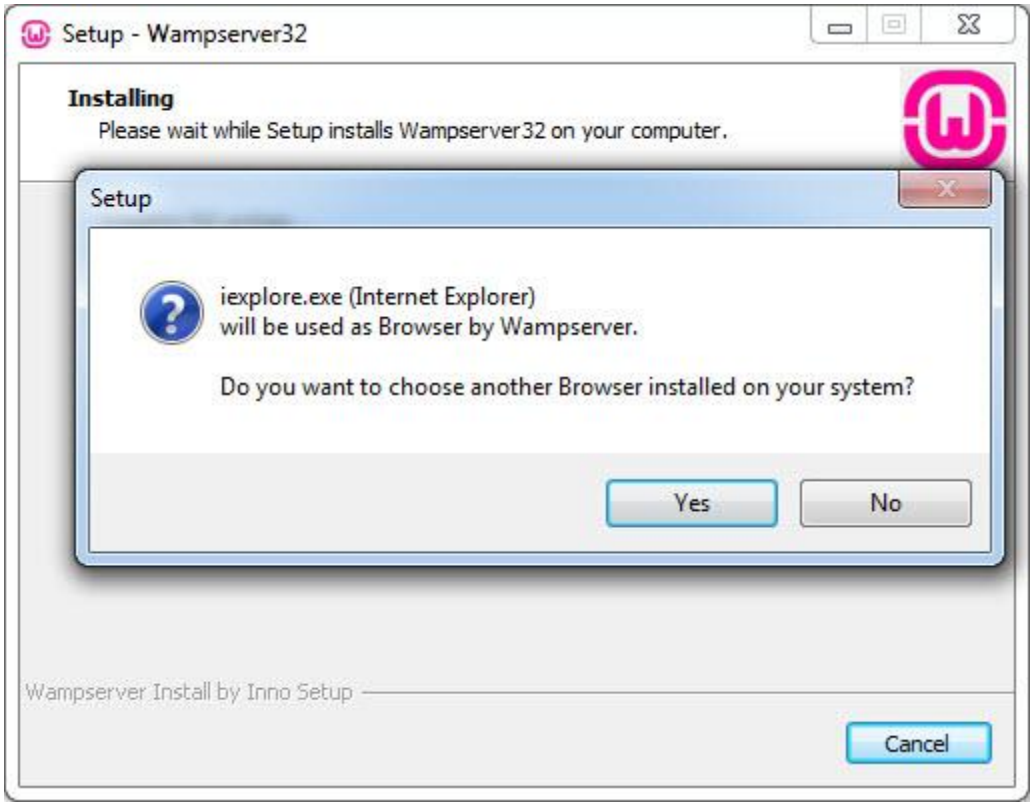


در صفحه زیر بر روی دکمه `Install` کلیک کرده تا وارد مرحله نصب شوید:

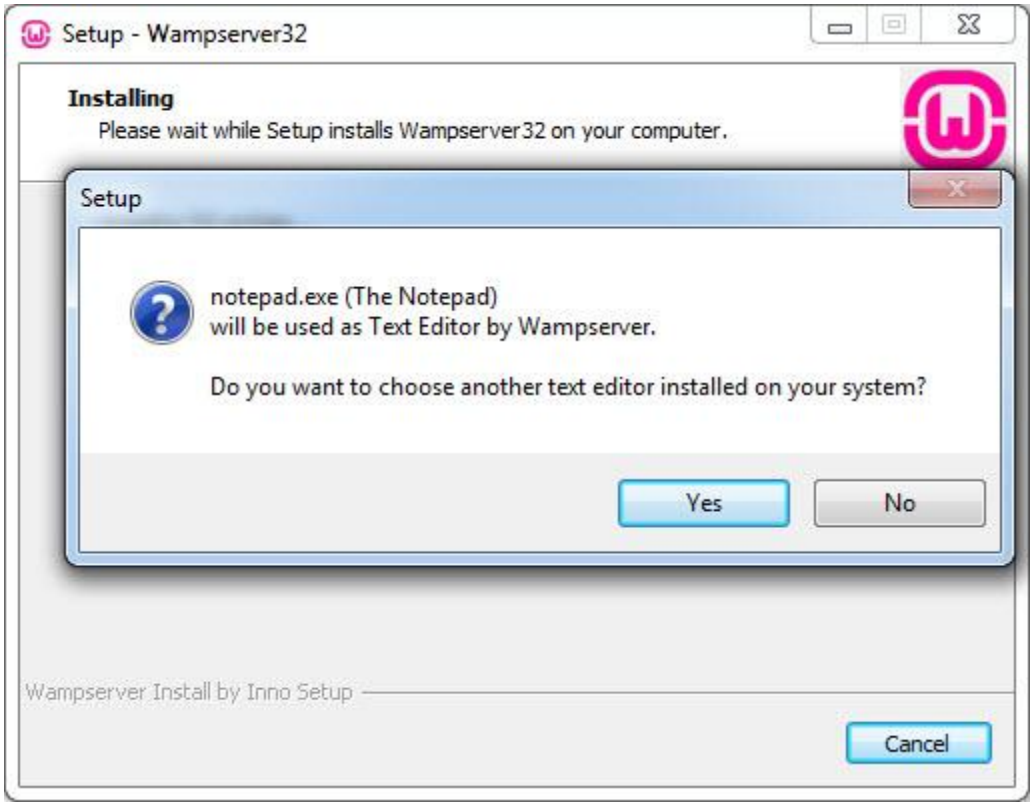




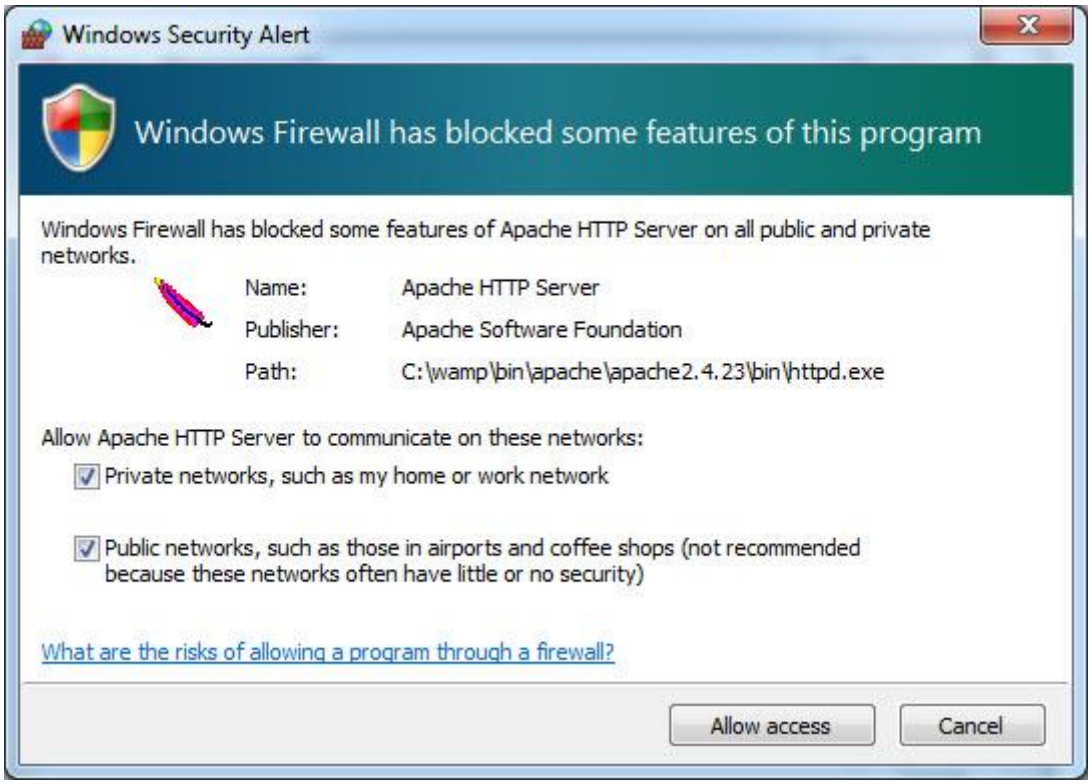
در صفحه زیر برنامه از شما می‌خواهد که مرورگر پیشفرض جهت اجرای نمایش خروجی برنامه‌ها را تغییر دهید که ما در اینجا از مرورگر پیشفرض استفاده کرده و دکمه NO را می‌زنیم:

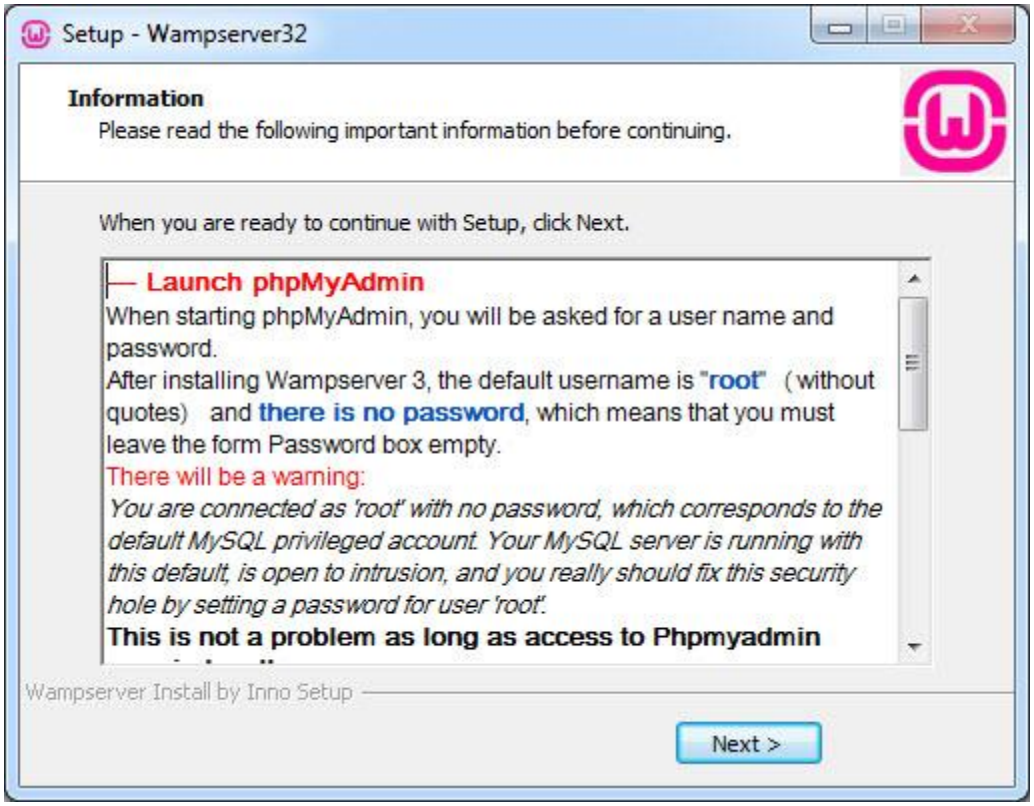


و در صفحه زیر هم از شما که در صورت تمایل اگر می‌خواهید از ویرایشگر متنی غیر از NotPad ویندوز استفاده کنید، آن را انتخاب کنید، که در این صفحه باز هم دکمه NO را می‌زنیم:



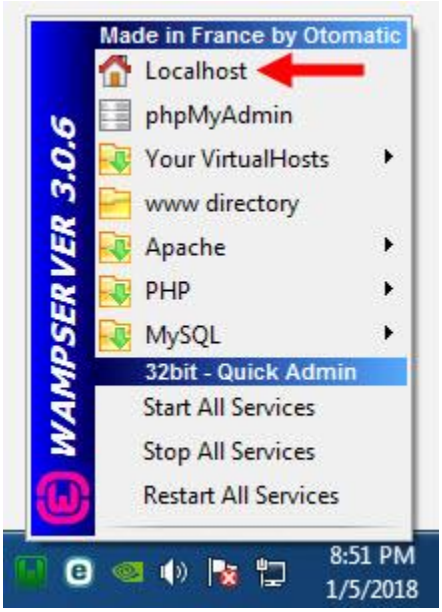
در صفحه زیر هر دو جعبه را تیک زده و دکمه `Allow Access` را کلیک کنید :



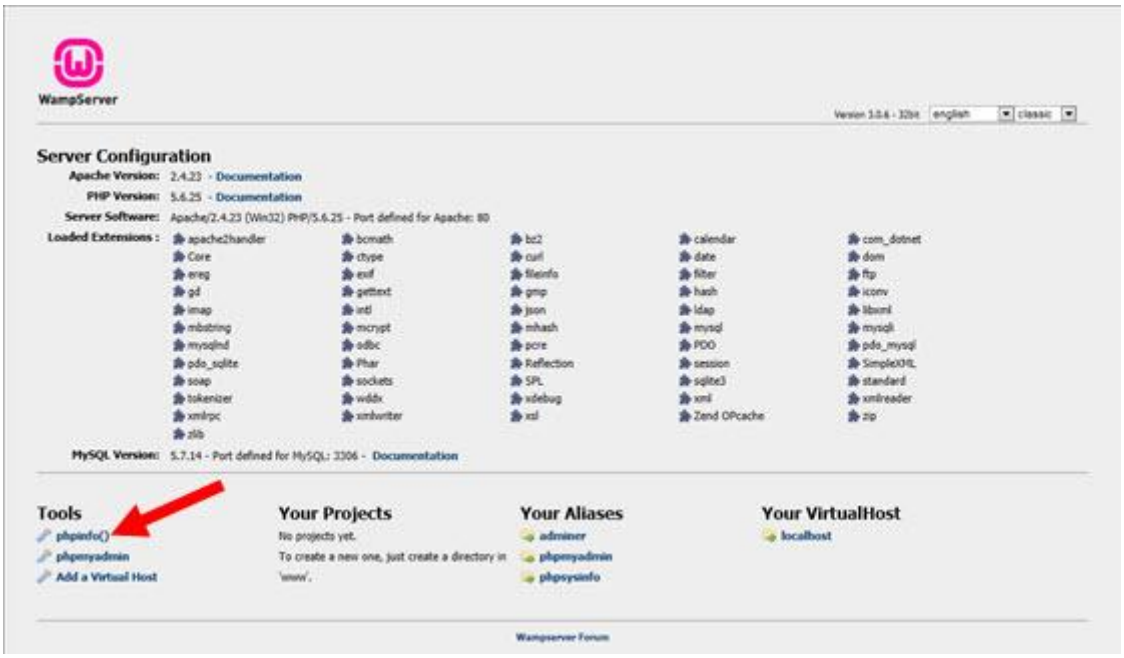


بعد از نصب آن یک آیکون جدید مانند شکل زیر در سمت راست و پایین صفحه مانیتور و درست کنار آیکون ساعت مشاهده خواهید کرد:

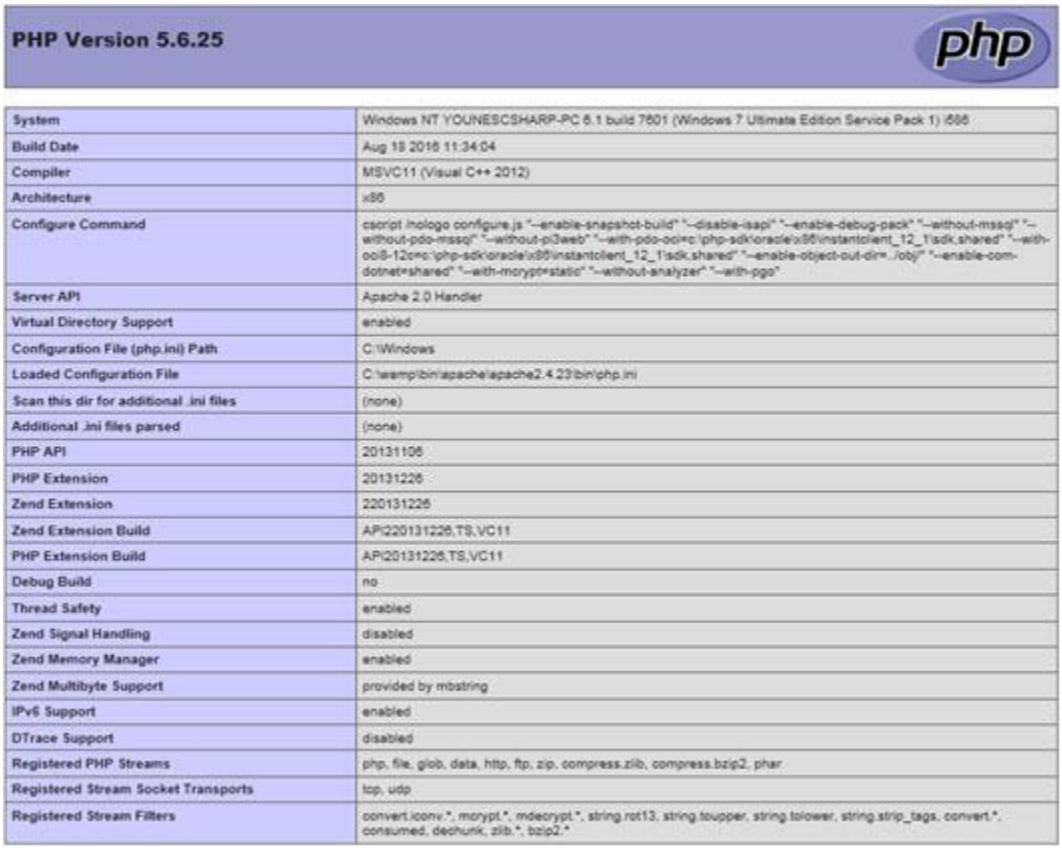
بر روی آیکونی که می‌بینید کلیک کنید تا یک منو باز شود. در منوی ظاهر شده می‌توانید سرور را متوقف کرده، از آن خارج شوید و صفحات پیکربندی را مشاهده نمایید. بر روی `localhost` کلیک کنید. مشاهده می‌کنید که یک صفحه ظاهر می‌شود:



`Localhost` به سروری که بر روی کامپیوتر شما نصب شده است اشاره می‌کند.



در قسمت Tools بر روی phpinfo() کلیک کنید. اگر همه چیز به خوبی پیش برود صفحه‌ای به شکل زیر مشاهده خواهید کرد که نشان دهنده نصب صحیح سرور بوده و شما می‌توانید شروع به کدنویسی کنید.



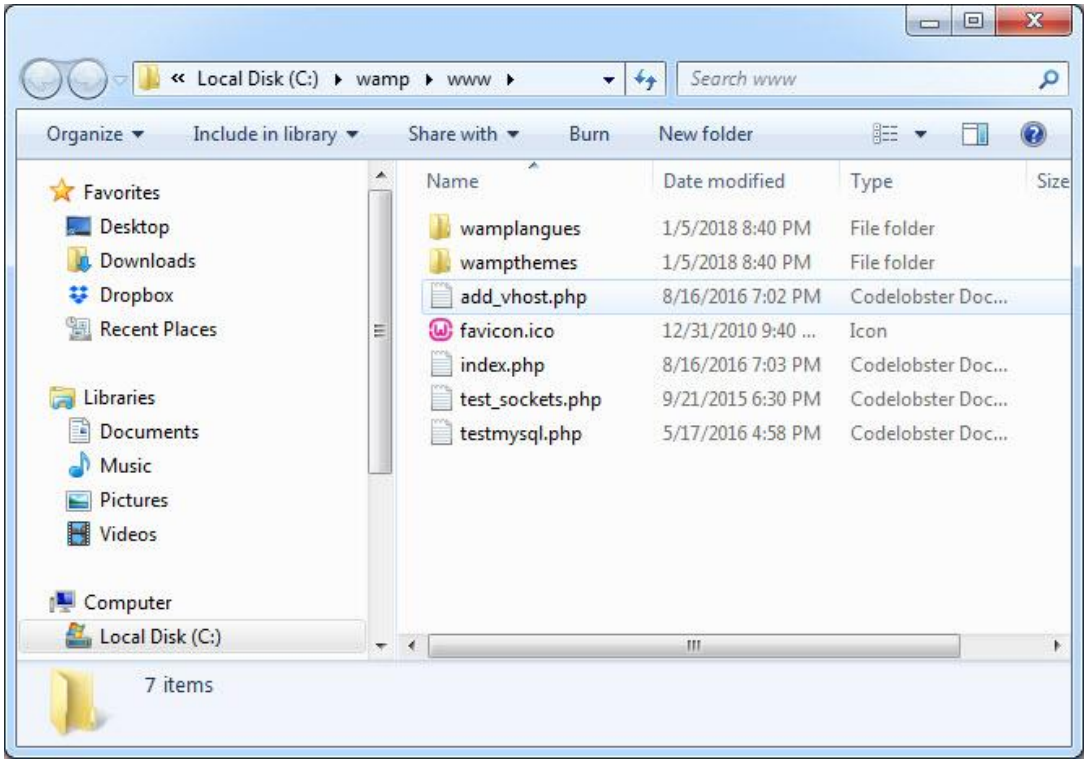
PHP Version 5.6.25

System	Windows NT YOUNESCSHARP-PC 6.1 build 7601 (Windows 7 Ultimate Edition Service Pack 1) i586
Build Date	Aug 19 2016 11:34:04
Compiler	MSVC11 (Visual C++ 2012)
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\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--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\apache2.4.23\bin\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,TS,VC11
PHP Extension Build	API20131226,TS,VC11
Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp
Registered Stream Filters	convert.iconv.*, mcrypt.*, mdecrypt.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, zlib.*, bzip2.*

ذخیره فایل‌های 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 و انتخاب گزینه www Directory قابل مشاهده است. به شکل زیر توجه کنید:

وقتی بر روی `www Directory` کلیک می‌کنید پنجره‌ای به شکل زیر ظاهر می‌شود. در داخل این پنجره ممکن است از قبل فایل یا پوشه‌هایی وجود داشته باشند:



پوشه `WWW` به صورت پیشفرض در مسیر زیر قرار دارد:

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

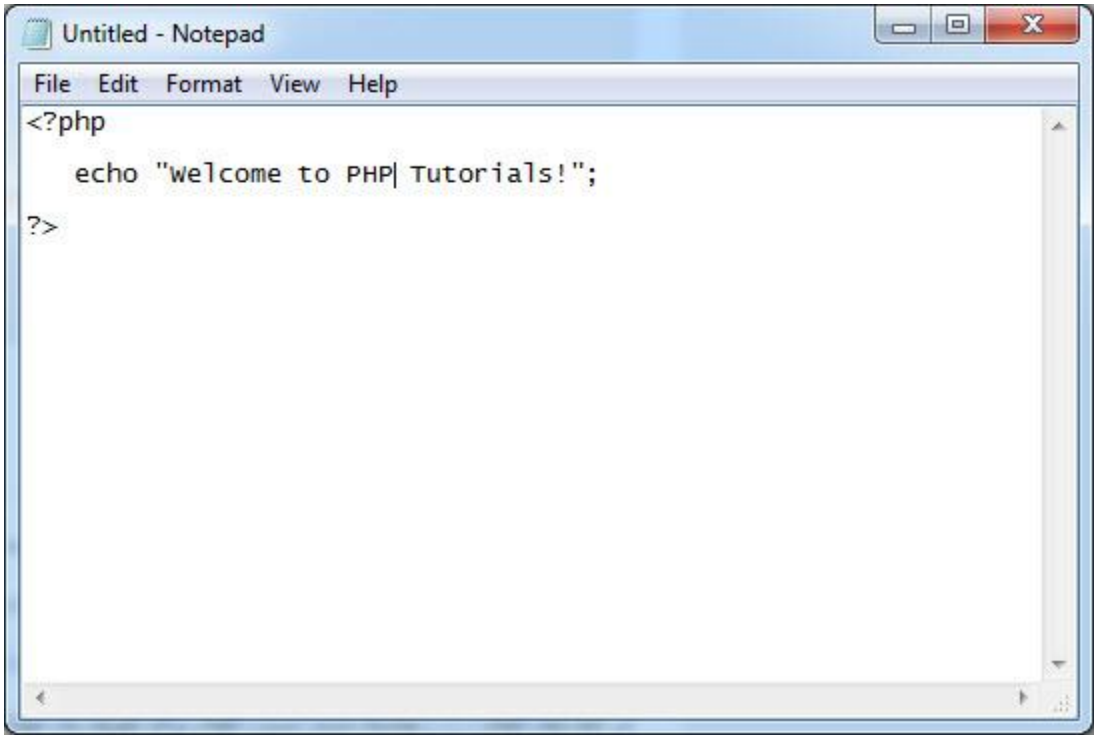
برای اجرای فایل‌های PHP باید آنها را در این پوشه ذخیره کنید.

ساخت یک برنامه ساده

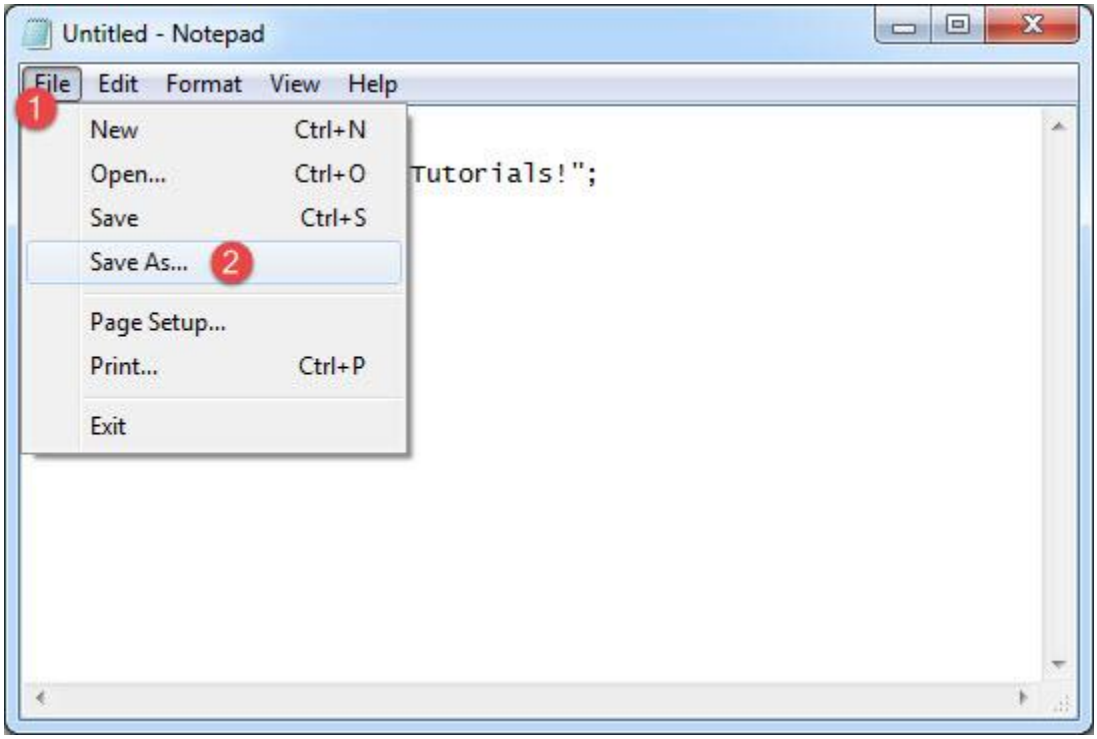
برای ایجاد یک برنامه ساده در PHP ابتدا برنامه Wampserver را ایجاد کرده و از سبز رنگ بودن آیکون آن  مطمئن شوید. سپس برنامه NotePad ویندوز را اجرا کرده:

و کدهای زیر را در داخل آن می‌نویسیم:

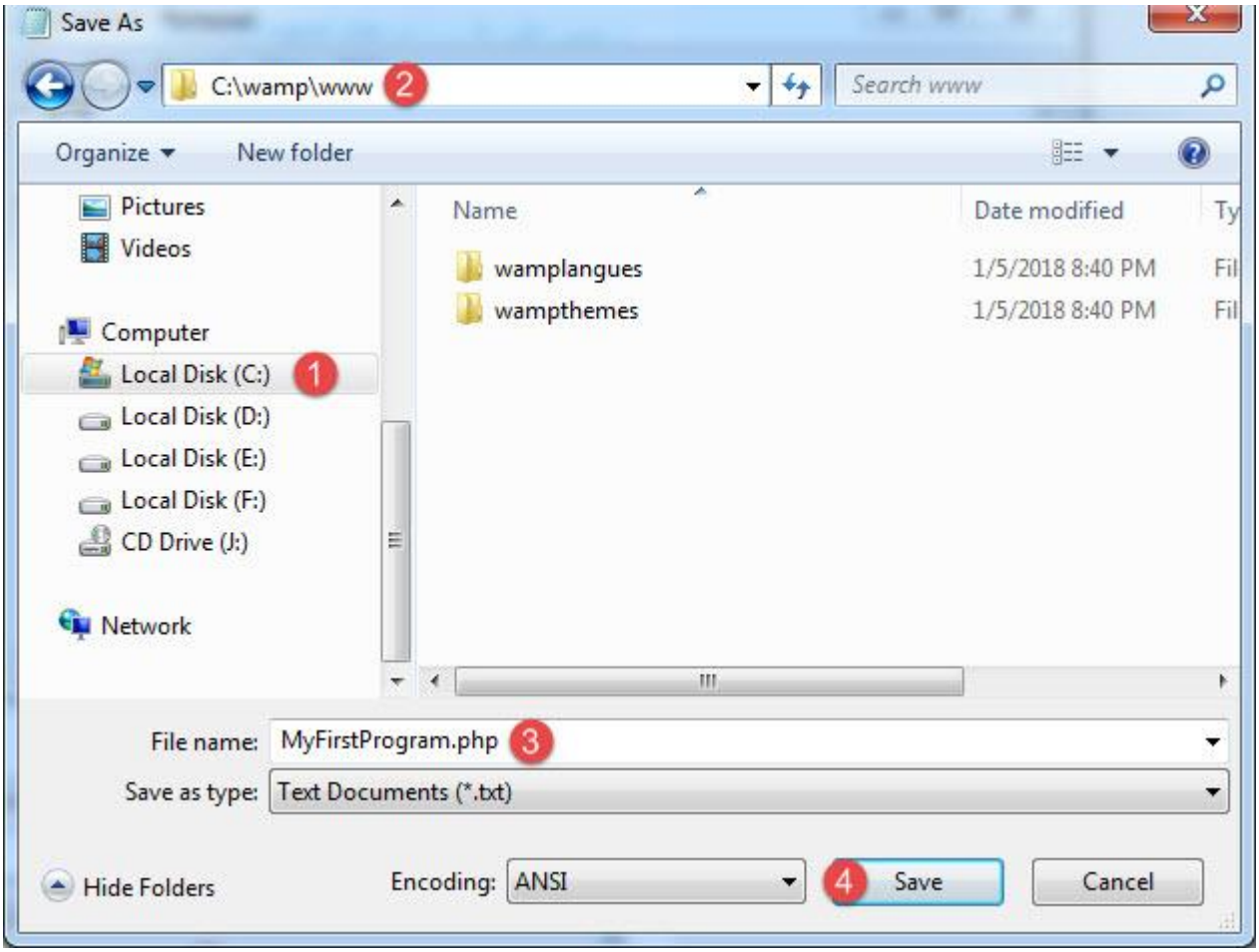
```
<?php
   echo "Welcome to PHP Tutorials!";
?>
```



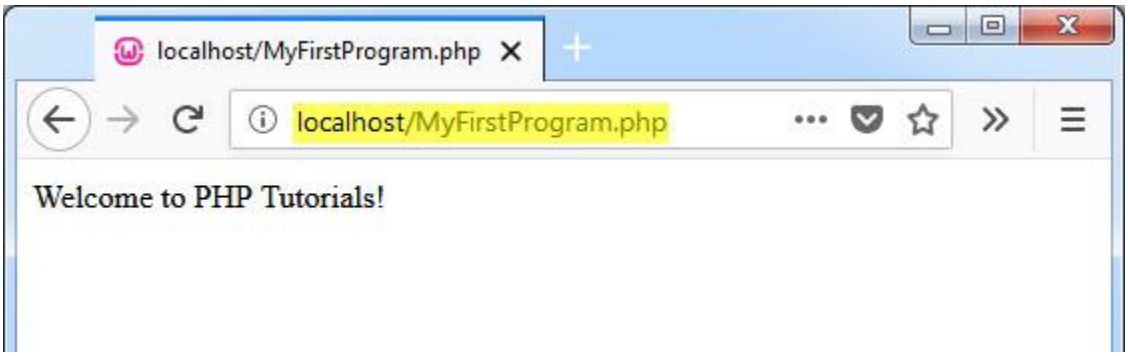
حال از منوی File بر روی گزینه Save as کلیک کنید:



و سپس فایل بالا را با نام MyFristProgram.php در مسیر C:\www\wamp به صورت زیر ذخیره کنید:



به این نکته توجه کنید که پسوند فایل حتماً باید به صورت php. باشد (عدد ۳ شکل بالا). حال نوبت به اجرای برنامه می‌رسد. برای اجرای کدهایی که نوشته‌ایم ابتدا مرورگر را باز کرده و سپس در نوار آدرس آن عبارت localhost/MyFirstProgram.php را نوشته و دکمه Enter را می‌زنیم :



همانطور که در شکل بالا مشاهده می‌کنید، جمله Welcome to PHP Tutorials! در مرورگر نمایش داده می‌شود که نشان دهنده اجرای صحیح کد است.

ساختار یک برنامه در PHP

مثال بالا ساده‌ترین برنامه‌ای است که شما می‌توانید در PHP بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است. هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. کدهای PHP در داخل تگ باز php?> و تگ بسته<? نوشته می‌شوند:

```
<?php
    // PHP code write here
?>
```

کدهایی که در داخل این دو تگ نوشته می‌شوند به سرور ارسال شده و پس از پردازش، خروجی آنها به مرورگر ارسال می‌شود. هر خط کد در PHP به یک سمیکالن (;) ختم می‌شود. اگر سمیکالن در آخر خط فراموش شود برنامه با خطا مواجه می‌شود. مثالی از یک خط کد در PHP به صورت زیر است:

```
echo "Welcome to PHP Tutorials!";
```

این خط کد پیغام Welcome to Visual PHP Tutorials! را در صفحه نمایش نشان می‌دهد. از کلمه echo برای چاپ یک رشته استفاده می‌شود. یک رشته گروهی از کاراکترها است، که به وسیله دابل کوتیشن (") محصور شده است. مانند: "Welcome to Visual PHP Tutorials!".

یک کاراکتر می‌تواند یک حرف، عدد، علامت یا ... باشد. PHP فضای خالی و خطوط جدید را نادیده می‌گیرد. بنابراین شما می‌توانید همه برنامه را در یک خط بنویسید. اما اینکار خواندن و اشکال زدایی برنامه را مشکل می‌کند. یکی از خطاهای معمول در برنامه نویسی فراموش کردن سمیکالن در پایان هر خط کد است. به مثال زیر توجه کنید:

```
echo
"Welcome to PHP Tutorials!";
```

PHP فضای خالی بالا را نادیده می‌گیرد و از کد بالا اشکال نمی‌گیرد. اما از کد زیر ایراد می‌گیرد:

```
echo;
"Welcome to PHP Tutorials!";
```

به سمیکالن آخر خط اول توجه کنید. برنامه با خطای نحوی مواجه می‌شود چون دو خط کد مربوط به یک برنامه هستند و شما فقط باید یک سمیکالن در آخر آن قرار دهید. همیشه به یاد داشته باشید که PHP به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال MAN و man در PHP با هم فرق دارند. رشته‌ها و توضیحات و کلمات کلیدی از این قاعده مستثنی هستند که در درسهای آینده توضیح خواهیم داد. مثلاً در کدهای زیر که کلمه کلیدی echo به صورت‌های مختلف نوشته شده است، ایجاد خطا نمی‌کنند:

```
<?php
    echo "Welcome to PHP Tutorials!";
    eCHo "Welcome to PHP Tutorials!";
    ECHO "Welcome to PHP Tutorials!";
?>
```

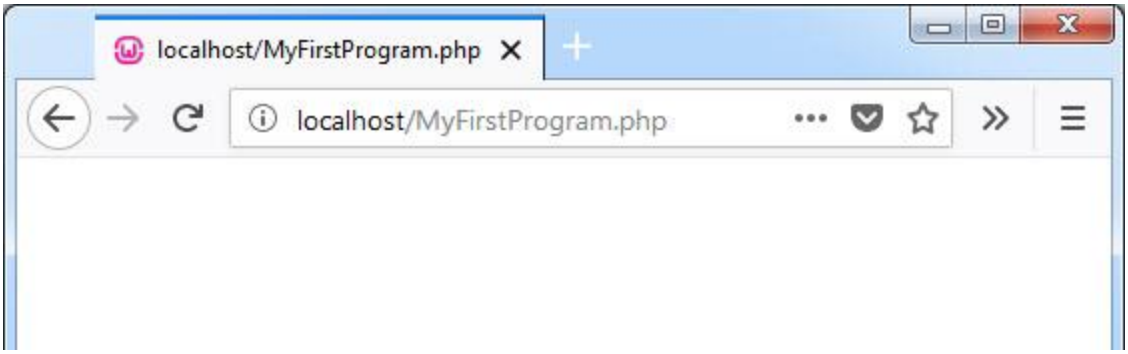
نکته دیگر اینکه، پسوند فایلهای PHP، باید به صورت php. باشد. حال که با خصوصیات و ساختار اولیه PHP آشنا شدید در درسهای آینده مطالب بیشتری از این زبان برنامه نویسی قدرتمند خواهید آموخت.

توضیحات

توضیحات در زبانهای برنامه نویسی بسیار مفید هستند. آن‌ها در به یادآوری وظایف کدها به شما کمک می‌کنند. توضیحات در PHP به سه صورت اعمال می‌شوند.

```
<?php

// single-line comment

# single-line comment

/*
multi-line
comment
*/

?>
```

استفاده از توضیحات در برنامه می‌تواند به شما و دیگران در فهم کدهایتان کمک کند. بدین صورت که در کارهای تیمی کسی که کدهای شما را می‌بیند با استفاده از توضیحاتی که در مورد کدها داده‌اید می‌فهمد که هر کد چه وظیفه‌ای دارد. دستورات بالا را در داخل فایل MyFirstProgram.php نوشته و ذخیره کنید. مشاهده می‌کنید که با اجرای آن دستورات بالا در مرورگر نمایش داده نمی‌شوند:



سایت های استاتیک و داینامیک

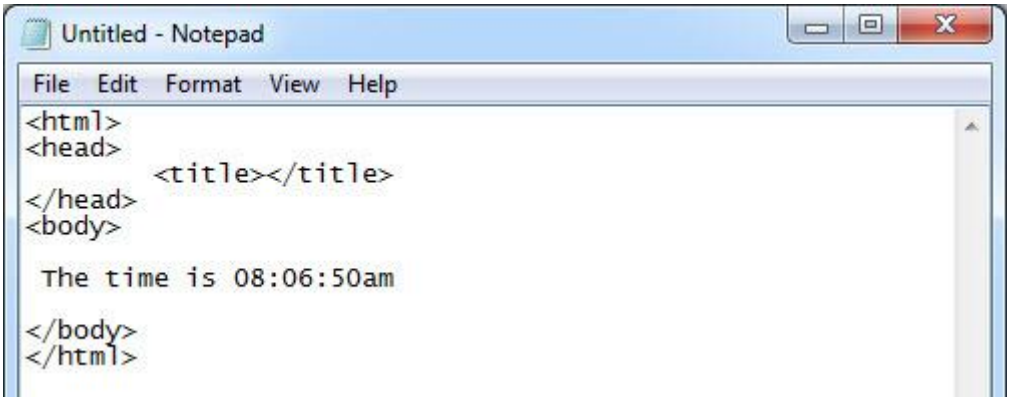
حال که با نحوه نوشتن، ذخیره و اجرای کدهای HTML و PHP آشنا شدید، بهتر است که شما را با دو نوع وب سایت استاتیک و داینامیک آشنا کنیم.

سایت های استاتیک یا ایستا به سایت های گفته می شود که محتوای آنها زیاد تغییر نمی کند. این نوع وب سایت ها با زبان نشانه گذاریHTML طراحی می شوند و برای انجام تغییرات در آنها باید یا مدیر سایت دانش برنامه نویسی داشته باشد و یا به طراح سایت بگوید که تغییرات را اعمال کند.

سایت های داینامیک به سایت های گفته می شود که محتوای آنها دائما در حال تغییر است. این نوع وب سایت ها دارای منوی مدیریت بوده و مدیر سایت برای تغییر در محتوای آنها می تواند بدون داشتن دانش برنامه نویسی، محتوای مورد نظر خود را در سایت قرار داده و یا ویرایش کند.

بهتر است که با یک مثال بسیار ساده این تفاوت را برای شما روشن کنیم. فرض کنید شما می خواهید در قسمتی از وب سایتتان، ساعت را نشان دهید. ابتدا این کار با با HTML انجام می دهیم. برنامه NotePad را باز کرده و کدهای زیر را در داخل آن بنویسید:

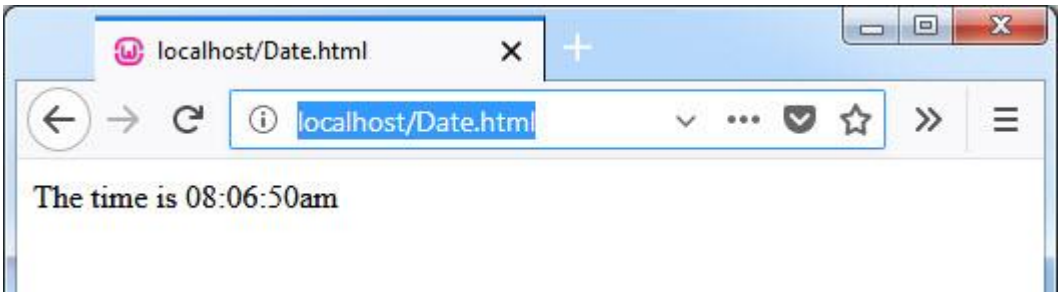
```
<html>
<head>
  <title></title>
</head>
<body>

 The time is 08:06:50am

</body>
</html>
```



حال از منوی File گزینه Save As را زده و فایل بالا را در پوشه www که در درس قبل توضیح دادیم، با نام Date.html ذخیره کنید. پسوند حتما html باشد، البته می توانید php هم بگذارید ولی شما از html استفاده کنید. حال مرورگر را باز کرده و آدرس زیر را در آن نوشته و دکمه Enter را بزنید:

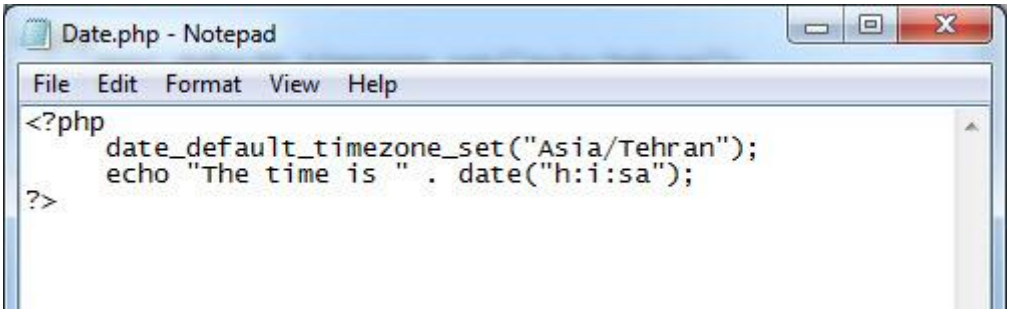
```
http://localhost/Date.html
```

با اجرای کد ساعت در سیستم نمایش داده می شود ولی همانی هست که شما در داخل کد بالا نوشته اید:



یعنی شما هر چند بار که دکمه Refresh مرورگر را بزنید، ساعت تغییر نمی کند .اگر شما بخواهید به جای ثانیه ۵۰، ثانیه ۵۱ نمایش داده شود باید فایل را باز کرده و کد را دستکاری کنید. یعنی به ازای هر تغییر در ساعت شما باید فایل را باز کرده و تغییرات را اعمال کنید و این معنای استاتیک و غیر داینامیک بودن سایت است. حال همین کار را با PHP انجام می دهیم. برنامه NotePad را باز کرده و کد زیر را در داخل آن بنویسید:

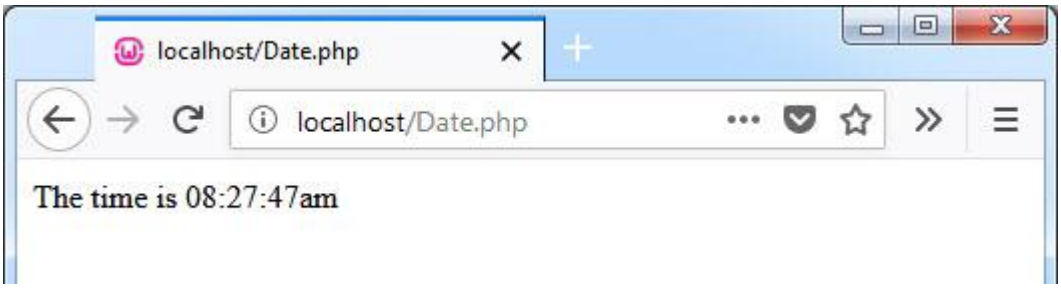
```php
<?php
     date_default_timezone_set("Asia/Tehran");
     echo "The time is " . date("h:i:sa");
?>
```



نگران کدهای نوشته شده در شکل بالا نباشید در آینده در مورد آنها توضیح می دهیم. حال همین فایل بالا را با نام Date.php در پوشه www ذخیره کنید. حتما حتما پسوند فایل باید php باشد. آدرس زیر را در مرورگر نوشته و دکمه Enter را بزنید:

```
http://localhost/Date.php
```

با اجرای کد بالا، ساعتی نمایش داده می شود که با ساعت سیستم شما یکی است:



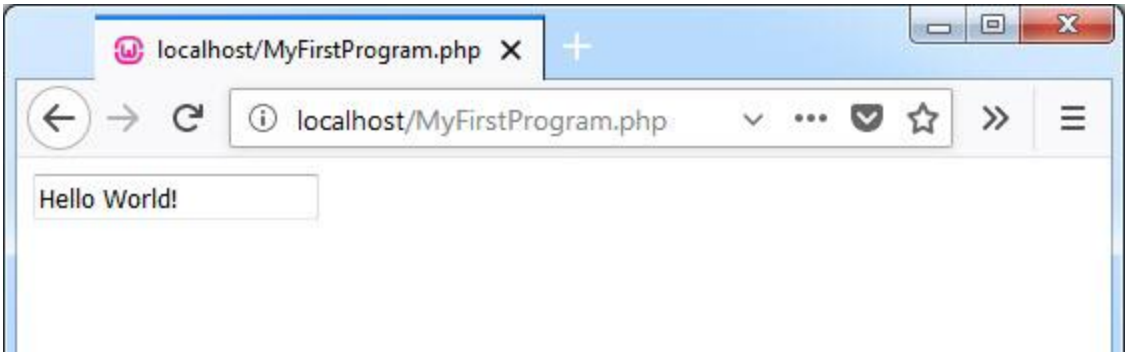
با هر بار زدن دکمه Refresh مرورگر و یا F5 صفحه کلید این ساعت تغییر می کند و لازم نیست که شما کد را دستکاری کنید، چون PHP این کار را به صورت خودکار برای شما انجام می دهد، و این معنای داینامیک یا پویا بودن سایت است. حال که با انواع سایت آشنا شدید، در درس بعد نحوه ادغام کدهای HTML و PHP را به شما آموزش می دهیم.

ادغام کدهای HTML و PHP

PHP بصورت Html embedded است و این بدین معناست که دستورات این زبان را می‌توان بین کدهای html نوشت. در یک فایل با پسوند .php می‌توان کدهای HTML و PHP را با هم ادغام کرد. یک مثال می‌زنیم. فرض کنید که می‌خواهید یک جعبه متن HTML را به وسیله دستور PHP ایجاد کنید برای این کار فایل MyFirstProgram.php ی، که در درس قبل ایجاد کردید را با ویرایشگر متن باز و کدهای قبلی آن را پاک کرده و سپس کد زیر را در داخل آن بنویسید:

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

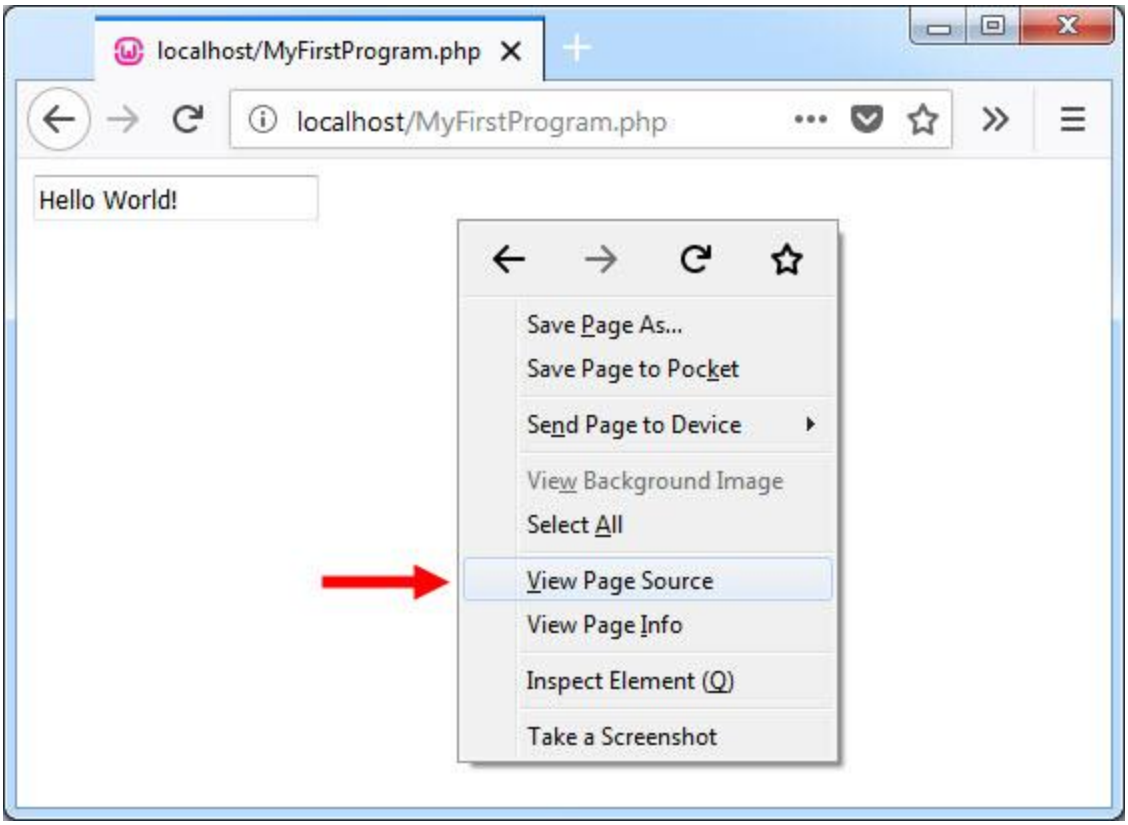
در کد بالا ما فقط با استفاده از یک تگ HTML یک جعبه متن را ایجاد کرده‌ایم. اگر برنامه را اجرا کنید، خروجی به صورت زیر خواهد بود:



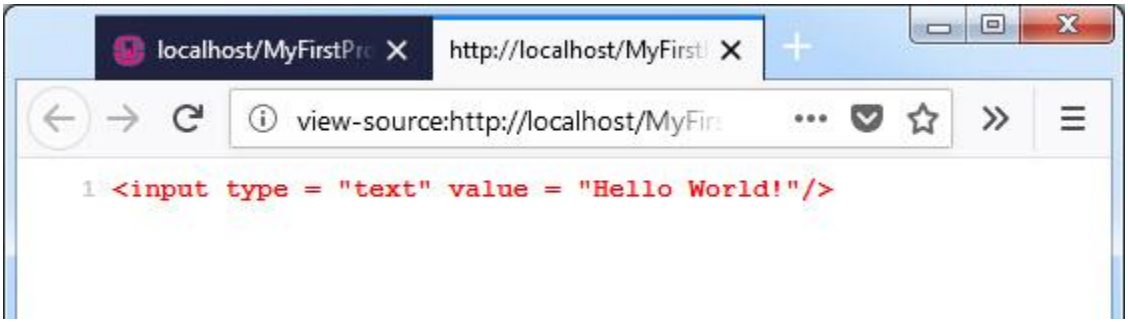
حال اگر بخواهیم همین جعبه متن را با کدهای PHP تولید کنیم، باید از دستور echo به صورت زیر استفاده شود:

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

با اجرای کد بالا مشاهده می‌کنید که خروجی همان است که در شکل بالا دیدیم. همانطور که در درس اول اشاره شده وب سرور یک فایل با پسوند HTML برای مرورگر می‌فرستد. چون فقط تگ‌های HTML برای مرورگر قابل تشخیص هستند. حال برای اطمینان از این گفته به سورس صفحه نگاه می‌کنیم. برای این منظور بر روی صفحه مرورگر راست کلیک کرده و گزینه View Page Source را انتخاب می‌کنیم:



با کلیک بر روی این گزینه، صفحه‌ای به صورت زیر نمایش داده می‌شود که همان کدهایی است که توسط وب سرور برای مرورگر ارسال شده‌اند. مشاهده می‌کنید که خبری از تگ‌های باز و بسته PHP نیست. چون این تگ‌ها فقط در داخل سرور پردازش شده و نتیجه به صورت HTML برای مرورگر ارسال می‌شود:



از کدهای PHP در داخل تگ‌های HTML هم می‌توان استفاده کرد. فرض کنید متن "Hello World" را یک بار با و بار دیگر بدون استفاده از PHP می‌خواهیم که در داخل جعبه متن بنویسیم:

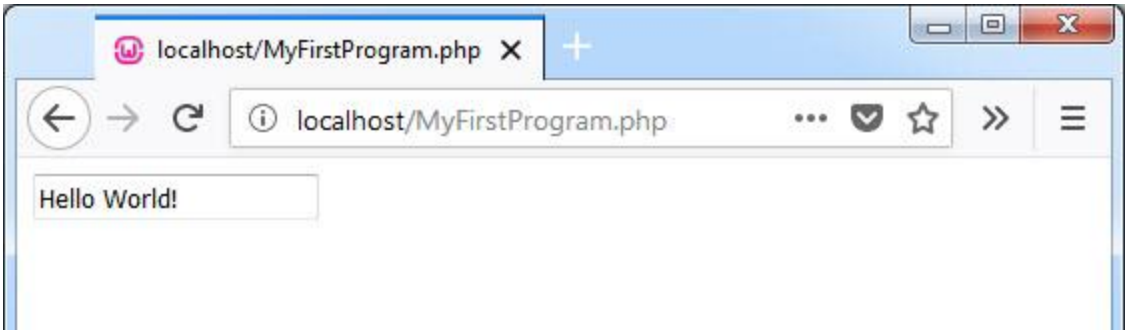
بدون استفاده از دستورPHP

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

با استفاده از دستورPHP

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

خروجی هر دو کد بالا به صورت زیر است:



کاراکترهای کنترلی

کاراکترهای کنترلی (escape characters) کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آنها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد. جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می‌دهد:

Escape	نتیجه
\n	حرکت به سطر بعد
\r	حرکت به ابتدای سطر جاری
\t	کاراکتر Tab (معادل ۸ کاراکترSpace)
\\	کاراکتر \
\’	کاراکتر ‘ (در رشته‌های محصور به گیومه تک)
\”	کاراکتر ” (در رشته‌های محصور به گیومه جفت)
\$	کاراکتر$
\[0-7]	کاراکتری که کد ASCII آن در مبنای ۸ در جلوی \ نوشته شده است
\x[0-F]	کاراکتری که کد ASCII آن در مبنای ۱۶ در جلوی \x نوشته شده است

ما برای استفاده از کاراکترهای کنترلی از بک اسلش (\) استفاده می‌کنیم. از آنجاییکه \ معنای خاصی به رشته‌ها می‌دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
<?php
   echo "We can print a \\ by using the \\\\ escape sequence.";
?>
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\، نشان دادن مسیر یک فایل در ویندوز است:

```php
<?php
    echo "C:\\Program Files\\Some Directory\\SomeFile.txt";
?>
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (") برای نشان دادن رشته‌ها استفاده می‌کنیم برای چاپ آن از \" استفاده می‌کنیم:

```php
<?php
    echo "I said, \"Motivate yourself!\".";
?>
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (') از \' استفاده می‌کنیم:

```php
<?php
    echo "The programmer\'s heaven.";
?>
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می‌شود:

```php
<?php
    echo "Left\tRight";
?>
```

```
Left    Right
```

برای مشاهده لیست مقادیر مبنای ۱۶ برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید:

```
http://www.ascii.cl/htmlcodes.htm
```

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می‌دهد. بیشترین خطا زمانی اتفاق می‌افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می‌کند.

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید. می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید. محتویات این ظرف می‌تواند پاک شود یا تغییر کند. هر متغیر دارای یک نام نیز هست، که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد. همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد. مقدار متغیر می‌تواند تهی نیز باشد.

متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده‌ای که در آن ذخیره می‌شود یکی است. متغیر دارای عمر نیز هست که از روی آن می‌توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد. و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می‌گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است. ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می‌کنیم. هنگامی که یک برنامه ایجاد می‌کنیم احتیاج به یک مکان برای ذخیره داده، مقادیر یا داده‌هایی که توسط کاربر وارد می‌شوند داریم. این مکان

همان متغیر است. برای این از کلمه متغیر استفاده می‌شود چون ما می‌توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می‌گیرند که برنامه در حال اجراست و وقتی شما برنامه را می‌بندید محتویات متغیرها نیز پاک می‌شود. قبلاً ذکر شد که به وسیله نام متغیر می‌توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

نامگذاری متغیرها

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

- در ابتدای نام متغیرها علامت $ می آید.
- نام متغیرها به حروف بزرگ و کوچک حساس است. یعنی متغیری با نام Myvar با متغیر myVar فرق دارد.
- نام متغیر می‌تواند شامل علامت زیر خط و عدد باشد ولی نمی‌تواند با عدد شروع شود.
- نام متغیر نمی‌تواند دارای فضای خالی و یا کلماتی باشد که برای PHP دارای معنی خاصی هستند.

نامهای مجاز:

```
$num1  $myNumber  $studentCount  $total       $first_name    $_minimum
$num2  $myChar    $average       $amountDue   $last_name     $_maximum
$name  $counter   $sum           $isLeapYear  $color_of_car  _age
```

نامهای غیر مجاز:

```
$123        $#numbers#  $#ofstudents  $1abc2
$123abc     $$money     $first name   $ty.np
$my number  $this&that  $last name    $1:00
```

اگر به نامهای مجاز در مثال بالا توجه کنید متوجه قراردادهای به کار رفته در نامگذاری آن‌ها خواهید شد. یکی از روش‌های نامگذاری، نامگذاری کوهان شتری است. در این روش که برای متغیرهای دو کلمه‌ای به کار می‌رود، اولین حرف اولین کلمه با حرف کوچک و اولین حرف دومین کلمه با حرف بزرگ نمایش داده می‌شود مانند : $myNumber. توجه کنید که اولین حرف کلمه Number با حرف بزرگ شروع شده است. مثال دیگر کلمه $numberOfStudents است. اگر توجه کنید بعد از اولین کلمه حرف اول سایر کلمات با حروف بزرگ نمایش داده شده است.

انواع داده

انواع داده در PHP شامل اعداد، کاراکترها و رشته‌ها و مقادیر بولی می‌باشند و دارای مجموعه مشخصی از مقادیر هستند و محدوده خاصی از اعداد را در خود ذخیره می‌کنند. در جدول زیر نه نوع داده‌ای 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 می‌شوند.

bool	مقدار داده‌های Boolean می‌تواند TRUE یا FALSE باشد.
string	رشته، یک توالی از کاراکترهاست
array	آرایه‌ها انواع خاصی از متغیرها به حساب می‌آیند که می‌توانند چندین داده را در قالب یک نام ذخیره کنند.
object	یک شیء نوع داده‌ای است که هم داده‌ها و هم اطلاعات مربوط به نحوه پردازش آنها را ذخیره می‌کند.
resource	مرجعی به یک منبع خارجی مثل DB می‌باشد.
callable	متدها و توابع
null	با استفاده از مقدار NULL، می‌توان نشان داد که یک متغیر مقدار ندارد.

استفاده از متغیرها

در مثال زیر نحوه تعریف و مقدار دهی متغیرها نمایش داده شده است :

```php
<?php
    //Declare and Assign values to variables
    $num1 = 1;
    $num2 = 2;
    $num3 = 3.54;
    $num4 = 4.12;
    $boolVal = true;
    $myChar = 'R';
    $message = "Hello World!";

    //Show the values of the variables
    echo "num1       =  " . $num1    .'<br/>';
    echo "num2       =  " . $num2    .'<br/>';
    echo "num3       =  " . $num3    .'<br/>';
    echo "num4       =  " . $num4    .'<br/>';
    echo "boolVal    =  " . $boolVal .'<br/>';
    echo "myChar     =  " . $myChar  .'<br/>';
    echo "message    =  " . $message .'<br/>';
?>
```

به یک نکته خیلی مهم توجه کنید:

PHP یک زبان با نوع داده‌ای 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 می‌شود که به آن اختصاص داده شده است. در کد بالا متغیر num2 به نوع صحیح و متغیر num4 به نوع اعشاری تبدیل می‌شود. چون مقادیری که به آنها اختصاص داده شده است از نوع اعداد صحیح و اعداد اعشاری می‌باشند. پس نوع متغیر از همان نوعی است که به آن اختصاص داده شده است .

تعریف متغیر

در خطوط ۹-۳ متغیرهایی با نام‌های متفاوت تعریف شده‌اند. برای تعریف یک متغیر در PHP ابتدا علامت $ و سپس یک نام برای آن در نظر می‌گیریم و در آخر سیمیکولن بگذاریم، مثلاً:

```
$number;
```

مقداردهی متغیرها

می‌توان فوراً بعد از تعریف متغیرها مقادیری را به آنها اختصاص داد (خطوط ۹-۳). این عمل را مقداردهی می‌نامند. در زیر نحوه مقدار دهی متغیرها نشان داده شده است:

```
$number = 7;
```

تعریف متغیر با مقدار دهی متغیرها متفاوت است. تعریف متغیر یعنی انتخاب نام برای متغیر ولی مقدار دهی یعنی اختصاص یک مقدار به متغیر.

محدوده متغیر

محدوده یک متغیر مشخص می‌کند که متغیر در کجای کد قابل دسترسی است. محدوده متغیرها انواعی دارد که در درس‌های بعدی با آنها آشنا می‌شوید. تشخیص محدوده متغیر بسیار مهم است چون به وسیله آن می‌فهمید که در کجای کد می‌توان از متغیر استفاده کرد. باید یاد آور شد اگر دو متغیر در یک محدوده نام یکسان باشند، دومین مقدار برای متغیر لحاظ می‌شود:

```
$num1 = 5;
$num1 = 10;
```

مثلاً در مثال بالا اگر مقدار num1 را چاپ کنید، عدد ۱۰ به شما نمایش داده می‌شود. از آنجاییکه PHP به بزرگی و کوچک بودن حروف حساس است می‌توان از این خاصیت برای تعریف چند متغیر هم نام ولی با حروف متفاوت (از لحاظ بزرگی و کوچکی) برای تعریف چند متغیر از یک نوع استفاده کرد مانند:

```
$num1;
$Num1;
$NUM1;
```

ثابت‌ها

ثابت‌ها انواعی از متغیرها هستند که مقدار آنها در طول برنامه تغییر نمی‌کند. ثابت‌ها حتماً باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می‌آید. بعد از این که به ثابت‌ها مقدار اولیه اختصاص داده شد، هرگز در زمان اجرای برنامه نمی‌توان آن را تغییر داد.

تعریف ثابت با کلمه کلیدی const و تابع define

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

```
const CONSTNAME= initial_value;
```

یا

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

مثال:

```php
<?php

    const NUMBER = 10;   //with const keyword

    define('PI', 3.14); // with define function

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

?>
```

```
10
3.14
```

همانطور که در مثال بالا مشاهده می‌کنید قبل از نام ثابت‌ها نباید از علامت $ و برای فراخوانی آنها در دستور echo نمی‌بایست از علامت " " استفاده کنیم. مقدار دادن به یک ثابت، که قبلاً مقدار دهی شده برنامه را با خطا مواجه می‌کند:

```php
<?php

    const NUMBER = 10;

    NUMBER = 12;

    echo NUMBER;

?>
```

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

نکته‌ی 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 سؤال برایتان پیش آمده باشد که دلیل استفاده از ثابت‌ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی‌کنند بهتر است که آنها را به صورت ثابت تعریف کنید. این کار هر چند کوچک کیفیت برنامه شما را بالا می‌برد.

عبارات و عملگرها

ابتدا با دو کلمه آشنا شوید:

- عملگر: نمادهایی هستند که اعمال خاص انجام می‌دهند.
- عملوند: مقادیری که عملگرها بر روی آنها عملی انجام می‌دهند.

مثلاً X+Y یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می‌آیند. زبانهای برنامه نویسی جدید دارای عملگرهایی هستند که از اجزاء معمول زبان به حساب می‌آیند. PHP دارای عملگرهای مختلفی از جمله عملگرهای ریاضی، تخصیصی، مقایسه‌ای، منطقی و بیتی می‌باشد. از عملگرهای ساده ریاضی می‌توان به عملگر جمع و تفریق اشاره کرد. سه نوع عملگر در PHP وجود دارد:

- یگانی - (Unary) به یک عملوند نیاز دارد.
- دودویی (- (Binary به دو عملوند نیاز دارد.
- سه تایی - (Ternary) به سه عملوند نیاز دارد.

انواع مختلف عملگر که در این بخش مورد بحث قرار می‌گیرند عبارت‌اند از:

- عملگرهای ریاضی
- عملگرهای تخصیصی
- عملگرهای مقایسه‌ای
- عملگرهای منطقی
- عملگرهای بیتی
- عملگر رشته

عملگرهای ریاضی

PHP از عملگرهای ریاضی برای انجام محاسبات استفاده می‌کند. جدول زیر عملگرهای ریاضی PHP را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
+	Binary	var1 = var2 + var3;	Var1 برابر است با حاصل جمع var2 و var3
-	Binary	var1 = var2 - var3;	Var1 برابر است با حاصل تفریق var2 و var3
*	Binary	var1 = var2 * var3;	Var1 برابر است با حاصلضرب var2 در var3
/	Binary	var1 = var2 / var3;	Var1 برابر است با حاصل تقسیم var2 بر var3
%	Binary	var1 = var2 % var3;	Var1 برابر است با باقیمانده تقسیم var2 و var3
+	Unary	var1 = +var2;	Var1 برابر است با مقدار var2
-	Unary	var1 = -var2;	Var1 برابر است با مقدار var2 ضربدر ۱-

مثال بالا در از نوع عددی استفاده شده است. اما استفاده از عملگرهای ریاضی برای نوع رشته‌ای نتیجه متفاوتی دارد. همچنین در جمع دو کاراکتر کامپایلر معادل عددی آنها را نشان می‌دهد. اگر از عملگر + برای رشته‌ها استفاده کنیم دو رشته را با هم ترکیب کرده و به هم می‌چسباند. دیگر عملگرهای PHP عملگرهای کاهش و افزایش هستند. این عملگرها مقدار ۱ را از متغیرها کم یا به آنها اضافه می‌کنند. از این متغیرها اغلب در حلقه‌ها استفاده می‌شود:

عملگر	دسته	مثال	نتیجه

++	Unary	var1 = ++var2;	مقدار var1 برابر است با var2 بعلاوه ۱
--	Unary	var1 = --var2;	مقدار var1 برابر است با var2 منهای ۱
++	Unary	var1 = var2++;	مقدار var1 برابر است با var2. به متغیر var2 یک واحد اضافه می‌شود.
--	Unary	var1 = var2--;	مقدار var1 برابر است با var2. از متغیر var2 یک واحد کم می‌شود.

به این نکته توجه داشته باشید که محل قرار گیری عملگر در نتیجه محاسبات تأثیر دارد. اگر عملگر قبل از متغیر var2 بیاید افزایش یا کاهش var1 اتفاق می‌افتد. چنانچه عملگرها بعد از متغیر var2 قرار بگیرند ابتدا var1 برابر var2 می‌شود و سپس متغیر var2 افزایش یا کاهش می‌یابد. به مثال‌های زیر توجه کنید:

```php
<?php
    $x = 0;
    $y = 1;

    $x = ++$y;

    echo('x = ' . $x).'<br/>';
    echo('y = ' . $y);
?>
```

```
x=2
y=2
```

```php
<?php
    $x = 0;
    $y = 1;

    $x = --$y;

    echo('x = ' . $x).'<br/>';
    echo('y = ' . $y);
?>
```

```
x=0
y=0
```

همانطور که در دو مثال بالا مشاهده می‌کنید، درج عملگرهای -- و ++ قبل از عملوند y باعث می‌شود که ابتدا یک واحد از y کم و یا یک واحد به y اضافه شود و سپس نتیجه در عملوند x قرار بگیرد. حال به دو مثال زیر توجه کنید:

```php
<?php
    $x = 0;
    $y = 1;

    $x = $y--;

    echo('x = ' . $x).'<br/>';
```

```
    echo('y = ' . $y);
?>
```

```
x=1
y=0
```

```
<?php
    $x = 0;
    $y = 1;

    $x = $y++;

    echo('x = ' . $x).'<br/>';
    echo('y = ' . $y);
?>
```

```
x=1
y=2
```

همانطور که در دو مثال بالا مشاهده می‌کنید، درج عملگرهای -- و ++ بعد از عملوند y باعث می‌شود که ابتدا مقدار y در داخل متغیر x قرار بگیرد و سپس یک واحد از y کم و یا یک واحد به ان اضافه شود.

عملگرهای تخصیصی

نوع دیگر از عملگرهای PHP عملگرهای جایگزینی نام دارند. این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می‌دهند. جدول زیر انواع عملگرهای تخصیصی در PHP را نشان می‌دهد:

عملگر	مثال	نتیجه
=	var1 = var2;	مقدار var1 برابر است با مقدار var2
+=	var1 += var2;	مقدار var1 برابر است با حاصل جمع var1 و var2
-=	var1 -= var2;	مقدار var1 برابر است با حاصل تفریق var1 و var2
*=	var1 *= var2;	مقدار var1 برابر است با حاصل ضرب var1 در var2
/=	var1 /= var2;	مقدار var1 برابر است با حاصل تقسیم var1 بر var2
%=	var1 %= var2;	مقدار var1 برابر است با باقیمانده تقسیم var1 بر var2

استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلاً دو کد زیر معادل هم هستند:

```
var1 += var2
var1  = var1 + var2
```

این حالت کدنویسی زمانی کارایی خود را نشان می‌دهد که نام متغیرها طولانی باشد. برنامه زیر چگونگی استفاده از عملگرهای تخصیصی و تأثیر آنها را بر متغیرها نشان می‌دهد.

```php
<?php
    $number = 10;

    echo 'Number = '. $number . '<br/>';

    $number += 10;
    echo 'Number = '. $number . '<br/>';

    $number -= 10;
    echo 'Number = '. $number . '<br/>';
?>
```

```
Number = 10
Number = 20
Number = 10
```

در برنامه از ۳ عملگر تخصیصی استفاده شده است. ابتدا یک متغیر و مقدار ۱۰ با استفاده از عملگر = به آن اختصاص داده شده است. سپس به آن با استفاده از عملگر =+ مقدار ۱۰ اضافه شده است. و در آخر به وسیله عملگر =- عدد ۱۰ از آن کم شده است.

عملگرهای مقایسه‌ای

از عملگرهای مقایسه‌ای برای مقایسه مقادیر استفاده می‌شود. نتیجه این مقادیر یک مقدار بولی (منطقی) است. این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار ۱ یا true و اگر نتیجه مقایسه اشتباه باشد مقدار تهی یا false را نشان می‌دهند. این عملگرها به طور معمول در دستورات شرطی به کار می‌روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می‌شوند. جدول زیر عملگرهای مقایسه‌ای در PHP را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
==	Binary	var1 = var2 == var3	var1 در صورتی true است که مقدار var2 با مقدارvar3 برابر باشد در غیر اینصورت false است.
!=	Binary	var1 = var2 != var3	var1 در صورتی true است که مقدار var2 با مقدارvar3 برابر نباشد در غیر اینصورت false است.
===	Binary	var1 = var2=== var3	var1 در صورتی true است که var2 با var3 هم از لحاظ مقدار و هم از لحاظ نوع یکی باشند، در غیر اینصورت false است.
!==	Binary	var1 = var2 !== var3	var1 در صورتی true است که var2 با var3 از لحاظ مقدار و نوع یکی نباشند، در غیر اینصورت false است.
<	Binary	var1 = var2 < var3	var1 در صورتی true است که مقدار var2 کوچک‌تر از var3 مقدار باشد در غیر اینصورت false است.
>	Binary	var1 = var2 > var3	var1 در صورتی true است که مقدار var2 بزرگ‌تر ازمقدار var3 باشد در غیر اینصورت false است.

<=	Binary	var1 = var2 <= var3	var1 در صورتی true است که مقدار var2 کوچک‌تر یا مساوی مقدار var3 باشد در غیر اینصورت false است.
>=	Binary	var1 = var2 >= var3	var1 در صورتی true است که مقدار var2 بزرگ‌تر یا مساوی var3 مقدار باشد در غیر اینصورت false است.

برنامه زیر نحوه عملکرد این عملگرها را نشان می‌دهد:

```php
<?php
    $firstnumber  = 10;
    $secondnumber = 5;

    echo ($firstnumber  == $secondnumber)  . '<br/>';
    echo ($firstnumber  != $secondnumber)  . '<br/>';
    echo ($firstnumber === $secondnumber)  . '<br/>';
    echo ($firstnumber !== $secondnumber)  . '<br/>';
    echo ($firstnumber   < $secondnumber)  . '<br/>';
    echo ($firstnumber   > $secondnumber)  . '<br/>';
    echo ($firstnumber  <= $secondnumber)  . '<br/>';
    echo ($firstnumber  >= $secondnumber)  . '<br/>';
?>
```

```
1

1

1

1
```

درمثال بالا ابتدا دو متغیر را که می‌خواهیم با هم مقایسه کنیم را ایجاد کرده و به آنها مقادیری اختصاص می‌دهیم. سپس با استفاده از یک عملگر مقایسه‌ای آنها را با هم مقایسه کرده و نتیجه را چاپ می‌کنیم. به این نکته توجه کنید که هنگام مقایسه دو متغیر باید از عملگر == به جای عملگر = استفاده شود. عملگر = عملگر تخصیصی است و در عبارتی مانندy = x مقدار y را در به x اختصاص می‌دهد. عملگر == عملگر مقایسه‌ای است که دو مقدار را با هم مقایسه می‌کند مانند x==y و اینطور خوانده می‌شود x برابر است با y.

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می‌کنند و نتیجه آنها نیز یک مقدار بولی است. از این عملگرها اغلب برای شرطهای پیچیده استفاده می‌شود. همانطور که قبلاً یاد گرفتید مقادیر بولی می‌توانند false یا true باشند. فرض کنید که var2 و var3 دو مقدار بولی هستند.

عملگر	مثال	توضیحات
&&	var1 = var2 && var3;	در صورتی مقدار var1 برابر true است که هم var2 و هم var3 مقدارشان true باشد.
and	var1 = var2 and var3;	در صورتی مقدار var1 برابر true است که هم var2 و هم var3 مقدارشان true باشد.

||	var1 = var2 || var3;	در صورتی مقدار var1 برابر true است که var2 یا var3 مقدارشان true باشد.
or	var1 = var2 or var3;	در صورتی مقدار var1 برابر true است که var2 یا var3 مقدارشان true باشد.
xor	var1 = var2 xor var3;	در صورتی مقدار var1 برابر true است که var2 یا var3 (نه هر دو) مقدارشان true باشد.
!	var1 = !var1;	در صورتی مقدار var1 برابر true است که var1 سمت راست مقدارش false باشد.

عملگر منطقی and یا (&&)

اگر مقادیر دو طرف عملگر and، true باشند عملگر and مقدار true را بر می‌گرداند. در غیر اینصورت اگر یکی از مقادیر یا هر دوی آنها false باشند مقدار false را بر می‌گرداند. در زیر جدول درستی عملگر and نشان داده شده است:

X && Y	Y	X
true	true	true
false	false	true
false	true	false
false	false	false

برای درک بهتر تأثیر عملگر and یاد آوری می‌کنم که این عملگر فقط در صورتی مقدار true را نشان می‌دهد که هر دو عملوند مقدارشان true باشد. در غیر اینصورت نتیجه تمام ترکیب‌های بعدی false خواهد شد. استفاده از عملگر and مانند استفاده از عملگرهای مقایسه‌ای است. به عنوان مثال نتیجه عبارت زیر درست (true) است اگر سن (age) بزرگ‌تر از ۱۸ و salary کوچک‌تر از ۱۰۰۰ باشد.

```
result = (age > 18) && (salary < 1000);
```

عملگر and زمانی کارامد است که ما با محدود خاصی از اعداد سر و کار داریم. مثلاً عبارت 10 <= x <= 100 بدین معنی است که x می‌تواند مقداری شامل اعداد ۱۰ تا ۱۰۰ را بگیرد. حال برای انتخاب اعداد خارج از این محدوده می‌توان از عملگر منطقی and به صورت زیر استفاده کرد.

```
inRange = (number <= 10) && (number >= 100);
```

عملگر منطقی or یا (||)

اگر یکی یا هر دو مقدار دو طرف عملگر or، درست (true) باشد، عملگر or مقدار true را بر می‌گرداند. جدول درستی عملگر or در زیر نشان داده شده است:

X || Y	Y	X
true	true	true
true	false	true
true	true	false

false	false	false

در جدول بالا مشاهده می‌کنید که عملگر or در صورتی مقدار false را بر می‌گرداند که مقادیر دو طرف آن false باشند. کد زیر را در نظر بگیرید. نتیجه این کد در صورتی درست (true) است که رتبه نهایی دانش آموز (finalGrade) بزرگ‌تر از ۷۵ یا یا نمره نهایی امتحان آن ۱۰۰ باشد.

```
isPassed = (finalGrade >= 75) || (finalExam == 100);
```

عملگر منطقی not یا (!)

برخلاف دو اپراتور or و and عملگر منطقی not یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. این عملگر یک مقدار یا اصطلاح بولی را نفی می‌کند. مثلاً اگر عبارت یا مقدار true باشد آنرا false و اگر false باشد آنراtrue می‌کند. جدول زیر عملکرد اپراتور not را نشان می‌دهد:

!X	X
false	true
true	false

نتیجه کد زیر در صورتی درست است که age (سن) بزرگ‌تر یا مساوی ۱۸ نباشد.

```
isMinor = !(age >= 18);
```

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می‌دهند که شکل باینری انواع داده‌ها را دستکاری کنید. برای درک بهتر این درس توصیه می‌شود که از لینک زیر شما سیستم باینری و نحوه تبدیل اعداد دهدهی به باینری را یاد بگیرید:

```
http://www.w3-farsi.com/?p=5698
```

در سیستم باینری (دودویی) که کامپیوتر از آن استفاده می‌کند وضعیت هر چیز یا خاموش است یا روشن. برای نشان دادن حالت روشن از عدد ۱ و برای نشان دادن حالت خاموش از عدد ۰ استفاده می‌شود. بنابراین اعداد باینری فقط می‌توانند صفر یا یک باشند. اعداد باینری را اعداد در مبنای ۲ و اعداد اعشاری را اعداد در مبنای ۱۰ می گویند. یک بیت نشان دهنده یک رقم باینری است و هر بایت نشان دهنده ۸ بیت است. به عنوان مثال برای یک داده از نوع int به ۳۲ بیت یا ۸ بایت فضا برای ذخیره آن نیاز داریم، این بدین معناست که اعداد از ۳۲ رقم ۰ و ۱ برای ذخیره استفاده می‌کنند. برای مثال عدد ۱۰۰ وقتی به عنوان یک متغیر از نوع int ذخیره می‌شود در کامپیوتر به صورت زیر خوانده می‌شود:

```
00000000000000000000000001100100
```

عدد ۱۰۰ در مبنای ده معادل عدد ۱۱۰۰۱۰۰ در مبنای ۲ است. در اینجا ۷ رقم سمت راست نشان دهنده عدد ۱۰۰ در مبنای ۲ است و مابقی صفرهای سمت راست برای پر کردن بیت‌هایی است که عدد از نوع int نیاز دارد. به این نکته توجه کنید که اعداد باینری از سمت راست به چپ خوانده می‌شوند. عملگرهای بیتی PHP در جدول زیر نشان داده شده‌اند:

عملگر	نام	دسته	مثال
&	بیتی AND	Binary	x = y & z;
|	بیتی OR	Binary	x = y | z;
^	بیتی XOR	Binary	x = y ^ z;
~	بیتی NOT	Unary	x = ~y;

عملگر بیتی (&)AND

عملگر بیتی AND کاری شبیه عملگر منطقی AND انجام می‌دهد با این تفاوت که این عملگر بر روی بیت‌ها کار می‌کند. اگر مقادیر دو طرف آن ۱ باشد مقدار ۱ را بر می‌گرداند و اگر یکی یا هر دو طرف آن صفر باشد، مقدار صفر را بر می‌گرداند. جدول درستی عمگر بیتی AND در زیر آمده است:

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

در زیر نحوه استفاده از عملگر بیتی AND آمده است:

```
$result = 5 & 3;

echo $result;
```

```
1
```

همانطور که در مثال بالا مشاهده می‌کنید نتیجه عملکرد عملگر AND بر روی دو مقدار ۵ و ۳ عدد یک می‌شود. اجازه بدهید ببینیم که چطور این نتیجه را به دست می‌آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
---------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد ۵ و ۳ به معادل باینری‌شان تبدیل می‌شوند. از آنجاییکه هر عدد صحیح ۳۲ بیت است از صفر برای پر کردن بیت‌های خالی استفاده می‌کنیم. با استفاده از جدول درستی عملگر بیتی AND می‌توان فهمید که چرا نتیجه عدد یک می‌شود.

عملگر بیتی (|)OR

اگر مقادیر دو طرف عملگر بیتی OR هر دو صفر باشند نتیجه صفر در غیر اینصورت ۱ خواهد شد. جدول درستی این عملگر در زیر آمده است:

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی OR در صورتی صفر است که عملوندهای دو طرف آن صفر باشند. اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد.

به مثال زیر توجه کنید:

```
$result = 7 | 9;

echo $result;
```

```
15
```

وقتی که از عملگر بیتی OR برای دو مقدار در مثال بالا (۷ و ۹) استفاده می‌کنیم نتیجه ۱۵ می‌شود. حال بررسی می‌کنیم که چرا این نتیجه به دست آمده است؟

```
7: 00000000000000000000000000000111
9: 00000000000000000000000000001001
-----------------------------------
15: 00000000000000000000000000001111
```

با استفاده از جدول درستی عملگر بیتی OR می‌توان نتیجه استفاده از این عملگر را تشخیص داد. عدد ۱۱۱۱ باینری معادل عدد ۱۵ صحیح است.

عملگر بیتی XOR (^)

جدول درستی این عملگر در زیر آمده است:

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند، نتیجه ۰، در غیر اینصورت نتیجه ۱ می‌شود. در مثال زیر تأثیر عملگر بیتی XOR را بر روی دو مقدار مشاهده می‌کنید:

```
$result = 5 ^ 7;

echo $result;
```

```
2
```

در زیر معادل باینری اعداد بالا (۵ و ۷) نشان داده شده است.

```
5: 00000000000000000000000000000101
7: 00000000000000000000000000000111
-----------------------------------
2: 00000000000000000000000000000010
```

با نگاه کردن به جدول درستی عملگر بیتی XOR، می‌توان فهمید که چرا نتیجه عدد ۲ می‌شود.

عملگر بیتی NOT (~)

این عملگر یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. در زیر جدول درستی این عملگر آمده است:

X	NOT X
1	0
0	1

عملگر بیتی NOT مقادیر بیت‌ها را معکوس می‌کند. در زیر چگونگی استفاده از این عملگر آمده است:

```
$result = ~7;

echo $result;
```

به نمایش باینری مثال بالا که در زیر نشان داده شده است توجه نمایید:

```
7: 00000000000000000000000000000111
------------------------------------
-8: 11111111111111111111111111111000
```

عملگر بیتی تغییر مکان (shift)

این نوع عملگرها به شما اجازه می‌دهند که بیت‌ها را به سمت چپ یا راست جا به جا کنید. دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می‌کنند. عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جایی بیت‌ها را نشان می‌دهد.

مثال	دسته	نام	عملگر
x = y << 2;	Binary	تغییر مکان به سمت چپ	<<
x = y >> 2;	Binary	تغییر مکان به سمت راست	>>

عملگر تغییر مکان به سمت چپ

این عملگر بیت‌های عملوند سمت چپ را به تعداد n مکان مشخص شده توسط عملوند سمت راست، به سمت چپ منتقل می‌کند. به عنوان مثال:

```
$result = 10 << 2;

echo $result;
```

```
40
```

در مثال بالا ما بیت‌های مقدار ۱۰ را دو مکان به سمت چپ منتقل کرده‌ایم، حال بیایید تأثیر این انتقال را بررسی کنیم:

```
10: 00000000000000000000000000001010
------------------------------------
40: 00000000000000000000000000101000
```

مشاهده می‌کنید که همه بیت‌ها به اندازه دو واحد به سمت چپ منتقل شده‌اند. در این انتقال دو صفر از صفرهای سمت چپ کم می‌شود و در عوض دو صفر به سمت راست اضافه می‌شود.

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت‌ها را به سمت راست جا به جا می‌کند. به عنوان مثال:

```php
$result = 100 >> 4;

echo $result;
```

```
6
```

با استفاده از عملگر تغییر مکان به سمت راست بیت‌های مقدار ۱۰۰ را به اندازه ۴ واحد به سمت چپ جا به جا می‌کنیم. اجازه بدهید تأثیر این جا به جایی را مورد بررسی قرار دهیم:

```
100: 00000000000000000000000001100100
-------------------------------------
  6: 00000000000000000000000000000110
```

هر بیت به اندازه ۴ واحد به سمت راست منتقل می‌شود، بنابراین ۴ بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می‌شود.

عملگر رشته

در رشته‌ها تنها عملگري که استفاده می‌شود نقطه "." می‌باشد که دو رشته را به همدیگه متصل می‌کند. به مثال زیر توجه کنید:

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

    echo "<br>";

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

```
Hello world!
Hello world!
```

تقدم عملگرها

تقدم عملگرها مشخص می‌کند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند. عملگرها در PHP در محاسبات دارای حق تقدم هستند. به عنوان مثال:

```php
$number = 1 + 2 * 3 / 1;
```

اگر ما حق تقدم عملگرها را رعایت نکنیم و عبارت بالا را از سمت چپ به راست انجام دهیم نتیجه ۹ خواهد شد (۳=۲+۱ سپس ۹=۳×۳ و در آخر ۹/۱=۹). اما PHP با توجه به تقدم عملگرها محاسبات را انجام می‌دهد. برای مثال عمل ضرب و تقسیم نسبت به جمع و تفریق تقدم دارند. بنابراین در مثال فوق ابتدا عدد ۲ ضربدر ۳ و سپس نتیجه آنها تقسیم بر ۱ می‌شود که نتیجه ۶ به دست می‌آید. در آخر عدد ۶ با ۱ جمع می‌شود و عدد ۷ حاصل می‌شود. در جدول زیر تقدم برخی از عملگرهای PHP آمده است:

اولویت	عملگر
کمترین	new
	( )

	[ ]
	! ++ -
	@
	/ * %
	+ - .
	< <= >= >
	== != === <>
	|| &&
	?:
	AND XOR OR
بیشترین	,

ابتدا عملگرهای با بالاترین و سپس عملگرهای با پایین‌ترین حق تقدم در محاسبات تأثیر می‌گذارند. به این نکته توجه کنید که تقدم عملگرها ++ و -- به مکان قرارگیری آنها بستگی دارد (در سمت چپ یا راست عملوند باشند). به عنوان مثال:

```php
<?php
    $number = 3;

    $number1 = 3 + ++$number;

    echo $number1;
?>
```

```
7
```

```php
<?php
    $number = 3;

    $number1 = 3 + $number++;

    echo $number1;
?>
```

```
6
```

در عبارت اول ابتدا به مقدار $number یک واحد اضافه شده و ۴ می‌شود و سپس مقدار جدید با عدد ۳ جمع می‌شود و در نهایت عدد ۷ به دست می‌آید. در عبارت دوم مقدار عددی ۳ به مقدار $number اضافه می‌شود و عدد ۶ به دست می‌آید. سپس این مقدار در متغیر $number1 قرار می‌گیرد. و در نهایت مقدار $number به ۴ افزایش می‌یابد. برای ایجاد خوانایی در تقدم عملگرها و انجام محاسباتی که در آنها از عملگرهای زیادی استفاده می‌شود از پرانتز استفاده می‌کنیم:

```php
$number = ( 1 + 2 ) * ( 3 / 4 ) % ( 5 - ( 6 * 7 ));
```

در مثال بالا ابتدا هر کدام از عباراتی که داخل پرانتز هستند مورد محاسبه قرار می‌گیرند. به نکته‌ای در مورد عبارتی که در داخل پرانتز سوم قرار دارد توجه کنید. در این عبارت ابتدا مقدار داخلی‌ترین پرانتز مورد محاسبه قرار می‌گیرد یعنی مقدار ۶ ضربدر ۷ شده و سپس از ۵ کم می‌شود. اگر دو یا

چند عملگر با حق تقدم یکسان موجود باشد ابتدا باید هر کدام از عملگرها را که در ابتدای عبارت می‌آیند مورد ارزیابی قرار دهید. به عنوان مثال :

```
$number = 3 * 2 + 8 / 4;
```

هر دو عملگر * و / دارای حق تقدم یکسانی هستند. بنابر این شما باید از چپ به راست آنها را در محاسبات تأثیر دهید. یعنی ابتدا ۳ را ضربدر ۲ می‌کنید و سپس عدد ۸ را بر ۴ تقسیم می‌کنید. در نهایت نتیجه دو عبارت را جمع کرده و در متغیر $number قرار می‌دهید.

رشته‌ها

رشته‌ها در PHP مجموعه‌ای از کاراکترهای متوالی هستند که بین دو گیومه (کوتِیشِن) تک ' ' یا جفت " " قرار می‌گیرند. مثال:

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

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

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

ادغام رشته‌ها

برای ادغام رشته‌ها در PHP از کاراکتر نقطه (.) استفاده می‌شود:

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

```
PHP Programming is full of enjoy.
```

در PHP می‌توانید رشته‌ها را با اعداد نیز ترکیب کنید:

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

```
Test5
```

تفاوت رشته‌های 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 رشته‌هایی که بین دو گیومه جفت قرار دارند، پردازش شده و بجای نام آنها، مقدارشان در عبارت مورد استفاده قرار می‌گیرد. درمقابل رشته‌های 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 مورد نظرتان، متغیری وجود ندارد، بهتر است آنرا در گیومه تک بگذارید تا سرعت پردازش کد شما افزایش یابد. همچنین باید دقت کنید که پردازش رشته‌های 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:

```php
$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 ۱۰ 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 رشته‌ها در صورت استفاده در عبارات محاسباتی، بطور خودکار به عدد تبدیل خواهند شد؛ بدین ترتیب که از ابتدای رشته، تا زمان رسیدن به اولین کاراکتر 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 = '52Hello';
$sum = 7 + $text;

echo $sum;
```

```
59
```

استفاده از Heredocs و Nowdocs

Heredocs و Nowdocs دو روش برای تعریف رشته‌های بزرگ در PHP می‌باشند. نحوه تعریف رشته در این دو روش به صورت زیر است:

```
<?php
$string = <<<HEREDOC
      ...
HEREDOC;
?>
```

```
<?php
$string = <<<'NOWDOC'
      ...
'NOWDOC';
?>
```

تفاوت این دو در این است که در HEREDOC متغیرها همانند دابل کوتیشن پردازش می‌شوند ولی در NOWDOC نه. به مثال‌های زیر توجه کنید:

HEREDOC

```
<?php
$heredoc = 'HEREDOC'

$string = <<<HEREDOC "The HEREDOC syntax is much cleaner to me and it is really useful ."
HEREDOC;
echo $string;
?>
```

```
The HEREDOC syntax is much cleaner to me and it is really useful .
```

NOWDOC

```
<?php
$nowdoc= 'NOWDOC'

$string = <<<'NOWDOC' "The $nowdoc syntax is much cleaner to me and it is really useful ."
'NOWDOC';
echo $string;
?>
```

```
The $nowdoc syntax is much cleaner to me and it is really useful .
```

به این نکات هم توجه کنید که:

- NOWDOC و HEREDOC را با حروف بزرگ بنویسید.
- NOWDOC باید در داخل تک کوتیشن قرار گیرد.
- شما به جای این دو کلمه هر کلمه دیگری می‌توانید به کار ببرید. فقط باید تگ ابتدا و انتها دقیقاً عین هم باشند.
- NOWDOC و HEREDOC پایانی باید در ابتدای یک خط قرار بگیرند و هیچ فاصله و کاراکتری نباید قبل از آنها قرار دهید.

آرایه‌ها

آرایه نوعی متغیر است که لیستی از آدرس‌های مجموعه‌ای از داده‌های هم نوع را در خود ذخیره می‌کند، البته در PHP داده‌ها هم نوع نباشند مهم نیست. تعریف چندین متغیر از یک نوع برای هدفی یکسان بسیار خسته کننده است. مثلاً اگر بخواهید صد متغیر از نوع اعداد صحیح تعریف

کرده و از آنها استفاده کنید. مطمئناً تعریف این همه متغیر بسیار کسالت آور و خسته کننده است. اما با استفاده از آرایه می‌توان همه آنها را در یک خط تعریف کرد. در زیر راهی ساده برای تعریف یک آرایه نشان داده شده است:

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

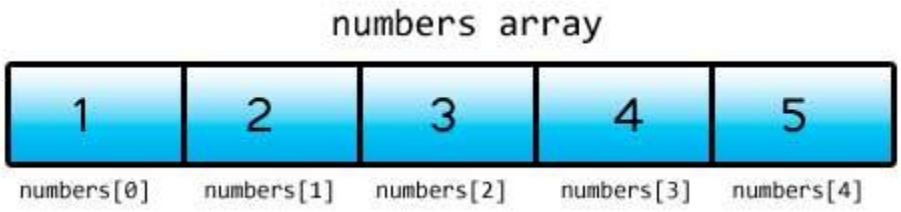
numbers نام آرایه را نشان می‌دهد. هنگام نامگذاری آرایه بهتر است که نام آرایه نشان دهنده نوع آرایه باشد. به عنوان مثال برای نامگذاری آرایه‌ای که اعداد را در خود ذخیره می‌کند از کلمه number استفاده کنید. با استفاده از دستور array یک آرایه ایجاد کرده‌ایم. حتی می‌توانیم به هنگام ایجاد آرایه مقادیر خانه‌های آن را نیز مشخص کنیم:

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

در این مثال ۵ مقدار در ۵ آدرس از فضای حافظه کامپیوتر شما ذخیره می‌شود. مثالا بالا را به روش زیر هم می‌توان پیاده سازی کرد:

```
$numbers = array();

$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 می‌شود. به عنوان مثال شما یک آرایه ۵ عضوی دارید، اندیس آرایه از ۰ تا ۴ می‌باشد چون طول آرایه ۵ است پس ۱-۵ برابر است با ۴. این بدان معناست که اندیس ۰ نشان دهنده اولین عضو آرایه است و اندیس ۱ 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 کرده‌اند معمولاً در گذاشتن اندیس دچار اشتباه می‌شوند و مثلاً ممکن است در مثال بالا اندیس‌ها را از ۱ شروع کنند.

انواع آرایه‌ها در PHP

در زبان PHP سه نوع آرایه وجود دارد:

- آرایه‌های عددی(Numeric Array)
- آرایه‌های انجمنی(Associative Array)
- آرایه‌های چند بعدی(Multidimensional Array)

آرایه‌های عددی (Numeric Array)

آرایه عددی یا Numeric Array آرایه ایست که در اکثر زبانهای برنامه نویسی وجود دارد. در این آرایه، به کل آرایه یک نام می‌دهیم بدین ترتیب هر یک از خانه‌های آرایه یک اندیس می‌گیرد و با آن اندیس می‌توانیم به یک خانه خاص آرایه دست بیابیم. به مثال زیر توجه کنید:

```php
<?php
    $name = array ('Jack' ,'Estephan' , 'Jordan');
    echo $name[2];
?>
```

```
Jordan
```

آرایه‌های انجمنی (Associative Array)

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

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

```
21
```

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

آرایه‌های چند بعدی (Multidimensional Array)

این نوع آرایه، آرایه‌ای است که هر عضو آن می‌تواند خود آرایه باشد. در حقیقت این نوع آرایه، آرایه‌ای از آرایه‌ها می‌باشد. حال این آرایه می‌تواند عددی باشد یا انجمنی. به مثال زیر توجه کنید:

```php
<?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 تفاوت‌های عمده‌ای که آرایه در PHP با آرایه در سایر زبانها دارد اینست که در آرایه‌های چند بعدی PHP لازم نیست تعداد اعضای بعدهای دوم به بعد با هم برابر باشند. و در آخر یاد آور می‌شویم که برای چاپ یک مقدار مورد نظر در آرایه‌های چند بعدی ابتدا نام آرایه اصلی سپس نام آرایه مورد نظر و بعد اندیس مقداری که قرار است چاپ شود را می‌نویسیم (مانند خط آخر مثال بالا).

دستورات شرطی

تقریباً همه زبانهای برنامه نویسی به شما اجازه اجرای کد را در شرایط مطمئن می‌دهند. 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 آن وقت با مشکل مواجه خواهید شد. PHP راه‌های مختلفی برای رفع این نوع مشکلات ارائه می‌دهد. در این بخش با مطالب زیر آشنا خواهید شد:

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

دستور if

می‌توان با استفاده از دستور if و یک شرط خاص که باعث ایجاد یک کد می‌شود یک منطق به برنامه خود اضافه کنید. دستور if ساده‌ترین دستور شرطی است که برنامه می‌گوید اگر شرطی برقرار است کد معینی را انجام بده. ساختار دستورif به صورت زیر است:

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

قبل از اجرای دستور if ابتدا شرط بررسی می‌شود. اگر شرط برقرار باشد یعنی درست باشد سپس کد اجرا می‌شود. شرط یک عبارت مقایسه‌ای است. می‌توان از عملگرهای مقایسه‌ای 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 مقدارnumber کمتر از ۱۰ و Goodbye World را اگر مقدار number از ۱۰ بزرگ‌تر باشد در صفحه نمایش می‌دهد:

```php
<?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 ۳ یک متغیر با نام number تعریف و مقدار ۵ به آن اختصاص داده شده است. وقتی به اولین دستور if در خط ۴ می‌رسیم برنامه تشخیص می‌دهد که مقدار number از ۱۰ کمتر است یعنی ۵ کوچک‌تر از ۱۰ است. منطقی است که نتیجه مقایسه درست می‌باشد بنابراین دستور if دستور را اجرا می‌کند (خط ۵) و پیغام Hello World چاپ می‌شود. حال مقدار number را به ۱۵ تغییر می‌دهیم (خط ۹). وقتی به دومین دستور if در خط ۱۰ می‌رسیم برنامه مقدار number را با ۱۰ مقایسه می‌کند و چون مقدار number یعنی ۱۵ از ۱۰ بزرگ‌تر است برنامه پیغام Goodbye World را چاپ می‌کند (خط ۱۱). به این نکته توجه کنید که دستور if را می‌توان در یک خط نوشت:

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

شما می‌توانید 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 ۱۰ بزرگ‌تر باشد دو پیغام چاپ می‌شود. حال اگر به عنوان مثال آکولاد را حذف کنیم و مقدار x از ۱۰ 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 می‌شود که بین دستورات فاصله بگذاریم.

```php
<?php

    $number = 5;
    if ($number > 10)
        echo ("x is greater than 10.").'<br/>';

        echo ("This is still part of the if statement.");

?>
```

```
This is still part of the if statement.
```

می‌بیند که دستور دوم (خط ۸) در مثال بالا جز دستور `if` نیست. اینجاست که چون ما فرض را بر این گذاشته‌ایم که مقدار x از ۱۰ کوچک‌تر است پس خط `This is still part of the if statement.` چاپ می‌شود. در نتیجه اهمیت وجود آکولاد مشخص می‌شود. 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 می‌کند. یکی از خطاهای معمول کسانی که برنامه نویسی را تازه شروع کرده‌اند قرار دادن سیمیکولن در سمت راست پرانتز `if` است. به عنوان مثال:

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

به یاد داشته باشید که `if` یک مقایسه را انجام می‌دهد و دستور اجرایی نیست. بنابراین برنامه شما با یک خطای منطقی مواجه می‌شود.

دستور if…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 و اگر شرط برقرار نبود دستورالعمل جایگزین اجرا می‌شود (بر خلاف دستور `if` که اگر شرط اول برقرار نباشد هیچ دستورالعملی اجرا نمی‌شود). نحوه‌ی استفاده از این دستور نیز مانند دستور `if` است با این تفاوت که بعد از بسته شدن آکولاد دستور if، دستور `else` اجرا می‌شود:

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

مثال درس قبلی را کامل‌تر می‌کنیم، می‌خواهیم اگر مقدار `number` از ۱۰ کمتر باشد پیغام `Hello World` و در غیر اینصورت پیام `Goodbye World` چاپ شود:

```php
<?php

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

```
    else
    {
        echo("Goodbye World.");
    }

?>
```

```
Goodbye World.
```

همانطور که مشاهده می‌کنید قسمت else اجرا می‌شود چون مقدار متغیر number از عدد ۱۰ بیشتر است.

دستور if…else if

این دستور این امکان را فراهم می‌کند تا در صورت عدم برقراری شرط دستور if، 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 می‌گردیم. ابتدا مقدار متغیر را چک می‌کنیم که آیا از مقدار ۱۰ کمتر است، سپس چک می‌کنیم که آیا از مقدار ۱۰ 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 مثال‌های بسیار ساده‌ای زده شد ولی برخی اوقات شما نیاز دارید تا موارد پیچیده‌تری را محاسبه کنید و از عملگرها بهره بگیرید.

عملگر سه تایی

عملگر شرطی (:?) در PHP مانند دستور شرطی if…else عمل می‌کند. 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 می‌دهد. خطی که به صورت پررنگ نمایش داده شده است به این صورت ترجمه می‌شود که: اگر مقدار متغیر number با عدد ۱۰ برابر بود پیغام This is Ten Number و در غیر اینصورت پیغام Error چاپ شود.

دستور Switch

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

```
switch (testVar)
{
    casecompareVal1:
    code to execute if testVar == compareVa11;
    break;
    casecompareVa12:
    code to execute if testVar == compareVa12;
    break;
    .
    .
    .
    casecompareVa1N:
    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 می‌دهید. این مقدار با هر یک از عبارت‌های case داخل بلوک switch مقایسه می‌شود. 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 breakتشخیص داده می‌شود که باعث می‌شود برنامه از دستور switch خارج شده و دستورات بعد از آن اجرا شوند. اگر این کلمه کلیدی از قلم بیوفتد برنامه با خطا مواجه می‌شود. دستور switch یک بخش default دارد. این دستور در صورتی اجرا می‌شود 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 نمی‌افتد. مکان این دستور هم مهم نیست اما بر طبق تعریف آن را در پایان دستورات می‌نویسند. 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 می‌کنید مقداری که به متغیر number اختصاص داده‌اید با مقادیر case مقایسه می‌شود 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 می‌شود. یکی دیگر از ویژگیهای دستور switch این است که شما می‌توانید از دو یا چند case برای نشان داده یک مجموعه کد استفاده کنید. در مثال زیر اگر مقدار number، ۱، ۲ یا ۳ باشد یک کد اجرا می‌شود. توجه کنید که caseها 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 می‌دهند 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. مثلاً شما مجبورید ۱۰ بار جمله "Hello World." را تایپ کنید مانند مثال زیر:

```php
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 می‌توانید با کپی کردن، این تعداد کد را راحت بنویسید ولی این کار در کل کیفیت کدنویسی را پایین می‌آورد. راه بهتر برای نوشتن کدهای بالا استفاده از حلقه‌ها است. ساختارهای تکرار در PHP عبارت‌اند از:

- while
- do while
- for
- foreach

حلقه While

ابتدایی‌ترین ساختار تکرار در PHP حلقه While است. ابتدا یک شرط را مورد بررسی قرار می‌دهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا می‌شوند. ساختار حلقه While به صورت زیر است:

```php
while(condition)
{
    //code to loop;
}
```

می‌بینید 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 true باشد سپس کدهای داخل بلوک While اجرا می‌شوند. اگر شرط غلط یا false باشد وقتی که برنامه به حلقه While برسد هیچکدام از کدها را اجرا نمی‌کند. 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 می‌گیرد 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:

```php
<?php

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

?>
```

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

برنامه بالا ۱۰ بار پیغام Hello World! را چاپ می‌کند. اگر از حلقه در مثال بالا استفاده نمی‌کردیم مجبور بودیم تمام ۱۰ خط را تایپ کنیم. اجازه دهید که نگاهی به کدهای برنامه فوق بیندازیم. ابتدا در خط ۳ یک متغیر تعریف و از آن به عنوان شمارنده حلقه استفاده شده است. سپس به آن مقدار ۱ را اختصاص می‌دهیم چون اگر مقدار نداشته باشد نمی‌توان در شرط از آن استفاده کرد. در خط ۴ حلقه While را وارد می‌کنیم. در حلقه While ابتدا مقدار اولیه شمارنده با ۱۰ مقایسه می‌شود که آیا از ۱۰ کمتر است یا با آن برابر است. نتیجه هر بار مقایسه ورود به بدنه حلقه While و چاپ پیغام است. همانطور که مشاهده می‌کنید بعد از هر بار مقایسه مقدار شمارنده یک واحد اضافه می‌شود (خط ۷). حلقه تا زمانی تکرار می‌شود که مقدار شمارنده از ۱۰ کمتر باشد.

اگر مقدار شمارنده یک بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز false نشود یک حلقه بین‌هایت به وجود می‌آید. به این نکته توجه کنید که در شرط بالا به جای علامت < از =>استفاده شده است. اگر از علامت < استفاده می‌کردیم کد ما ۹ بار تکرار می‌شد چون مقدار اولیه ۱ است و هنگامی که شرط به ۱۰ برسد false می‌شود چون ۱۰ < ۱۰ نیست. اگر می‌خواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (true) باشد.

```php
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 می‌شود و سپس شرط مورد بررسی قرار می‌گیرد. ساختار حلقه do while به صورت زیر است:

```
do
{
    //code to repeat;
} while (condition);
```

همانطور که مشاهده می‌کنید شرط در آخر ساختار قرار دارد. این بدین معنی است که کدهای داخل بدنه حداقل یکبار اجرا می‌شوند. برخلاف حلقه while که اگر شرط نادرست باشد دستورات داخل بدنه اجرا نمی‌شوند. 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 می‌کنید ابتدا در خط ۶ یک بار رشته Hello World چاپ می‌شود و سپس یک واحد به متغیر number در خط ۷ اضافه می‌شود. در این خط مقدار متغیر number عدد ۲ است و سپس عدد ۲ در خط ۹ با عدد ۱۰ مقایسه می‌شود و چون از عدد ۱۰ بزرگ‌تر نیست شرط نادرست و حلقه متوقف می‌شود. در نتیجه حداقل یکبار حلقه اجرا می‌شود. اما همین مثال در صورتیکه شرط درست باشد نتیجه‌ای به صورت زیر خواهد داشت:

```
<?php

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

?>
```

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

حلقه for

یکی دیگر از ساختارهای تکرار حلقه for است. این حلقه عملی شبیه به حلقه while انجام می‌دهد 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 می‌دهیم. شمارنده فقط در داخل حلقه for قابل دسترسی است.

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

عملگر (operation) که مقدار اولیه متغیر را کاهش یا افزایش می‌دهد. 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 می‌کنیم و آن را با مقدار ۱ مقدار دهی اولیه می‌کنیم. سپس با استفاده از شرط آن را با مقدار ۱۰ مقایسه می‌کنیم که آیا کمتر است یا مساوی؟

توجه کنید که قسمت سوم حلقه (i++) فوراً اجرا نمی‌شود. کد اجرا می‌شود و ابتدا رشته Hello World! را چاپ می‌کند. آنگاه یک واحد به مقدار i اضافه شده و مقدار i برابر ۲ می‌شود و بار دیگر i با عدد ۱۰ مقایسه می‌شود و این حلقه تا زمانی که مقدار شرط true شود ادامه می‌یابد.

اگر بخواهید معکوس بازه‌ای از اعداد را پیاده سازی کنید یعنی اعداد از بزرگ به کوچک چاپ شوند باید به صورت زیر عمل کنید:

```
for ($i = 10; $i > 0; $i--)
{
    echo $i . '<br/>';
}
```

```
10
9
8
7
6
```

```
5
4
3
2
1
```

کد بالا اعداد را از ۱۰ به ۱ چاپ می‌کند (از بزرگ به کوچک). مقدار اولیه شمارنده را ۱۰ می‌دهیم و با استفاده از عملگر کاهش (--) برنامه‌ای که شمارش معکوس را انجام می‌دهد ایجاد می‌کنیم.

حلقه foreach

حلقه foreach یکی دیگر از ساختارهای تکرار در PHP می‌باشد که مخصوصاً برای آرایه‌ها و مجموعه‌ها طراحی شده است. حلقه foreach با هر بار گردش در بین اجزاء، مقادیر هر یک از آنها را در داخل یک متغیر موقتی قرار می‌دهد و شما می‌توانید بواسطه این متغیر به مقادیر دسترسی پیدا کنید. در زیر نحوه استفاده از حلقه foreach آمده است:

```
foreach (array as temporaryVar)
{
    //code to execute;
}
```

array نام آرایه است. سپس کلمه کلیدی as و بعد از آن temporaryVar (این نام کاملاً اختیاری است) متغیری که مقادیر اجزای آرایه را در خود نگهداری می‌کند. در زیر نحوه استفاده از حلقه foreach آمده است:

```
<?php

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

?>
```

```
1
2
3
4
5
```

در برنامه آرایه‌ای با ۵ جزء تعریف شده و مقادیر ۱ تا ۵ در آنها قرار داده شده است (خط ۳). در خط ۹ حلقه foreach شروع می‌شود. ما یک متغیر موقتی تعریف کرده‌ایم که اعداد آرایه را در خود ذخیره می‌کند. در هر بار تکرار از حلقه foreach متغیر موقتی tempNumber، مقادیر عددی را از آرایه استخراج می‌کند. حلقه foreach مقادیر اولین تا آخرین جزء آرایه را در اختیار ما قرار می‌دهد. حلقه foreach برای دریافت هر یک از مقادیر آرایه کاربرد دارد. بعد از گرفتن مقدار یکی از اجزای آرایه، مقدار متغیر موقتی را چاپ می‌کنیم. روش دیگر استفاده از حلقه foreach که معمولاً برای آرایه‌های انجمنی و عددی به کار می‌رود به صورت زیر می‌باشد:

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

برای درک بهتر روش بالا به مثال زیر توجه کنید:

```php
<?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 می‌کنید یک آرایه در خط ۳ ایجاد کرده‌ایم که شامل کلید و مقدار است. برای به دست آوردن کلیدها از حالت دوم حلقه foreach استفاده کرده‌ایم. البته یاد آور می‌شویم که این روش زمانی مفید است که ما به اندیس‌ها و یا کلیدها در آرایه نیاز داشته باشیم مثلاً در مورد آرایه مثال اول هم می‌توانیم اندیس اعداد را به این روش به دست بیاوریم:

```php
<?php

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

?>
```

```
0 => 1
1 => 2
2 => 3
3 => 4
4 => 5
```

خارج شدن از حلقه با استفاده از break و continue

گاهی اوقات با وجود درست بودن شرط می‌خواهیم حلقه متوقف شود. سؤال 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 می‌توان بخشی از حلقه را رد کرد و به مرحله بعد رفت. برنامه زیر نحوه استفاده از continue و break را نشان می‌دهد:

```php
<?php

    echo 'Demonstrating the use of break.'.'<br/>';

    for ($x = 1; $x < 10; $x++)
    {
        if ($x == 5)
            break;

            echo $x . '<br/>';
    }

    echo '<br/>'. 'Demonstrating the use of continue.'.'<br/>';
```

```
    for ($x = 1; $x < 10; $x++)
    {
        if ($x == 5)
            continue;

            echo $x . '<br/>';
    }

?>
```

```
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 حلقه‌هایwhile و do…while استفاده می‌شد، نتیجه یکسانی به دست می‌آمد. همانطور که در شرط برنامه (خط ۷) آمده است وقتی که مقدار x به عدد ۵ رسید سپس دستور break اجرا شود (خط ۸). حلقه بلافاصله متوقف می‌شود حتی اگر شرط 10 > x برقرار باشد. از طرف دیگر در خط ۱۷ حلقه for فقط برای یک تکرار خاص متوقف شده و سپس ادامه می‌یابد). وقتی مقدار x برابر ۵ شود حلقه از ۵ رد شده و مقدار ۵ را چاپ نمی‌کند و بقیه مقادیر چاپ می‌شوند.

تابع

توابع به شما اجازه می‌دهند که یک رفتار یا وظیفه را تعریف کنید و مجموعه‌ای از کدها هستند که در هر جای برنامه می‌توان از آنها استفاده کرد.

توابع در PHP و اکثر زبانهای برنامه نویسی بر دو نوع‌اند:

- توابع از پیش تعریف شده
- توابعی که توسط کاربر تعریف می‌شوند.

ساده‌ترین 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:

```
<?php

    function PrintMessage()
```

```
    {
        echo 'Hello World!';
    }

    PrintMessage ();

?>
```

در خطوط ۶-۳ یک تابع تعریف کرده‌ایم. همانطور که مشاهده می‌کنید در خط ۳ و برای تعریف تابع از کلمه کلیدی function سپس نام تابع و بعد از آن پرانتز باز و بسته استفاده کرده‌ایم. نام تابع ما PrintMessage() است. به این نکته توجه کنید که در نامگذاری تابع از روش پاسکال (حرف اول هر کلمه بزرگ نوشته می‌شود) استفاده کرده‌ایم. این روش نامگذاری قراردادی است و می‌توان از این روش استفاده نکرد، اما پیشنهاد می‌شود که از این روش برای تشخیص توابع استفاده کنید. بهتر است در نامگذاری توابع از کلماتی استفاده شود که کار آن تابع را مشخص می‌کند مثلاً نام‌هایی مانند GoToBed یا OpenDoor. همچنین به عنوان مثال اگر مقدار برگشتی (در درس‌های آینده توضیح می‌دهیم) تابع یک مقدار بولی باشد، می‌توانید اسم تابع خود را به صورت یک کلمه سوالی انتخاب کنید مانند IsLeapyear یا IsTeenager. ولی از گذاشتن علامت سؤال در آخر اسم تابع خودداری کنید. دو پرانتزی که بعد از نام تابع می‌آید نشان دهنده آن است که نام متعلق به یک تابع است. بعد از پرانتزها دو آکولاد قرار می‌دهیم که بدنه تابع را تشکیل می‌دهد و کدهایی را که می‌خواهیم اجرا شوند را در داخل این آکولادها می‌نویسیم. در خط ۸ تابع را صدا می‌زنیم. 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() می‌رود. مثلاً وقتی ما تابع PrintMessage() را در خط ۸ صدا می‌زنیم برنامه از خط ۸ به خط ۳، یعنی جایی که تابع تعریف شده می‌رود و کدهای آن را اجرا می‌کند.

مقدار برگشتی از یک تابع

توابع می‌توانند مقدار برگشتی از هر نوع داده‌ای داشته باشند. این مقادیر می‌توانند 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 می‌زنید و از او می‌خواهید که کار یک سند را به پایان برساند. سپس از او می‌خواهید 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 می‌کنید مقدار بازگشتي از تابع را جلوي دستور return می‌نویسیم. مثال زیر یک تابع که دارای مقدار برگشتی است را نشان می‌دهد:

```
<?php

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

```
        return $sum;
    }

    $result = CalculateSum();
    echo $result;

?>
```

```
15
```

در خطوط ۵ و ۶ مثال فوق، دو متغیر تعریف و مقدار دهی شده‌اند. توجه کنید که این متغیرها، متغیرهای محلی هستند. و این بدان معنی است که این متغیرها در سایر توابع قابل دسترسی نیستند و فقط در تابعی که در آن تعریف شده‌اند قابل استفاده هستند. در خط ۷ جمع دو متغیر در متغیر sum قرار می‌گیرد. در خط ۹ مقدار برگشتی sum توسط دستور return فراخوانی می‌شود. در خط ۱۲ یک متغیر به نام result تعریف می‌کنیم و تابع ()CalculateSum را فراخوانی می‌کنیم.

تابع ()CalculateSum مقدار ۱۵ را بر می‌گرداند که در داخل متغیر result ذخیره می‌شود. در خط ۱۳ مقدار ذخیره شده در متغیر result چاپ می‌شود. 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 ۱۵ است، در حالیکه می‌توانستیم به راحتی یک متغیر تعریف کرده و مقدار ۱۵ 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 می‌خواهیم 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:

```
1   <?php
2
3       function GetNumber()
4       {
5           $number = 11 ;
6
7           if ($number > 10)
8           {
9               return $number;
10          }
11          else
12          {
13              return 0;
14          }
15      }
16
17      $result = GetNumber();
18      echo $result;
19
20  ?>
```

در خطوط ۳-۱۶ یک تابع با نام ()GetNumber تعریف شده است. در خط ۵ متغیری با مقدار ۱۱ مقداردهی شده است که در خط خط ۷ با مقدار ۱۰ مقایسه می‌شود و چون مقدار این متغیر از ۱۰ بیشتر است پس دستور return اول مقدار ۱۱ را برمی گرداند. حال اگر مقدار این متغیر از ۱۰ کمتر باشد دستور return مربوط به قسمت else اجرا و مقدار صفر چاپ می‌شود. که از کاربر یک عدد بزرگ‌تر از ۱۰ را می‌خواهد. اگر قسمت else دستور if و یا دستور return را از آن حذف کنیم در هنگام اجرای برنامه نتیجه چاپ نمی‌شود. چون اگر شرط دستور if نادرست باشد برنامه به قسمت else می‌رود تا مقدار صفر را بر گرداند و چون قسمت else حذف شده است برنامه هیچ مقداری را چاپ نمی‌کند و همچنین

اگر دستور return حذف شود چون برنامه نیاز به مقدار برگشتی دارد برنامه هیچ مقداری را چاپ نمی‌کند. و آخرین مطلبی که در این درس می‌خواهیم به شما آموزش دهیم این است که شما می‌توانید از یک تابع که مقدار برگشتی ندارد خارج شوید. استفاده از return باعث خروج از بدنه تابع و اجرای کدهای بعد از آن می‌شود.

```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 می‌شود.

پارامترها و آرگومانها

پارامترها داده‌های خامی هستند که متد آنها را پردازش می‌کند و سپس اطلاعاتی را که به دنبال آن هستید در اختیار شما قرار می‌دهد. فرض کنید پارامترها مانند اطلاعاتی هستند که شما به یک کارمند می‌دهید که بر طبق آنها کارش را به پایان برساند. یک متد می‌تواند هر تعداد پارامتر داشته باشد. هر پارامتر می‌تواند 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 می‌گیرند. بر اساس کاری که متد انجام می‌دهد می‌توان تعداد پارامترهای زیادی به متد اضافه کرد. بعد از فراخوانی یک متد باید آرگومانهای آن را نیز تأمین کنید. آرگومان‌ها مقادیری هستند که به پارامترها اختصاص داده می‌شوند. ترتیب ارسال آرگومانها به پارامترها مهم است. اجازه بدهید که یک مثال بزنیم:

```php
<?php

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

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

?>
```

در برنامه بالا یک متد به نام CalculateSum() (خطوط ۶-۳) تعریف شده است که وظیفه آن جمع مقدار دو عدد است. متد دارای دو پارامتر است که اعداد را به آنها ارسال می‌کنیم. در بدنه متد دستور return نتیجه جمع دو عدد را بر می‌گرداند. در خط ۸ دو عدد ۵ و ۱۰ را به عنوان آرگومان به متد ارسال می‌کنیم. بعد از ارسال مقادیر ۵ و ۱۰ به پارامترها، پارامترها آنها را دریافت می‌کنند. به این نکته نیز توجه کنید که نام پارامترها طبق قرارداد به شیوه کوهان شتری یا camelCasing (حرف اول دومین کلمه بزرگ نوشته می‌شود) نوشته می‌شود. در داخل بدنه متد (خط ۵) دو مقدار با هم جمع می‌شوند و در خط ۹ نتیجه چاپ می‌شود. دانستن مبانی مقادیر برگشتی و ارسال آرگومانها باعث می‌شود که شما متدهای کارآمدتری تعریف کنید. تکه کد زیر نشان می‌دهد که شما حتی می‌توانید 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 ۵ است و به عنوان آرگومان به متد AnotherMethod() ارسال می‌شود خروجی کد بالا هم عدد ۵ است.

پارامترهای اختیاری

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

```php
<?php

    function PrintMessage($String = "Welcome to PHP Tutorials!")
    {
        echo $String . '<br/>';
    }

    PrintMessage();
    PrintMessage("Learn PHP Today!");

?>
```

```
Welcome to PHP Tutorials!
Learn PHP Today!
```

متد PrintMessage() (خطوط ۶-۳) یک پارامتر اختیاری دارد. برای تعریف یک پارامتر اختیاری می‌توان به آسانی و با استفاده از علامت = یک مقدار را به یک پارامتر اختصاص داد (مثال بالا خط ۳). دو بار متد را فراخوانی می‌کنیم. در اولین فراخوانی (خط ۸) ما آرگومانی به متد ارسال

نمی‌کنیم بنابراین متد از مقدار پیشفرض (Welcome to PHP Tutorials!) استفاده می‌کند. در دومین فراخوانی (خط ۹) یک پیغام (آرگومان) به متد ارسال می‌کنیم که جایگزین مقدار پیشفرض پارامتر می‌شود.

ارسال آرگومان به روش ارجاع و مقدار

آرگومان‌ها را می‌توان به کمک ارجاع ارسال کرد. این بدان معناست که شما آدرس متغیری را ارسال می‌کنید نه مقدار آن را. ارسال با ارجاع زمانی مفید است که شما بخواهید یک آرگومان که دارای مقدار بزرگی است (مانند یک آبجکت) را ارسال کنید. در این حالت وقتی که آرگومان ارسال شده را در داخل متد اصلاح می‌کنیم مقدار اصلی آرگومان در خارج از متد هم تغییر می‌کند. اما در روش مقدار مقدار اصلی آرگومان در خارج از متد تغییر نمی‌کند. در زیر دستورالعمل پایه‌ای تعریف پارامترها که در آنها به جای مقدار از آدرس استفاده شده است نشان داده شده:

```
function FunctionName(& param1)
{
    //code to execute;
}
```

همانطور که در کد بالا مشاهده می‌کنید باید قبل از پارامتری که قرار است به روش ارجاع مقداری به آن ارسال شود علامت (&) قرار داده شود.

اجازه دهید که تفاوت بین ارسال با ارجاع و ارسال با مقدار آرگومان را با یک مثال توضیح دهیم.

```
1   <?php
2       function ModifyNumberVal($number)
3       {
4           $number += 10;
5           echo 'Value of number inside method is '.$number.'<br/>';
6       }
7
8       function ModifyNumberRef(&$number)
9       {
10          $number += 10;
11          echo 'Value of number inside method is '.$number.'<br/>';
12      }
13
14      $num = 5;
15
16      echo 'num = '.$num;
17
18      echo '<br/><br/>';
19
20      echo 'Passing num by value to method ModifyNumberVal() ...<br/>';
21      ModifyNumberVal($num);
22      echo 'Value of num after exiting the method is '.$num.'<br/>';
23
24      echo '<br/><br/>';
25
26      echo 'Passing num by ref to method ModifyNumberRef() ...<br/>';
27      ModifyNumberRef($num);
28      echo 'Value of num after exiting the method is  '. $num.'<br/>';
29  ?>
```

```
num = 5

Passing num by value to method ModifyNumberVal() ...
Value of number inside method is 15.
Value of num after exiting the method is 5.
```

```
Passing num by ref to method ModifyNumberRef() ...
Value of number inside method is 15.
```

در برنامه بالا دو متد که دارای یک هدف یکسان هستند تعریف شده‌اند و آن اضافه کردن عدد ۱۰ به مقداری است که به آنها ارسال می‌شود. اولین متد (خطوط ۶-۲) دارای یک پارامتر است که نیاز به یک مقدار آرگومان دارد. وقتی که متد را صدا می‌زنیم و آرگومانی به آن اختصاص می‌دهیم (خط ۲۱)، کپی آرگومان به پارامتر متد ارسال می‌شود. بنابراین مقدار اصلی متغیر خارج از متد ($num) هیچ ارتباطی به پارامتر متد ندارد. سپس مقدار ۱۰ را به متغیر پارامتر (number) اضافه کرده و نتیجه را چاپ می‌کنیم (خطوط ۵ و ۴). برای اثبات اینکه متغیر $num هیچ تغییری نکرده است مقدار آن را یکبار دیگر چاپ کرده و مشاهده می‌کنیم که تغییری نکرده است (خط ۲۲). دومین متد (خطوط ۱۲-۸) نیاز به یک مقدار با ارجاع دارد. در این حالت به جای اینکه یک کپی از مقدار به عنوان آرگومان به ان ارسال شود آدرس متغیر به آن ارسال می‌شود. حال پارامتر به مقدار اصلی متغیر که زمان فراخوانی متد به آن ارسال می‌شود دسترسی دارد. وقتی که ما مقدار متغیر پارامتری که شامل آدرس متغیر اصلی است را تغییر می‌دهیم (خط ۱۰) در واقع مقدار متغیر اصلی در خارج از متد را تغییر داده‌ایم. در نهایت مقدار اصلی متغیر را وقتی که از متد خارج شدیم را نمایش می‌دهیم و مشاهده می‌شود که مقدار آن واقعاً تغییر کرده است.

محدوده متغیر

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

```php
<?php

    function firstLocalVariable()
    {
        $number = 10;
        echo $number;
    }

    function secondLocalVariable()
    {
        $number = 5;
        echo $number;
    }

    firstLocalVariable ();
    echo '<br/>';
    secondLocalVariable ();

?>
```

```
10
5
```

مشاهده می‌کنید که حتی اگر ما دو متغیر با نام یکسان تعریف کنیم (خطوط ۵ و ۱۱) که دارای محدوده‌های متفاوتی هستند، می‌توان 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)

متغیرهای محلی

متغیرهایی که داخل توابع تعریف می‌شوند محلی هستند و فقط داخل همان تابع قابل استفاده‌اند. به مثال زیر توجه کنید:

```php
<?php

    function LocalVariable()
    {
        $number = 10;
        echo $number;
    }

    LocalVariable ();
    echo $number;

?>
```

```
10
Notice: Undefined variable: number in C:\wamp\www\test.php on line 10
```

همانطور که مشاهده می‌کنید با فراخوانی متد در خط ۹ مقدار متغیر number چاپ می‌شود ولی در خط ۱۰ که سعی در چاپ مقدار این متغیر داریم با پیغام خطا مواجه می‌شویم چون طول عمر این متغیر تا زمانی است که تابع به پایان نرسیده است. با پایان تابع متغیر و مقدار آن هم از بین می‌رود در نتیجه در خارج از تابع نمی‌توان مقدار آن را چاپ کرد.

متغیرهای سراسری

متغیرهایی که در بیرون تابع تعریف می‌شوند 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 شده‌اند 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. روش دیگر برای دسترسی به متغیرهای سراسری استفاده از آرایه فوق سراسری $GLOBALS است. یعنی کد بالا را به صورت زیر هم می‌توان نوشت:

```php
<?php

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

    function GlobalVariable()
    {
        $GLOBALS["Sum"] = $GLOBALS["firstNumber"] + $GLOBALS["secondNumber"];
    }

    GlobalVariable ();
    echo $Sum;

?>
```

در مورد آرایه‌های فوق سراسری در درس‌های بعد توضیح می‌دهیم.

متغیرهای ایستا

با اتمام اجرای تابع تمام متغیرها و آن تابع از بین می‌شوند، 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 می‌کنید هر بار که تابع فراخوانی می‌شود، متغیر firstNumber که به صورت static تعریف شده، مقدار قبلی خود را حفظ می‌کند. بنابراین با هر بار فراخوانی، مقداری آن به اضافه ۱ شده و ذخیره می‌گردد. در خطوط ۸-۳ یک متد و در داخل آن یک متغیر از نوع ایستا (static) تعریف شده است (خط ۵). در خطوط ۱۲-۱۰ سه بار متد را فراخوانی کرده‌ایم. در فراخوانی اول مقدار ۱۰ چاپ می‌شود. در خط ۷ یک واحد به این متغیر اضافه می‌شود و این مقدار در فراخوانی دوم چاپ می‌شود (مقدار ۱۱). در فراخوانی سوم هم یک واحد به مقدار قبلی اضافه شده (۱+۱۱) و این مقدار یعنی ۱۲ چاپ می‌شود.

بازگشت (Recursion)

بازگشت فرایندی است که در آن متد مدام خود را فراخوانی می‌کند تا زمانی که به یک مقدار مورد نظر برسد. بازگشت یک مبحث پیچیده در برنامه نویسی است و تسلط به آن کار را حتی نیست. به این نکته هم توجه کنید که بازگشت باید در یک نقطه متوقف شود وگرنه برای بی نهایت بار، متد، خود را فراخوانی می‌کند. در این درس یک مثال ساده از بازگشت را برای شما توضیح می‌دهیم. فاکتوریل یک عدد صحیح مثبت (n!) شامل حاصل ضرب همه اعداد مثبت صحیح کوچک‌تر یا مساوی آن می‌باشد. به فاکتوریل عدد ۵ توجه کنید.

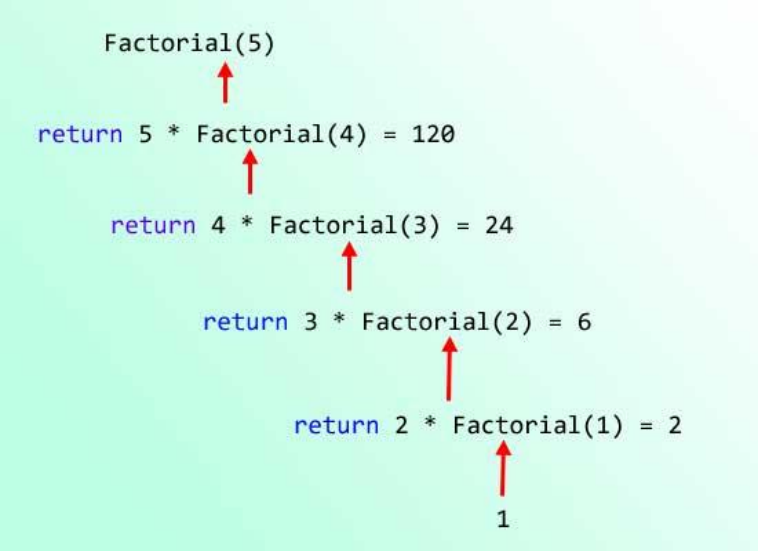
```
5! = 5 * 4 * 3 * 2 * 1 = 120
```

بنابراین برای ساخت یک متد بازگشتی باید به فکر توقف آن هم باشیم. بر اساس توضیح بازگشت، فاکتوریل فقط برای اعداد مثبت صحیح است. کوچک‌ترین عدد صحیح مثبت ۱ است. در نتیجه از این مقدار برای متوقف کردن بازگشت استفاده می‌کنیم.

```php
<?php
    function Factorial($number)
    {
        if ($number == 1)
            return 1;

            return $number * Factorial($number - 1);
    }

    echo Factorial(5);
?>
```

```
120
```

متد مقدار بزرگی را بر می‌گرداند چون محاسبه فاکتوریل می‌تواند خیلی بزرگ باشد. متد یک آرگومان که یک عدد است و می‌تواند در محاسبه مورد استفاده قرار گیرد را می‌پذیرد. در داخل متد یک دستور if می‌نویسیم و در خط ۴ می گوییم که اگر آرگومان ارسال شده برابر ۱ باشد سپس مقدار ۱ را برگردان، در غیر اینصورت به خط بعد برو. این شرط باعث توقف تکرارها نیز می‌شود. در خط ۷ مقدار جاری متغیر number در عددی یک واحد کمتر از خودش (number - 1) ضرب می‌شود. در این خط متد Factorial خود را فراخوانی می‌کند و آرگومان آن در این خط همان number - 1 است. مثلاً اگر مقدار جاری number عدد ۱۰ باشد یعنی اگر ما بخواهیم فاکتوریل عدد ۱۰ را به دست بیاوریم آرگومان متد Factorial در اولین ضرب ۹ خواهد بود. فرایند ضرب تا زمانی ادامه می‌یابد که آرگومان ارسال شده با عدد ۱ برابر نشود. شکل زیر فاکتوریل عدد ۵ را نشان می‌دهد.



کد بالا را به وسیله یک حلقه for نیز می‌توان نوشت:

```php
<?php
    $factorial = 1;

    for ( $counter = 5; $counter >= 1; $counter-- )
        $factorial *= $counter;

    echo $factorial;
?>
```

```
120
```

این کد از کد معادل بازگشتی آن آسان‌تر است. از بازگشت در زمینه‌های خاصی در علوم کامپیوتر استفاده می‌شود. استفاده از بازگشت حافظه زیادی اشغال می‌کند پس اگر سرعت برای شما مهم است از آن استفاده نکنید.

سربارگذاری متدها

در درس‌های قبل با چگونگی ایجاد متدها آشنا شدید. در این درس می‌خواهیم شما را با یک مفهوم دیگر درباره متدها به نام سربارگذاری متدها (Method Overloading) آشنا کنیم. در PHP تعریف دو متد با نام یکسان که دارای تعداد پارامترهای متفاوتی باشند امکان پذیر نیست. به مثال زیر توجه کنید:

```php
<?php

    function ShowMessage()
    {
        echo 'Hello World !';
    }

    function ShowMessage($string)
    {
        echo 'Hello '.$string;
    }
```

```
    ShowMessage();

?>
```

```
(!) Fatal error: Cannot redeclare Person::showMessage() in C:\wamp\www\Tuts\index.php on line 11
```

با اجرای کد بالا با خطا مواجه می‌شوید، چون PHP نمی‌داند که شما کدام متد را فراخوانی کرده‌اید. در PHP توابعی وجود دارند که توسط توسعه دهندگان این زبان برای مقاصد خاصی تعریف شده‌اند و همراه با نصب PHP به صورت توکار وجود دارند. با این توابع، توابع از پیش تعریف شده (Predefined functions) می گویند. مثلاً از برخی از این توابع برای به دست آوردن طول یک رشته، به دست آوردن تعداد عناصر موجود در یک آرایه، کار با تاریخ و ساعت، کار با پوشه‌ها و فایل‌ها ... استفاده می‌شود. قدرت PHP در همین توابع از پیش تعریف شده است و تعداد آنها در هر نسخه جدید از PHP تغییر کرده و بیشتر می‌شود. یکی از این متدها، متد func_get_args() است، که می‌توان با استفاده از آن یک متد با تعداد پارامترهای متفاوت ایجاد کرد:

```
<?php
    function showMessage()
    {
        $string = func_get_args();
        foreach($string as $str)
        {
            echo $str;
        }
    }

    showMessage('1');
    echo '<br/>';
    showMessage('1','2');
    echo '<br/>';
    showMessage('1','2', '3');
?>
```

```
1
12
123
```

همانطور که در کد بالا مشاهده می‌کنید با استفاده از این متد توانستیم در هربار فراخوانی متد ShowMessage() تعداد پارامترهای متفاوتی به آن ارسال کنیم.