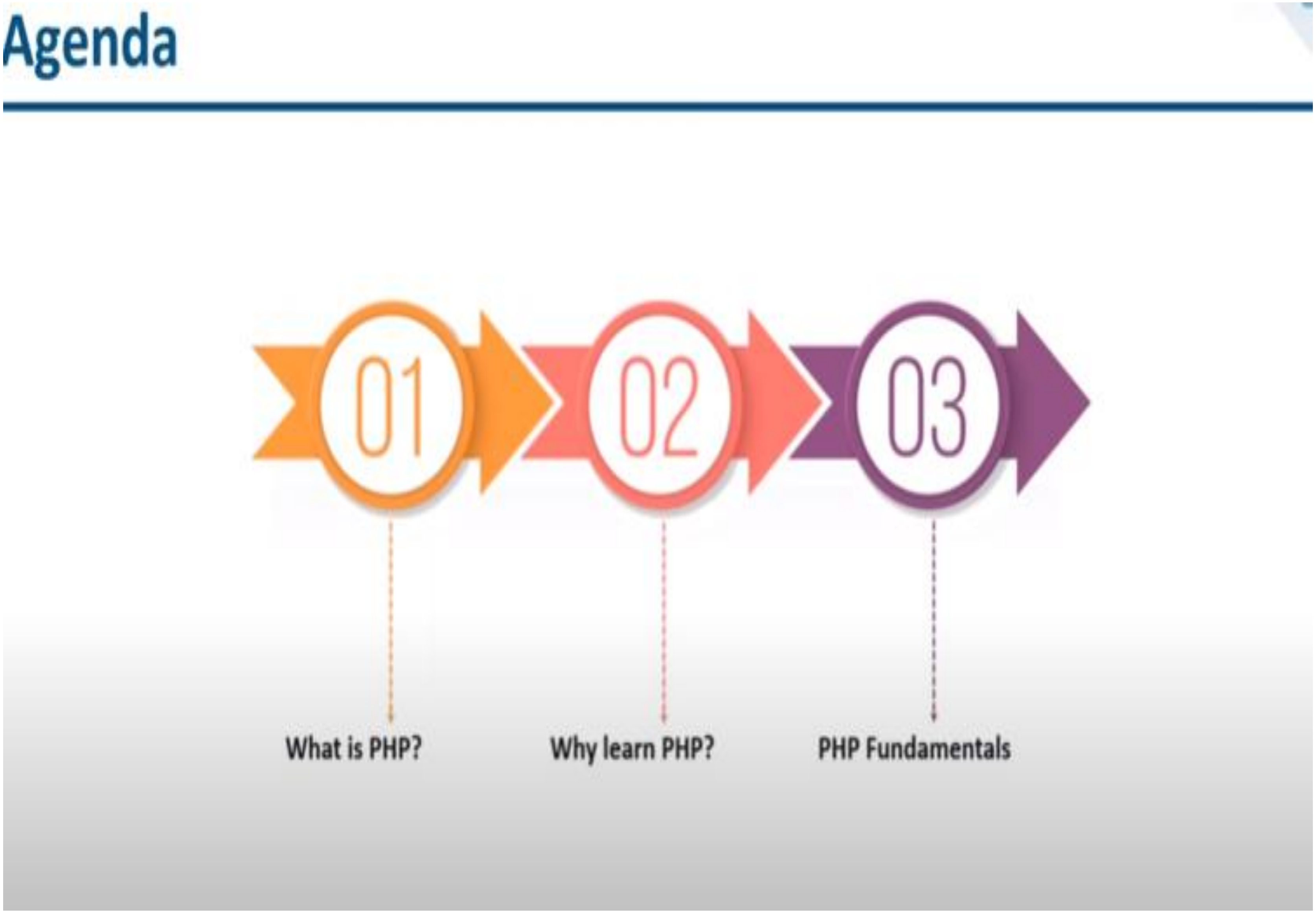


Agenda

01

What is PHP?

02

Why learn PHP?

03

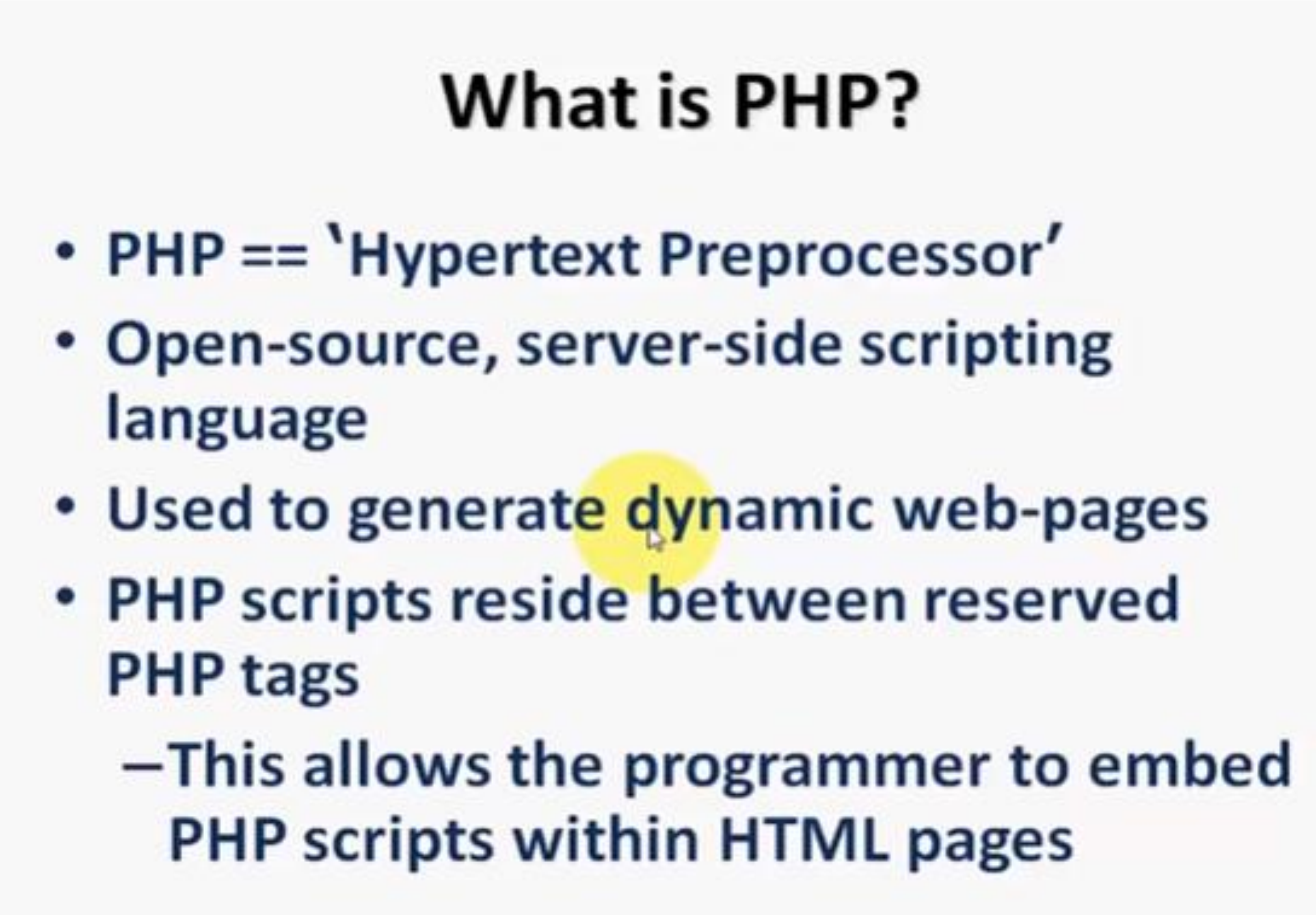
PHP Fundamentals

What is PHP?

PHP is a server side scripting language, and a powerful tool for making dynamic and interactive Web pages

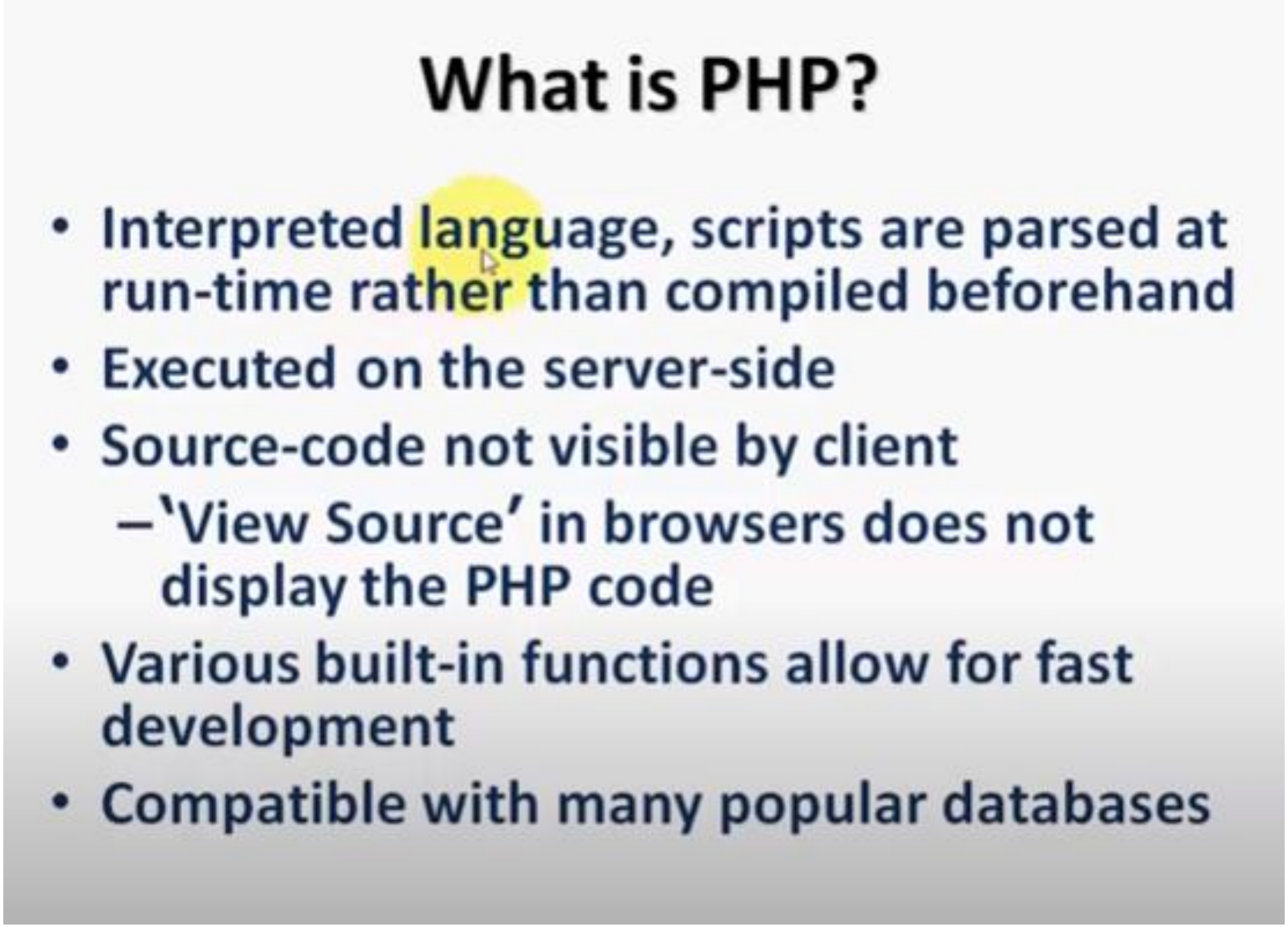
Hypertext Pre-processor

Used for server side logic



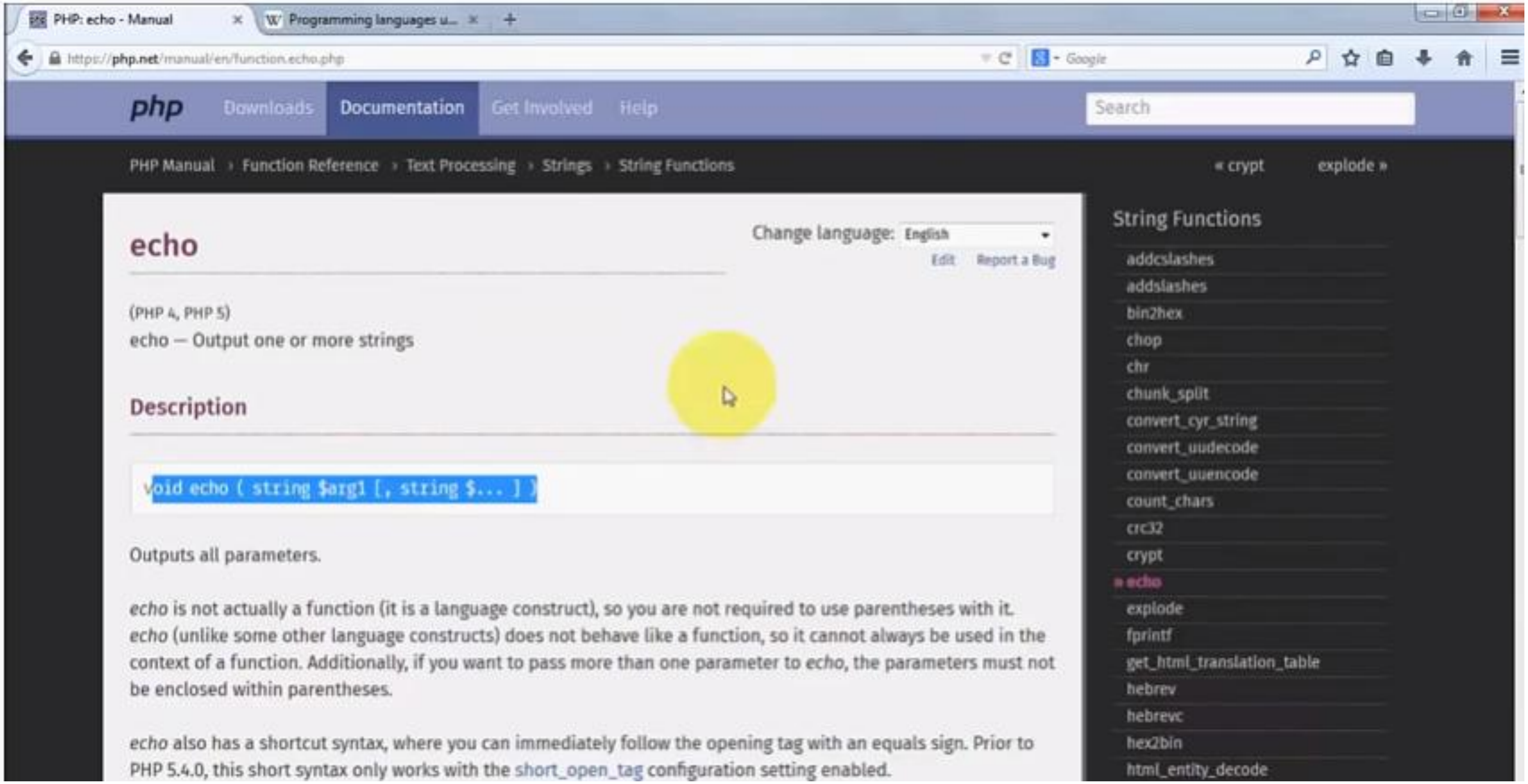
What is PHP?

- PHP == ‘Hypertext Preprocessor’
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
  - This allows the programmer to embed PHP scripts within HTML pages



What is PHP?

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client
  - 'View Source' in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases



echo

(PHP 4, PHP 5)

echo — Output one or more strings

Description

```
void echo ( string $arg1 [, string $... ] )
```

Outputs all parameters.

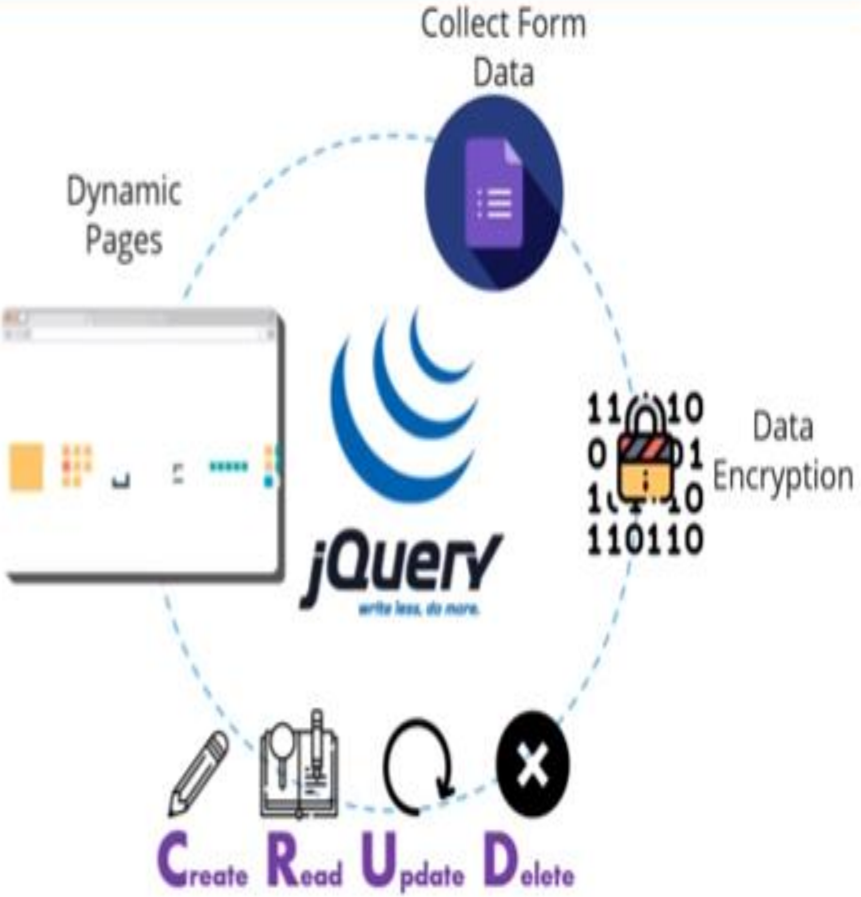
echo is not actually a function (it is a language construct), so you are not required to use parentheses with it. *echo* (unlike some other language constructs) does not behave like a function, so it cannot always be used in the context of a function. Additionally, if you want to pass more than one parameter to *echo*, the parameters must not be enclosed within parentheses.

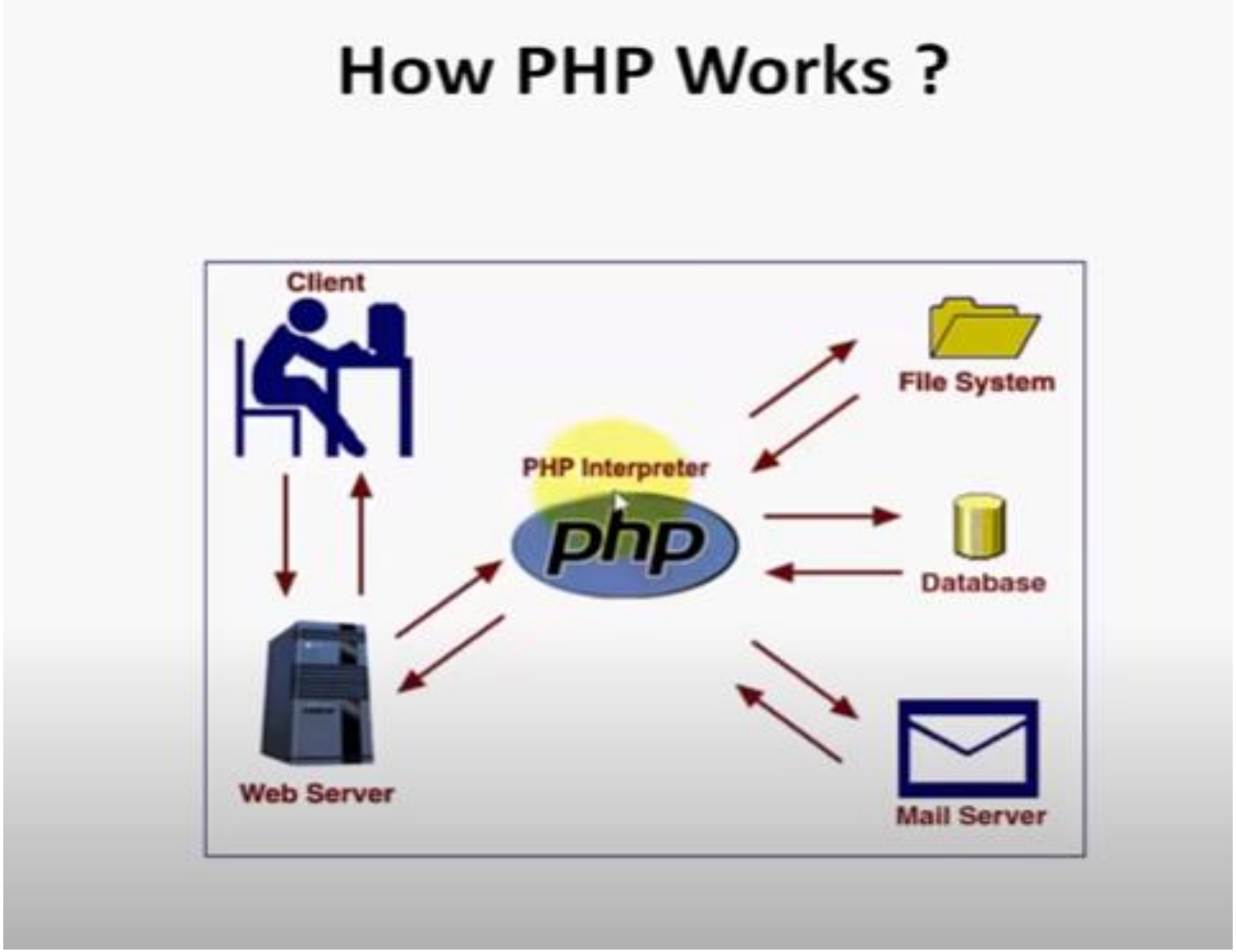
echo also has a shortcut syntax, where you can immediately follow the opening tag with an equals sign. Prior to PHP 5.4.0, this short syntax only works with the short_open_tag configuration setting enabled.

What can PHP do?

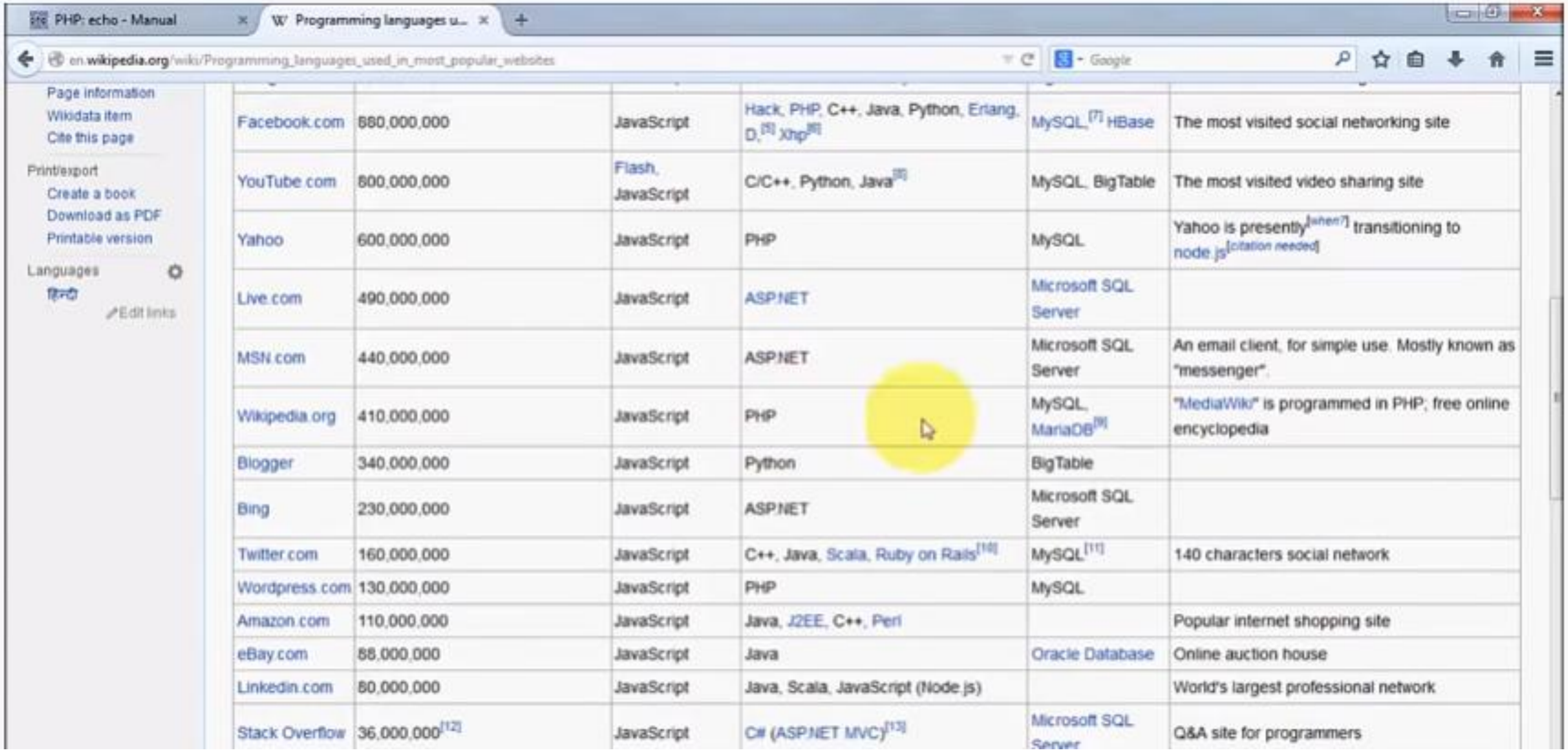
PHP Features

- Easy to Learn
- Supports different types of DB
- Cross Platform
- Supports multiple server types

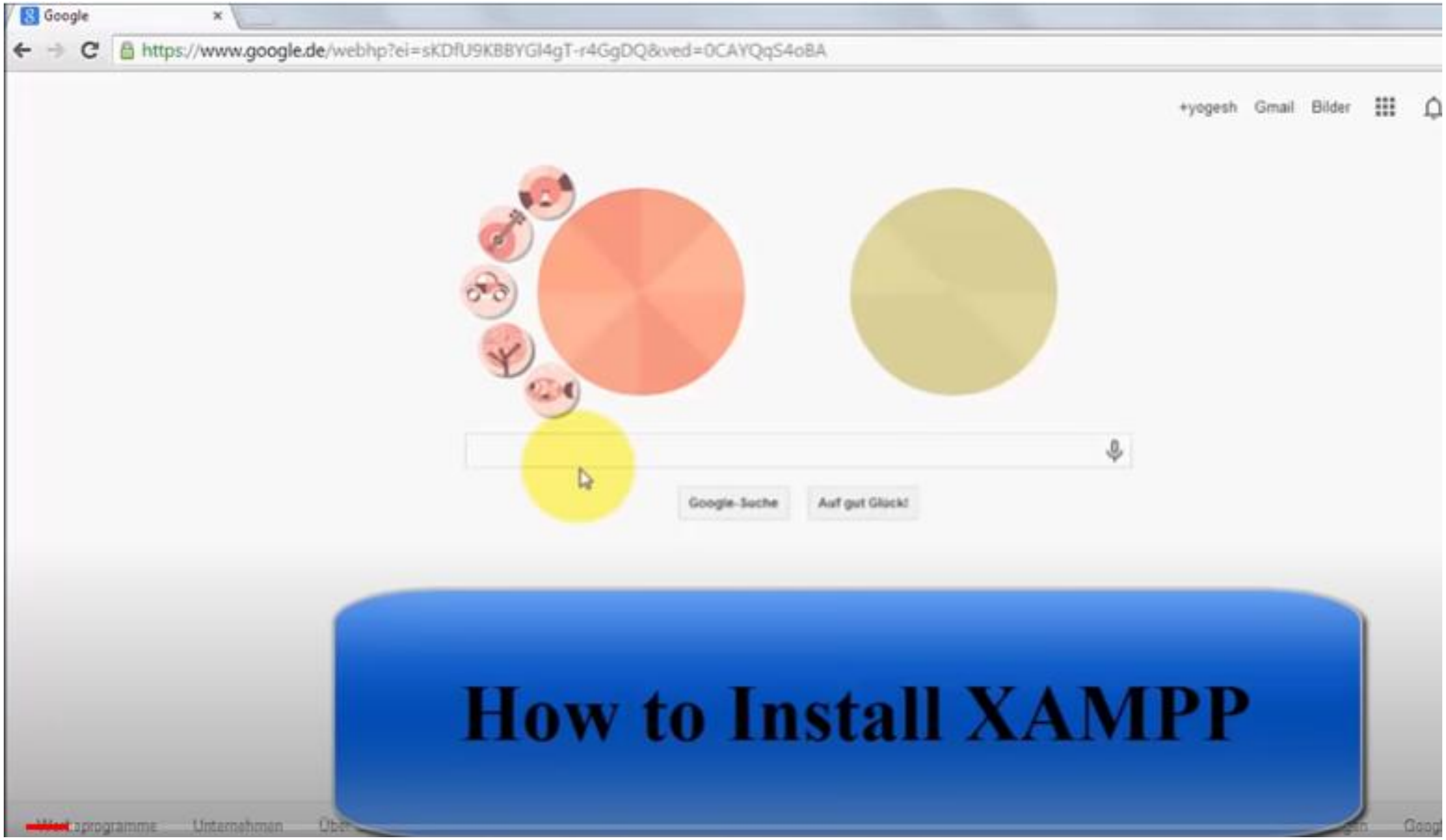




How PHP Works ?

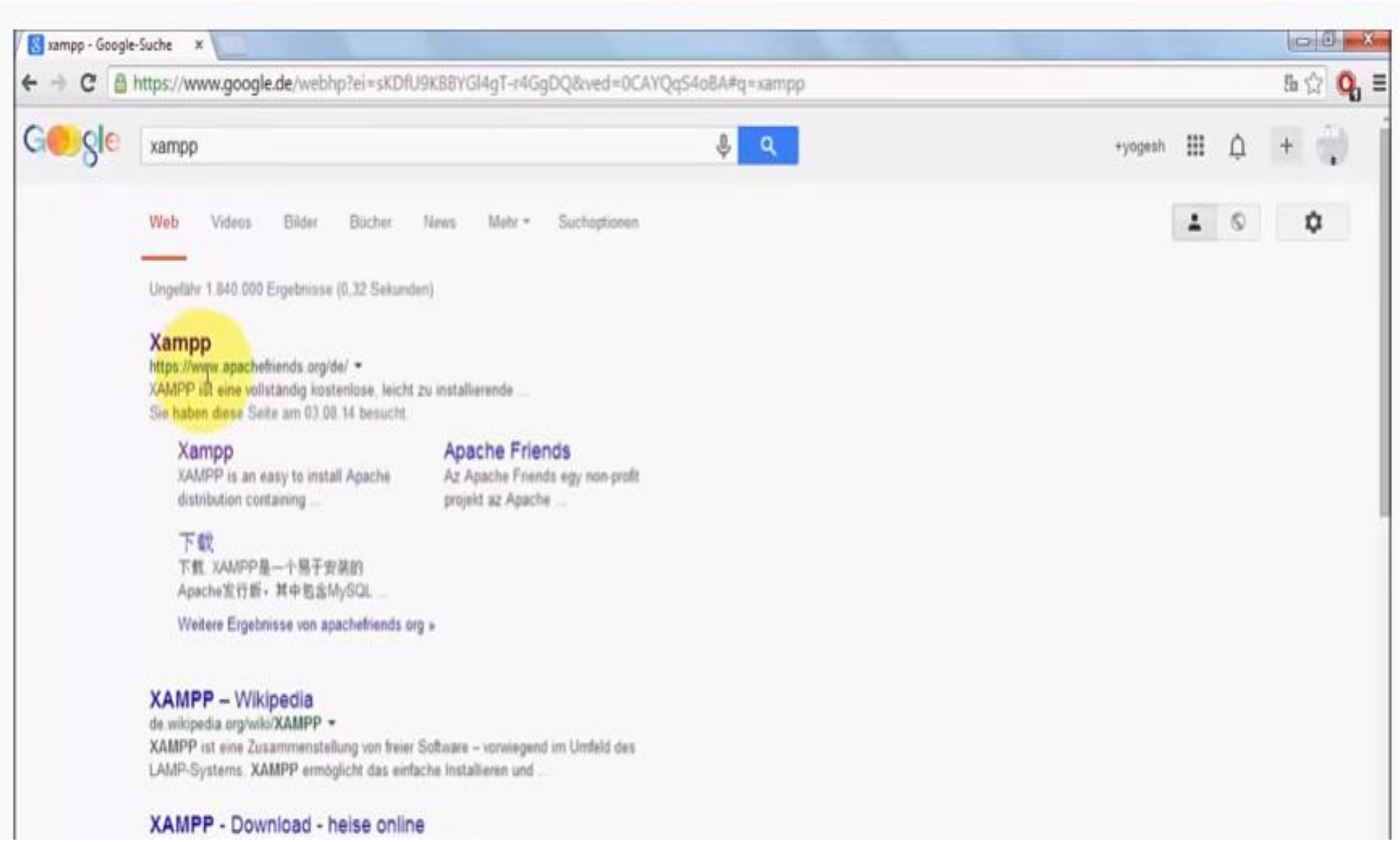


Facebook.com	880,000,000	JavaScript	Hack, PHP, C++, Java, Python, Erlang, D,[5] Xhp[6]	MySQL,[7] HBase	The most visited social networking site
YouTube.com	800,000,000	Flash, JavaScript	C/C++, Python, Java[8]	MySQL, BigTable	The most visited video sharing site
Yahoo	600,000,000	JavaScript	PHP	MySQL	Yahoo is presently[when?] transitioning to node.js[citation needed]
Live.com	490,000,000	JavaScript	ASP.NET	Microsoft SQL Server	
MSN.com	440,000,000	JavaScript	ASP.NET	Microsoft SQL Server	An email client, for simple use. Mostly known as "messenger".
Wikipedia.org	410,000,000	JavaScript	PHP	MySQL, MariaDB[9]	"MediaWiki" is programmed in PHP; free online encyclopedia
Blogger	340,000,000	JavaScript	Python	BigTable	
Bing	230,000,000	JavaScript	ASP.NET	Microsoft SQL Server	
Twitter.com	160,000,000	JavaScript	C++, Java, Scala, Ruby on Rails[10]	MySQL[11]	140 characters social network
Wordpress.com	130,000,000	JavaScript	PHP	MySQL	
Amazon.com	110,000,000	JavaScript	Java, J2EE, C++, Perl		Popular internet shopping site
eBay.com	88,000,000	JavaScript	Java	Oracle Database	Online auction house
Linkedin.com	80,000,000	JavaScript	Java, Scala, JavaScript (Node.js)		World's largest professional network
Stack Overflow	36,000,000[12]	JavaScript	C# (ASP.NET MVC)[13]	Microsoft SQL Server	Q&A site for programmers



https://www.google.de/webhp?ei=sKDfU9KBBYGI4gT-r4GgDQ&ved=0CAYQqS4oBA

How to Install XAMPP


xampp - Google-Suche
https://www.google.de/webhp?ei=sKDfU9KB8YGl4gT-r4GgDQ&ved=0CAYQqS4oBA#q=xampp
xampp
+yogesh
Web
Videos
Bilder
Bücher
News
Mehr
Suchoptionen

Ungefähr 1.840.000 Ergebnisse (0,32 Sekunden)

Xampp
https://www.apachefriends.org/de/
XAMPP ist eine vollständig kostenlose, leicht zu installierende ...
Sie haben diese Seite am 03.08.14 besucht.

Xampp
XAMPP is an easy to install Apache distribution containing ...

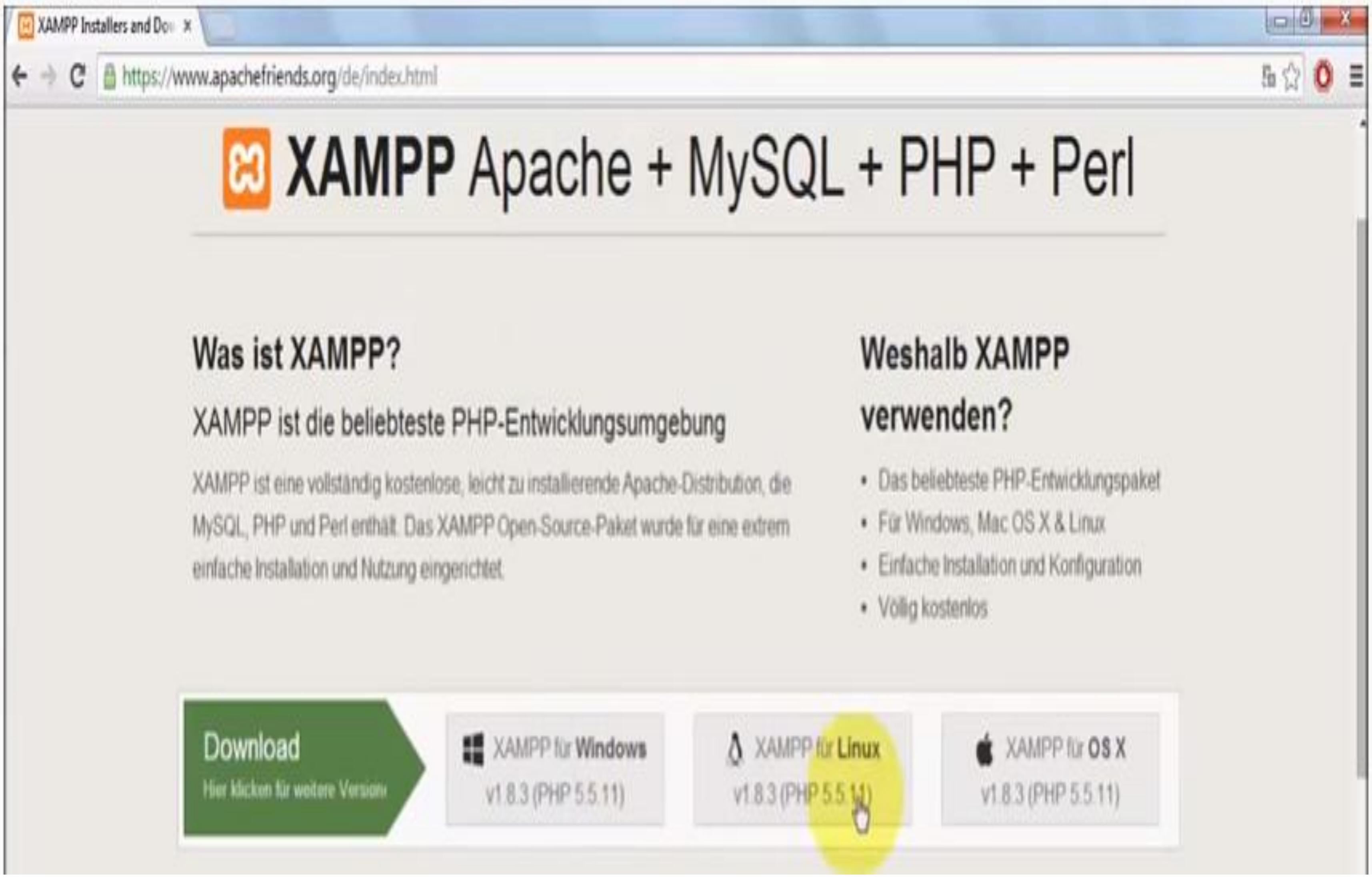
Apache Friends
Az Apache Friends egy non-profit projekt az Apache ...

下載
下載. XAMPP是一个易于安装的 Apache发行版，其中包含MySQL ...

Weitere Ergebnisse von apachefriends.org »

XAMPP – Wikipedia
de.wikipedia.org/wiki/XAMPP
XAMPP ist eine Zusammenstellung von freier Software – vorwiegend im Umfeld des LAMP-Systems. XAMPP ermöglicht das einfache Installieren und ...

XAMPP - Download - heise online



XAMPP Apache + MySQL + PHP + Perl

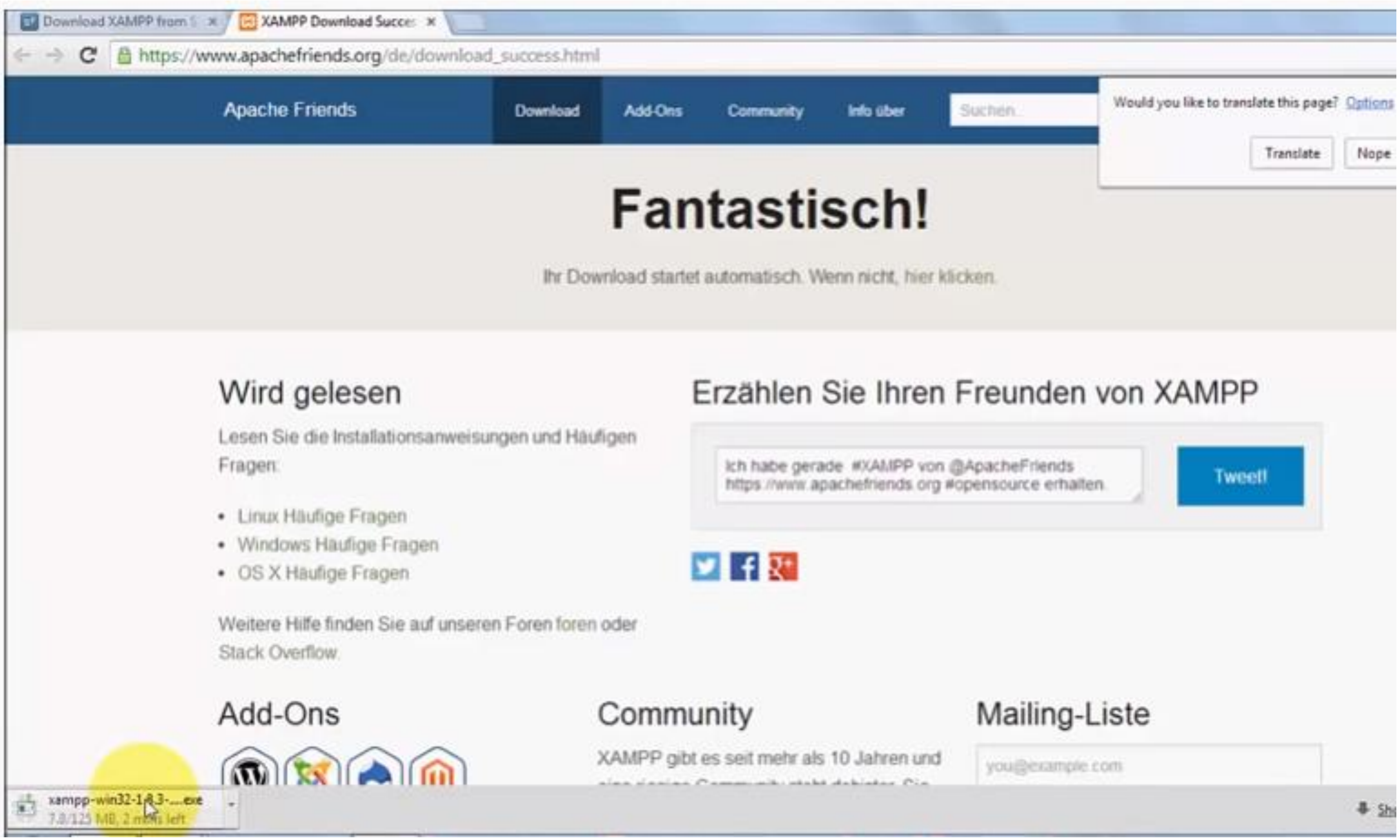
Was ist XAMPP?

XAMPP ist die beliebteste PHP-Entwicklungsumgebung

XAMPP ist eine vollständig kostenlose, leicht zu installierende Apache-Distribution, die MySQL, PHP und Perl enthält. Das XAMPP Open-Source-Paket wurde für eine extrem einfache Installation und Nutzung eingerichtet.

Weshalb XAMPP verwenden?

- Das beliebteste PHP-Entwicklungspaket
- Für Windows, Mac OS X & Linux
- Einfache Installation und Konfiguration
- Völlig kostenlos



Fantastisch!

Ihr Download startet automatisch. Wenn nicht, hier klicken.

Wird gelesen

Lesen Sie die Installationsanweisungen und Häufigen Fragen:

- Linux Häufige Fragen
- Windows Häufige Fragen
- OS X Häufige Fragen

Weitere Hilfe finden Sie auf unseren Foren foren oder Stack Overflow.

Erzählen Sie Ihren Freunden von XAMPP

Ich habe gerade #XAMPP von @ApacheFriends https://www.apachefriends.org #opensource erhalten

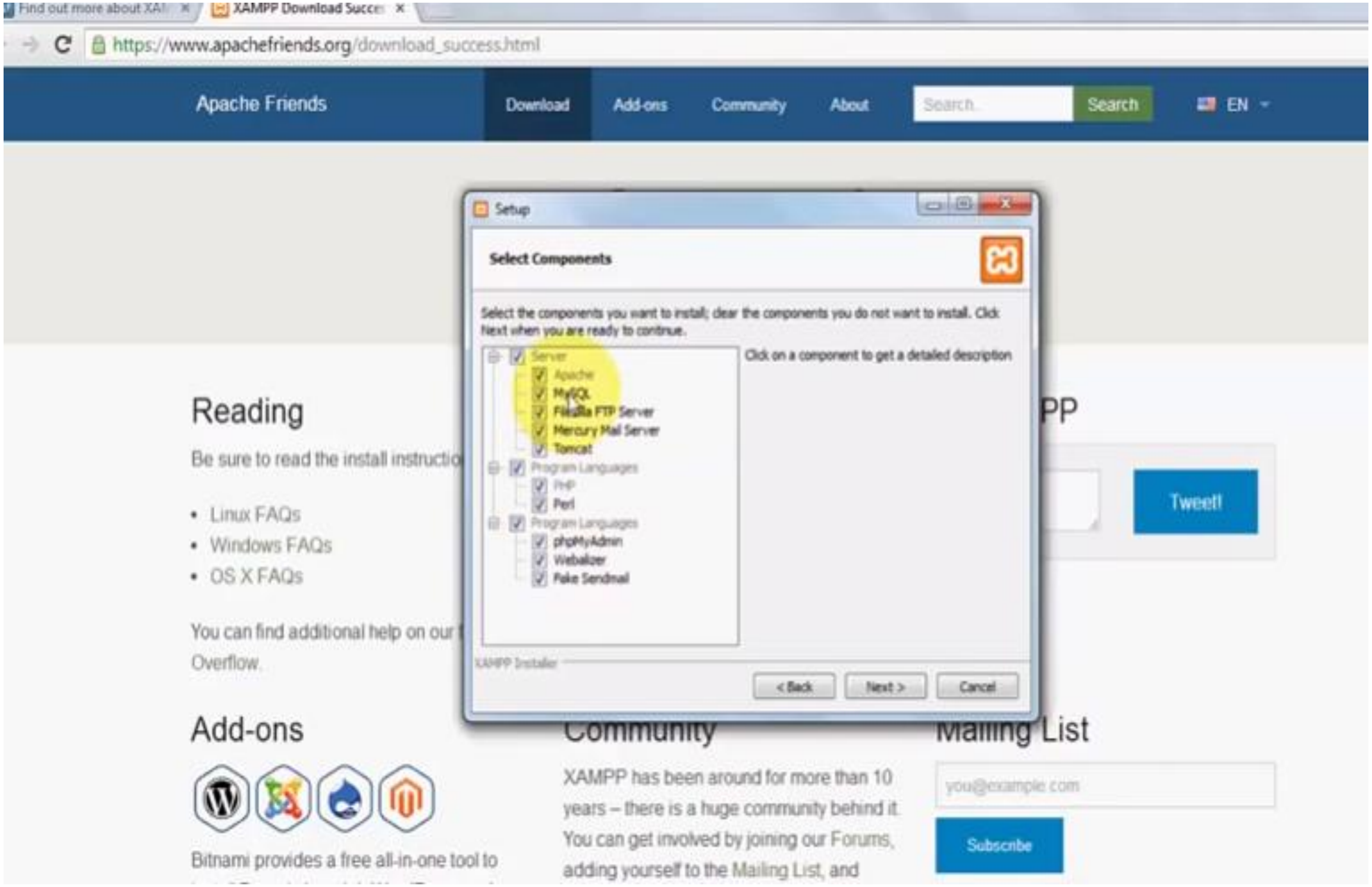
Add-Ons

Community

XAMPP gibt es seit mehr als 10 Jahren und

Mailing-Liste

you@example.com



Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

Reading

Be sure to read the install instructio

- Linux FAQs
- Windows FAQs
- OS X FAQs

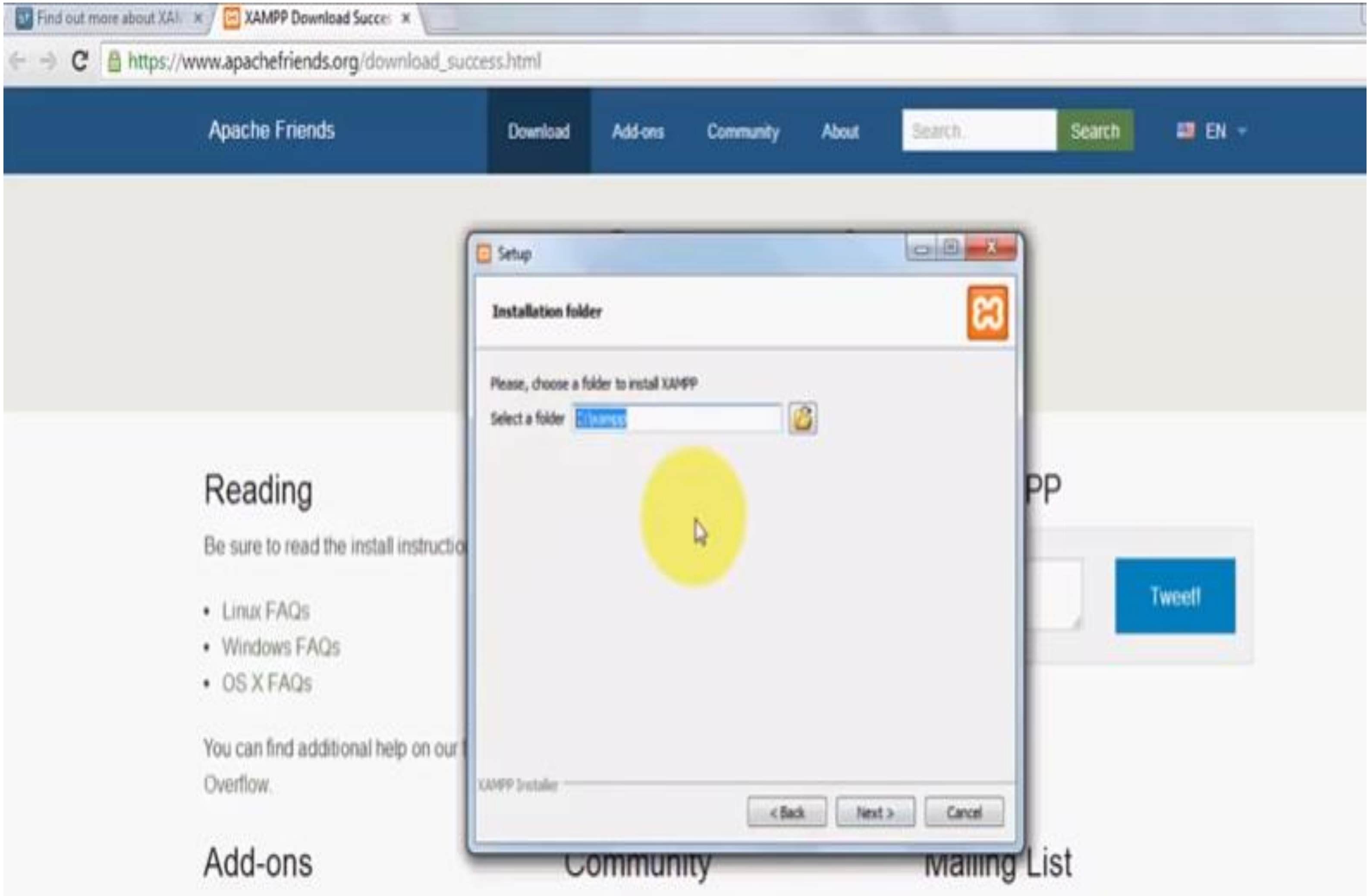
You can find additional help on our f Overflow.

Add-ons

Bitnami provides a free all-in-one tool to

Community

XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, adding yourself to the Mailing List, and



Installation folder

Please, choose a folder to install XAMPP

Select a folder C:\xampp

XAMPP Installer

Reading

Be sure to read the install instructio

- Linux FAQs
- Windows FAQs
- OS X FAQs

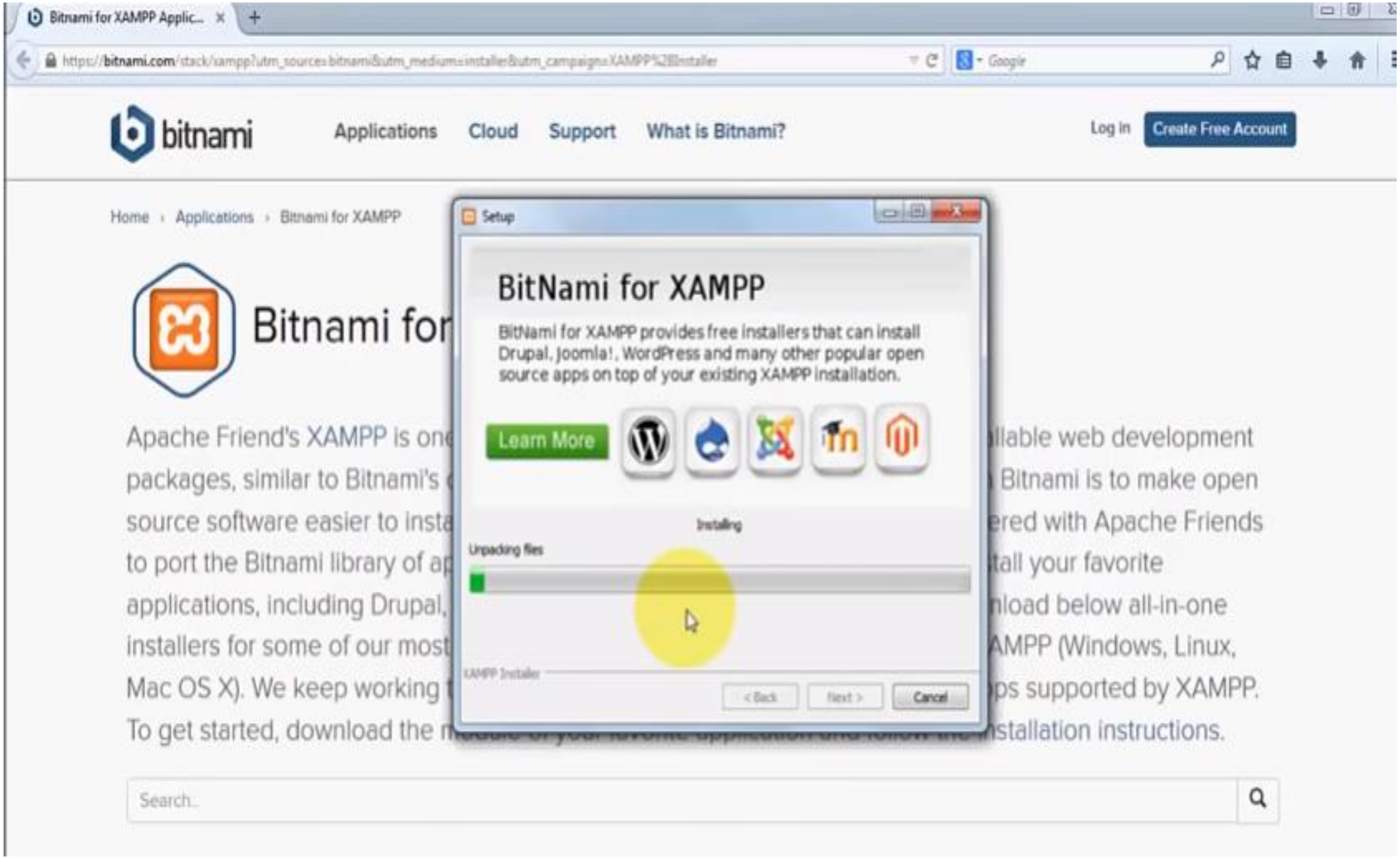
You can find additional help on our
Overflow.

PP

Add-ons

Community

Mailing List

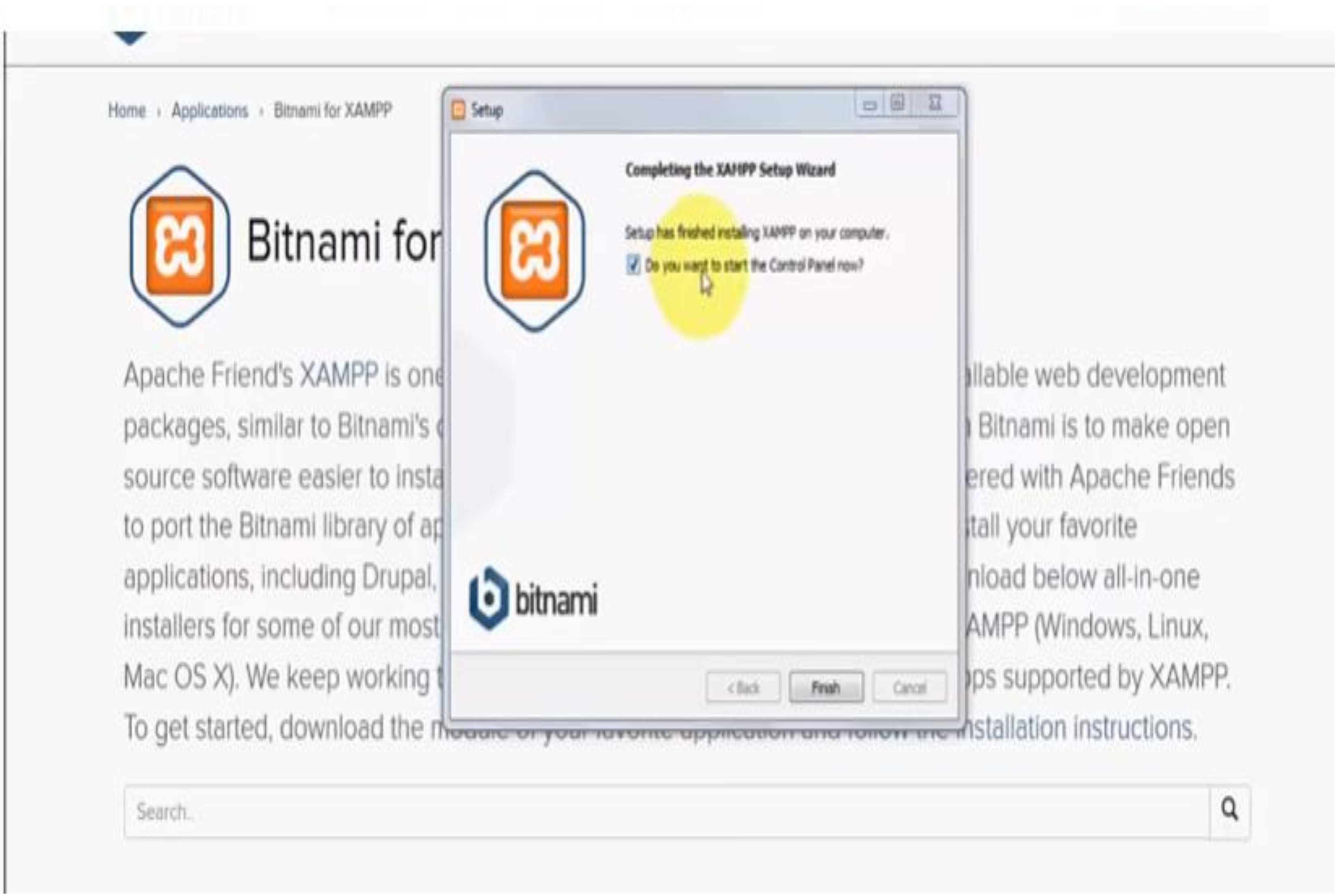

Bitnami for XAMPP Applic...
https://bitnami.com/stack/xampp?utm_source=bitnami&utm_medium=installer&utm_campaign=XAMPP%2BInstaller
Google
bitnami
Applications
Cloud
Support
What is Bitnami?
Log in
Create Free Account
Home › Applications › Bitnami for XAMPP
Setup
Learn More
Installing
Unpacking files
XAMPP Installer
< Back
Next >
Cancel
Search...

Bitnami for

BitNami for XAMPP

BitNami for XAMPP provides free installers that can install Drupal, Joomla!, WordPress and many other popular open source apps on top of your existing XAMPP installation.

Apache Friend's XAMPP is one … ilable web development
packages, similar to Bitnami's … Bitnami is to make open
source software easier to insta… ered with Apache Friends
to port the Bitnami library of ap… tall your favorite
applications, including Drupal, … nload below all-in-one
installers for some of our most … AMPP (Windows, Linux,
Mac OS X). We keep working … ps supported by XAMPP.
To get started, download the m… stallation instructions.



Home › Applications › Bitnami for XAMPP

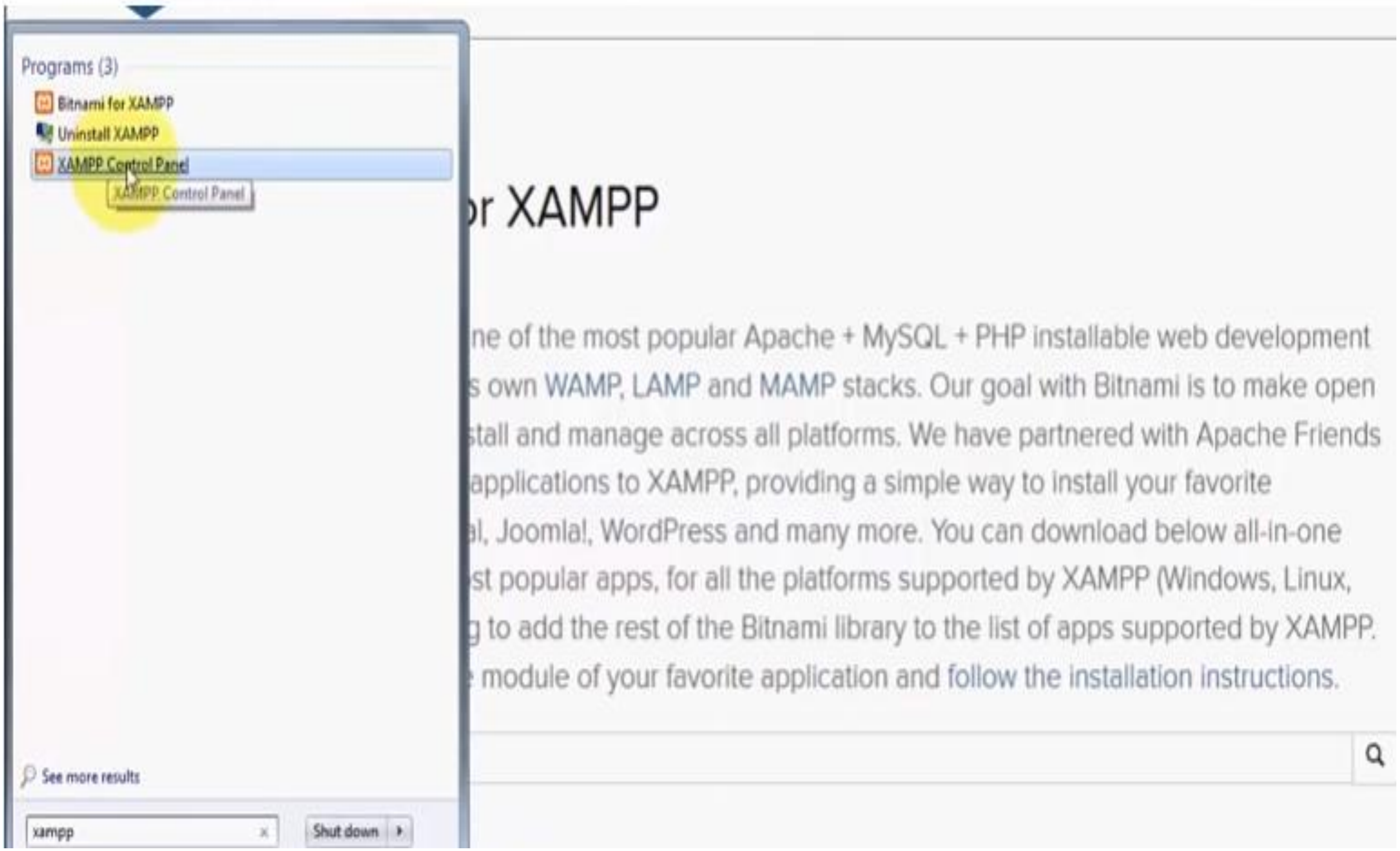
Bitnami for

Apache Friend's XAMPP is one … ailable web development packages, similar to Bitnami's … Bitnami is to make open source software easier to insta… ered with Apache Friends to port the Bitnami library of ap… tall your favorite applications, including Drupal, … nload below all-in-one installers for some of our most … AMPP (Windows, Linux, Mac OS X). We keep working … pps supported by XAMPP. To get started, download the module of your favorite application and follow the installation instructions.

Completing the XAMPP Setup Wizard

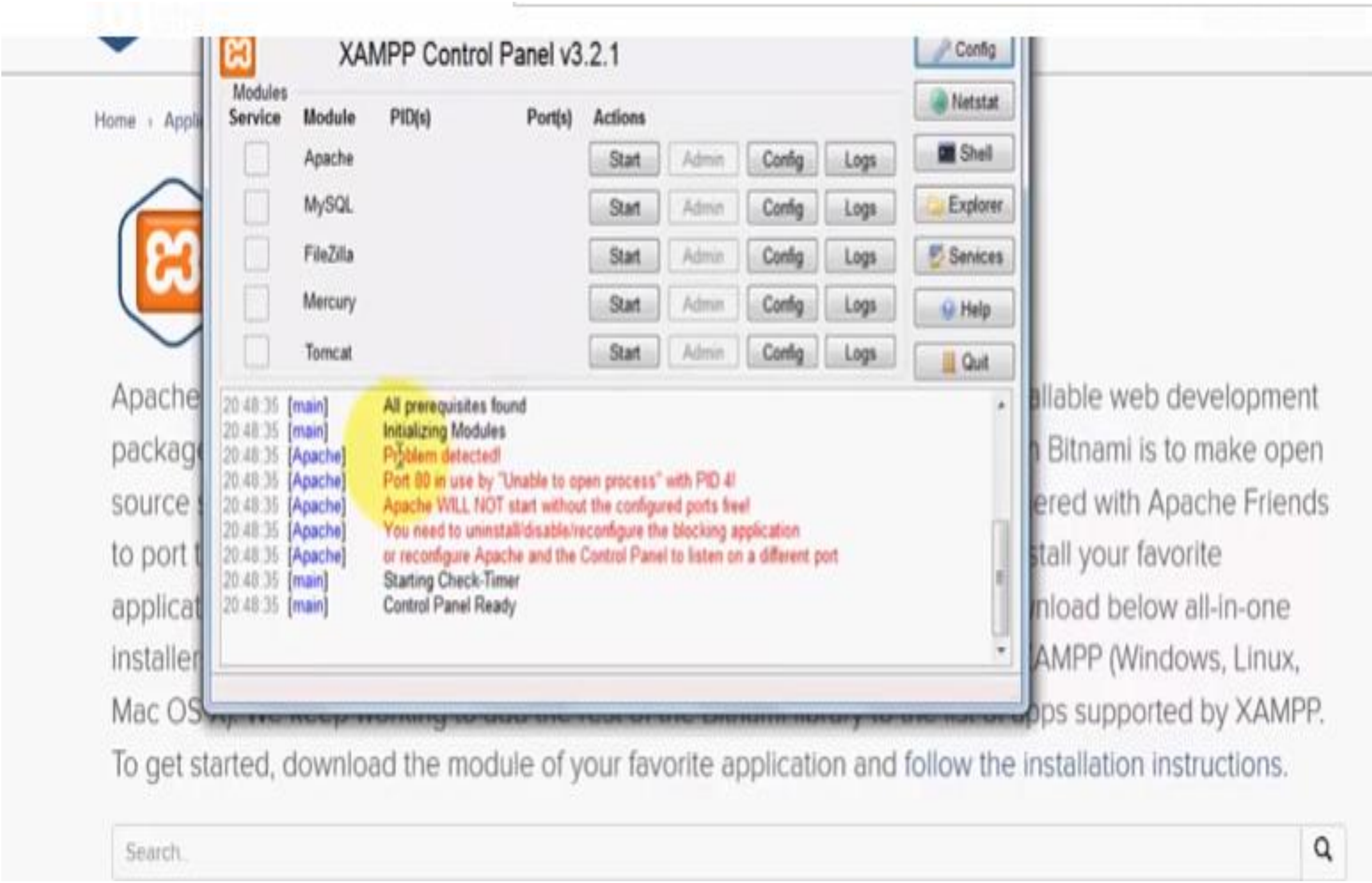
Setup has finished installing XAMPP on your computer.

☑ Do you want to start the Control Panel now?



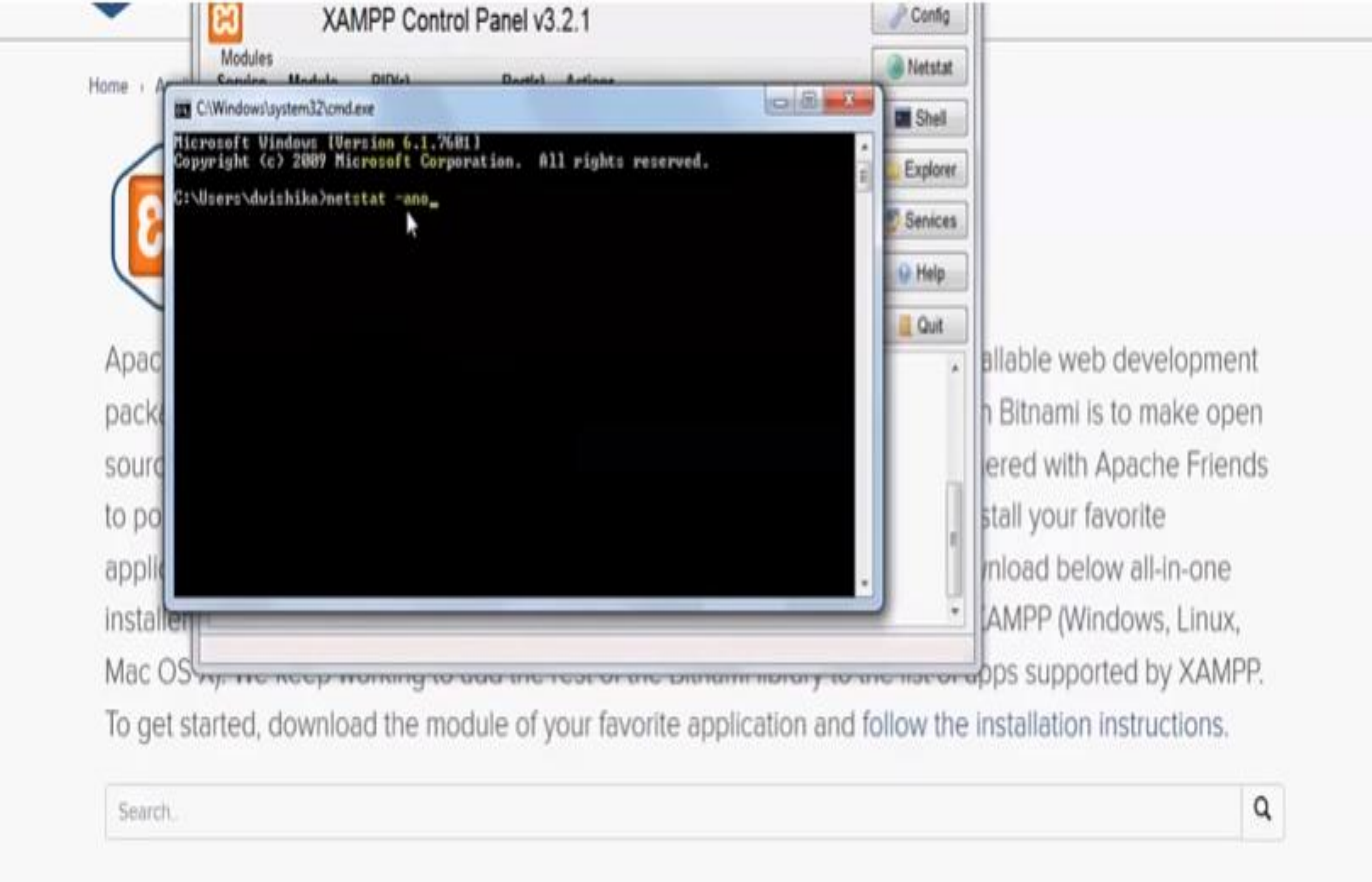
r XAMPP

ne of the most popular Apache + MySQL + PHP installable web development
s own WAMP, LAMP and MAMP stacks. Our goal with Bitnami is to make open
tall and manage across all platforms. We have partnered with Apache Friends
applications to XAMPP, providing a simple way to install your favorite
l, Joomla!, WordPress and many more. You can download below all-in-one
st popular apps, for all the platforms supported by XAMPP (Windows, Linux,
g to add the rest of the Bitnami library to the list of apps supported by XAMPP.
module of your favorite application and follow the installation instructions.



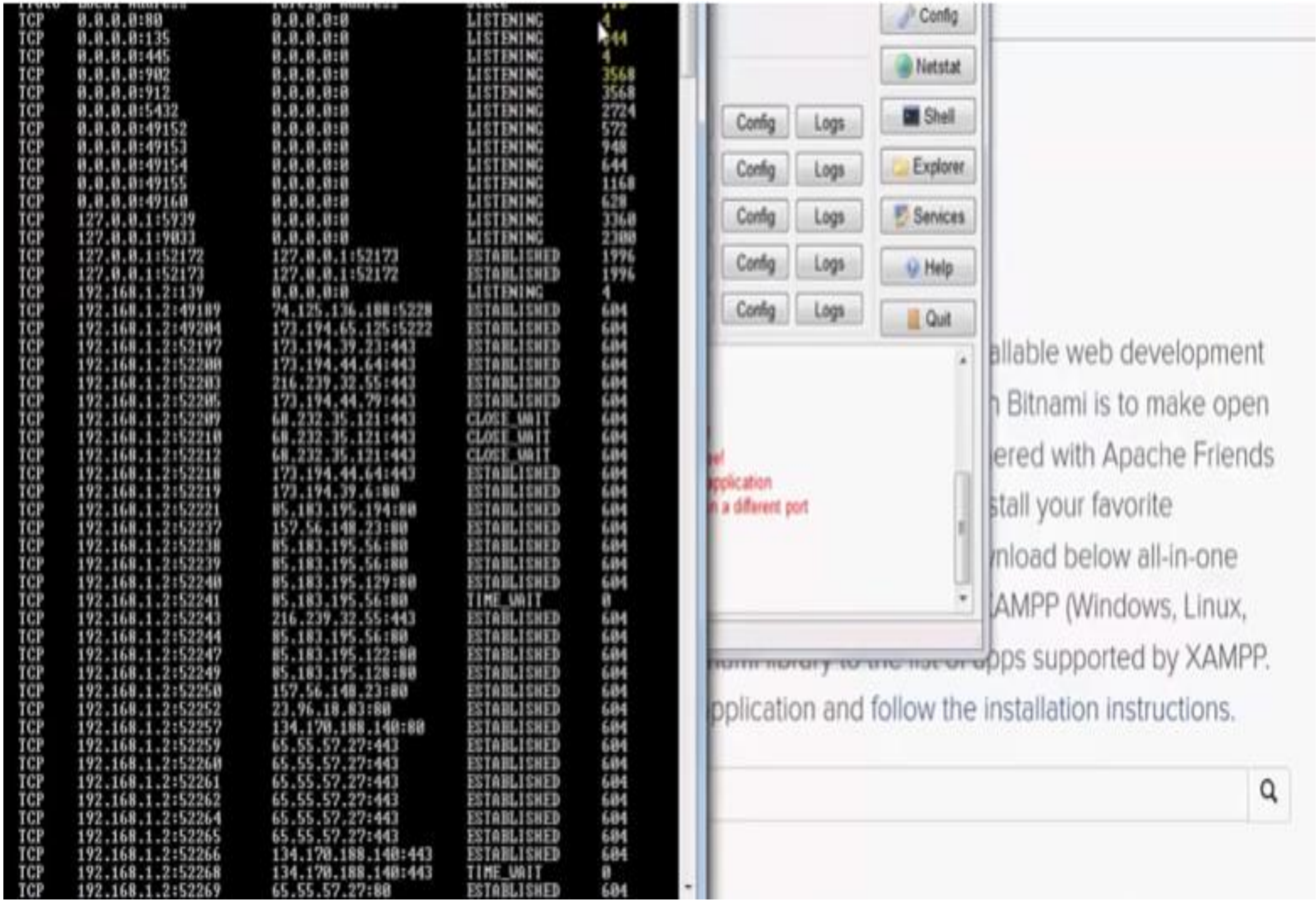
```
20:48:35  [main]     All prerequisites found
20:48:35  [main]     Initializing Modules
20:48:35  [Apache]   Problem detected!
20:48:35  [Apache]   Port 80 in use by "Unable to open process" with PID 4!
20:48:35  [Apache]   Apache WILL NOT start without the configured ports free!
20:48:35  [Apache]   You need to uninstall/disable/reconfigure the blocking application
20:48:35  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
20:48:35  [main]     Starting Check-Timer
20:48:35  [main]     Control Panel Ready
```

Apache … llable web development
packag … Bitnami is to make open
source … ered with Apache Friends
to port t … tall your favorite
applicat … nload below all-in-one
installer … AMPP (Windows, Linux,
Mac OS … ps supported by XAMPP.
To get started, download the module of your favorite application and follow the installation instructions.


XAMPP Control Panel v3.2.1
Modules
Service Module PID(s) Port(s) Actions
Config
Netstat
Shell
Explorer
Services
Help
Quit
Home › A
C:\Windows\system32\cmd.exe
Search..

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\dwishika>netstat -ano_
```

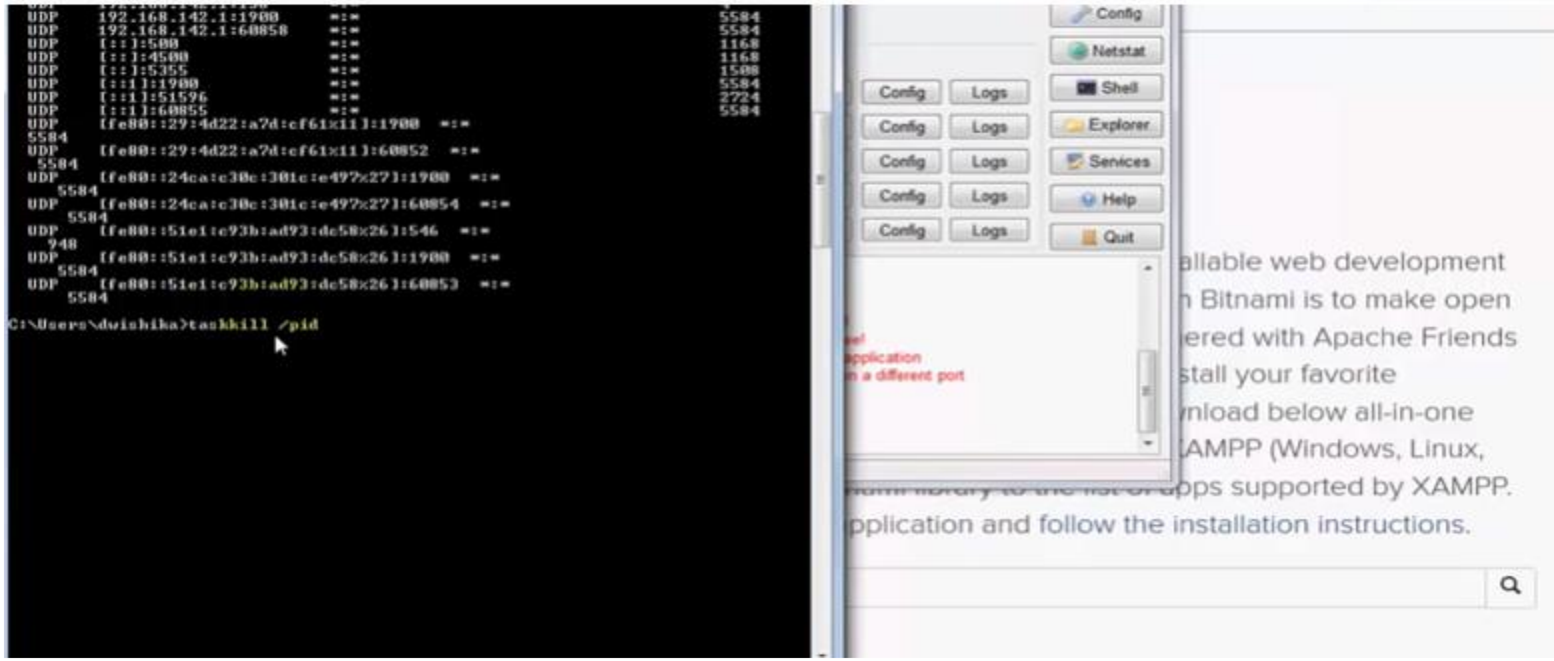
Apac ailable web development
pack Bitnami is to make open
sourc ered with Apache Friends
to po stall your favorite
appli nload below all-in-one
installer AMPP (Windows, Linux,
Mac OS X). pps supported by XAMPP.
To get started, download the module of your favorite application and follow the installation instructions.



```
  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       644
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:902            0.0.0.0:0              LISTENING       3568
  TCP    0.0.0.0:912            0.0.0.0:0              LISTENING       3568
  TCP    0.0.0.0:5432           0.0.0.0:0              LISTENING       2724
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING       572
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING       948
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING       644
  TCP    0.0.0.0:49155          0.0.0.0:0              LISTENING       1160
  TCP    0.0.0.0:49160          0.0.0.0:0              LISTENING       628
  TCP    127.0.0.1:5939         0.0.0.0:0              LISTENING       3360
  TCP    127.0.0.1:9033         0.0.0.0:0              LISTENING       2300
  TCP    127.0.0.1:52172        127.0.0.1:52173        ESTABLISHED     1996
  TCP    127.0.0.1:52173        127.0.0.1:52172        ESTABLISHED     1996
  TCP    192.168.1.2:139        0.0.0.0:0              LISTENING       4
  TCP    192.168.1.2:49189      74.125.136.188:5228    ESTABLISHED     604
  TCP    192.168.1.2:49204      173.194.65.125:5222    ESTABLISHED     604
  TCP    192.168.1.2:52197      173.194.39.23:443      ESTABLISHED     604
  TCP    192.168.1.2:52200      173.194.44.64:443      ESTABLISHED     604
  TCP    192.168.1.2:52203      216.239.32.55:443      ESTABLISHED     604
  TCP    192.168.1.2:52205      173.194.44.79:443      ESTABLISHED     604
  TCP    192.168.1.2:52209      68.232.35.121:443      CLOSE_WAIT      604
  TCP    192.168.1.2:52210      68.232.35.121:443      CLOSE_WAIT      604
  TCP    192.168.1.2:52212      68.232.35.121:443      CLOSE_WAIT      604
  TCP    192.168.1.2:52218      173.194.44.64:443      ESTABLISHED     604
  TCP    192.168.1.2:52219      173.194.39.6:80        ESTABLISHED     604
  TCP    192.168.1.2:52221      85.183.195.194:80      ESTABLISHED     604
  TCP    192.168.1.2:52237      157.56.148.23:80       ESTABLISHED     604
  TCP    192.168.1.2:52238      85.183.195.56:80       ESTABLISHED     604
  TCP    192.168.1.2:52239      85.183.195.56:80       ESTABLISHED     604
  TCP    192.168.1.2:52240      85.183.195.129:80      ESTABLISHED     604
  TCP    192.168.1.2:52241      85.183.195.56:80       TIME_WAIT       0
  TCP    192.168.1.2:52243      216.239.32.55:443      ESTABLISHED     604
  TCP    192.168.1.2:52244      85.183.195.56:80       ESTABLISHED     604
  TCP    192.168.1.2:52247      85.183.195.122:80      ESTABLISHED     604
  TCP    192.168.1.2:52249      85.183.195.128:80      ESTABLISHED     604
  TCP    192.168.1.2:52250      157.56.148.23:80       ESTABLISHED     604
  TCP    192.168.1.2:52252      23.96.18.83:80         ESTABLISHED     604
  TCP    192.168.1.2:52257      134.170.188.140:80     ESTABLISHED     604
  TCP    192.168.1.2:52259      65.55.57.27:443        ESTABLISHED     604
  TCP    192.168.1.2:52260      65.55.57.27:443        ESTABLISHED     604
  TCP    192.168.1.2:52261      65.55.57.27:443        ESTABLISHED     604
  TCP    192.168.1.2:52262      65.55.57.27:443        ESTABLISHED     604
  TCP    192.168.1.2:52264      65.55.57.27:443        ESTABLISHED     604
  TCP    192.168.1.2:52265      65.55.57.27:443        ESTABLISHED     604
  TCP    192.168.1.2:52266      134.170.188.140:443    ESTABLISHED     604
  TCP    192.168.1.2:52268      134.170.188.140:443    TIME_WAIT       0
  TCP    192.168.1.2:52269      65.55.57.27:80         ESTABLISHED     604
```

ef
pplication
n a different port

allable web development
h Bitnami is to make open
ered with Apache Friends
stall your favorite
nload below all-in-one
AMPP (Windows, Linux,
pps supported by XAMPP.
pplication and follow the installation instructions.

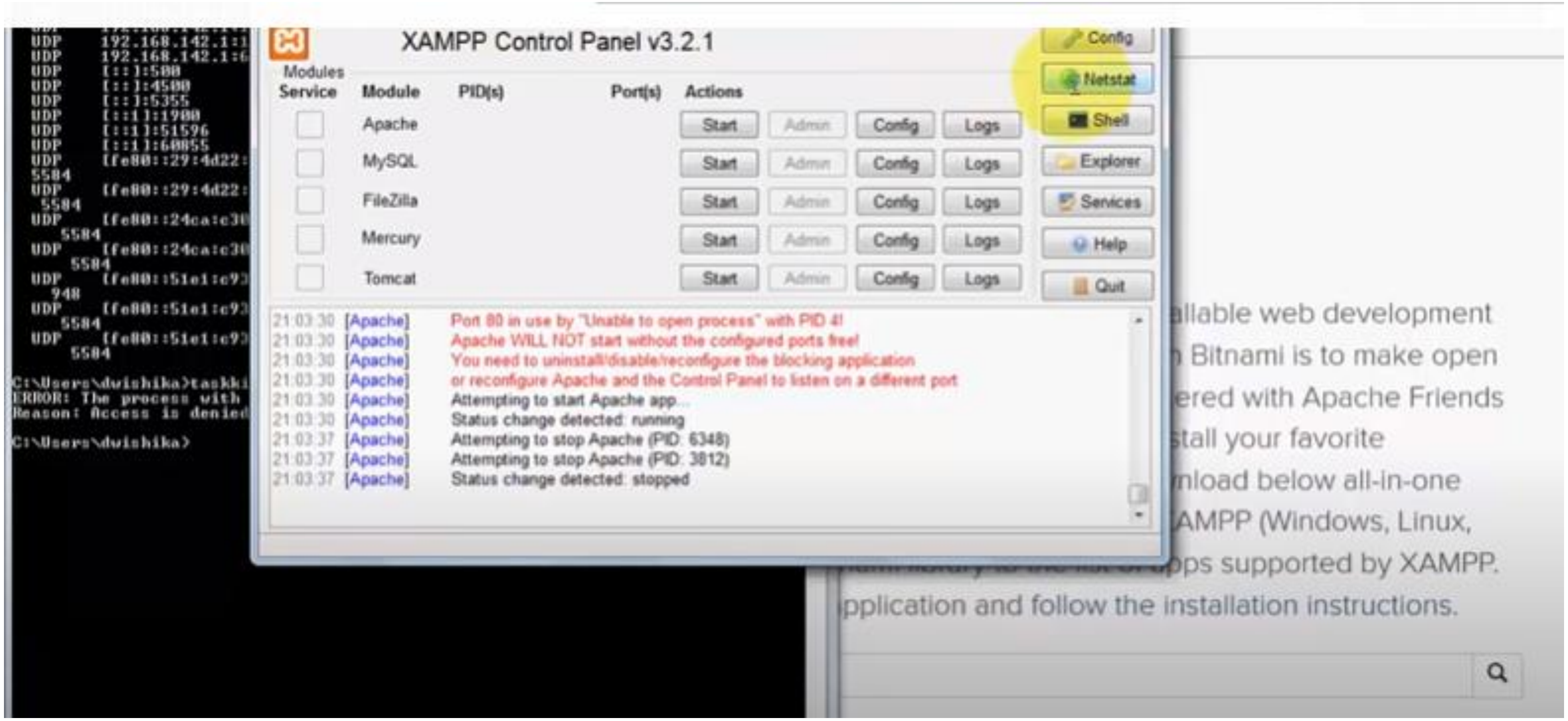


```
  UDP    192.168.142.1:1900     *:*                                    5584
  UDP    192.168.142.1:60858    *:*                                    5584
  UDP    [::]:500               *:*                                    1168
  UDP    [::]:4500              *:*                                    1168
  UDP    [::]:5355              *:*                                    1508
  UDP    [::1]:1900             *:*                                    5584
  UDP    [::1]:51596            *:*                                    2724
  UDP    [::1]:60855            *:*                                    5584
  UDP    [fe80::29:4d22:a7d:cf61%11]:1900  *:*
  5584
  UDP    [fe80::29:4d22:a7d:cf61%11]:60852  *:*
   5584
  UDP    [fe80::24ca:c30c:301c:e497%27]:1900  *:*
      5584
  UDP    [fe80::24ca:c30c:301c:e497%27]:60854  *:*
       5584
  UDP    [fe80::51e1:c93b:ad93:dc58%26]:546  *:*
     948
  UDP    [fe80::51e1:c93b:ad93:dc58%26]:1900  *:*
      5584
  UDP    [fe80::51e1:c93b:ad93:dc58%26]:60853  *:*
       5584

C:\Users\dwishika>taskkill /pid
```

ef
pplication
n a different port

allable web development
n Bitnami is to make open
ered with Apache Friends
stall your favorite
vnload below all-in-one
AMPP (Windows, Linux,
pps supported by XAMPP.
pplication and follow the installation instructions.

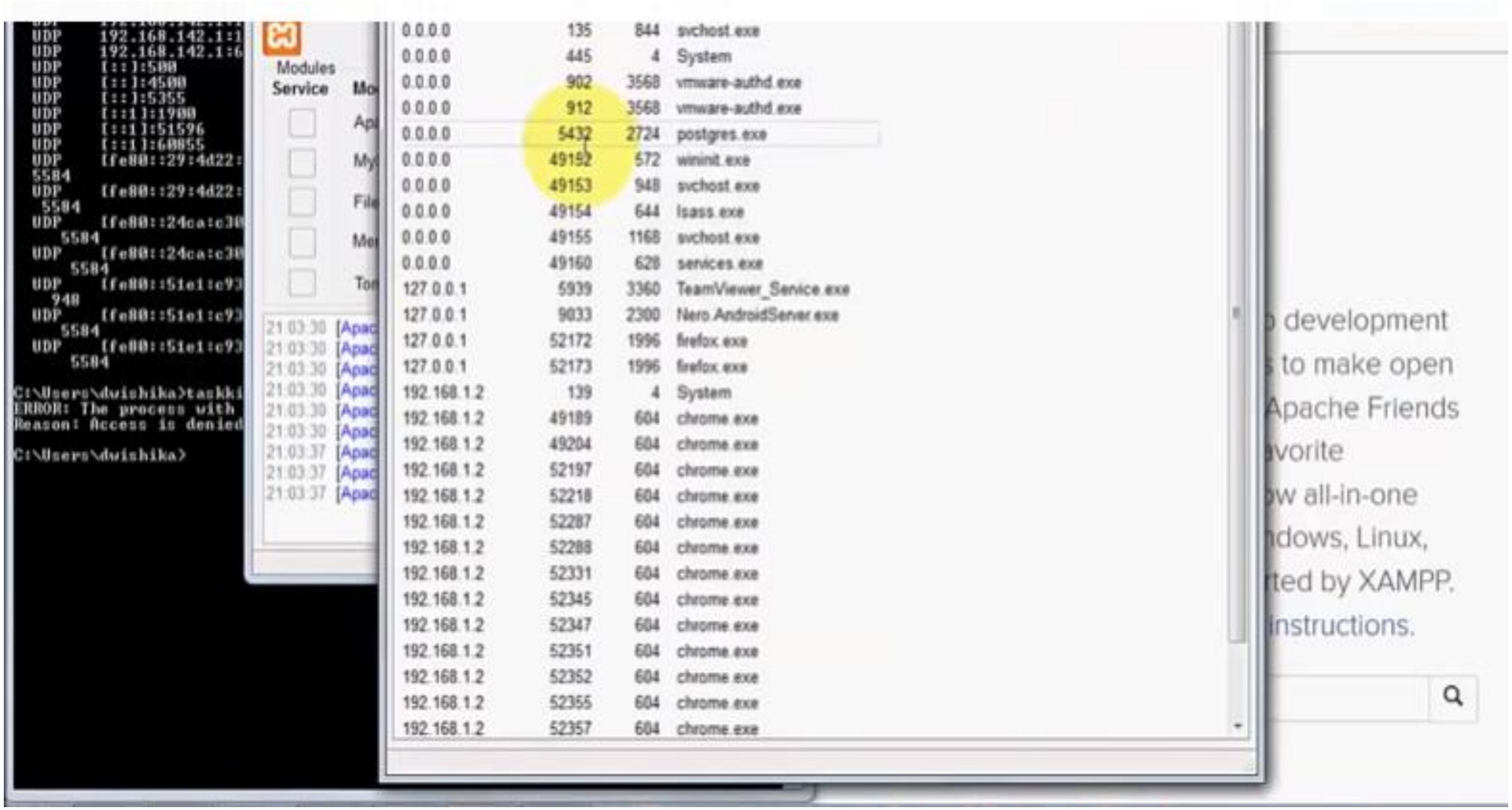
Taskkill /pid 4



```
  UDP    192.168.142.1:1
  UDP    192.168.142.1:6
  UDP    [::]:500
  UDP    [::]:4500
  UDP    [::]:5355
  UDP    [::1]:1900
  UDP    [::1]:51596
  UDP    [::1]:60055
  UDP    [fe80::29:4d22:
  5584
  UDP    [fe80::29:4d22:
   5584
  UDP    [fe80::24ca:c30
      5584
  UDP    [fe80::24ca:c30
       5584
  UDP    [fe80::51e1:c93
    948
  UDP    [fe80::51e1:c93
      5584
  UDP    [fe80::51e1:c93
       5584

C:\Users\dwishika>taskki
ERROR: The process with
Reason: Access is denied

C:\Users\dwishika>
```

```
21:03:30  [Apache]   Port 80 in use by "Unable to open process" with PID 4!
21:03:30  [Apache]   Apache WILL NOT start without the configured ports free!
21:03:30  [Apache]   You need to uninstall/disable/reconfigure the blocking application
21:03:30  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
21:03:30  [Apache]   Attempting to start Apache app...
21:03:30  [Apache]   Status change detected: running
21:03:37  [Apache]   Attempting to stop Apache (PID: 6348)
21:03:37  [Apache]   Attempting to stop Apache (PID: 3812)
21:03:37  [Apache]   Status change detected: stopped
```

ailable web development
Bitnami is to make open
ered with Apache Friends
stall your favorite
nload below all-in-one
AMPP (Windows, Linux,
pps supported by XAMPP.
pplication and follow the installation instructions.

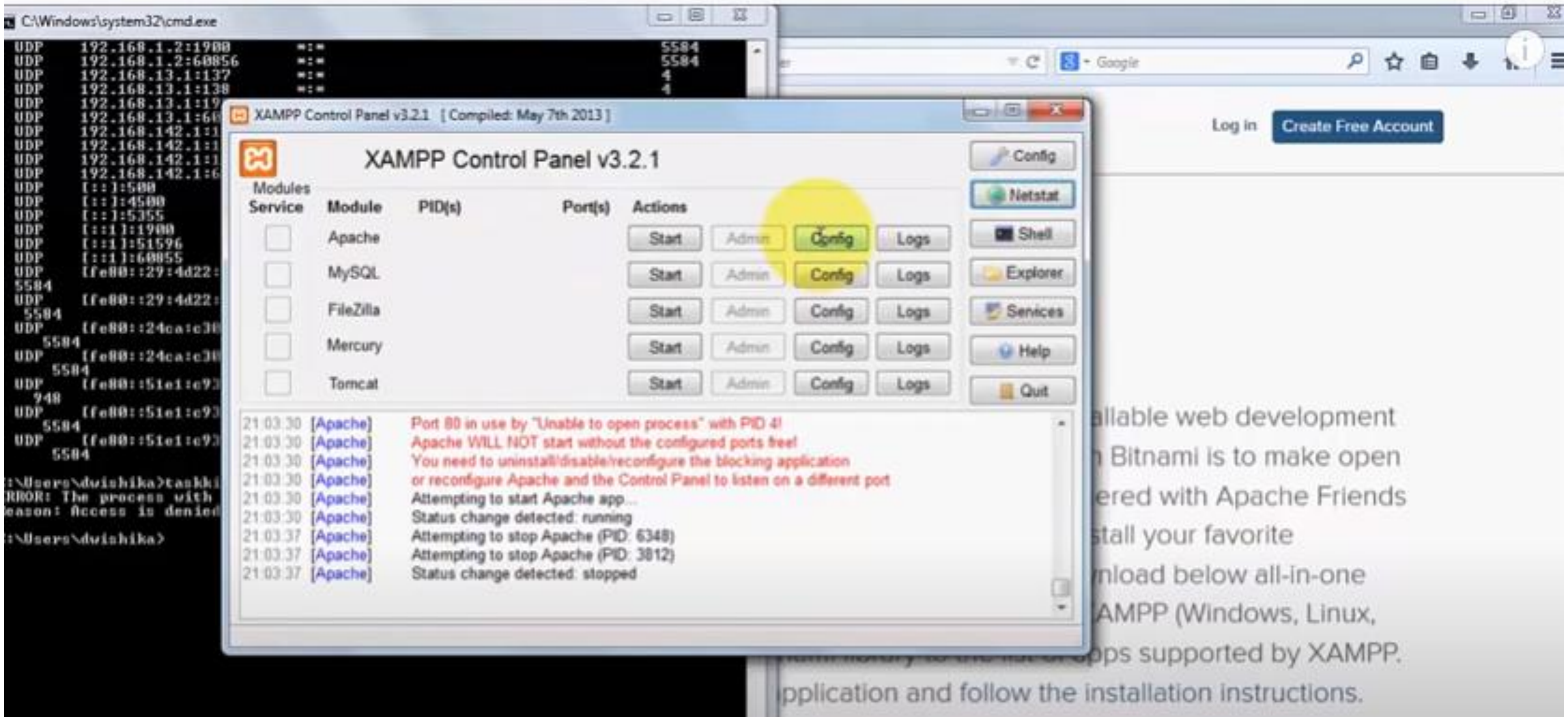


```
  UDP    192.168.142.1:1
  UDP    192.168.142.1:6
  UDP    [::]:500
  UDP    [::]:4500
  UDP    [::]:5355
  UDP    [::1]:1900
  UDP    [::1]:51596
  UDP    [::1]:60855
  UDP    [fe80::29:4d22:
  5584
  UDP    [fe80::29:4d22:
   5584
  UDP    [fe80::24ca:c30
     5584
  UDP    [fe80::24ca:c30
      5584
  UDP    [fe80::51e1:c93
    948
  UDP    [fe80::51e1:c93
     5584
  UDP    [fe80::51e1:c93
      5584

C:\Users\dwishika>taskki
ERROR: The process with
Reason: Access is denied

C:\Users\dwishika>
```

```
21:03:30 [Apac
21:03:30 [Apac
21:03:30 [Apac
21:03:30 [Apac
21:03:30 [Apac
21:03:30 [Apac
21:03:37 [Apac
21:03:37 [Apac
21:03:37 [Apac
```

```
0.0.0.0          135    844  svchost.exe
0.0.0.0          445      4  System
0.0.0.0          902   3568  vmware-authd.exe
0.0.0.0          912   3568  vmware-authd.exe
0.0.0.0         5432   2724  postgres.exe
0.0.0.0        49152    572  wininit.exe
0.0.0.0        49153    948  svchost.exe
0.0.0.0        49154    644  lsass.exe
0.0.0.0        49155   1168  svchost.exe
0.0.0.0        49160    628  services.exe
127.0.0.1       5939   3360  TeamViewer_Service.exe
127.0.0.1       9033   2300  Nero.AndroidServer.exe
127.0.0.1      52172   1996  firefox.exe
127.0.0.1      52173   1996  firefox.exe
192.168.1.2      139      4  System
192.168.1.2    49189    604  chrome.exe
192.168.1.2    49204    604  chrome.exe
192.168.1.2    52197    604  chrome.exe
192.168.1.2    52218    604  chrome.exe
192.168.1.2    52287    604  chrome.exe
192.168.1.2    52288    604  chrome.exe
192.168.1.2    52331    604  chrome.exe
192.168.1.2    52345    604  chrome.exe
192.168.1.2    52347    604  chrome.exe
192.168.1.2    52351    604  chrome.exe
192.168.1.2    52352    604  chrome.exe
192.168.1.2    52355    604  chrome.exe
192.168.1.2    52357    604  chrome.exe
```

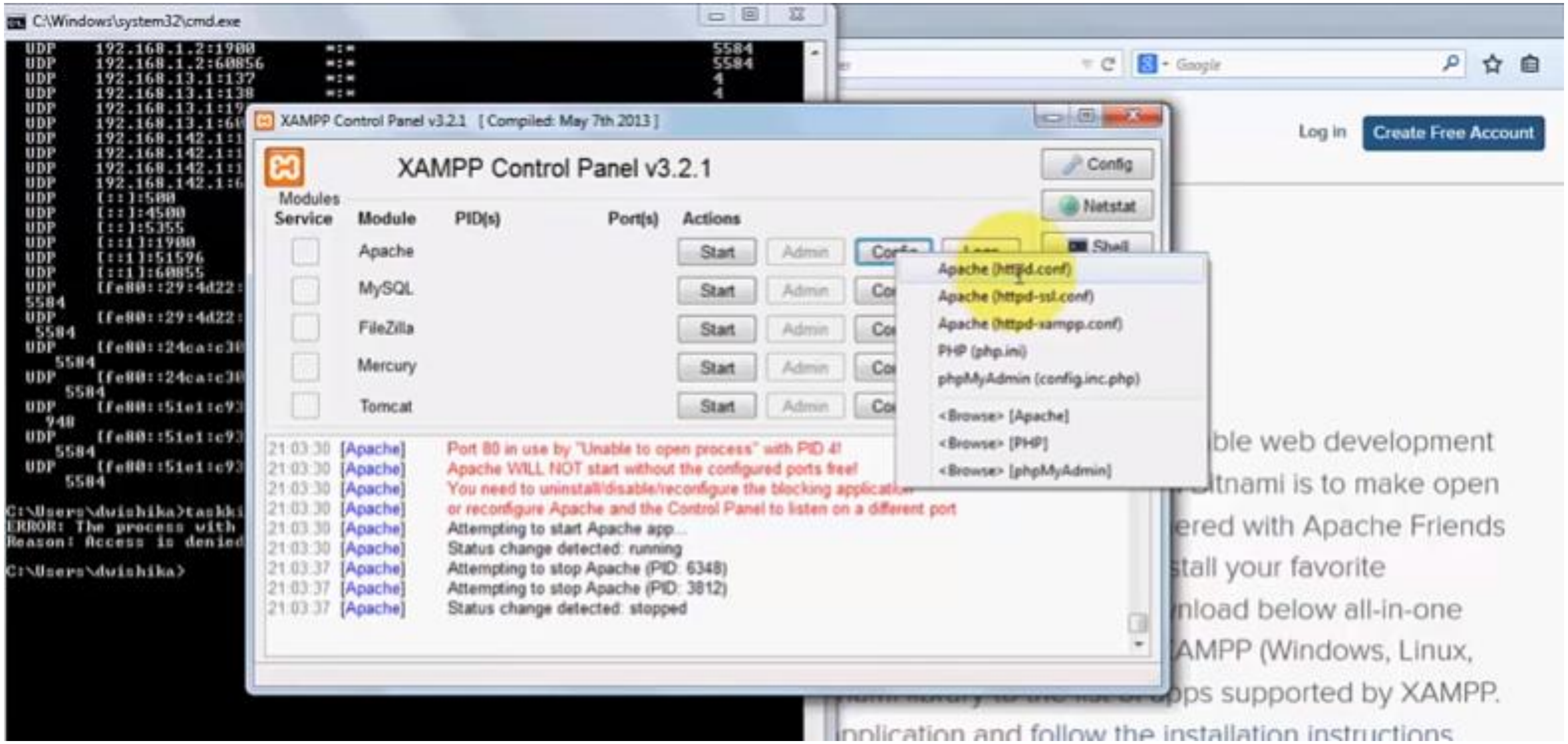
development
to make open
Apache Friends
vorite
w all-in-one
dows, Linux,
ted by XAMPP.
instructions.



```
UDP    192.168.1.2:1900        *:*                    5584
UDP    192.168.1.2:60856       *:*                    5584
UDP    192.168.13.1:137        *:*                    4
UDP    192.168.13.1:138        *:*                    4
UDP    192.168.13.1:19
UDP    192.168.13.1:60
UDP    192.168.142.1:1
UDP    192.168.142.1:1
UDP    192.168.142.1:1
UDP    192.168.142.1:6
UDP    [::]:500
UDP    [::]:4500
UDP    [::]:5355
UDP    [::1]:1900
UDP    [::1]:51596
UDP    [::1]:60855
UDP    [fe80::29:4d22:
5584
UDP    [fe80::29:4d22:
 5584
UDP    [fe80::24ca:c30
   5584
UDP    [fe80::24ca:c30
    5584
UDP    [fe80::51e1:c93
  948
UDP    [fe80::51e1:c93
   5584
UDP    [fe80::51e1:c93
    5584

:\Users\dwishika>taskki
RROR: The process with
eason: Access is denied

:\Users\dwishika>
```

```
21:03:30  [Apache]    Port 80 in use by "Unable to open process" with PID 4!
21:03:30  [Apache]    Apache WILL NOT start without the configured ports free!
21:03:30  [Apache]    You need to uninstall/disable/reconfigure the blocking application
21:03:30  [Apache]    or reconfigure Apache and the Control Panel to listen on a different port
21:03:30  [Apache]    Attempting to start Apache app...
21:03:30  [Apache]    Status change detected: running
21:03:37  [Apache]    Attempting to stop Apache (PID: 6348)
21:03:37  [Apache]    Attempting to stop Apache (PID: 3812)
21:03:37  [Apache]    Status change detected: stopped
```

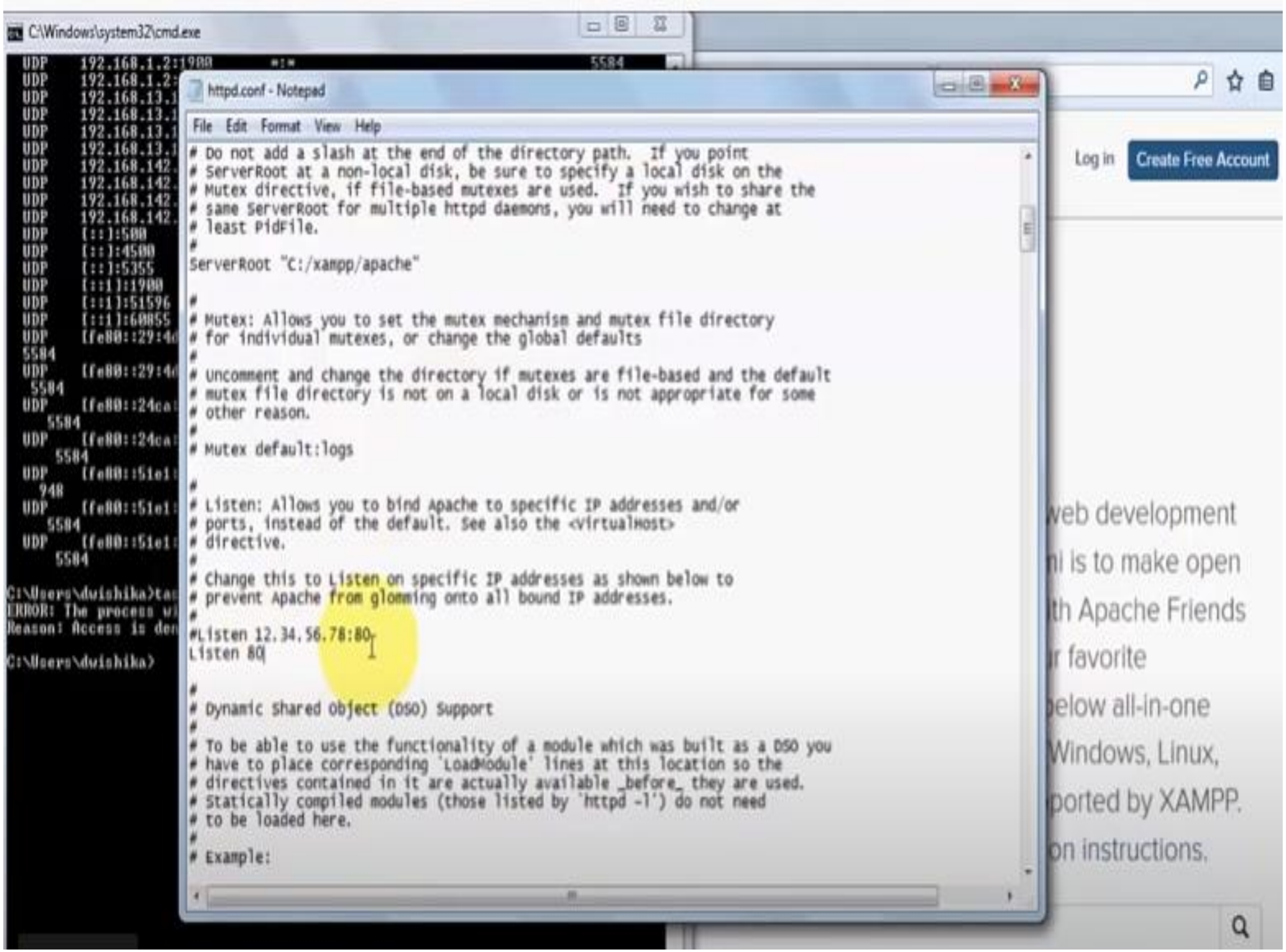
ailable web development
Bitnami is to make open
ered with Apache Friends
stall your favorite
nload below all-in-one
AMPP (Windows, Linux,
pps supported by XAMPP.
pplication and follow the installation instructions.

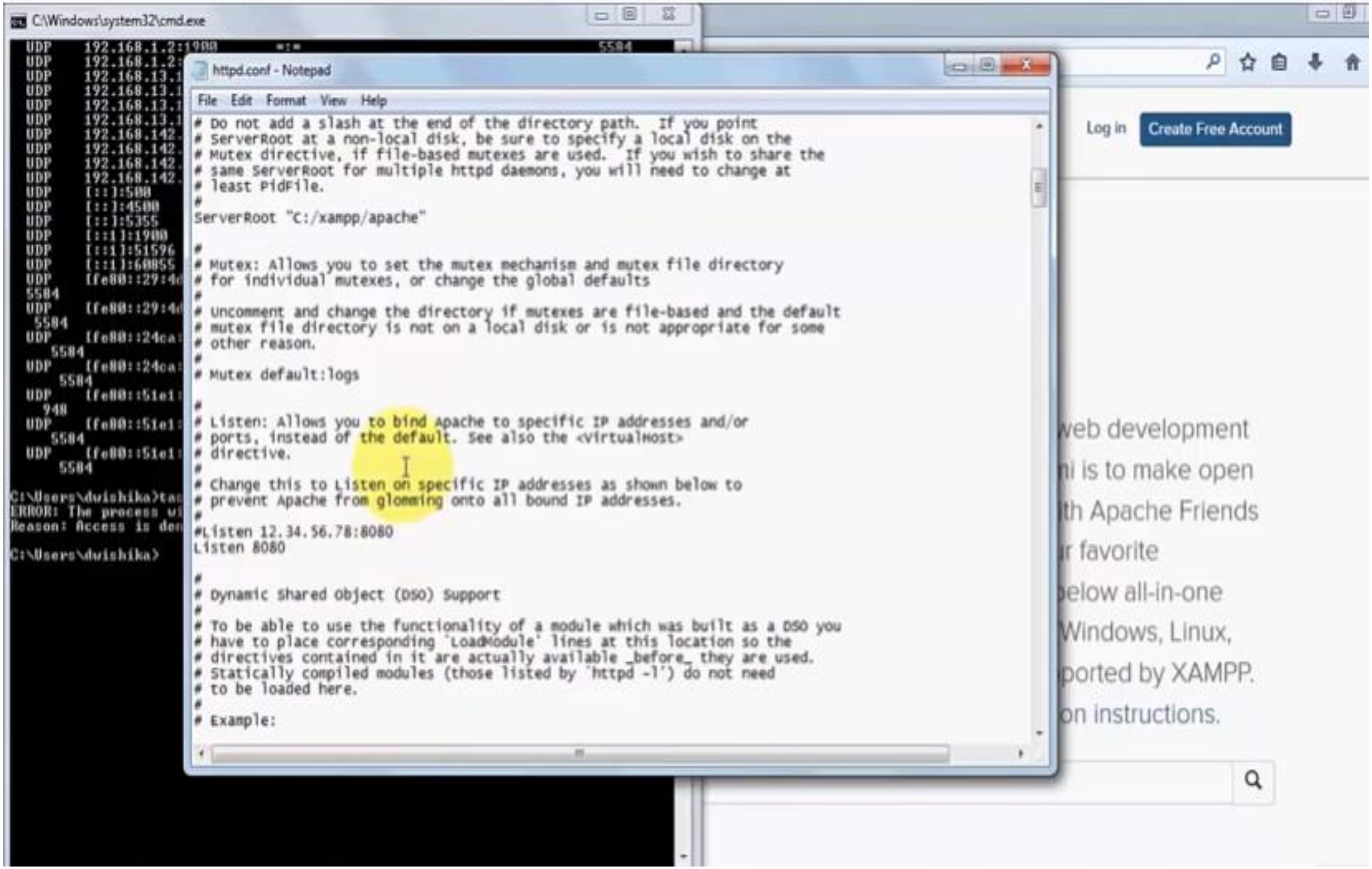


```
  UDP    192.168.1.2:1900       *:*                                    5584
  UDP    192.168.1.2:60856      *:*                                    5584
  UDP    192.168.13.1:137       *:*                                    4
  UDP    192.168.13.1:138       *:*                                    4
  UDP    192.168.13.1:19
  UDP    192.168.13.1:60
  UDP    192.168.142.1:1
  UDP    192.168.142.1:1
  UDP    192.168.142.1:1
  UDP    192.168.142.1:6
  UDP    [::]:500
  UDP    [::]:4500
  UDP    [::]:5355
  UDP    [::1]:1900
  UDP    [::1]:51596
  UDP    [::1]:60855
  UDP    [fe80::29:4d22:
  5584
  UDP    [fe80::29:4d22:
   5584
  UDP    [fe80::24ca:c30
     5584
  UDP    [fe80::24ca:c30
      5584
  UDP    [fe80::51e1:c93
    940
  UDP    [fe80::51e1:c93
     5584
  UDP    [fe80::51e1:c93
      5584

C:\Users\dwishika>taskki
ERROR: The process with
Reason: Access is denied

C:\Users\dwishika>
```

```
21:03:30  [Apache]   Port 80 in use by "Unable to open process" with PID 4!
21:03:30  [Apache]   Apache WILL NOT start without the configured ports free!
21:03:30  [Apache]   You need to uninstall/disable/reconfigure the blocking applicati
21:03:30  [Apache]   or reconfigure Apache and the Control Panel to listen on a different port
21:03:30  [Apache]   Attempting to start Apache app...
21:03:30  [Apache]   Status change detected: running
21:03:37  [Apache]   Attempting to stop Apache (PID: 6348)
21:03:37  [Apache]   Attempting to stop Apache (PID: 3812)
21:03:37  [Apache]   Status change detected: stopped
```

ble web development
itnami is to make open
ered with Apache Friends
stall your favorite
nload below all-in-one
AMPP (Windows, Linux,
pps supported by XAMPP.
pplication and follow the installation instructions





```
  UDP    192.168.1.2:1900       *:*                    5584
  UDP    192.168.1.2:
  UDP    192.168.13.1
  UDP    192.168.13.1
  UDP    192.168.13.1
  UDP    192.168.13.1
  UDP    192.168.142.
  UDP    192.168.142.
  UDP    192.168.142.
  UDP    192.168.142.
  UDP    [::]:500
  UDP    [::]:4500
  UDP    [::]:5355
  UDP    [::1]:1900
  UDP    [::1]:51596
  UDP    [::1]:60855
  UDP    [fe80::29:4d
 5584
  UDP    [fe80::29:4d
  5584
  UDP    [fe80::24ca:
    5584
  UDP    [fe80::24ca:
     5584
  UDP    [fe80::51e1:
   948
  UDP    [fe80::51e1:
    5584
  UDP    [fe80::51e1:
     5584

C:\Users\dwishika>tas
ERROR: The process wi
Reason: Access is den

C:\Users\dwishika>
```

```
# Do not add a slash at the end of the directory path.  If you point
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used.  If you wish to share the
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
ServerRoot "C:/xampp/apache"


#
# Mutex: Allows you to set the mutex mechanism and mutex file directory
# for individual mutexes, or change the global defaults
#
# Uncomment and change the directory if mutexes are file-based and the default
# mutex file directory is not on a local disk or is not appropriate for some
# other reason.
#
# Mutex default:logs


#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:8080
Listen 8080


#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Statically compiled modules (those listed by `httpd -l') do not need
# to be loaded here.
#
# Example:
```

Php Demo

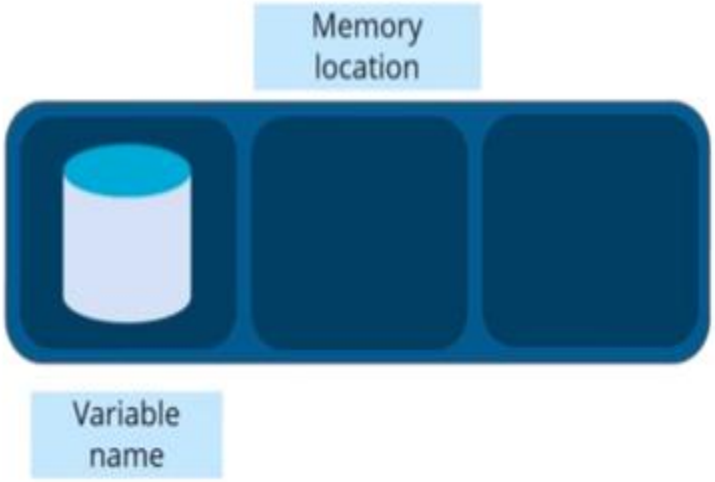
- Samp1.php

Variables

Variable is a name given to a memory location which acts as a container for storing data.

Syntax:

```
$age = 22;
```



All variables in PHP start with a leading dollar sign like $variable_name. To assign a variable, use the = operator, with the name of the variable on the left and the expression to be evaluated on the right.

Rules for PHP variables

- Variable declarations starts with $, followed by the name of the variable
- Variable names can only start with an upper or lowercase letter or an underscore (_)
- Variable names can only contain letters, numbers, or underscores (A-z, 0-9, and _). Other special characters like + - % ( ) . & are invalid
- Variable names are case sensitive

Some examples of allowed variable names:

- $my_variable
- $anotherVariable
- $the2ndVariable

Predefined Variables

- PHP has several special keywords that, while they are "valid" variable names, cannot be used for your variables. The reason for this is that the language itself has already defined those variables and they have are used for special purposes. Several examples are listed below, for a complete list see the PHP documentation site.
- $this
- $_GET
- $_POST
- $_SERVER
- $_FILES

Programs Demo

Var1.php

Predefined variable Demo

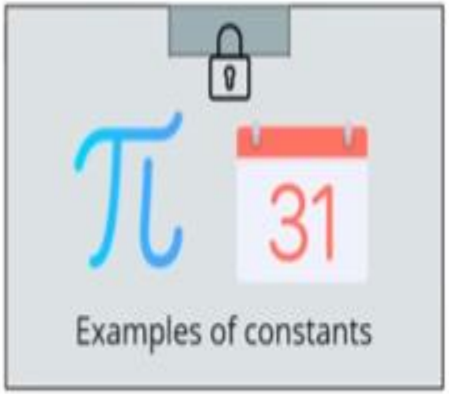
- Predefvar2.php

Constants

Constants are fixed values that do not change during execution time.

Syntax:

```
define ('PI', 3.1415926);
```



Examples of constants

Constants are a type of variable in PHP. The define() function to set a constant takes three arguments - the key name, the key's value, and a Boolean (true or false) which determines whether the key's name is case-insensitive (false by default).

A constant's value cannot be altered once it is set. It is used for values which rarely change (for example a database password OR API key).

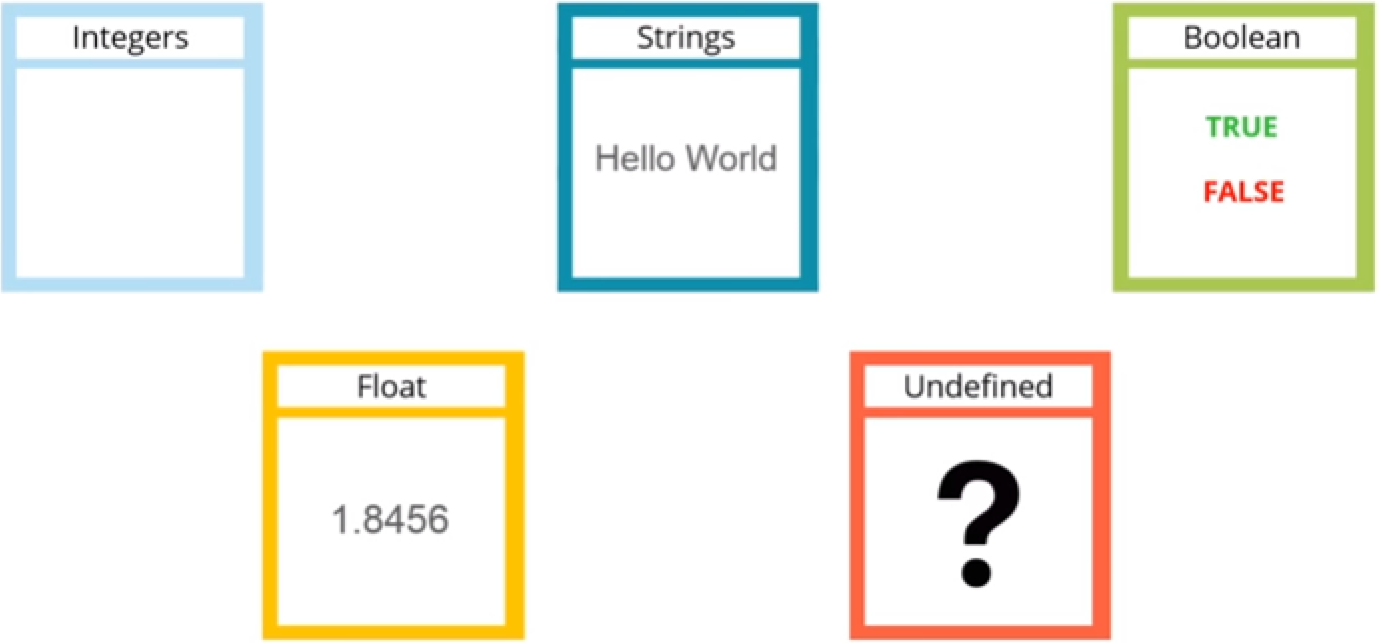
Scope

It is important to know that unlike variables, constants ALWAYS have a global scope and can be accessed from any function in the script.

Programs demo

Const1.php

Primitive Data Types



Integers

Strings

Hello World

Boolean

TRUE

FALSE

Float

1.8456

Undefined

?

Variables can store data of different types such as:

- String ("Hello")
- Integer (5)
- Float (also called double) (1.0)
- Boolean (1 or 0)
- Array (array("I", "am", "an", "array"))
- Object
- NULL
- Resource

Strings

- A string is a sequence of characters. It can be any text inside quotes (single or double):
- $x = "Hello!";
- $y = 'Hello!';

Integers

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

Rules for integers:

- Integers must have at least one digit
- Integers must not have a decimal point
- Integers can be either positive or negative
- $x = 5;

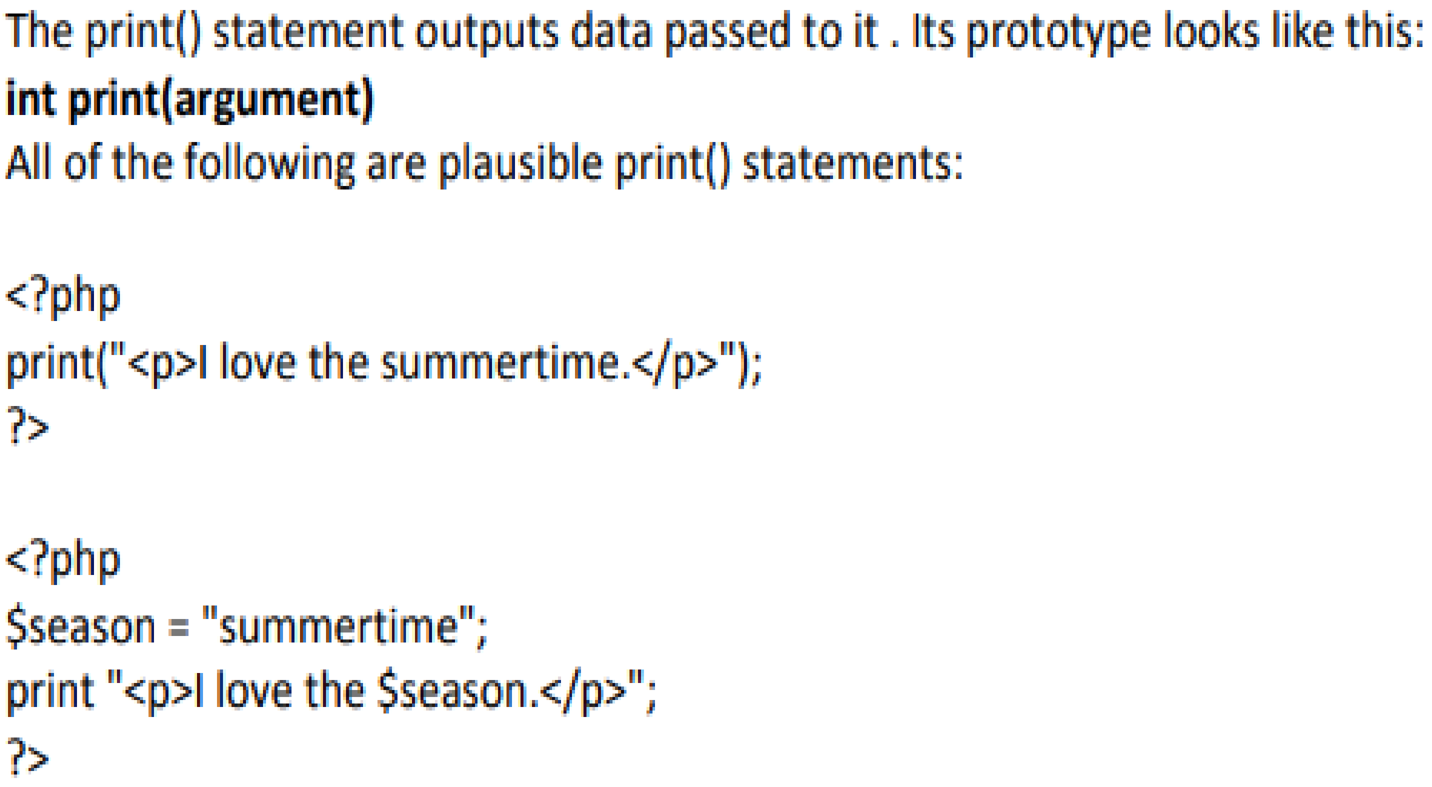
Floats

- A float, or floating point number, is a number with a decimal point.
- $x = 5.01;

Booleans

- A Boolean represents two possible states: TRUE or FALSE. Booleans are often used in conditional testing.
- $x = true; $y = false;

SAMPLE SNIPPETS BEFORE DOING PROGRAMS



The print() statement outputs data passed to it . Its prototype looks like this:

int print(argument)

All of the following are plausible print() statements:

```
<?php
print("<p>I love the summertime.</p>");
?>

<?php
$season = "summertime";
print "<p>I love the $season.</p>";
?>
```

Alternatively, you could use the echo() statement for the same purposes as print().echo()'s prototype looks like this:
void echo(string argument1 [, ...string argumentN])

To use echo(), just provide it with an argument just as was done with print():

```
echo "I love the summertime.";
```

As you can see from the prototype, echo() is capable of outputting multiple strings. Here's an example:

```
<?php
$heavyweight = "Lennox Lewis";
$lightweight = "Floyd Mayweather";
echo $heavyweight, " and ", $lightweight, " are great fighters.";
?>
```

This code produces the following:

```
Lennox Lewis and Floyd Mayweather are great fighters.
```

The printf() statement is ideal when you want to output a blend of static text and dynamic information stored within one or several variables. It's ideal for two reasons.

- First, it neatly separates the static and dynamic data into two distinct sections, allowing for easy maintenance.
- Second, printf() allows you to wield considerable control over how the dynamic information is rendered to the screen in terms of its type, precision, alignment, and position.

Its prototype looks like this:
integer printf(string format [, mixed args])
For example, suppose you wanted to insert a single dynamic integer value into an otherwise static string:

```
printf("Bar inventory: %d bottles of tonic water.", 100);
```

Commonly Used Type Specifiers

%b Argument considered an integer; presented as a binary number

%c Argument considered an integer; presented as a character corresponding to that ASCII value

%d Argument considered an integer; presented as a signed decimal number

%f Argument considered a floating-point number; presented as a floating-point number

%o Argument considered an integer; presented as an octal number

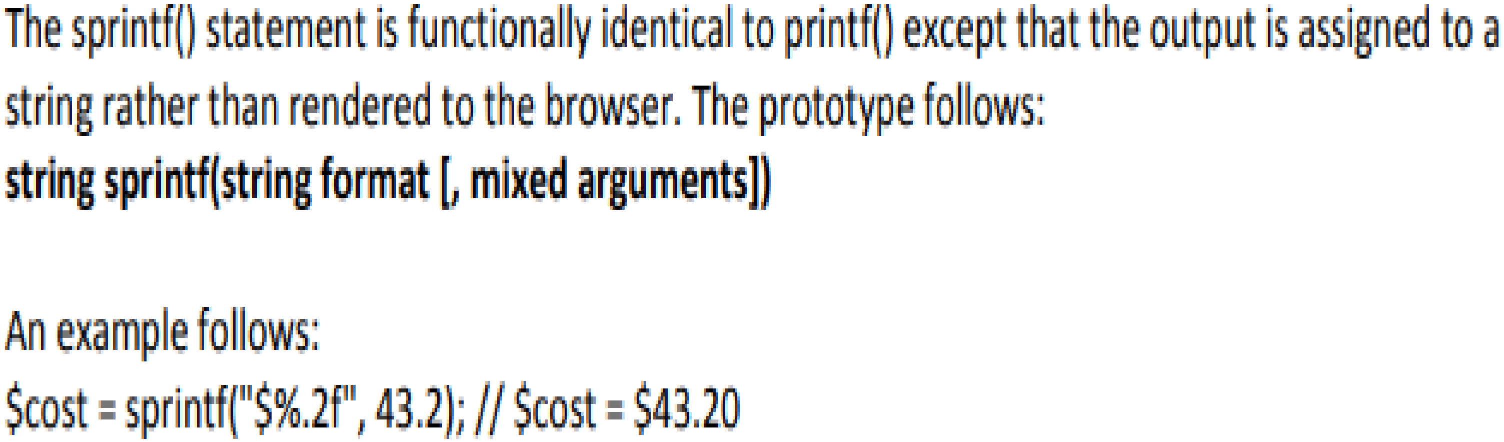
%s Argument considered a string; presented as a string

%u Argument considered an integer; presented as an unsigned decimal number

%x Argument considered an integer; presented as a lowercase hexadecimal number

%X Argument considered an integer; presented as an uppercase hexadecimal number

Sprintf()

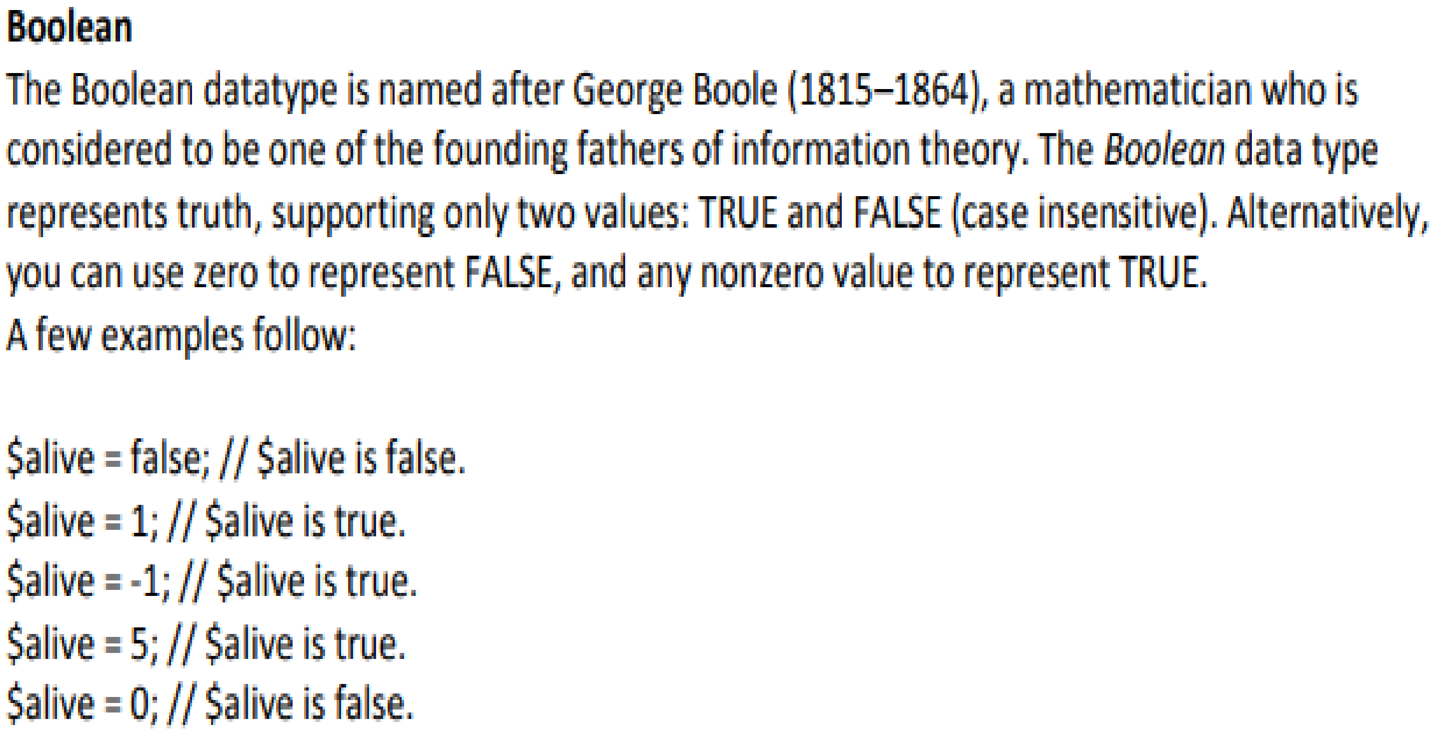


The sprintf() statement is functionally identical to printf() except that the output is assigned to a string rather than rendered to the browser. The prototype follows:

string sprintf(string format [, mixed arguments])

An example follows:

$cost = sprintf("$%.2f", 43.2); // $cost = $43.20

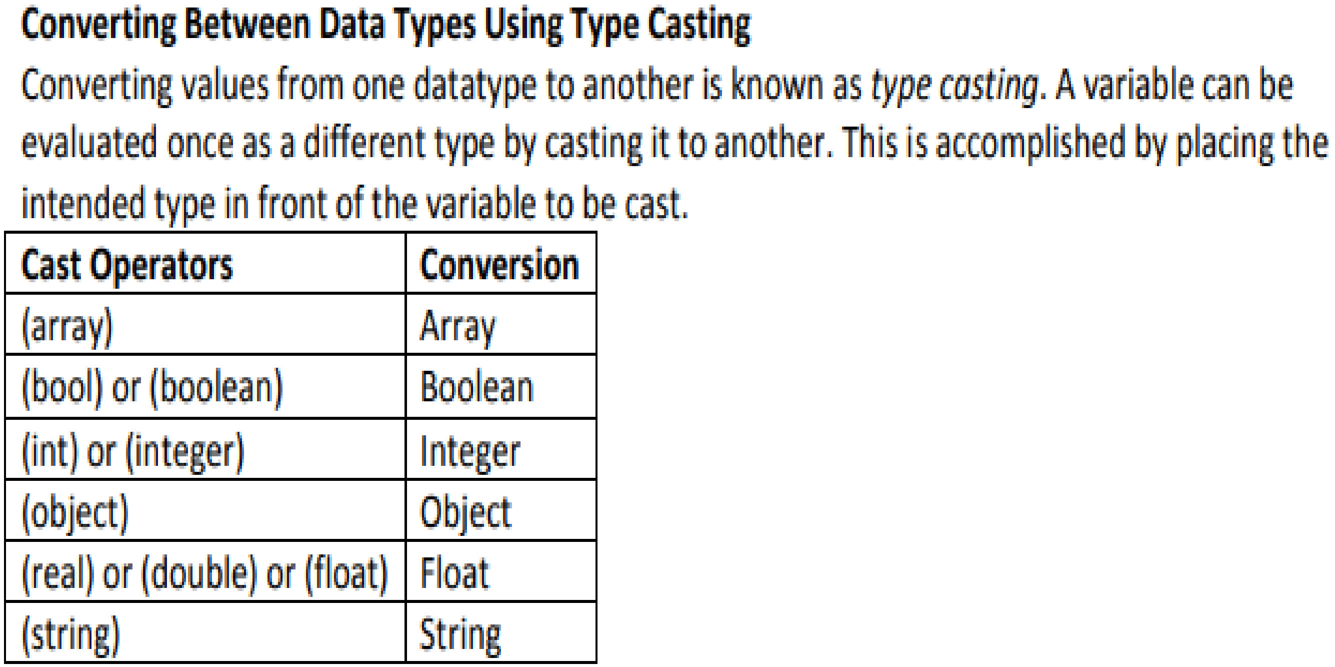


Boolean

The Boolean datatype is named after George Boole (1815–1864), a mathematician who is considered to be one of the founding fathers of information theory. The *Boolean* data type represents truth, supporting only two values: TRUE and FALSE (case insensitive). Alternatively, you can use zero to represent FALSE, and any nonzero value to represent TRUE.

A few examples follow:

```
$alive = false; // $alive is false.
$alive = 1; // $alive is true.
$alive = -1; // $alive is true.
$alive = 5; // $alive is true.
$alive = 0; // $alive is false.
```



Converting Between Data Types Using Type Casting

Converting values from one datatype to another is known as *type casting*. A variable can be evaluated once as a different type by casting it to another. This is accomplished by placing the intended type in front of the variable to be cast.

Cast Operators	**Conversion**
(array)	Array
(bool) or (boolean)	Boolean
(int) or (integer)	Integer
(object)	Object
(real) or (double) or (float)	Float
(string)	String

Let’s consider several examples. Suppose you’d like to cast an integer as a double:

```
$score = (double) 13; // $score = 13.0
```

Type casting a double to an integer will result in the integer value being rounded down, regardless of the decimal value. Here’s an example:

```
$score = (int) 14.8; // $score = 14
```

What happens if you cast a string datatype to that of an integer? Let’s find out:

```
$sentence = "This is a sentence";
echo (int) $sentence; // returns 0
```

Adapting Data Types with Type Juggling

Because of PHP's lax attitude toward type definitions, variables are sometimes automatically cast to best fit the circumstances in which they are referenced. Consider the following snippet:

```
<?php
$total = 5; // an integer
$count = "15"; // a string
$total += $count; // $total = 20 (an integer)
?>
```

```
<?php
$total = "45 fire engines";
$incoming = 10;
$total = $incoming + $total; // $total = 55
?>
```

The integer value at the beginning of the original $total string is used in the calculation. However, if it begins with anything other than a numerical representation, the value is 0.

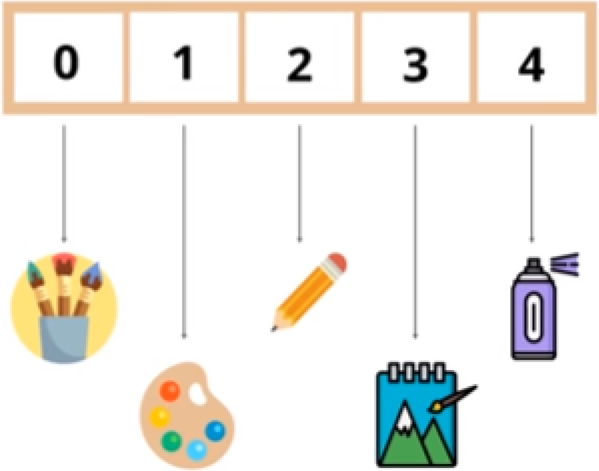
Consider another example:

```
<?php
$total = "1.0";
if ($total) echo "We're in positive territory!";
?>
```

In this example, a string is converted to Boolean type in order to evaluate the *if* statement.

Reference Data Types - Arrays

An array is a data structure that contains a list of elements. These elements are all of the same data type, such as an integer or string.



Syntax:

```
$chocolates = array('Eclairs', 'Mars', 'KitKat');

```

For example, if you were interested in creating a list of U.S. states, you could use a numerically indexed array, like so:

```
$state[0] = "Alabama";
$state[1] = "Alaska";
$state[2] = "Arizona";
...
$state[49] = "Wyoming";
```

But what if the project required correlating U.S. states to their capitals? Rather than base the keys on a numerical index, you might instead use an associative index, like this:

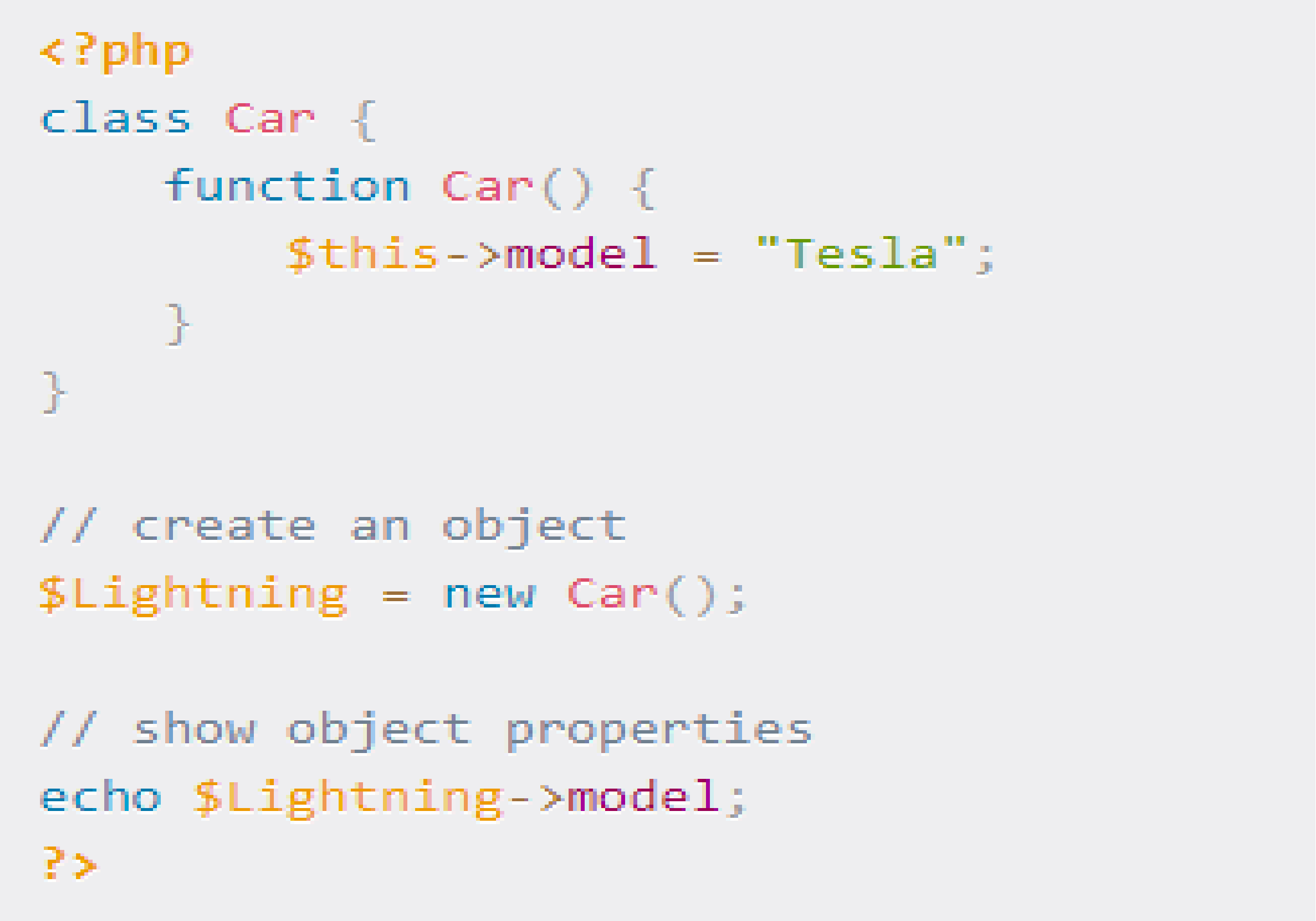
```
$state["Alabama"] = "Montgomery";
$state["Alaska"] = "Juneau";
$state["Arizona"] = "Phoenix";
...
$state["Wyoming"] = "Cheyenne";
```

NULL

- Null is a special data type that can only have the value null. Variables can be declared with no value or emptied by setting the value to null. Also, if a variable is created without being assigned a value, it is automatically assigned null.
- **<?php** // Assign the value "Hello!" to greeting $greeting = "Hello!"; // Empty the value greeting by setting it to null $greeting = null; **?>**

Classes and Objects

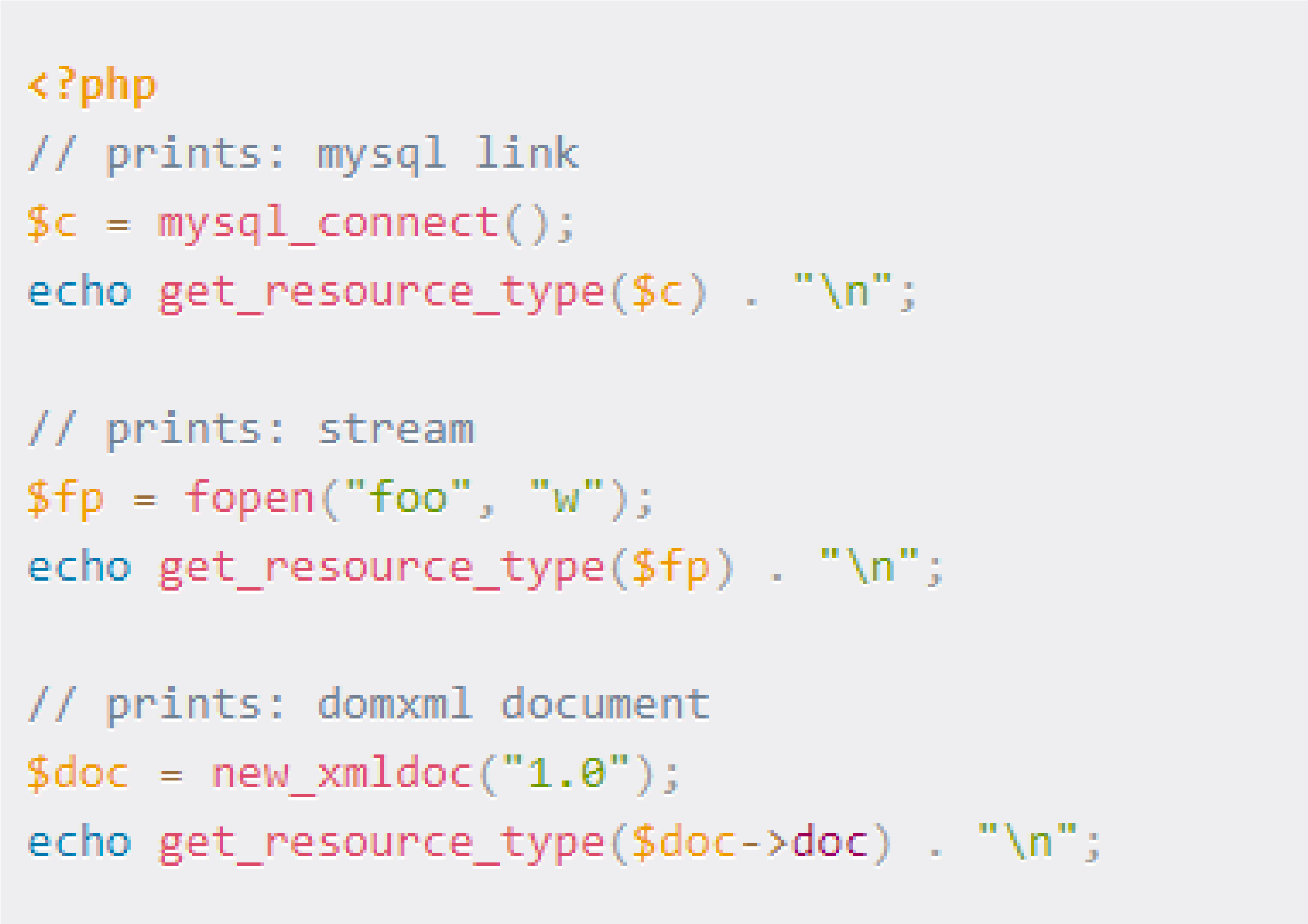
- A class is a data structure useful for modeling things in the real world, and can contain properties and methods. Objects are instances a class, and are a convenient way to package values and functions specific to a class.



```
<?php
class Car {
    function Car() {
        $this->model = "Tesla";
    }
}

// create an object
$Lightning = new Car();

// show object properties
echo $Lightning->model;
?>
```

PHP Resource

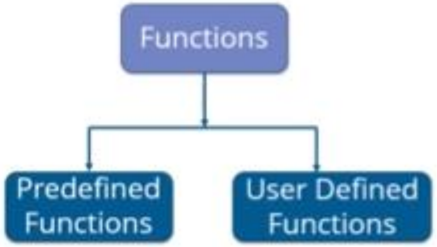
- A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions. You can use get*resource*type() function to see resource type.



```
<?php
// prints: mysql link
$c = mysql_connect();
echo get_resource_type($c) . "\n";

// prints: stream
$fp = fopen("foo", "w");
echo get_resource_type($fp) . "\n";

// prints: domxml document
$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc) . "\n";
```

Reference Data Types - Functions

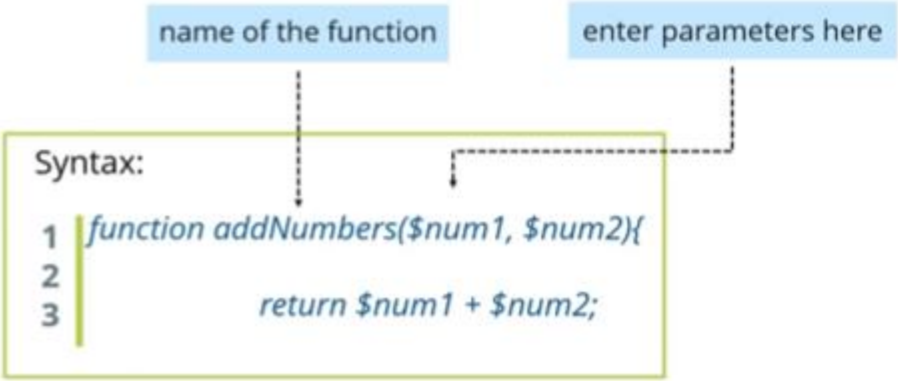
A **function** is a block of organized, reusable code that is used to perform a single, related action.



Functions

Predefined Functions

User Defined Functions

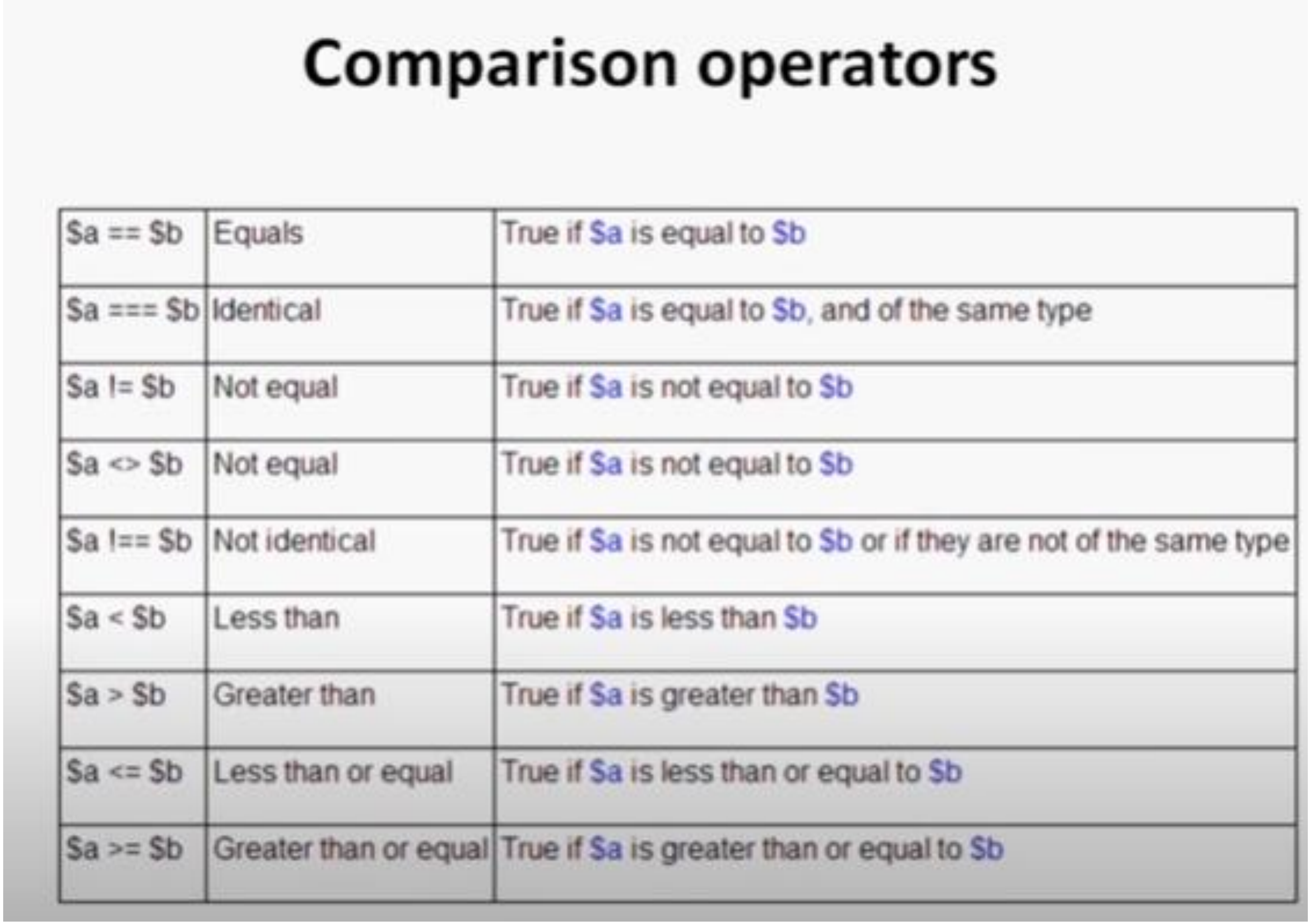


name of the function

enter parameters here

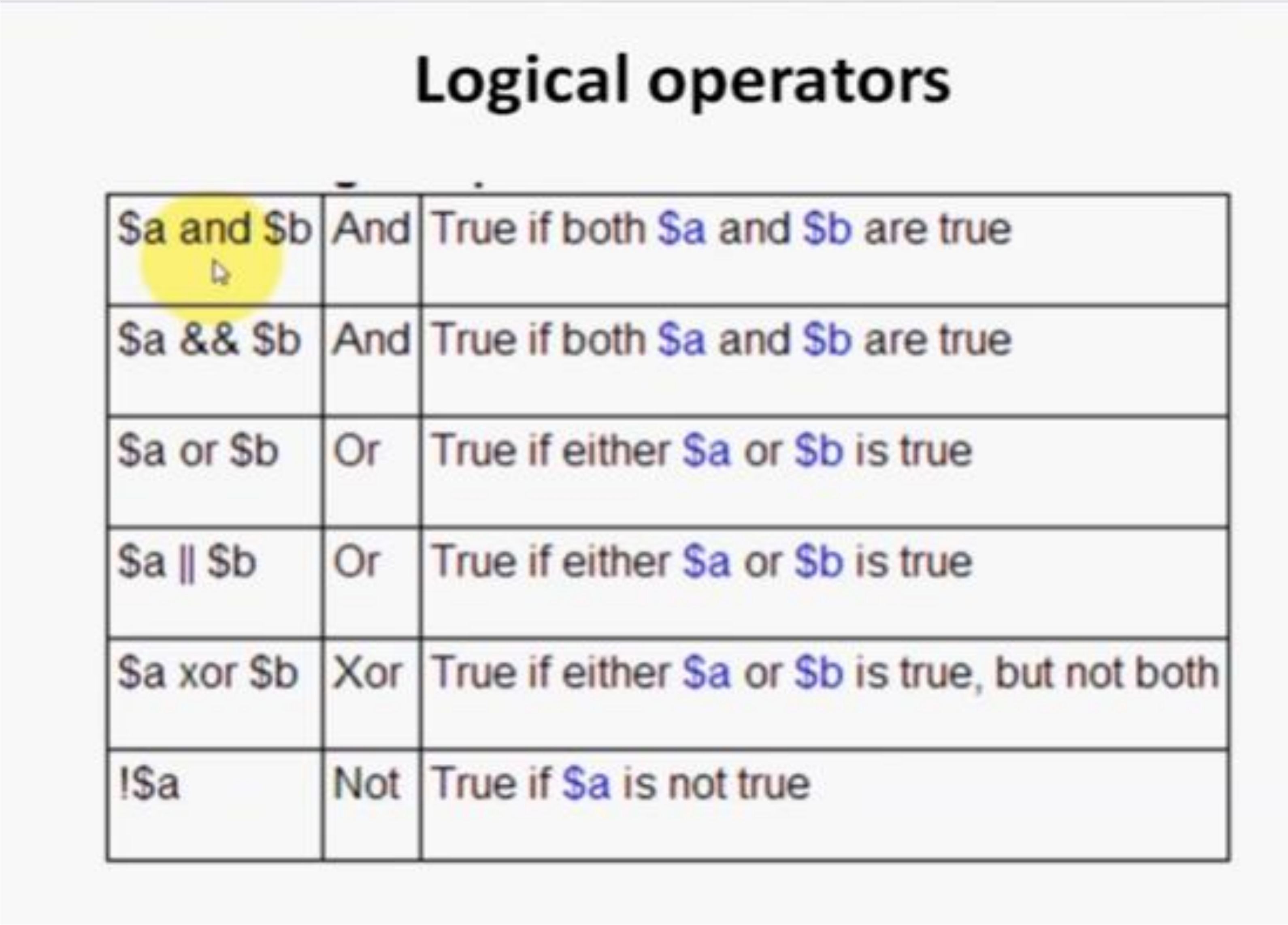
Syntax:

```
function addNumbers($num1, $num2){

            return $num1 + $num2;
```



Comparison operators

$a == $b	Equals	True if $a is equal to $b
$a === $b	Identical	True if $a is equal to $b, and of the same type
$a != $b	Not equal	True if $a is not equal to $b
$a <> $b	Not equal	True if $a is not equal to $b
$a !== $b	Not identical	True if $a is not equal to $b or if they are not of the same type
$a < $b	Less than	True if $a is less than $b
$a > $b	Greater than	True if $a is greater than $b
$a <= $b	Less than or equal	True if $a is less than or equal to $b
$a >= $b	Greater than or equal	True if $a is greater than or equal to $b



Logical operators

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

Reference Assignment

PHP 4 introduced the ability to assign variables by reference, which essentially means that you can create a variable that refers to the same content as another variable does. Therefore, a change to any variable referencing a particular item of variable content will be reflected among all other variables referencing that same content. You can assign variables by reference by appending an ampersand (&) to the equal sign. Let's consider an example:

```
<?php
$value1 = "Hello";
$value2 =& $value1; // $value1 and $value2 both equal "Hello"
$value2 = "Goodbye"; // $value1 and $value2 both equal "Goodbye"
?>
```

An alternative reference-assignment syntax is also supported, which involves appending the
ampersand to the front of the variable being referenced. The following example adheres to this new
syntax:

```
<?php
$value1 = "Hello";
$value2 = &$value1; // $value1 and $value2 both equal "Hello"
$value2 = "Goodbye"; // $value1 and $value2 both equal "Goodbye"
?>
```

```
<?php
  $x = 4;

  function assignx () {
    $x = 0;
    print "\$x inside function is $x. <br />";
  }

  assignx();
  print "\$x outside of function is $x. <br />";
?>
```

This will produce the following result –

```
$x inside function is 0.
$x outside of function is 4.
```

Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword **GLOBAL** in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example –

```
 <?php
   $somevar = 15;

  function addit() {
    GLOBAL $somevar;
    $somevar++;

    print "Somevar is $somevar";
  }

  addit();
?>
```

This will produce the following result −

```
Somevar is 16
```

The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

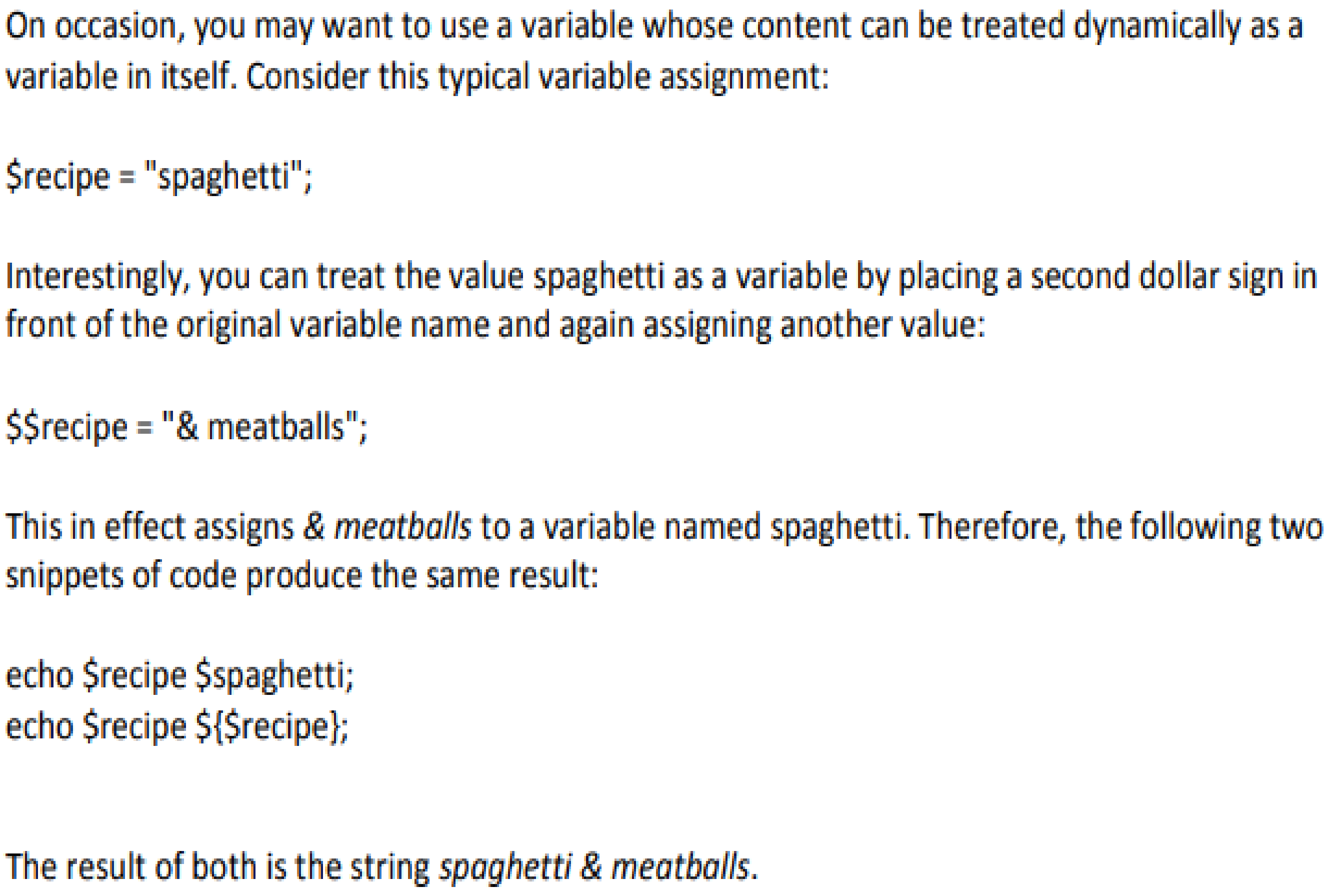
You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

```
<?php
  function keep_track() {
    STATIC $count = 0;
    $count++;
    print $count;
    print "<br />";
  }

  keep_track();
  keep_track();
  keep_track();
?>
```

This will produce the following result –

```
1
2
3
```

Variable Variables



On occasion, you may want to use a variable whose content can be treated dynamically as a variable in itself. Consider this typical variable assignment:

```
$recipe = "spaghetti";
```

Interestingly, you can treat the value spaghetti as a variable by placing a second dollar sign in front of the original variable name and again assigning another value:

```
$$recipe = "& meatballs";
```

This in effect assigns *& meatballs* to a variable named spaghetti. Therefore, the following two snippets of code produce the same result:

```
echo $recipe $spaghetti;
echo $recipe ${$recipe};
```

The result of both is the string *spaghetti & meatballs*.

String Interpolation

To offer developers the maximum flexibility when working with string values, PHP offers a means for both literal and figurative interpretation. For example, consider the following string:

The $animal jumped over the wall.\n

You might assume that $animal is a variable and that \n is a newline character, and therefore both should be interpreted accordingly. However, what if you want to output the string exactly as it is written, or perhaps you want the newline to be rendered but want the variable to display in its literal form ($animal), or vice versa? All of these variations are possible in PHP, depending on how the strings are enclosed and whether certain key characters are escaped through a predefined sequence.

Double Quotes

Strings enclosed in double quotes are the most commonly used in PHP scripts because they offer the most flexibility. This is because both variables and escape sequences will be parsed accordingly.

Consider the following example:

```
<?php
$sport = "boxing";
echo "Jason's favorite sport is $sport.";
?>
```

This example returns the following:

```
Jason's favorite sport is boxing.
```

Escape sequences are also parsed. Consider this example:

```
<?php
$output = "This is one line.\nAnd this is another line.";
echo $output;
?>
```

This returns the following (as viewed from within the browser source):

```
This is one line.
And this is another line.
```

Single Quotes

Enclosing a string within single quotes is useful when the string should be interpreted exactly as stated. This means that both variables and escape sequences will not be interpreted when the string is parsed.

For example, consider the following single-quoted string:

```
print 'This string will $print exactly as it\'s \n declared.';
```

This produces the following:

```
This string will $print exactly as it's \n declared.
```

Curly Braces

While PHP is perfectly capable of interpolating variables representing scalar data types, you'll find that variables representing complex data types such as arrays or objects cannot be so easily parsed when embedded in an echo() or print() string. You can solve this issue by delimiting the variable in curly braces, like this:

```
echo "The capital of Ohio is {$capitals['ohio']}.";
```

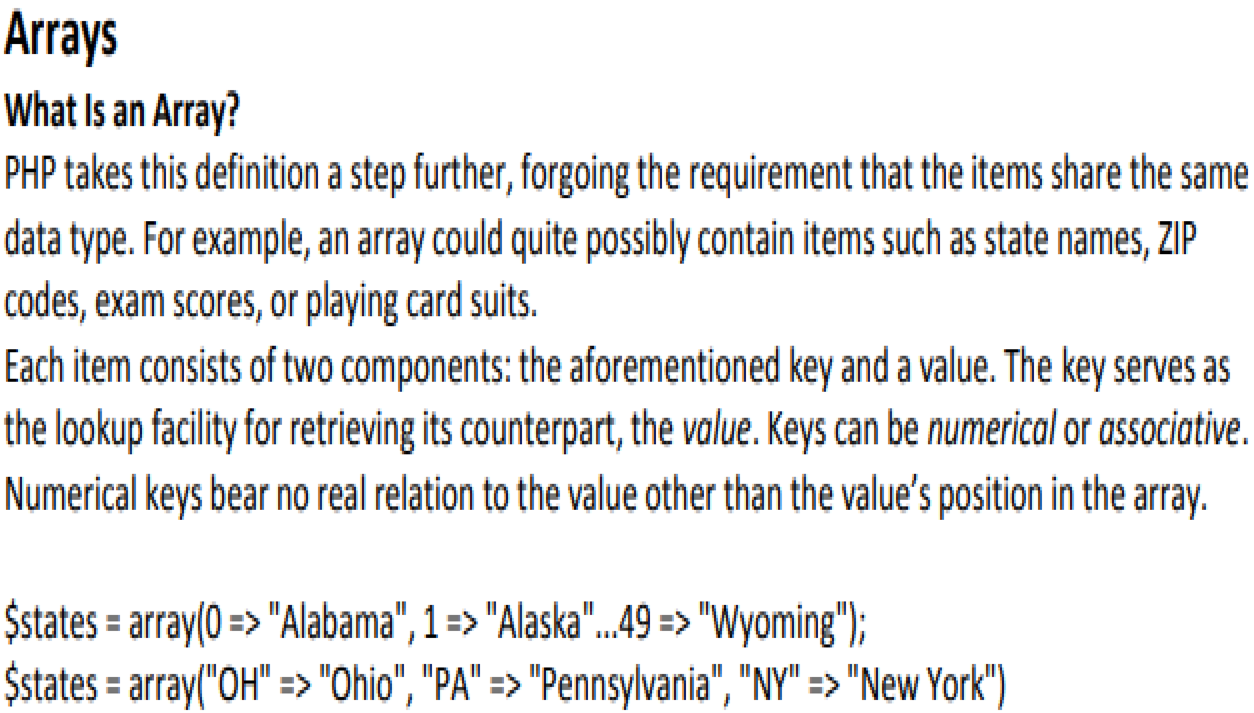
Heredoc

Heredoc syntax offers a convenient means for outputting large amounts of text. Rather than delimiting strings with double or single quotes, two identical identifiers are employed. An example follows:

```
<?php
$website = "http://www.romatermini.it";
echo <<<EXCERPT
<p>Rome's central train station, known as <a href = "$website">Roma Termini</a>,
was built in 1867. Because it had fallen into severe disrepair in the late 20th
century, the government knew that considerable resources were required to
rehabilitate the station prior to the 50-year <i>Giubileo</i>.</p>
EXCERPT;
?>
```

PROGRAMS DEMO

PHP ARRAY



Arrays

What Is an Array?

PHP takes this definition a step further, forgoing the requirement that the items share the same data type. For example, an array could quite possibly contain items such as state names, ZIP codes, exam scores, or playing card suits.

Each item consists of two components: the aforementioned key and a value. The key serves as the lookup facility for retrieving its counterpart, the *value*. Keys can be *numerical* or *associative*. Numerical keys bear no real relation to the value other than the value's position in the array.

```
$states = array(0 => "Alabama", 1 => "Alaska"...49 => "Wyoming");
$states = array("OH" => "Ohio", "PA" => "Pennsylvania", "NY" => "New York")
```

Creating an Array

Unlike other languages, PHP doesn’t require that you assign a size to an array at creation time. In fact, because it’s a loosely typed language, PHP doesn’t even require that you declare the array before using it.

```
$state[0] = "Delaware";
```

Interestingly, if you intend for the index value to be numerical and ascending, you can omit the index value at creation time:

```
$state[] = "Pennsylvania";
$state[] = "New Jersey";
...
$state[] = "Hawaii";
```

Populating Arrays with a Predefined Value Range

The range() function provides an easy way to quickly create and fill an array consisting of a range of low and high integer values. An array containing all integer values in this range is returned. Its prototype looks like this:

array range(int *low*, int *high* [, int *step*])

For example, suppose you need an array consisting of all possible face values of a die:

```
$die = range(1, 6);
// Same as specifying $die = array(1, 2, 3, 4, 5, 6)
```

Testing for an Array

When you incorporate arrays into your application, you'll sometimes need to know whether a particular variable is an array. A built-in function, is_array(), is available for accomplishing this task. Its prototype follows:

boolean is_array(mixed *variable*)

The is_array() function determines whether variable is an array, returning TRUE if it is and FALSE otherwise.

An example follows:

```
$states = array("Florida");
$state = "Ohio";
printf("\$states is an array: %s <br />", (is_array($states) ? "TRUE" : "FALSE"));
```

Outputting an Array

The most common way to output an array's contents is by iterating over each key and echoing the corresponding value. For instance, a foreach statement does the trick nicely:

```
$states = array("Ohio", "Florida", "Texas");
foreach ($states AS $state) {
echo "{$state}<br />";
}
```

Adding and Removing Array Elements

PHP provides a number of functions for both growing and shrinking an array. Some of these functions are provided as a convenience to programmers who wish to mimic various queue implementations (FIFO, LIFO, etc.), as reflected by their names (push, pop, shift, and unshift).

Adding a Value to the Front of an Array

The array_unshift() function adds elements to the front of the array. All preexisting numerical keys are modified to reflect their new position in the array, but associative keys aren’t affected. Its prototype follows:

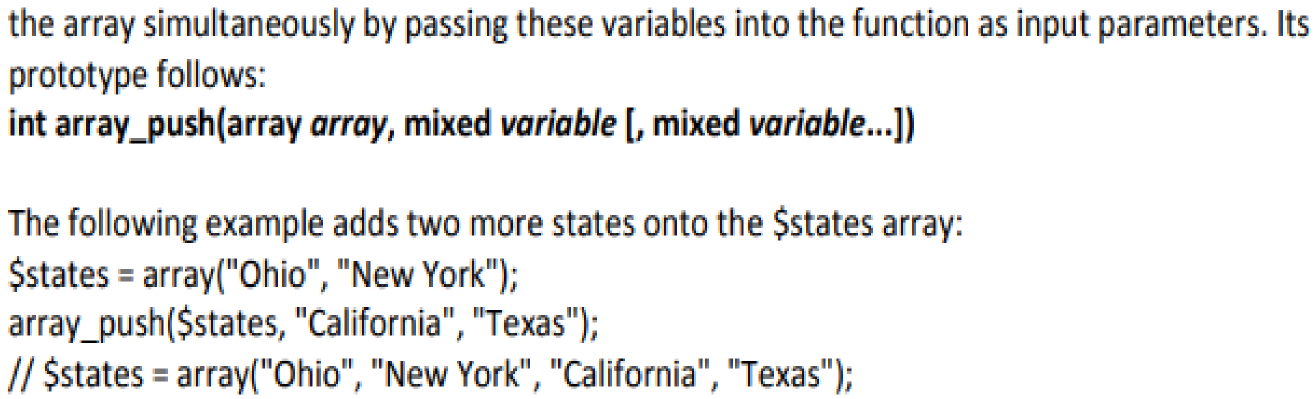
int array_unshift(array *array*, mixed *variable* [, mixed *variable*...])

The following example adds two states to the front of the $states array:

```
$states = array("Ohio", "New York");
array_unshift($states, "California", "Texas");
// $states = array("California", "Texas", "Ohio", "New York");
```

Adding a Value to the End of an Array

The array_push() function adds a value to the end of an array, returning the total count of elements in the array after the new value has been added. You can push multiple variables onto



the array simultaneously by passing these variables into the function as input parameters. Its prototype follows:

int array_push(array *array*, mixed *variable* [, mixed *variable*...])

The following example adds two more states onto the $states array:

```
$states = array("Ohio", "New York");
array_push($states, "California", "Texas");
// $states = array("Ohio", "New York", "California", "Texas");
```

Removing a Value from the Front of an Array

The array_shift() function removes and returns the first item found in an array. If numerical keys are used, all corresponding values will be shifted down, whereas arrays using associative keys will not be affected. Its prototype follows:

mixed array_shift(array *array*)

The following example removes the first state from the $states array:

```
$states = array("Ohio", "New York", "California", "Texas");
$state = array_shift($states);
// $states = array("New York", "California", "Texas")
// $state = "Ohio"
```

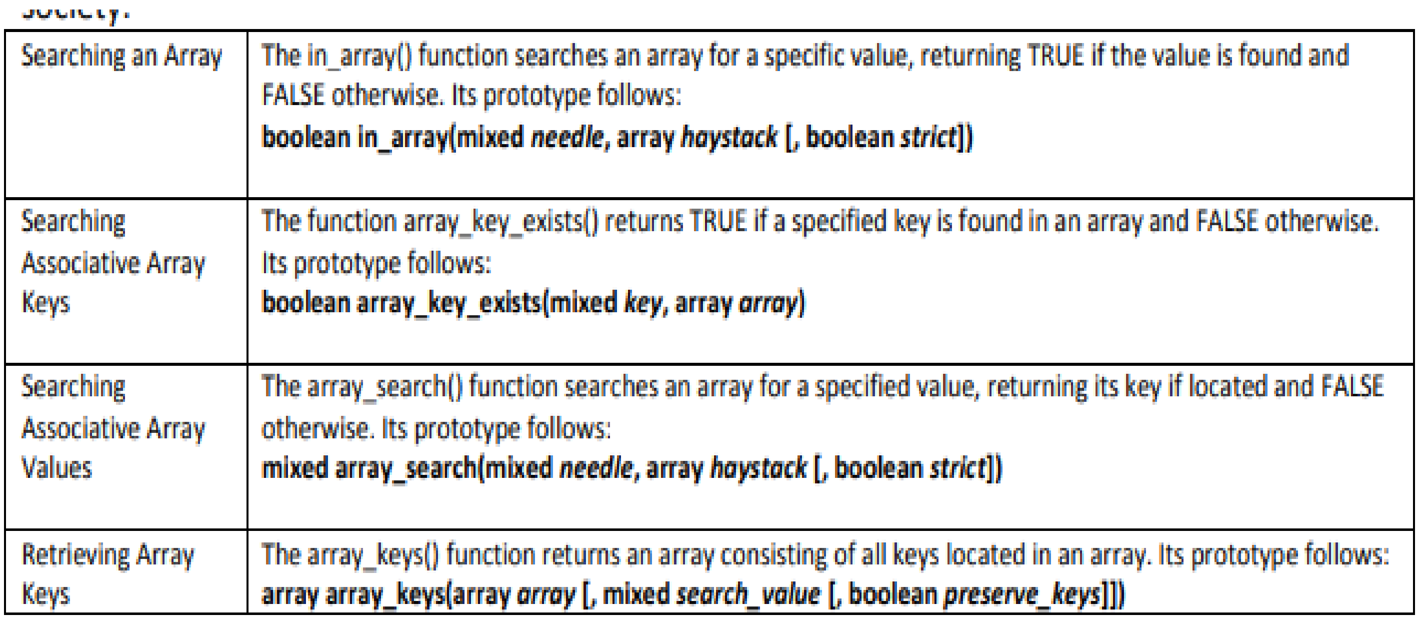
Removing a Value from the End of an Array

The array_pop() function removes and returns the last element from an array. Its prototype follows:

mixed array_pop(array *array*)

The following example removes the last state from the $states array:

```
$states = array("Ohio", "New York", "California", "Texas");
$state = array_pop($states);
// $states = array("Ohio", "New York", "California"
// $state = "Texas"
```



society.

Searching an Array	The in_array() function searches an array for a specific value, returning TRUE if the value is found and FALSE otherwise. Its prototype follows: **boolean in_array(mixed *needle*, array *haystack* [, boolean *strict*])**
Searching Associative Array Keys	The function array_key_exists() returns TRUE if a specified key is found in an array and FALSE otherwise. Its prototype follows: **boolean array_key_exists(mixed *key*, array *array*)**
Searching Associative Array Values	The array_search() function searches an array for a specified value, returning its key if located and FALSE otherwise. Its prototype follows: **mixed array_search(mixed *needle*, array *haystack* [, boolean *strict*])**
Retrieving Array Keys	The array_keys() function returns an array consisting of all keys located in an array. Its prototype follows: **array array_keys(array *array* [, mixed *search_value* [, boolean *preserve_keys*]])**

Retrieving Array Values	The array_values() function returns all values located in an array, automatically providing numeric indexes for the returned array. Its prototype follows: **array array_values(array *array*)**

Traversing Arrays

The need to travel across an array and retrieve various keys, values, or both is common, so it’s not a surprise that PHP offers numerous functions suited to this need. Many of these functions do double duty: retrieving the key or value residing at the current pointer location, and moving the pointer to the next appropriate location.

Retrieving the Current Array Key	The key() function returns the key located at the current pointer position of the provided array. Its prototype follows: **mixed key(array *array*)**
Retrieving the Current Array Value	The current() function returns the array value residing at the current pointer position of the array. Its prototype follows: **mixed current(array *array*)**
Retrieving the Current Array Key and Value	The each() function returns the current key/value pair from the array and advances the pointer one position. Its prototype follows: **array each(array *array*)**

Moving the Array Pointer

Several functions are available for moving the array pointer.

Moving the Pointer to the Next Array Position	The next() function returns the array value residing at the position immediately following that of the current array pointer. Its prototype follows: **mixed next(array *array*)**
Moving the Pointer to the Previous Array Position	The prev() function returns the array value residing at the location preceding the current pointer location, or FALSE if the pointer resides at the first position in the array. Its prototype follows: **mixed prev(array *array*)**
Moving the Pointer to the First Array Position	The reset() function serves to set an array pointer back to the beginning of the array. Its prototype follows: **mixed reset(array *array*)**
Moving the Pointer to the Last Array Position	The end() function moves the pointer to the last position of an array, returning the last element. Its prototype follows: **mixed end(array *array*)**

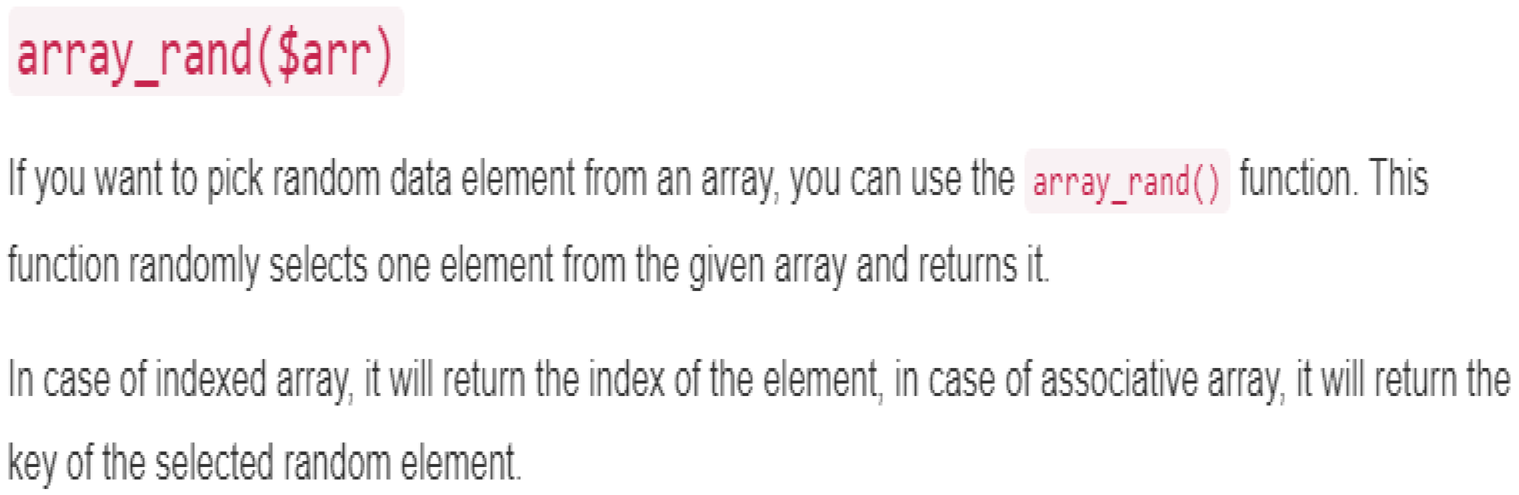
`array_map('function_name', $arr)`

If you want to perform certain operation on all the values stored in an array, you can do it by iterating over the array using a `for` loop or `foreach` and performing the required operation on all the values of the array.

Or, you can use the function `array_map()`. All we have to do is define a separate function to which we will provide the values stored in the array one by one(one at a time) and it will perform the operation on the values. Let's have an example,

`array_flip($arr)`

This function interchange the keys and the values of a PHP associative array.



`array_rand($arr)`

If you want to pick random data element from an array, you can use the `array_rand()` function. This function randomly selects one element from the given array and returns it.

In case of indexed array, it will return the index of the element, in case of associative array, it will return the key of the selected random element.

```
array_slice($arr, $offset, $length)
```

This function is used to create a subset of any array. Using this function, we define the starting point(`$offset`, which is the array index from where the subset starts) and the length(or, the number of elements required in the subset, starting from the offset).

Let's take an example,

- PHP comes with a number of built-in functions designed specifically for sorting array elements in different ways like alphabetically or numerically in ascending or descending order. Here we'll explore some of these functions most commonly used for sorting arrays.
- sort() and rsort() — For sorting indexed arrays
- asort() and arsort() — For sorting associative arrays by value
- ksort() and krsort() — For sorting associative arrays by key

Passing Array Values to a Function

The array_walk() function will pass each element of an array to the user-defined function. This is useful when you need to perform a particular action based on each array element. If you intend to actually modify the array key/value pairs, you'll need to pass each key/value to the function as a reference. Its prototype follows:

boolean array_walk(array &*array*, callback *function* [, mixed *userdata*])

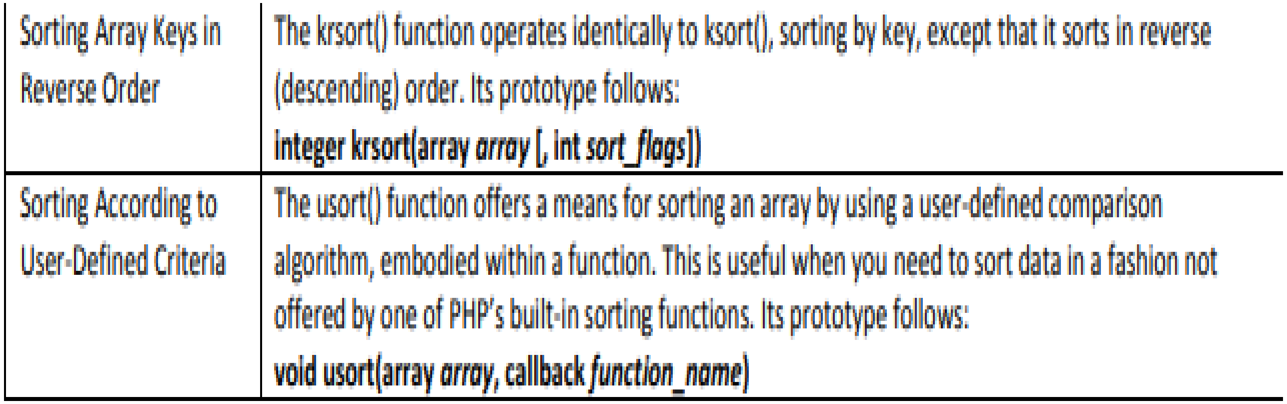
Determining Array Size and Uniqueness

A few functions are available for determining the number of total and unique array values.

Determining the Size of an Array	The count() function returns the total number of values found in an array. Its prototype follows: **integer count(array *array* [, int *mode*])**
Counting Array Value Frequency	The array_count_values() function returns an array consisting of associative key/value pairs. Its prototype follows: **array array_count_values(array *array*)**
Determining Unique Array Values	The array_unique() function removes all duplicate values found in an array, returning an array consisting of solely unique values. Its prototype follows: **array array_unique(array *array* [, int *sort_flags* = SORT_STRING])**

Sorting Arrays

Reversing Array Element Order	The array_reverse() function reverses an array's element order. Its prototype follows: **array array_reverse(array *array* [, boolean *preserve_keys*])**
Flipping Array Keys and Values	The array_flip() function reverses the roles of the keys and their corresponding values in an array. Its prototype follows: **array array_flip(array *array*)**
Sorting an Array	The sort() function sorts an array, ordering elements from lowest to highest value. Its prototype follows: **void sort(array *array* [, int *sort_flags*])**
Sorting an Array While Maintaining Key/Value Pairs	The asort() function is identical to sort(), sorting an array in ascending order, except that the key/value correspondence is maintained. Its prototype follows: **void asort(array *array* [, integer *sort_flags*])**
Sorting an Array in Reverse Order	The rsort() function is identical to sort(), except that it sorts array items in reverse (descending) order.Its prototype follows: **void rsort(array *array* [, int *sort_flags*])**
Sorting an Array in Reverse Order While Maintaining Key/Value Pairs	Like asort(), arsort() maintains key/value correlation. However, it sorts the array in reverse order. Its prototype follows: **void arsort(array *array* [, int *sort_flags*])**
Sorting an Array Naturally	The natsort() function is intended to offer a sorting mechanism comparable to the mechanisms that people normally use. Its prototype follows: **void natsort(array *array*)**
Case-Insensitive Natural Sorting	The function natcasesort() is functionally identical to natsort(), except that it is case insensitive: **void natcasesort(array *array*)**
Sorting an Array by Key Values	The ksort() function sorts an array by its keys, returning TRUE on success and FALSE otherwise. Its prototype follows: integer ksort(array *array* [, int *sort_flags*])

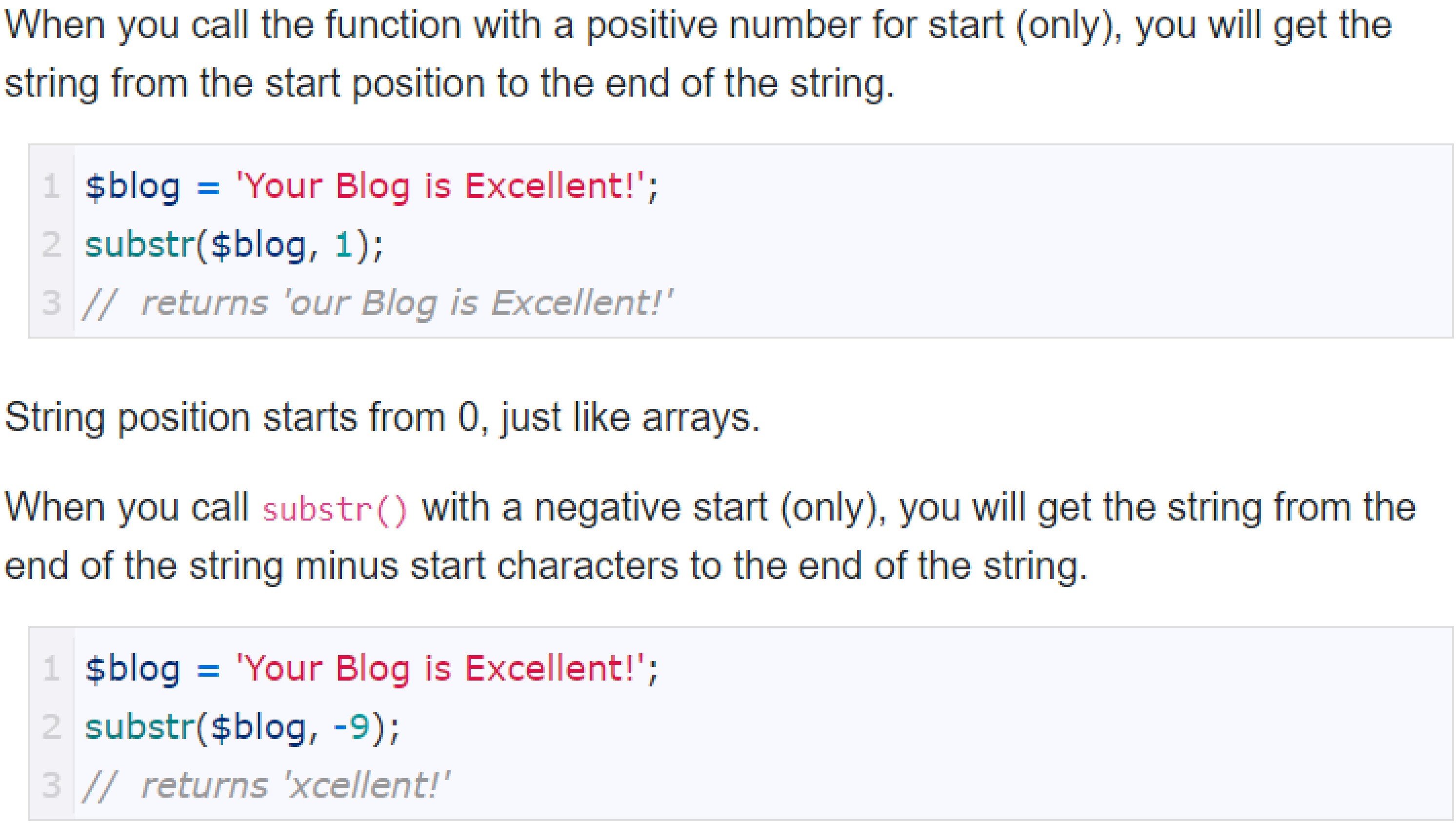


Sorting Array Keys in Reverse Order	The krsort() function operates identically to ksort(), sorting by key, except that it sorts in reverse (descending) order. Its prototype follows: **integer krsort(array *array* [, int *sort_flags*])**
Sorting According to User-Defined Criteria	The usort() function offers a means for sorting an array by using a user-defined comparison algorithm, embodied within a function. This is useful when you need to sort data in a fashion not offered by one of PHP's built-in sorting functions. Its prototype follows: **void usort(array *array*, callback *function_name*)**

Merging, Slicing, Splicing, and Dissecting Arrays

Merging Arrays	The array_merge() function merges arrays together, returning a single, unified array. The resulting array will begin with the first input array parameter, appending each subsequent array parameter in the order of appearance. Its prototype follows: **array array_merge(array *array1*, array *array2* [, array *arrayN*])**
Recursively Appending Arrays	The array_merge_recursive() function operates identically to array_merge(), joining two or more arrays together to form a single, unified array. The difference between the two functions lies in the way that this function behaves when a string key located in one of the input arrays already exists within the resulting array. Note that array_merge() will simply overwrite the preexisting key/value pair, replacing it with the one found in the current input array, while array_merge_recursive() will instead merge the values together, forming a new array with the preexisting key as its name. Its prototype follows: **array array_merge_recursive(array *array1*, array *array2* [, array arrayN])**
Combining Two Arrays	The array_combine() function produces a new array consisting of a submitted set of keys and corresponding values. Its prototype follows: **array array_combine(array *keys*, array *values*)** Both input arrays must be of equal size, and neither can be empty.
Slicing an Array	The array_slice() function returns a section of an array based on a starting and ending offset value. Its prototype follows: **array array_slice(array *array*, int *offset* [, int *length* [, boolean *preserve_keys*]])**
Splicing an Array	The array_splice() function removes all elements of an array found within a specified range, returning those removed elements in the form of an array. Its prototype follows: **array array_splice(array *array*, int *offset* [, int *length* [, array *replacement*]])**
Calculating an Array Intersection	The array_intersect() function returns a key-preserved array consisting only of those values present in the first array that are also present in each of the other input arrays. Its prototype follows: **array array_intersect(array *array1*, array *array2* [, *arrayN*])**
Calculating Associative Array Intersections	The function array_intersect_assoc() operates identically to array_intersect(), except that it also considers array keys in the comparison. Therefore, only key/value pairs located in the first array that arealso found in all other input arrays will be returned in the resulting array. Its prototype follows: **array array_intersect_assoc(array *array1*, array *array2* [, arrayN])**

Calculating Array Differences	Essentially the opposite of array_intersect(), the function array_diff() returns those values located in the first array that are not located in any of the subsequent arrays: **array array_diff(array *array1*, array *array2* [, arrayN])**
Calculating Associative Array Differences	The function array_diff_assoc() operates identically to array_diff(), except that it also considers array keys in the comparison. Therefore, only key/value pairs located in the first array but not appearing in any of the other input arrays will be returned in the result array. Its prototype follows: **array array_diff_assoc(array *array1*, array *array2* [, array *arrayN*])**



When you call the function with a positive number for start (only), you will get the string from the start position to the end of the string.

```
$blog = 'Your Blog is Excellent!';
substr($blog, 1);
//  returns 'our Blog is Excellent!'
```

String position starts from 0, just like arrays.

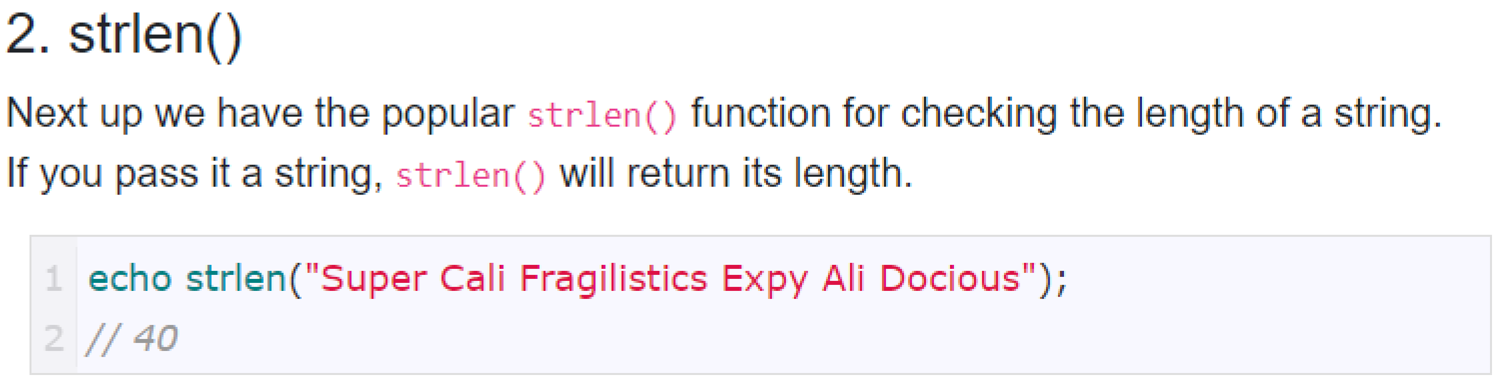
When you call `substr()` with a negative start (only), you will get the string from the end of the string minus start characters to the end of the string.

```
$blog = 'Your Blog is Excellent!';
substr($blog, -9);
//  returns 'xcellent!'
```

The length parameter can be used to specify either a number of characters to return if it is positive, or the end character of the return sequence if it is negative.

```php
$blog = 'Your Blog is Excellent!';
substr($blog, 0, 4);
// returns 'Your'

substr($blog, 5, -13);
//returns 'Blog'
```

5 signifies the starting character point (B) and -13 determines the ending point (count 13 places backwards starting from the end of the string).



2. strlen()

Next up we have the popular `strlen()` function for checking the length of a string. If you pass it a string, `strlen()` will return its length.

```
echo strlen("Super Cali Fragilistics Expy Ali Docious");
// 40
```

Another function which enables display of the number of words in any specific string is str_word_count(). This function is also useful in validation of input fields.

Syntax

```
Str_word_count(string)
```

Strrev() is used for reversing a string. You can use this function to get the reverse version of any string.

Syntax

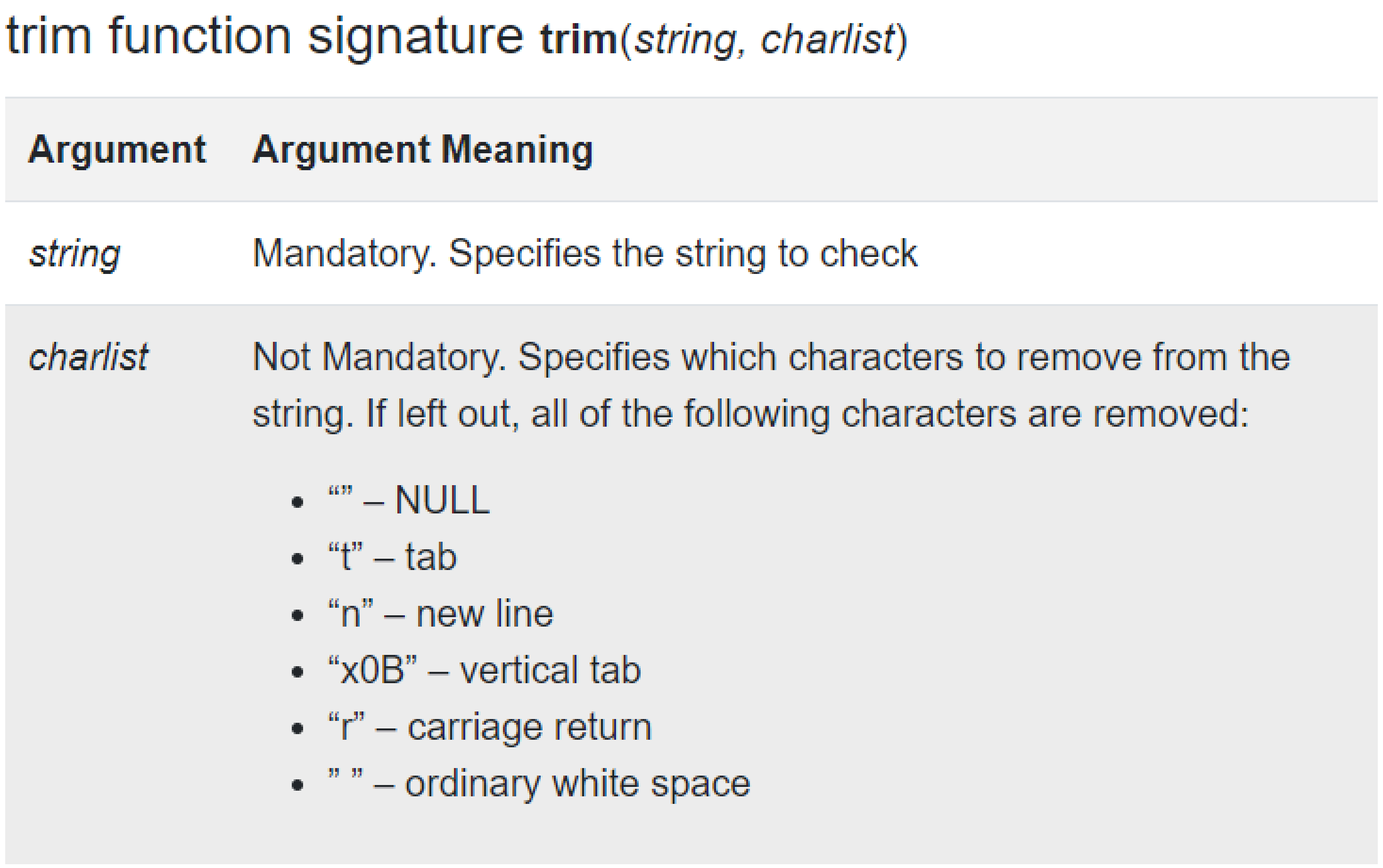
> Strev(string)

Strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return “False”. Strops() is most commonly used in validating input fields like email.

Syntax

```
Strpos(string,text);
```

- trim() **definition:** The trim() function cleans a string of all whitespace or other specified characters.
- **Usage:** Returns **string** with every whitespace character in **charlist** stripped from the beginning and end of the string.
- You can specify a range of characters to strip using .. within the string.

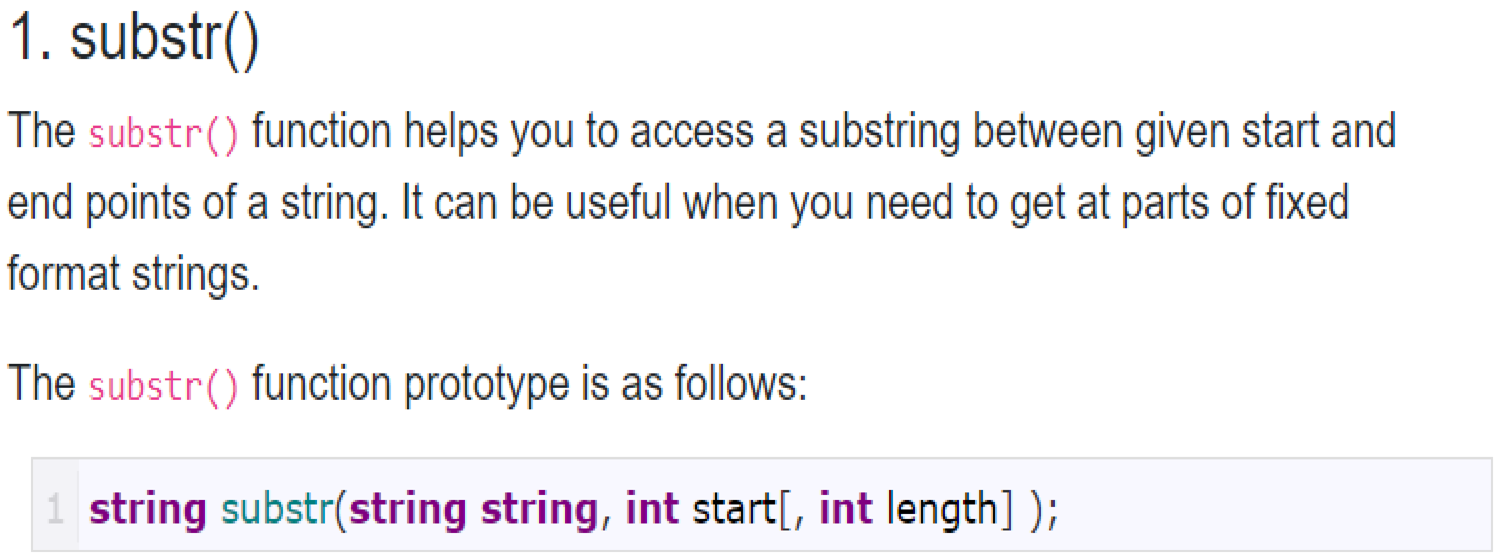


trim function signature **trim**(*string, charlist*)

Argument	**Argument Meaning**
string	Mandatory. Specifies the string to check
charlist	Not Mandatory. Specifies which characters to remove from the string. If left out, all of the following characters are removed: • “” – NULL • “t” – tab • “n” – new line • “x0B” – vertical tab • “r” – carriage return • ” ” – ordinary white space

- stripos() **definition:** The stripos() function finds the position of the first occurrence of a string inside another string.
- **Usage:** Returns the position of the first occurrence of **find** in string using case-insensitive comparison. If specified, the function begins its search at position **start**. Returns false if **find** is not found.
- stripos function signature **stripos**(*string, find, start*)

- strstr() **definition:** The strstr() function searches for the first occurrence of a string inside another string.
- **Usage:** Returns the portion of **string** from the first occurrence of **search** until the end of **string**, or from the first occurrence of **search** until the beginning of **string** if **before_search** is specified and true. If **search** is not found, the function returns false. If **search** contains more than one character, only the first is used.
- strstr function signature **strstr**(*string, search, before_search*)



1. substr()

The `substr()` function helps you to access a substring between given start and end points of a string. It can be useful when you need to get at parts of fixed format strings.

The `substr()` function prototype is as follows:

```
string substr(string string, int start[, int length] );
```

Outline

26.1 Introduction
26.2 PHP
26.3 String Processing and Regular Expressions
26.4 Viewing Client/Server Environment Variables
26.5 Form Processing and Business Logic
26.6 Verifying a Username and Password
26.7 Connecting to a Database
26.8 Cookies
26.9 Dynamic Content in PHP
26.10 Operator Precedence
26.11 Web Resources

Objectives

In this chapter, you will learn:

- To understand PHP data types, operators, arrays and control structures.
- To understand string processing and regular expressions in PHP.
- To construct programs that process form data.
- To be able to read and write client data using cookies.
- To construct programs that interact with MySQL databases.

first.php
(1 of 1)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 26.1: first.php -->
<!-- Our first PHP script -->

<?php
   $name = "LunaTic";    // declaration
?>

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>A simple PHP document</title
   </head>

   <body style = "font-size: 2em">
      <p>
         <strong>

            <!-- print variable name’s value -->
            Welcome to PHP, <?php print( "$name" ); ?>!
         </strong>
      </p>
   </body>
</html>
```

Scripting delimiters

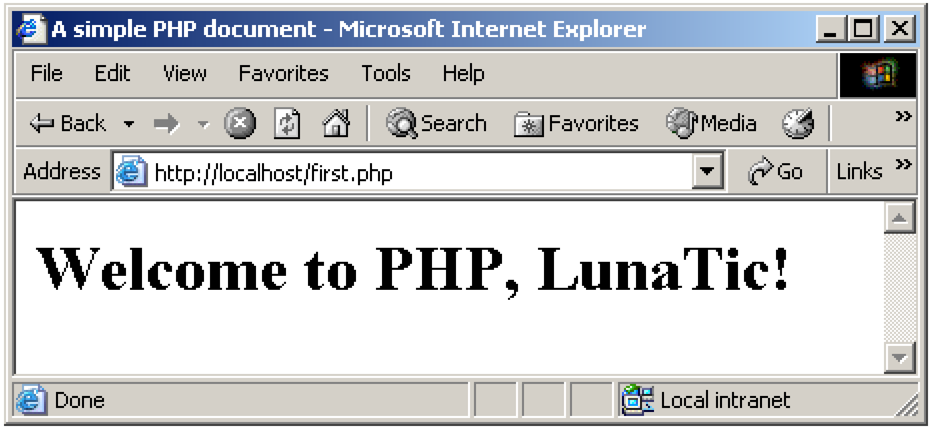
Declare variable `$name`

Single-line comment

Function `print` outputs the value of variable `$name`

2 PHP

Fig. 26.1 Simple PHP program.


A simple PHP document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/first.php Go Links
Done
Local intranet

Welcome to PHP, LunaTic!

2 PHP

- Variables
  - Can have different types at different times
  - Variable names inside strings replaced by their value
  - Type conversions
    - `settype` function
    - Type casting
  - Concatenation operator
    - `.` (period)
    - Combine strings

2 PHP

Data type	Description
`int, integer`	Whole numbers (i.e., numbers without a decimal point).
`float, double`	Real numbers (i.e., numbers containing a decimal point).
`string`	Text enclosed in either single (' ') or double (" ") quotes.
`bool, Boolean`	True or false.
`array`	Group of elements of the same type.
`object`	Group of associated data and methods.
`Resource`	An external data source.
`NULL`	No value.

Fig. 26.2 PHP data types.

data.php
(1 of 3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.3: data.php              -->
<!-- Demonstration of PHP data types -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>PHP data types</title>
   </head>

   <body>

      <?php

         // declare a string, double and integer
         $testString = "3.5 seconds";
         $testDouble = 79.2;
         $testInteger = 12;
      ?>

```

Assign a string to variable `$testString`

Assign a double to variable

Assign an integer to variable `$testInteger`

data.php

```
      <!-- print each variable’s value -->
      <?php print( $testString ); ?> is a string.<br />
      <?php print( $testDouble ); ?> is a double.<br />
      <?php print( $testInteger ); ?> is an integer.<br />

      <br />
      Now, converting to other types:<br />
      <?php

         // call function settype to convert variable
         // testString to different data types
         print( "$testString" );
         settype( $testString, "double" );
         print( " as a double is $testString <br />" );
         print( "$testString" );
         settype( $testString, "integer" );
         print( " as an integer is $testString <br />" );
         settype( $testString, "string" );
         print( "Converting back t
            $testString <br

         $data = "98.6 degr
```

Print each variable’s value

Call function `settype` to

a

Call function `settype` to convert the data type of variable `$testString` to an

Convert variable `$testString` back to a string

Outline

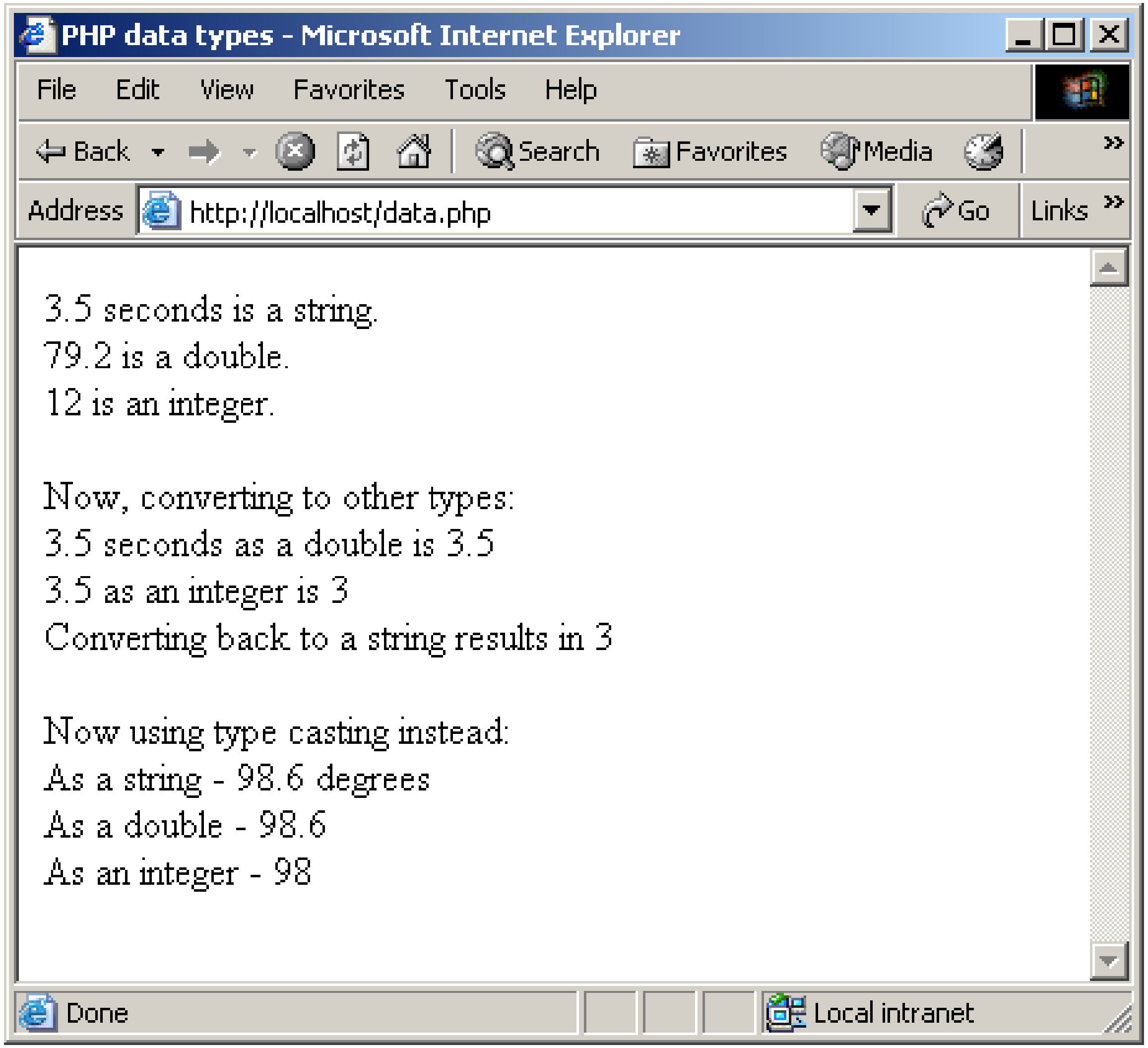
data.php
(3 of 3)

```

         // use type casting to cast variables to a
         // different type
         print( "Now using type casting instead: <br />
            As a string - " . (string) $data .
            "<br />As a double - " . (double) $data .
            "<br />As an integer - " . (integer) $data );
      ?>
   </body>
</html>
```

Use type casting to cast variable `$data` to different types

2 PHP

Fig. 26.3 Type conversion.


PHP data types - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/data.php
Go Links
Done
Local intranet

```
3.5 seconds is a string.
79.2 is a double.
12 is an integer.

Now, converting to other types:
3.5 seconds as a double is 3.5
3.5 as an integer is 3
Converting back to a string results in 3

Now using type casting instead:
As a string - 98.6 degrees
As a double - 98.6
As an integer - 98
```

2 PHP

- Arithmetic operators
  - Assignment operators
    - Syntactical shortcuts
    - Before being assigned values, variables have value `undef`
- Constants
  - Named values
  - `define` function

operators.php
(1 of 3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.4: operators.php   -->
<!-- Demonstration of operators -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using arithmetic operators</title>
   </head>

   <body>
      <?php
         $a = 5;
         print( "The value of variable a is $a <br />" );

         // define constant VALUE
         define( "VALUE", 5 );

         // add constant VALUE to variable $a
         $a = $a + VALUE;
         print( "Variable a after adding constant VALUE
            is $a <br />" );

```

Define constant VALUE.

Add constant VALUE to variable $a.

Outline

php

is less than 50

```
      // multiply variable $a by 2
      $a *= 2;
      print( "Multiplying variable

      // test if variable $a
      if ( $a < 50 )
         print( "Variable a is less than 50 <br />" );

      // add 40 to variable $a
      $a += 40;
      print( "Variable a after ad

      // test if variable $a is 50 or less
      if ( $a < 51 )
         print( "Variable a is still 50 or less<br />" );

      // test if variable $a is between 50 and 100, inclusive
      elseif ( $a < 101 )
         print( "Variable a is now between 50 and 100,
            inclusive<br />" );
      else
         print( "Variable a is now greater than 100
            <br />" );

```

Multiply variable $a by two using the multiplication assignment operator *=.

Print if variable $a is less than 50.

Add 40 to variable $a using the addition assignment operator **+=**.

operators.php
(3 of 3)

```
         // print an uninitialized variable
         print( "Using a variable before initializing:
            $nothing <br />" );

         // add constant VALUE to an uninitialized variable
         $test = $num + VALUE;
         print( "An uninitialized variable plus constant
            VALUE yields $test <br />" );

         // add a string to an int
         $str = "3 dollars";
         $a += $str;
         print( "Adding a string to variable a yields $a
            <br />" );
      ?>
   </body>
</html>
```

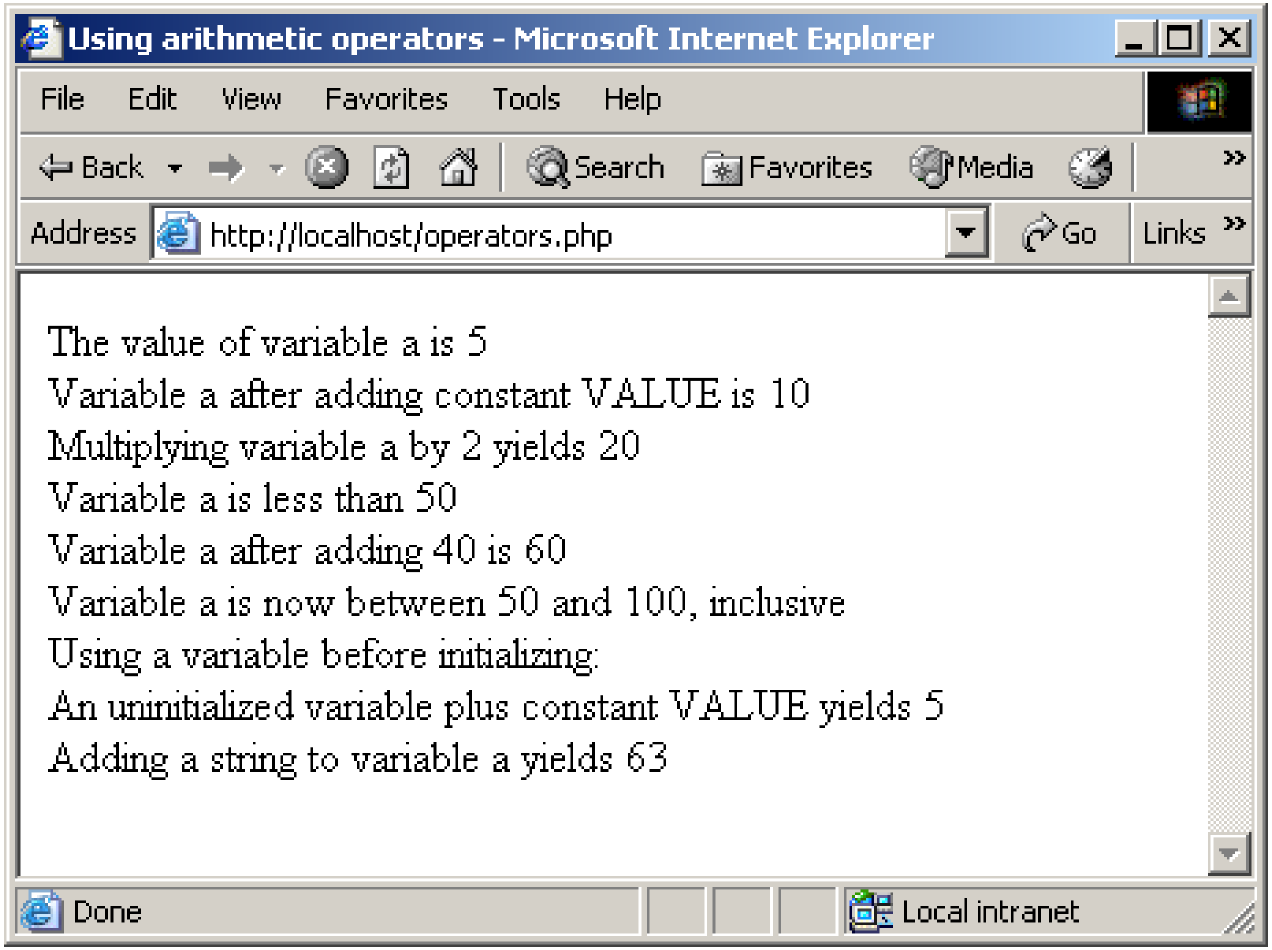
Print an uninitialized variable ($nothing).

variable.

Add a string to an integer.

2 PHP

Fig. 26.4 Using PHP's arithmetic operators.


Using arithmetic operators - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/operators.php Go Links
Done
Local intranet

```
The value of variable a is 5
Variable a after adding constant VALUE is 10
Multiplying variable a by 2 yields 20
Variable a is less than 50
Variable a after adding 40 is 60
Variable a is now between 50 and 100, inclusive
Using a variable before initializing:
An uninitialized variable plus constant VALUE yields 5
Adding a string to variable a yields 63
```

PHP

- Keywords
  - Reserved for language features
  - `if…elseif…else`
- Arrays
  - Group of related data
    - Elements
  - Name plus braces and index
    - Indices start at zero
  - `count` function
  - `array` function

PHP

- Arrays, cont.
  - Built-in iterators
    - Maintain pointer to element currently referenced
    - `reset`
    - `key`
    - `next`
    - `foreach` loops

PHP

PHP keywords					
and	do	for	include	require	true
break	else	foreach	list	return	var
case	elseif	function	new	static	virtual
class	extends	global	not	switch	xor
continue	false	if	or	this	while
default					

Fig. 26.5 PHP keywords.

arrays.php
(1 of 3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.6: arrays.php -->
<!-- Array manipulation    -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Array manipulation</title>
    </head>

    <body>
       <?php

          // create array first
          print( "<strong>Creating the first array</strong>
             <br />" );
          $first[ 0 ] = "zero";
          $first[ 1 ] =
          $first[ 2 ] =
          $first[] = "three";

          // print each element's index and value
          for ( $i = 0; $i < count( $first ); $i++ )
             print( "Element $i is $first[$i] <br />" );
```

Create the array `$first` by assigning a value to an array element.

Assign a value to the array, omitting the index. Appends

Use a `for` loop to print out each element's index and value. Function `count` returns the total number of elements in the array.

```
      print( "<br /><strong>Creating the s
         </strong><br />" );

      // call function array to create arr
      $second = array( "zero", "one", "two", "three" );
      for ( $i = 0; $i < count( $second ); $i++ )
         print( "Element $i is $second[$i] <br />" );

      print( "<br /><strong>Creating the third array
         </strong><br />" );

      // assign values to non-numerical indices
      $third[ "ArtTic" ] = 2
      $third[ "LunaTic" ] =
      $third[ "GalAnt" ] = 23;

      // iterate through the array elements and print each
      // element's name and value
      for ( reset( $third ); $element = key( $third );
         next( $third ) )
         print( "$element is $third[$element] <br />" );

```

Call function `array` to create an array that contains the arguments passed to it. Store the array in variable `$second`.

(2 of 3)

Assign values to non-numerical indices in array `$third`.

Function `reset` sets the internal pointer to the first element of the array.

Function key returns the index of the element which the internal pointer references.

Function `next` moves the internal pointer to the next element.

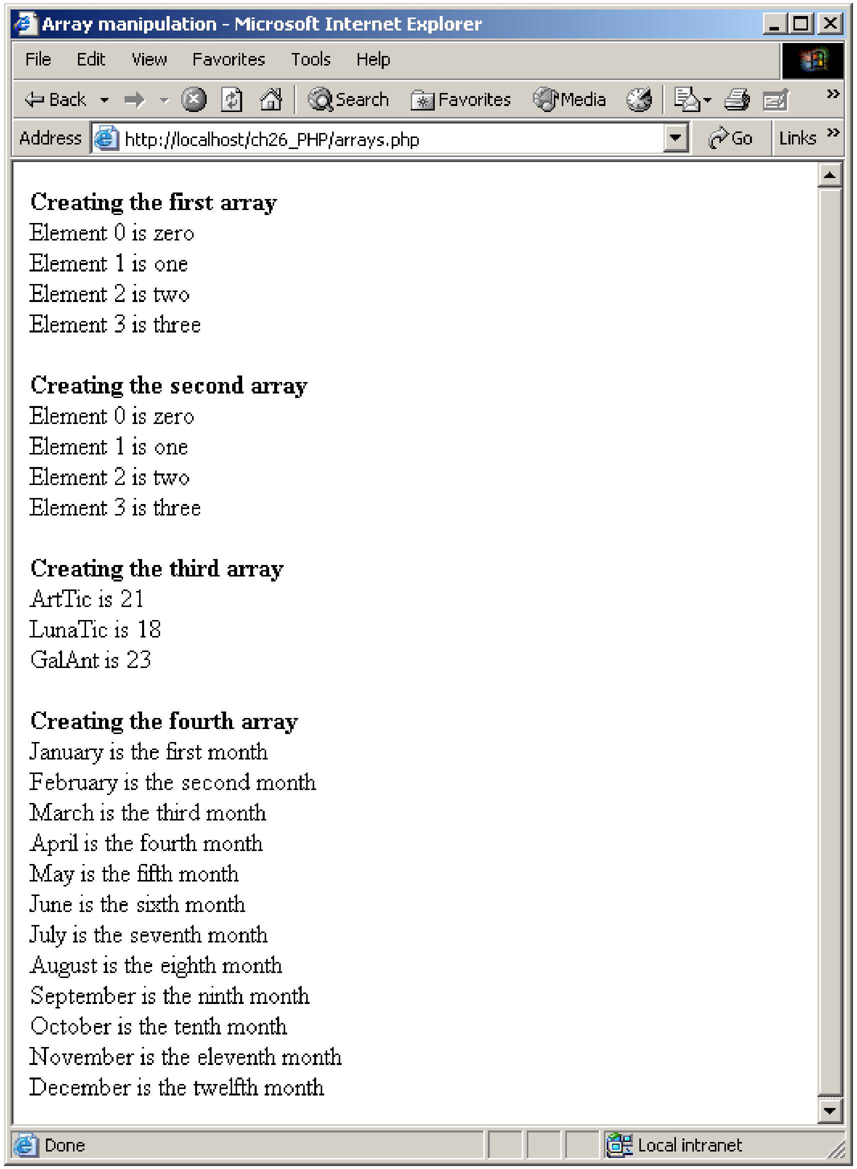
arrays.php
(3 of 3)

```
         print( "<br /><strong>Creating the fourth array
            </strong><br />" );

         // call function array to create array fourth using
         // string indices
         $fourth = array(
            "January"   => "first",    "February" => "second",
            "March"     => "third",
            "May"       => "fifth",
            "July"      => "seventh"
            "September" => "ninth",
            "November"  => "eleventh
            );

         // print each element's name and value
         foreach ( $fourth as $element => $value )
            print( "$element is the $value month <br />" );
      ?>
   </body>
</html>
```

Operator **=>** is used in function `array` to assign each element a string index. The value to the left of the operator is the array index, and the value to the right is the element's value.

2 PHP

Fig. 26.6 Array manipulation.


Array manipulation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/ch26_PHP/arrays.php
Go Links
Done
Local intranet

```
Creating the first array
Element 0 is zero
Element 1 is one
Element 2 is two
Element 3 is three

Creating the second array
Element 0 is zero
Element 1 is one
Element 2 is two
Element 3 is three

Creating the third array
ArtTic is 21
LunaTic is 18
GalAnt is 23

Creating the fourth array
January is the first month
February is the second month
March is the third month
April is the fourth month
May is the fifth month
June is the sixth month
July is the seventh month
August is the eighth month
September is the ninth month
October is the tenth month
November is the eleventh month
December is the twelfth month
```

26.3 String Processing and Regular Expressions

- String processing
  - Equality and comparison two important operations
  - `strcmp` function
    - Returns –1 if string 1 < string 2
    - Returns 0 if string 1 = string 2
    - Returns 1 if string 1 > string 2
  - Relational operators

compare.php
(1 of 2)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.7: compare.php -->
<!-- String Comparison      -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>String Comparison</title>
   </head>

   <body>
      <?php

         // create array fruits
         $fruits = array( "apple

         // iterate through each
         for ( $i = 0; $i < coun

            // call function strcmp to compare the array element
            // to string "banana"
            if ( strcmp( $fruits[ $i ], "banana" ) < 0 )
               print( $fruits[ $i ]." is less than banana " );
```

Use a `for` loop to iterate through each array element.

Function `strcmp` compares two strings. If the first string alphabetically precedes the second, then `–1` is returned. If the strings are equal, `0` is returned. If the first string alphabetically follows the second, then `1` is returned.

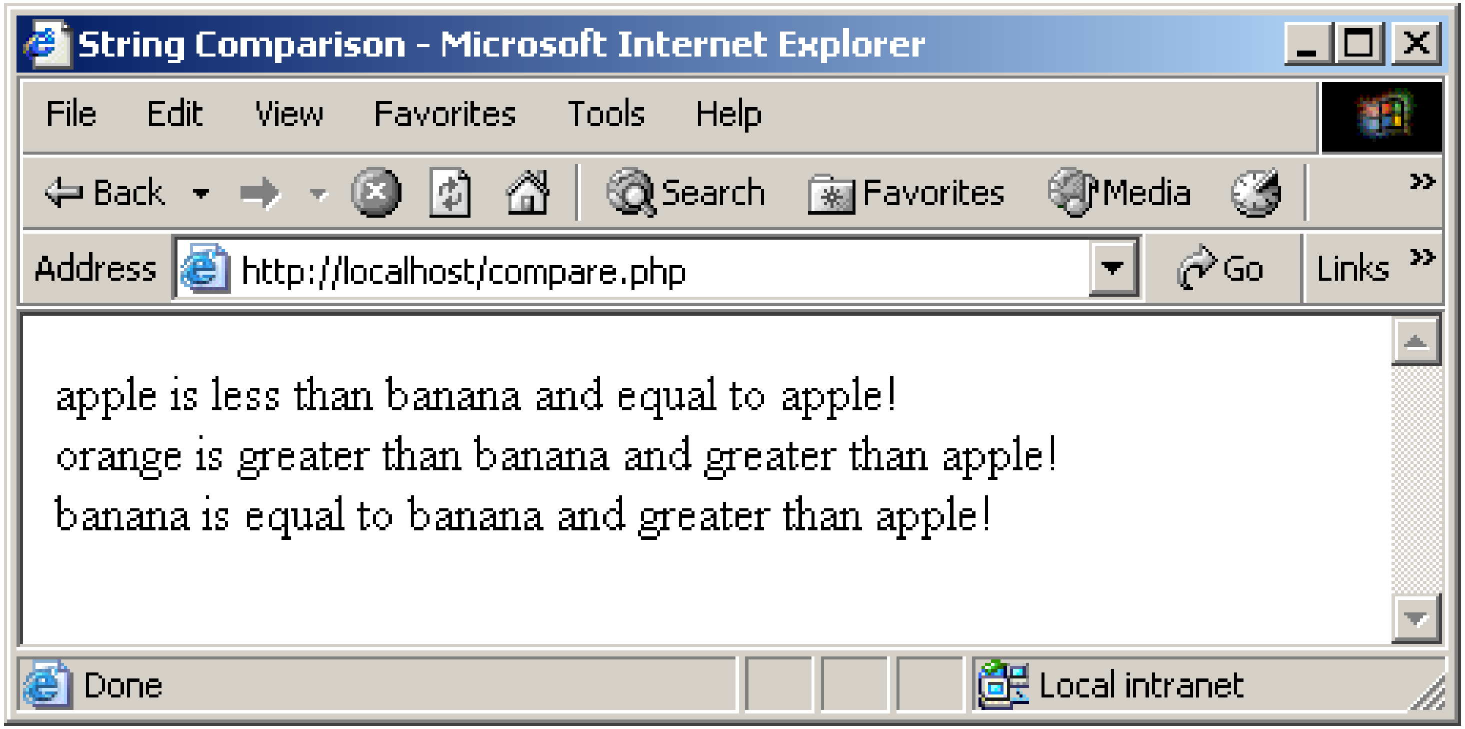
.php

```
            elseif ( strcmp( $fruits[ $i ], "banana" ) > 0 )
               print( $fruits[ $i ].
                  " is greater than banana " );
            else
               print( $fruits[ $i ]." i

            // use relational operators to compare each element
            // to string "apple"
            if ( $fruits[ $i ] < "apple" )
              print( "and less than apple! <br />" );
            elseif ( $fruits[ $i ] > "apple" )
              print( "and greater than apple! <br />" );
            elseif ( $fruits[ $i ] == "apple" )
               print( "and equal to apple! <br />" );

         }
      ?>
   </body>
</html>
```

Use relational operators to compare each array element to string “apple”.

3 String Processing and Regular Expressions

Fig. 26.7 Using the string comparison operators.


String Comparison - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/compare.php Go Links
Done
Local intranet

```
apple is less than banana and equal to apple!
orange is greater than banana and greater than apple!
banana is equal to banana and greater than apple!
```

3 String Processing and Regular Expressions

- Regular expressions
  - Pattern matching templates
  - `ereg` function
    - POSIX
  - `preg_match` function
    - Perl
  - `ereg_replace` function
- Building regular expressions
  - Metacharacters
    - `$`, `.`, `^`
  - Brackets `[ ]`

expression.php
(1 of 3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.8: expression.php -->
<!-- Using regular expressions -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Regular expressions</title>
   </head>

   <body>
      <?php
         $search = "Now is th
         print( "Test string is: '$search'<br /><br />" );

         // call function ereg to search for pattern 'Now'
         // in variable search
         if ( ereg( "Now", $search ) )
            print( "String 'Now' was found.<br />" );

```

Function `ereg` searches for the literal characters `Now` inside variable `$search`.

Outline

expression.php

(2 of 3)

```
      // search for pattern 'Now' in the beginning of
      // the string
      if ( ereg( "^Now", $search ) )
         print( "String

            of the line.

      // search for patt
      if ( ereg( "Now$", $search ) )
         print( "String 'Now' was found at the end
            of the line.<br />" );

      // search for any word ending in 'ow'
      if ( ereg( "[[:<:]]([a-zA-Z]*ow)[[:>:]]", $search,
         $match ) )
         print( "Word found ending in 'ow': " .
            $match[ 1 ] . "<br />" );

      // search for any words b
      print( "Words beginning w

      while ( eregi( "[[:<:]](t[[
         $search, $match ) ) {
         print( $match[ 1 ] . " " );
```

The dollar sign special character ($) search for the

The caret special character (^) matches the beginning of a string. Function ereg searches the beginning of the string for pattern Now .

The
m
m

The special bracket expressions [[:<:]] and

Placing a pattern in parentheses stores the matched string in the array that is specified in the third argument to function ereg.

The pattern used in this example, [[:<:]](t[[:alpha:]]+)[[:>:]], matches any
owed by one or
ha:]]
recognizes any alphabetic character.

Function eregi is used to specify case insensitive pattern matches.

Outline

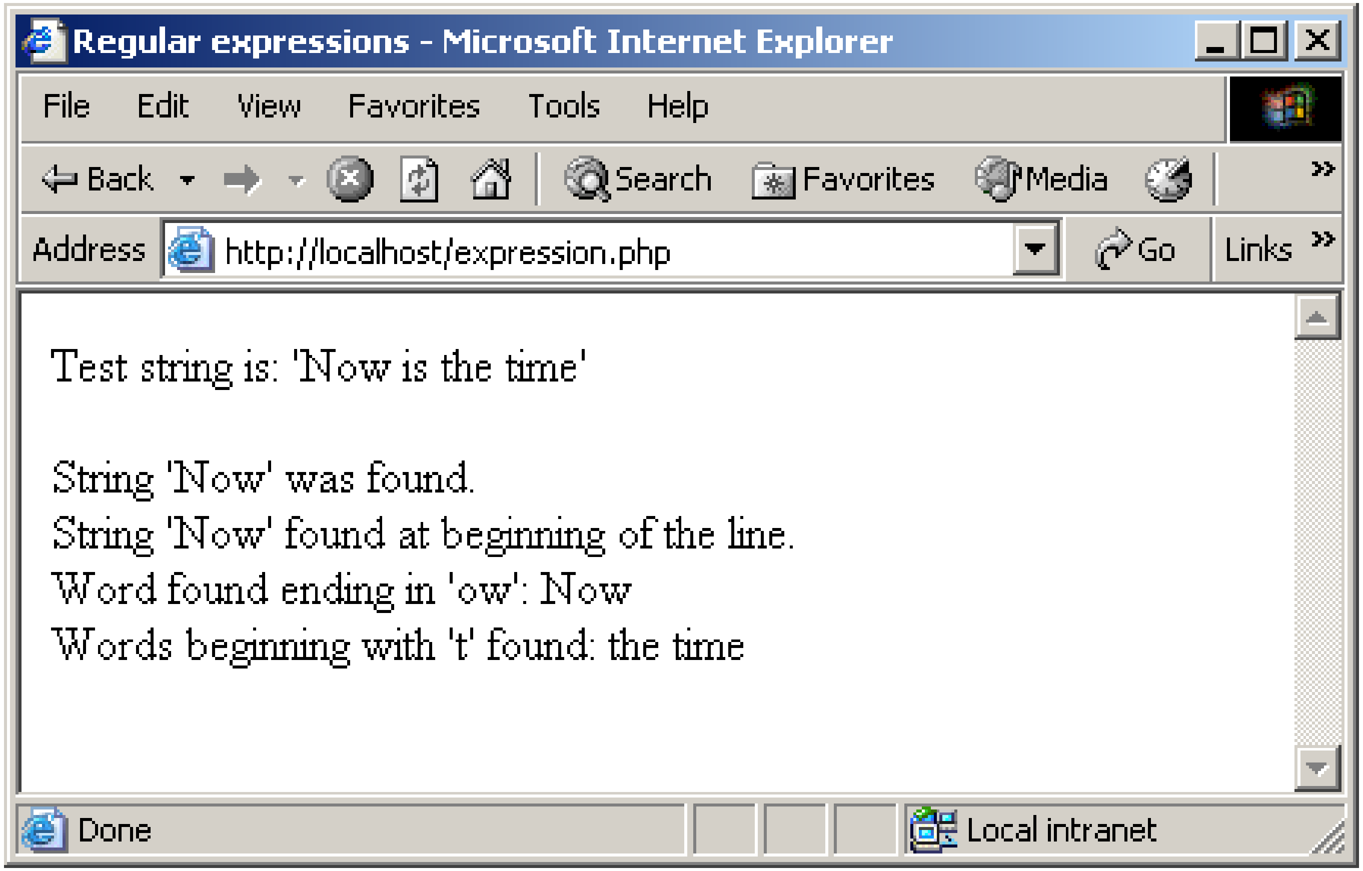
hp

```
            // remove the first occurrence of a word beginning
            // with 't' to find other instances in the string
            $search = ereg_replace( $match[ 1 ], "", $search );
         }

         print( "<br />" );
      ?>
   </body>
</html>
```

After printing a match of a word beginning with `t`, function `ereg_replace` is called to remove the word from the string. This is necessary be because to find multiple instances of a given pattern, the first matched instance must first be removed. Function `ereg_replace` takes three arguments: the pattern to match, a string to replace the matched string and the string to search.

3 String Processing and Regular Expressions

Fig. 26.8 Regular expressions in PHP.



Regular expressions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/expression.php
Go Links
Done
Local intranet

```
Test string is: 'Now is the time'

String 'Now' was found.
String 'Now' found at beginning of the line.
Word found ending in 'ow': Now
Words beginning with 't' found: the time
```

3 String Processing and Regular Expressions

Quantifier	Matches
{n}	Exactly n times.
{m,n}	Between m and n times inclusive.
{n,}	n or more times.
+	One or more times (same as {1,}).
*	Zero or more times (same as {0,}).
?	Zero or one time (same as {0,1}).

Fig. 26.9 Some PHP quantifiers.

3 String Processing and Regular Expressions

Character class	Description
alnum	Alphanumeric characters (i.e., letters [a-zA-Z] or digits [0-9]).
alpha	Word characters (i.e., letters [a-zA-Z]).
digit	Digits.
space	Whitespace.
lower	Lowercase letters.
upper	Uppercase letters.

Fig. 26.10 Some PHP character classes.

4 Viewing Client/Server Environment Variables

- Environment variables
  - Provide information about execution environment
    - Type of Web browser
    - Type of server
    - Details of HTTP connection
  - Stored as array in PHP
    - `$_ENV`

4 Viewing Client/Server Environment Variables

Variable name	Description
`$_SERVER`	Data about the currently running server.
`$_ENV`	Data about the client’s environment.
`$_GET`	Data posted to the server by the `get` method.
`$_POST`	Data posted to the server by the `post` method.
`$_COOKIE`	Data contained in cookies on the client’s computer.
`$GLOBALS`	Array containing all global variables.

Fig. 26.11 Some useful global arrays.

env.php
(1 of 1)

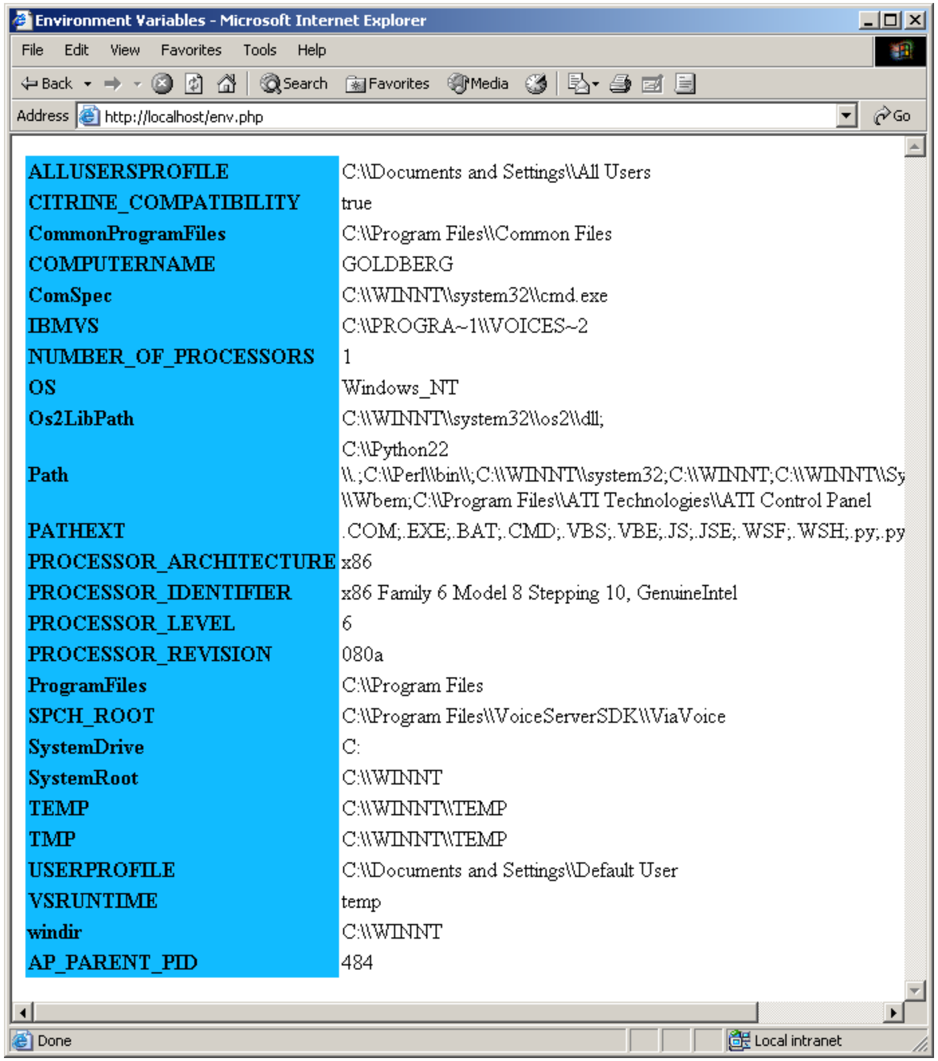
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.11: env.php                       -->
<!-- Program to display environment variables -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Environment Variables</title>
   </head>

   <body>
      <table border = "0" cellpadding = "2" cellspacing = "0"
         width = "100%">
      <?php

         // print the key and value for each element
         // in the $_ENV array
         foreach ( $_ENV as $key => $value )
            print( "<tr><td bgcolor = \"#11bbff\">
               <strong>$k
               <td>$valu
      ?>
      </table>
   </body>
</html>
```

The `foreach` loop is used to print out the keys and values for each element in the $_ENV array.

PHP stores environment variables and their values in the `$_ENV` array.

4 Viewing Client/Server Environment Variables

Fig. 26.12 Displaying environment variables.



Environment Variables - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Go
Done
Local intranet

Address http://localhost/env.php

ALLUSERSPROFILE	C:\\Documents and Settings\\All Users
CITRINE_COMPATIBILITY	true
CommonProgramFiles	C:\\Program Files\\Common Files
COMPUTERNAME	GOLDBERG
ComSpec	C:\\WINNT\\system32\\cmd.exe
IBMVS	C:\\PROGRA~1\\VOICES~2
NUMBER_OF_PROCESSORS	1
OS	Windows_NT
Os2LibPath	C:\\WINNT\\system32\\os2\\dll;
Path	C:\\Python22 \\.;C:\\Perl\\bin\\;C:\\WINNT\\system32;C:\\WINNT;C:\\WINNT\\Sy \\Wbem;C:\\Program Files\\ATI Technologies\\ATI Control Panel
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.py;.py
PROCESSOR_ARCHITECTURE	x86
PROCESSOR_IDENTIFIER	x86 Family 6 Model 8 Stepping 10, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	080a
ProgramFiles	C:\\Program Files
SPCH_ROOT	C:\\Program Files\\VoiceServerSDK\\ViaVoice
SystemDrive	C:
SystemRoot	C:\\WINNT
TEMP	C:\\WINNT\\TEMP
TMP	C:\\WINNT\\TEMP
USERPROFILE	C:\\Documents and Settings\\Default User
VSRUNTIME	temp
windir	C:\\WINNT
AP_PARENT_PID	484

5 Form Processing and Business Logic

- Form processing
  - `action` property
    - Where to send form data
  - `method` property
    - `post`
  - Each element has unique name

form.html
(1 of 4)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.13: form.html                  -->
<!-- Form for use with the form.php program -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sample form to take user input in XHTML</title>
   </head>

   <body>

      <h1>This is a sample regist
      Please fill in all fields a

      <!-- post form data to form.php -->
      <form method = "post" action = "form.php">
         <img src = "images/user.gif" alt = "User" /><br />
         <span style = "color: blue">
            Please fill out the fields below.<br />
         </span>

```

The `action` attribute of the `form` element indicates that when the user clicks **Register**, the form data will be posted to `form.php`.

Outline

html

(2 of 4)

```
      <!-- create four text boxes for user input -->
      <img src = "images/fname.gif"
      <input type = "text" name = "

      <img src = "images/lname.gif"
      <input type = "text" name = "lname" /><br />

      <img src = "images/email.gif" alt = "Email" />
      <input type = "text" name = "email" /><br />

      <img src = "images/phone.gif" alt = "Phone" />
      <input type = "text" name = "phone" /><br />

      <span style = "font-size: 10pt">
         Must be in the form (555)555-5555</span>
      <br /><br />

      <img src = "images/downloads.gif"
         alt = "Publications" /><br />

      <span style = "color: blue">
         Which book would you like information about?
      </span><br />

```

A unique `name` (e.g., `email`) is assigned to each of the form's `input` fields. When **Register** is clicked, each field's `name` and `value` are sent to the Web server.

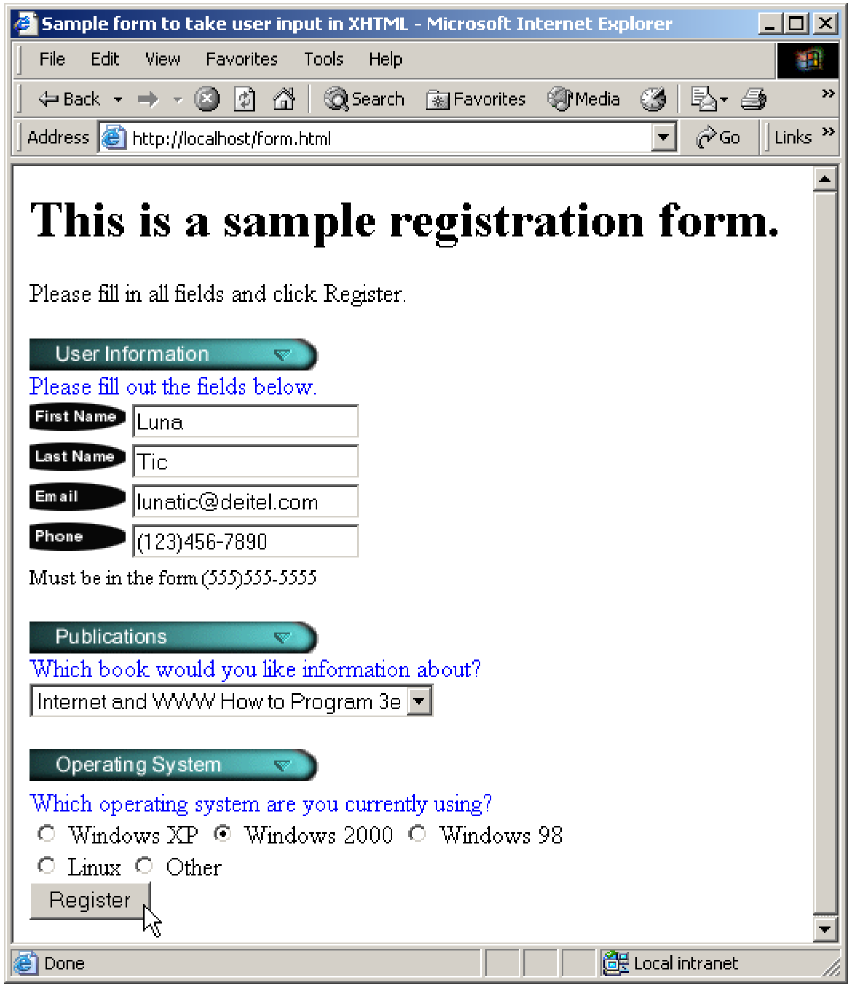
form.html
(3 of 4)

```
         <!-- create drop-down list containing book names -->
         <select name = "book">
            <option>Internet and WWW How to Program 3e</option>
            <option>C++ How to Program 4e</option>
            <option>Java How to Program 5e</option>
            <option>XML How to Program 1e</option>
         </select>
         <br /><br />

         <img src = "images/os.gif" alt = "Operating System" />
         <br /><span style = "color: blue">
            Which operating system are you currently using?
         <br /></span>

         <!-- create five radio buttons -->
         <input type = "radio" name = "os" value = "Windows XP"
            checked = "checked" />
            Windows XP

         <input type = "radio" name = "os" value =
            "Windows 2000" />
            Windows 2000

         <input type = "radio" name = "os" value =
            "Windows 98" />
            Windows 98<br />

         <input type = "radio" name = "os" value = "Linux" />
            Linux

         <input type = "radio" name = "os" value = "Other" />
            Other<br />

          <!-- create a submit button -->
         <input type = "submit" value = "Register" />
      </form>

   </body>
</html>
```

form.html
(4 of 4)

5 Form Processing and Business Logic

Fig. 26.13 XHTML form for gathering user input.



Sample form to take user input in XHTML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/form.html
Go Links
Done
Local intranet

This is a sample registration form.

Please fill in all fields and click Register.

User Information

Please fill out the fields below.

First Name	Luna
Last Name	Tic
Email	lunatic@deitel.com
Phone	(123)456-7890

Must be in the form (555)555-5555

Publications

Which book would you like information about?

Internet and WWW How to Program 3e

Operating System

Which operating system are you currently using?

○ Windows XP ● Windows 2000 ○ Windows 98
○ Linux ○ Other

Register

5 Form Processing and Business Logic

- Business logic
  - Confirm that valid information was entered
  - `extract` function
    - Creates variables corresponding to each key-value pair in array
    - Easily retrieve all values sent to PHP page
  - Regular expressions very helpful
  - Do checks on client side where possible
    - JavaScript
    - Conserves server resources
- Ending a script
  - `die` function
    - Remember to close all HTML tags

form.php
(1 of 4)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.14: form.php                 -->
<!-- Read information sent from form.html -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Form Validation</title>
    </head>

    <body style = "font-family: arial,sans-se

         <?php
            extract( $_POST

            // determine whether phone number is valid and print
            // an error message if not
            if ( !ereg( "^\([0-9]{3}\)[0-9]{3}-[0-9]{4}$",
               $phone ) ){

```

termine whether the
user is valid.

ing

The parentheses in the expression must be followed by three digits ([0-9]{3}), a closing parenthesis, three digits, a literal hyphen and four additional digits.

We access the phone field's value from form.html by using variable $phone.

```
      print( "<p><span style = \"color: red;
         font-size: 2em\">
         INVALID PHONE NUMBER</span><br />
         A valid phone number must be in the form
         <strong>(555)555-5555</strong><br />
         <span style = \"color: blue\">
         Click the Back button,
         number and resubmit.<br /><br />
         Thank You.</span></p></body></html>" );

      die(); // terminate script execution
   }
?>

<p>Hi
   <span style = "color: blue">
      <strong>
         <?php print( "$fname" ); ?>
      </strong>
   </span>.
   Thank you for completing the survey.<br />

```

Function die terminates script execution

form.php
(2 of 4)

form.php
(3 of 4)

```
         You have been added to the
         <span style = "color: blue">
            <strong>
               <?php print( "$book " ); ?>
            </strong>
         </span>
         mailing list.
      </p>
      <strong>The following information has been saved
          in our database:</strong><br />

      <table border = "0" cellpadding = "0" cellspacing = "10">
         <tr>
            <td bgcolor = "#ffffaa">Name </td>
            <td bgcolor = "#ffffbb">Email</td>
            <td bgcolor = "#ffffcc">Phone</td>
            <td bgcolor = "#ffffdd">OS</td>
         </tr>

         <tr>
            <?php

```

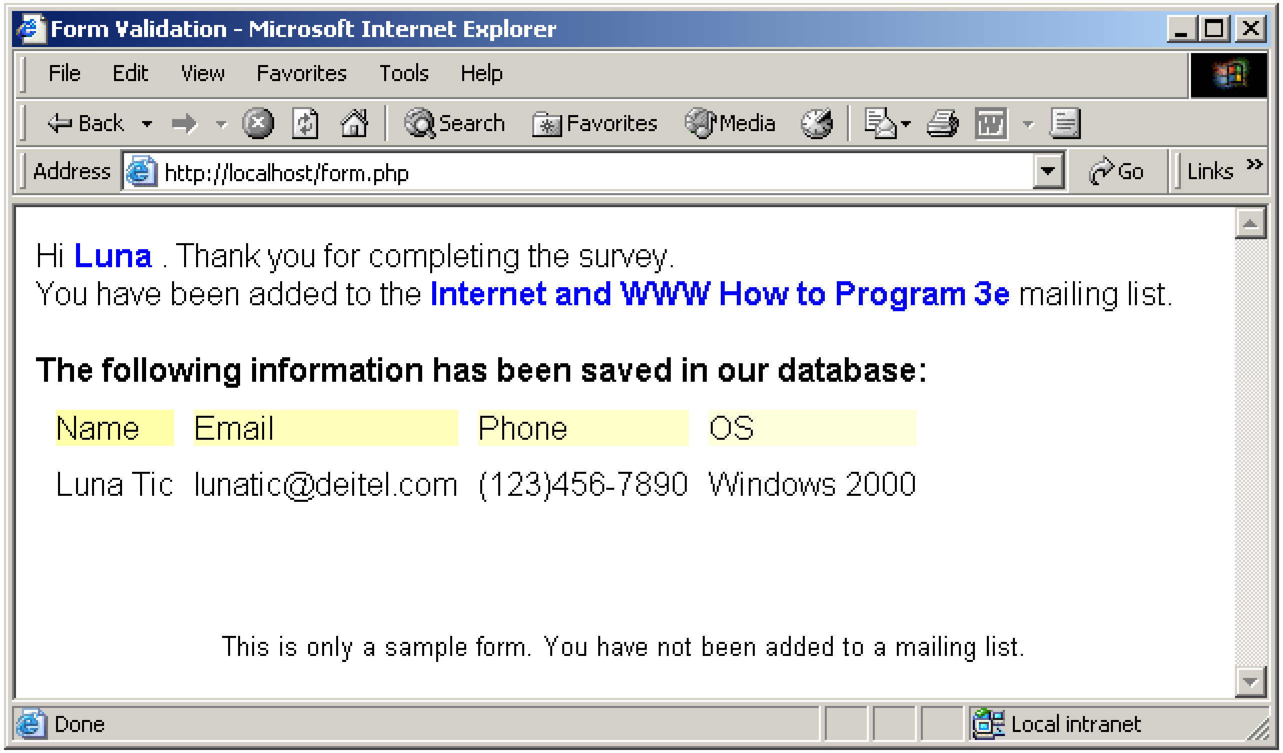
Outline

```
               // print each form field's value
               print( "<td>$fname $lname</td>
                  <td>$email</td>
                  <td>$phone</td>
                  <td>$os</td>" );
            ?>
         </tr>
      </table>

      <br /><br /><br />
      <div style = "font-size: 10pt; text-align: center">
         This is only a sample form.
         You have not been added to a mailing list.
      </div>
   </body>
</html>
```

form.php
(4 of 4)

5 Form Processing and Business Logic

Fig. 26.14 Obtaining user input through forms.


Form Validation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/form.php
Go Links
Done
Local intranet

Hi **Luna** . Thank you for completing the survey.
You have been added to the **Internet and WWW How to Program 3e** mailing list.

The following information has been saved in our database:

Name	Email	Phone	OS
Luna Tic	lunatic@deitel.com	(123)456-7890	Windows 2000

This is only a sample form. You have not been added to a mailing list.

6 Verifying a Username and Password

- Private website
  - Only accessible to certain individuals
  - Encrypt username and password data when sending, storing and retrieving for increased security
- Implementing password checking
  - Login information stored in file
    - `fopen` function
    - Read, write, append modes
  - Store data using `fputs`
    - `\n` newline character
  - Close files when done
    - `fclose` function

6 Verifying a Username and Password

- Implementing password checking, cont.
  - Trim newline character
    - `chop` function
  - Split string into substrings given a certain delimiter
    - `split` function
  - If username/password match list, allow access

Outline

password.html (1 of 4)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.15: password.html                        -->
<!-- XHTML form sent to password.php for verification -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Verifying a username and a password.</title>

      <style type = "text/css">
         td { background-color: #DDDDDD }
      </style>
   </head>

   <body style = "font-family: arial">
      <p style = "font-size: 13pt">
         Type in your username and password below.
         <br />
         <span style = "color: #0000FF; font-size: 10pt;
            font-weight: bold">
            Note that password will be sent as plain text
         </span>
      </p>

      <!-- post form data to password.php -->
      <form action = "password.php" method = "post">
         <br />

         <table border = "0" cellspacing = "0"
            style = "height: 90px; width: 123px;
            font-size: 10pt" cellpadding = "0">

            <tr>
               <td colspan = "3">
                  <strong>Username:</strong>
               </td>
            </tr>

            <tr>
               <td colspan = "3">
                  <input size = "40" name = "USERNAME"
                     style = "height: 22px; width: 115px" />
               </td>
            </tr>

```

Form data is posted to password.php.

```
            <tr>
               <td colspan = "3">
                  <strong>Password:</strong>
               </td>
            </tr>

            <tr>
               <td colspan = "3">
                  <input size = "40" name = "PASSWORD"
                     style = "height: 22px; width: 115px"
                     type = "password" />
               <br/></td>
            </tr>

            <tr>
               <td colspan = "1">
                  <input type = "submit" name = "Enter"
                     value = "Enter" style = "height: 23px;
                     width: 47px" />
               </td>
               <td colspan = "2">
                  <input type = "submit" name = "NewUser"
                     value = "New User"
                     style = "height: 23px" />
               </td>
```

Outline

password.html (3 of 4)

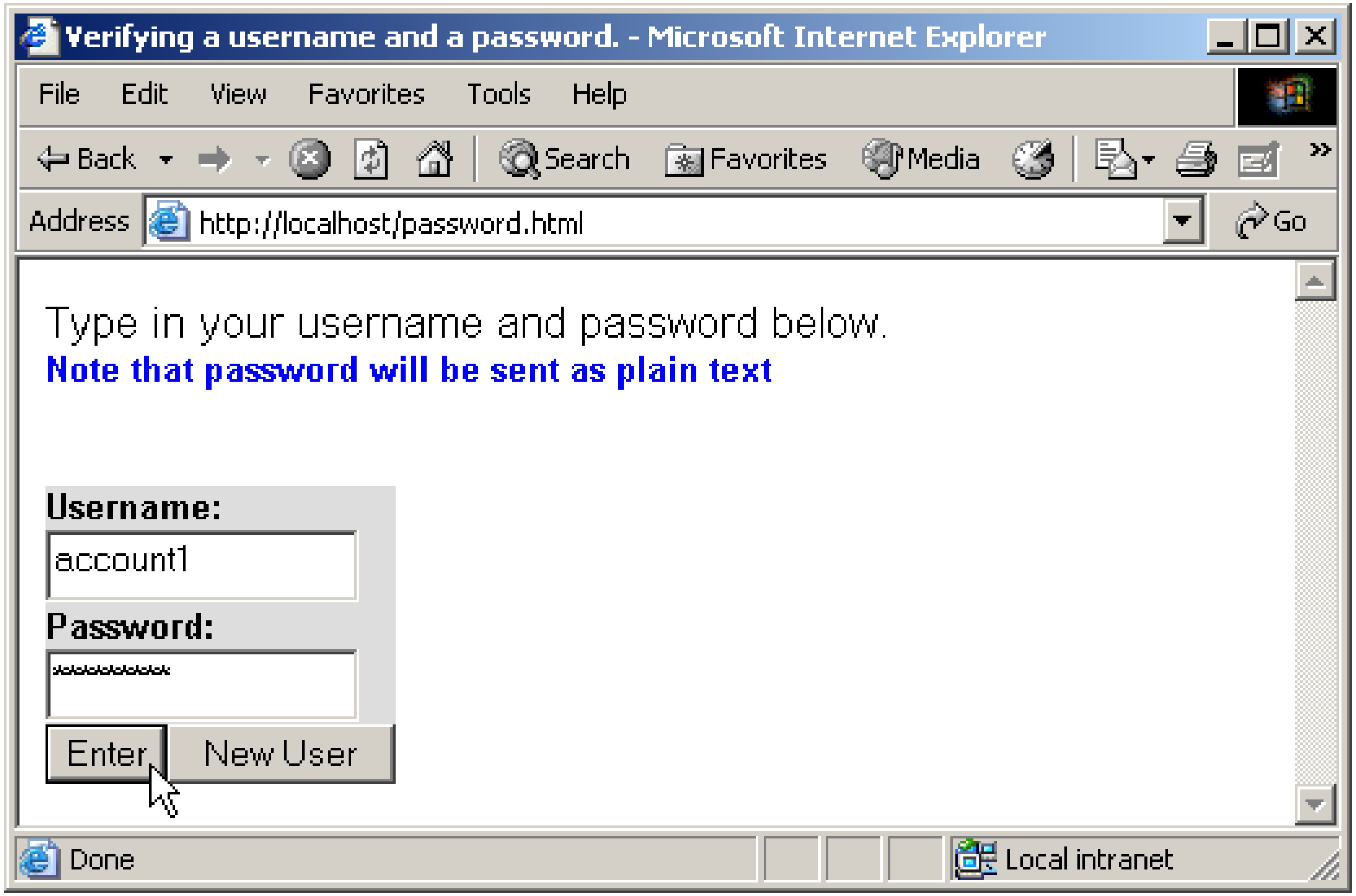
Outline

```
                  </tr>
               </table>
            </form>
         </body>
      </html>
```

password.html
(4 of 4)

6 Verifying a Username and Password

Fig. 26.15 XHTML form for obtaining a username and password.



Verifying a username and a password. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go
Done
Local intranet

http://localhost/password.html

Type in your username and password below.

Note that password will be sent as plain text

Username:

account1

Password:

****

Enter New User

password.php
(1 of 7)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.16: password.php                          -->
<!-- Searching a database for usernames and passwords. -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <?php
         extract( $_POST );

         // check if user has left USERNAME or PASSWORD field blank
         if ( !$USERNAME || !$PASSWORD ) {
            fieldsBlank();
            die();
         }

         // check if the New User button w
         if ( isset( $NewUser ) ) {

            // open password.txt for writing using append mode
            if ( !( $file = fopen( "password.txt",
               "a" ) ) ) {

```

Variable names, when preceded by the logical negation operator (`!`), return `true` if they are empty or set to `0`. This checks if a user has submitted a form without specifying a username or password.

Function `fieldsBlank` is called if the user has

Function `isset` tests whether the user has pressed the **New User** button, indicating that a new user must be added.

To add a new user, we open the file `password.txt` in `append` mode and assign the file handle that is returned to variable `$file`.

password.php

```
            // print error message and terminate script
            // execution if file cannot be opened
            print( "<title>Error</title></head><body>
              Could not open password file
              </body></html>" );
            die();
         }

         // write username and password to file and
         // call function userAdded
         fputs( $file, "$USERNAME,$PASSWORD\n" );
         userAdded( $USERNAME );
      }
      else {

         // if a new user
         // for reading
         if ( !( $file = fopen( "password.txt",
            "r" ) ) ) {
            print( "<title>Error</title></head>
               <body>Could not open password file
               </body></html>" );
            die();
         }

```

Print an error message and terminate script execution if the file cannot be opened.

Function userAdded is called to print a message to the user to indicate that the username and password were added to the file.

Outline

password.php
(3 of 7)

```
      $userVerified = 0;

      // read each line in fi
      // and passwor
      while ( !feof(



         $

         // split us
         $field = sp

         // verify username
         if ( $USERNAME == $field[ 0 ] ) {
            $userVerified = 1;

            // call function checkPassword to
            // user's password
            if ( checkPassword( $PASSWORD, $f
               == true )
               accessGranted( $USERNAME );
            else
               wrongPassword();
```

Before entering the while loop, variable

Function fgets reads a line from the text file. The result is assigned to variable $line.

The while l
lines in the f

Function chop removes the newline character
from the end of the line

Function split is called to separate the string at the specified delimiter (in this case, a comma). The
resulting

The username entered by the user is tested against the one returned in the text file (stored in the first element of the array). If they match, variable $userVerified is set to 1.

If function checkPassword returns true, function accessGranted is called to notify the client that permission has been granted. Otherwise, function wrongPassword is called.

Function checkPassword is called to verify the user's password. Variable $PASSWORD and array $field are passed to the function.

Outline

rd.php
(4 of 7)

```
      }
   }

   // close text
   fclose( $file

   // call function accessDenied if username has
   // not been verified
   if ( !$userVerified )
      accessDenied();
}

 // verify user password and return a boolean
function checkPassword( $userpassword, $filedata )
{
   if ( $userpassword == $filedata[ 1 ] )
      return true;
   else
      return false;
}

```

tion

If variable `$userVerified` has not been set to a value other than `0`, function `accessDenied` is called to notify the client that access has been denied.

Function `checkPassword` compares the user’s password to the password in the file. If they match, `true` is returned, whereas `false` is returned if they do not.

```
    // print a message indicating the user has been added
   function userAdded( $name )
   {
      print( "<title>Thank You</title></head>
         <body style = \"fo
         font-size: 1em; co
         <strong>You have been added
         to the user list, $name.
         <br />Enjoy the site.</strong

   }

    // print a message indicating permission
    // has been granted
   function accessGranted( $name )
   {
      print( "<title>Thank You</title></head>
         <body style = \"font-family: arial;
         font-size: 1em; color: blue\">
         <strong>Permission has been
         granted, $name. <br />
         Enjoy the site.</strong>" );
   }

```

Function `userAdded` prints a message to the client indicating that the user has been added.

Function `accessGranted` prints a message to the client indicating that permission has been granted.

password.php
(5 of 7)

Outline

.php

(6 of 7)

```
      // print a message indicating password is invalid
      function wrongPassword()
      {
         print( "<title>Access Denied
            <body style = \"font-family: arial;
            font-size: 1em; color: red\">
            <strong>You entered an invalid
            password.<br />Access has
            been denied.</strong>" );
      }

      // print a message indicating access has been denied
      function accessDenied()
      {
         print( "<title>Access Denied</title></head>
            <body style = \"font-family: arial;
            font-size: 1em; color: red\">
            <strong>
            You were denied access to this server.
            <br /></strong>" );
      }

```

Function `wrongPassword` prints a message to the client indicating that the password is invalid.

Function `accessDenied` prints a message to the client indicating that access has been denied.

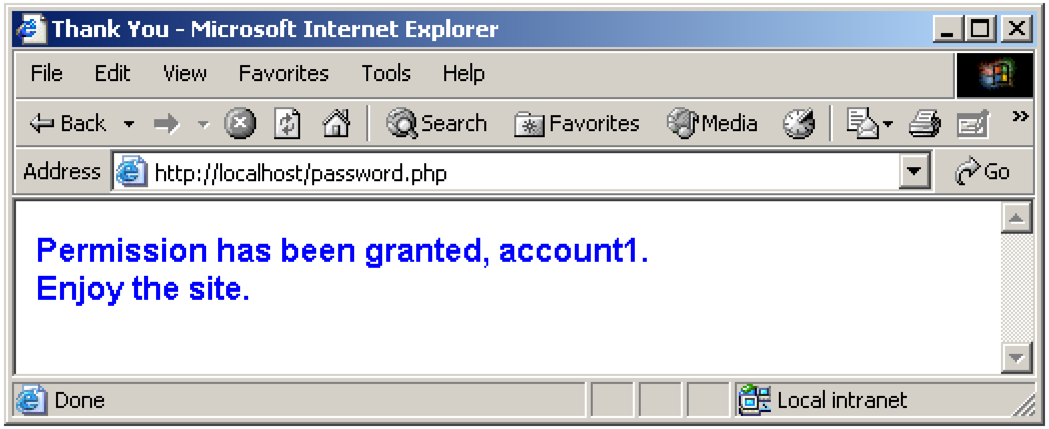
```
          // print a message indicating that fields
          // have been left blank
         function fieldsBlank()
         {
            print( "<title>Access Denied</title></head>
               <body style = \"font-family: arial;
               font-size: 1em; color: red\">
               <strong>
               Please fill in all form fields.
               <br /></strong>" );
         }
      ?>
   </body>
</html>
```

Function `fieldsBlank` prints a message to the client indicating that all form fields have not been completed.

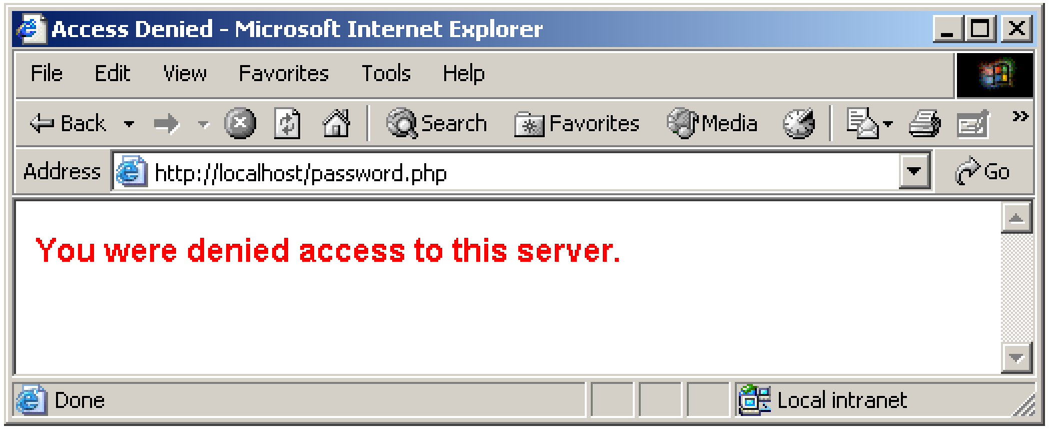
password.php
(7 of 7)

6 Verifying a Username and Password

Fig. 26.16 Verifying a username and password.


Thank You - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/password.php
Go
Done
Local intranet

Permission has been granted, account1.
Enjoy the site.


Access Denied - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/password.php
Go
Done
Local intranet

You were denied access to this server.

```
account1,password1
account2,password2
account3,password3
account4,password4
account5,password5
account6,password6
account7,password7
account8,password8
account9,password9
account10,password10
```

password.txt
(1 of 1)

7 Connecting to a Database

- Databases
  - Store and maintain data
  - MySQL is a free database product
  - PHP supports many database operations
    - Access databases from Web pages

data.html
(1 of 2)

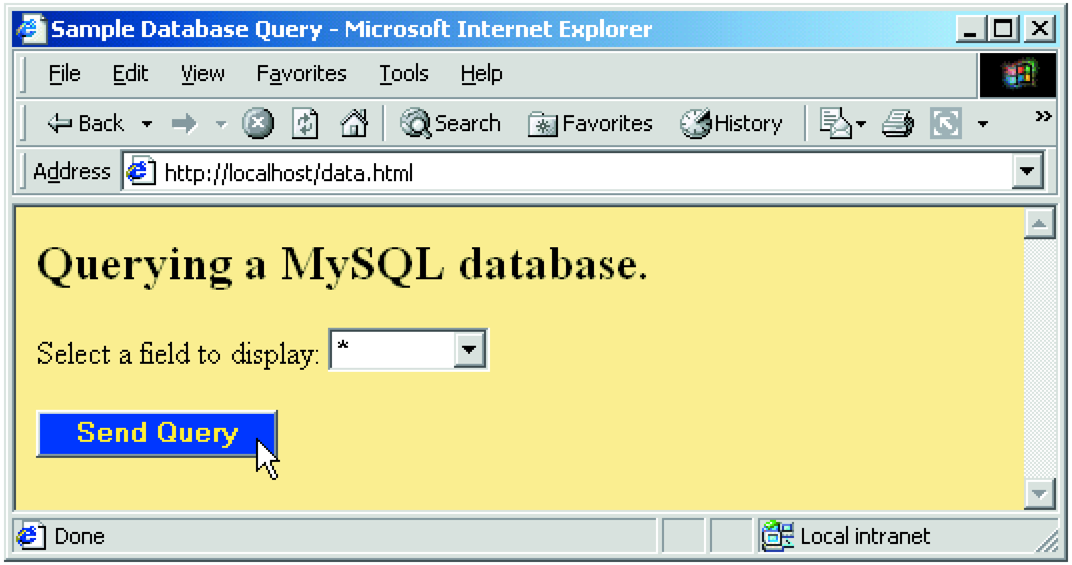
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.18: data.html     -->
<!-- Querying a MySQL Database -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sample Database Query</title>
   </head>

   <body style = "background-color: #F0E68C">
      <h2 style = "font-family: arial color: blue">
         Querying a MySQL database.
      </h2>

      <form method = "post" action = "database.php">
         <p>Select a field to display:

            <!-- add a select box containing options -->
            <!-- for SELECT query                    -->
            <select name = "select">
               <option selected = "selected">*</option>
               <option>ID</option>
               <option>Title</option>
               <option>Category</option>
               <option>ISBN</option>
            </select>
         </p>

         <input type = "submit" value = "Send Query"
            style = "background-color: blue;
            color: yellow; font-weight: bold" />
      </form>
   </body>
</html>
```

Select box containing options for a SELECT query.

(2 of 2)

7 Connecting to a Database

Fig. 26.18 Form to query a MySQL database.



Sample Database Query - Microsoft Internet Explorer

http://localhost/data.html

Querying a MySQL database.

Select a field to display: *

7 Connecting to a Database

- Interacting with databases
  - SQL
    - Structured Query Language
    - Used to manipulate databases
  - Several useful functions
    - `mysql_connect`
    - `mysql_select_db`
    - `mysql_query`
    - `mysql_error`
    - `mysql_fetch_row`
    - `mysql_close`

database.php
(1 of 3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.19: database.php        -->
<!-- Program to query a database and -->
<!-- send results to the client.     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Search Results</title>
   </head>

   <body style = "font-family: arial, sans-serif"
      style = "background-color: #F0E68C">
      <?php

         extract( $_POST );

         // build SELECT query
         $query = "SELECT " . $select . " F

         // Connect to MySQL
         if ( !( $database = mysql_connect( "localhost",
            "httpd", "" ) ) )
            die( "Could not connect to database" );
```

Build the select query and assign the string to variable `$query`.

Function `mysql_connect` returns a database handle which represents PHP’s connection to a database. If this connection is not made, function `die` is called to terminate script execution.

Outline

se.php

(2 of 3)

```
         // open Products database
         if ( !mysql_select_db( "Products", $
            die( "Could not open Products dat

         // query Products database
         if ( !( $
            print(
            die( mysql_error() );
         }
      ?>

      <h3 style = "color: blue">
      Search Results</h3>

      <table border = "1" cellpadding = "3" cellspacing = "2"
         style = "background-color: #ADD8E6">

         <?php

            // fetch each record in result se
            for ( $counter = 0;
               $row = mysql_fetch_row( $resul
               $counter++ ){

```

Function `mysql_query` returns an object containing the result set of the query, which we assign to variable `$result`.

Function `mysql_select_db` is called to specify the database to be queried.

The `for` loop iterates through each record in the result set while constructing an XHTML table from the results. Variable `$counter` is incremented by one for each row retrieved.

Function `mysql_fetch_row` returns an array containing the elements of each row in the result set of our query (`$result`).

database.php
(3 of 3)

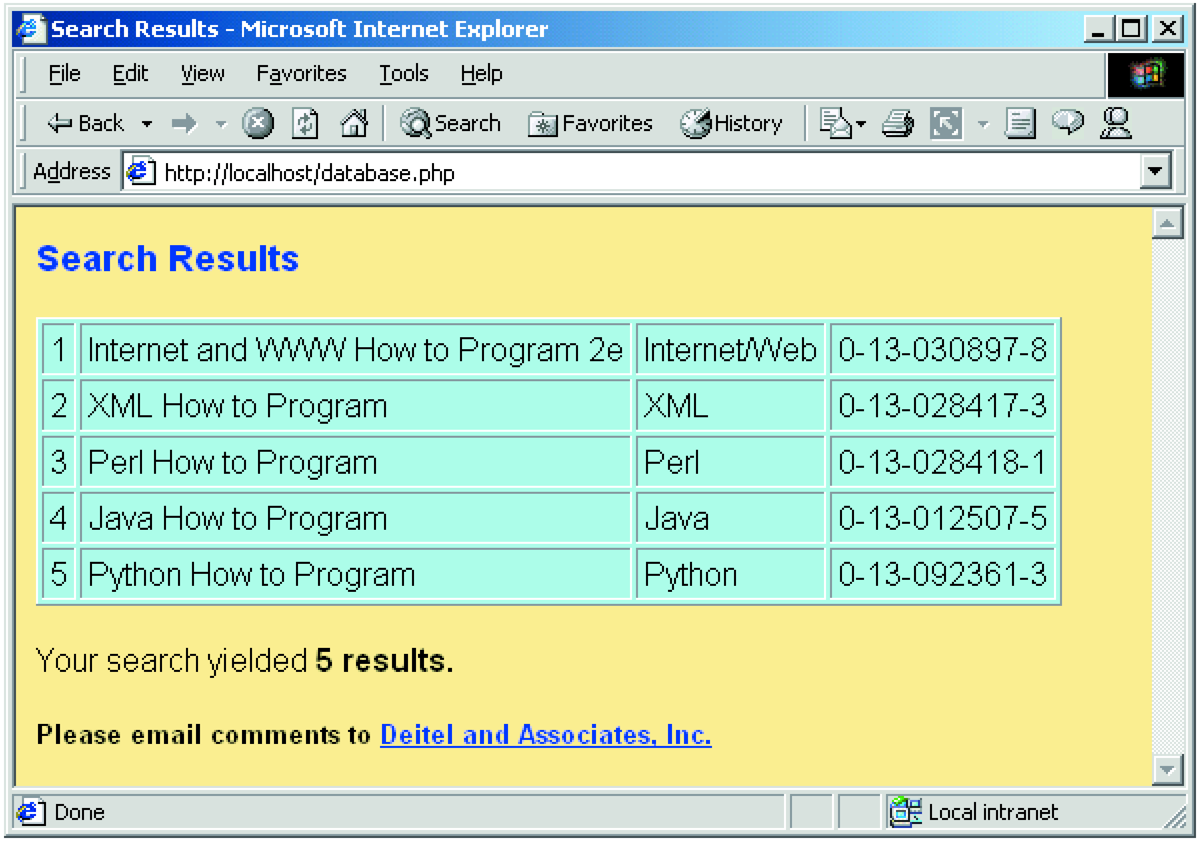
```
               // build table to display results
               print( "<tr>" );

               foreach ( $row as $key => $value )
                  print( "<td>$value</td>" );

               print( "</tr>" );
            }

            mysql_close( $databas
         ?>

      </table>

      <br />Your search yielded <strong>
      <?php print( "$counter" ) ?> results.<br /><br /></strong>

      <h5>Please email comments to
         <a href = "mailto:deitel@deitel.com">
            Deitel and Associates, Inc.
         </a>
      </h5>

   </body>
</html>
```

The `foreach` loop iterates through the array containing the elements of each row an

The total number of results are printed to the client.

7 Connecting to a Database

Fig. 26.19 Querying a database and displaying the results.



Search Results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost/database.php
Done
Local intranet

Search Results

1	Internet and WWW How to Program 2e	Internet/Web	0-13-030897-8
2	XML How to Program	XML	0-13-028417-3
3	Perl How to Program	Perl	0-13-028418-1
4	Java How to Program	Java	0-13-012507-5
5	Python How to Program	Python	0-13-092361-3

Your search yielded **5 results.**

Please email comments to Deitel and Associates, Inc.

8 Cookies

- Cookies
  - Store information on client computer
  - Track preferences and other information
  - Stored as text files on hard drive
  - Never store sensitive information, such as credit card numbers, in a cookie
    - Security risk
- Cookies and PHP
  - `setcookie` function
    - Name
    - Value
    - Expiration date

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.20: cookies.html -->
<!-- Writing a Cookie         -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Writing a cookie to the client computer</title>
   </head>

   <body style = "font-family: arial, sans-serif;
      background-color: #99CCFF">

      <h2>Click Write Cookie to save your cookie data.</h2>

```

cookies.html
(1 of 2)

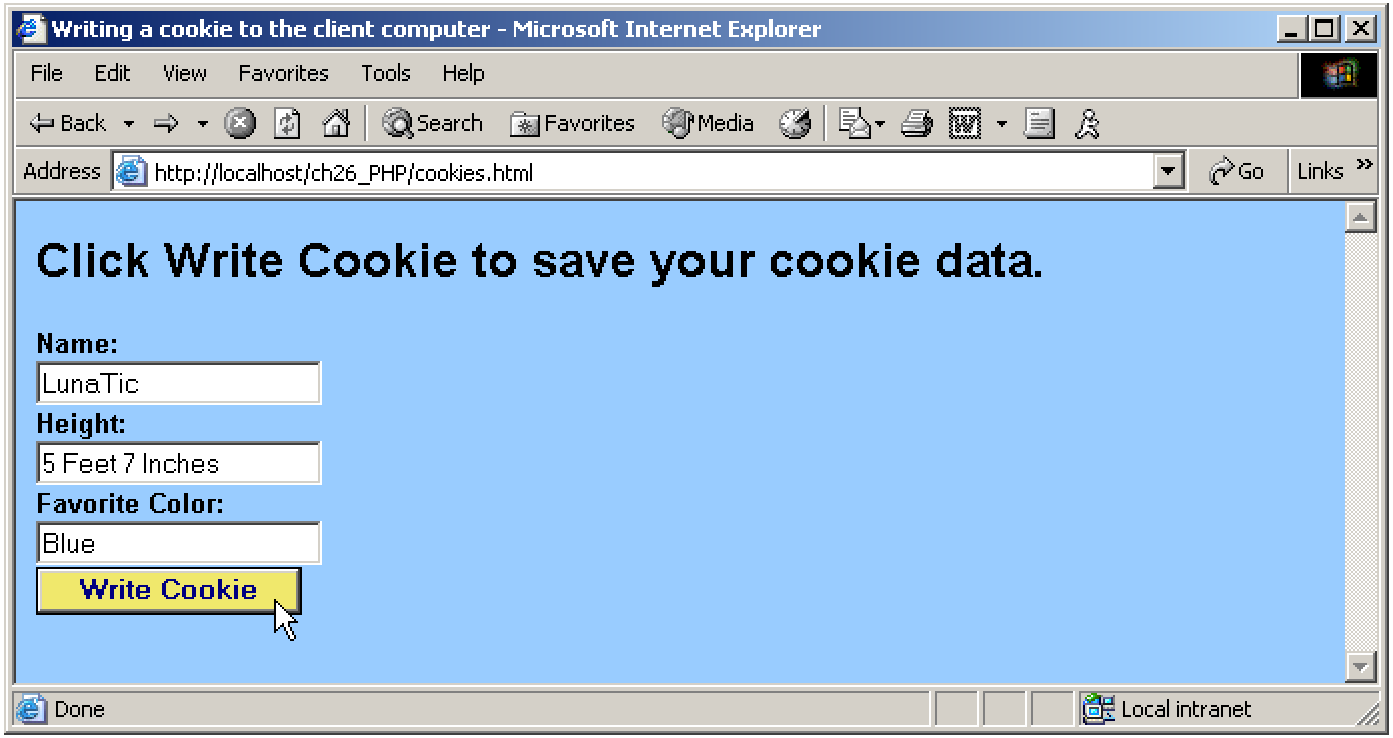
cookies.html
(2 of 2)

```
      <form method = "post" action = "cookies.php"
            style = "font-size: 10pt">
         <strong>Name:</strong><br />
         <input type = "text" name = "NAME" /><br />

         <strong>Height:</strong><br />
         <input type = "text" name = "HEIGHT" /><br />

         <strong>Favorite Color:</strong><br />
         <input type = "text" name = "COLOR" /><br />

         <input type = "submit" value = "Write Cookie"
            style = "background-color: #F0E86C; color: navy;
            font-weight: bold" /></p>
      </form>
   </body>
</html>
```

Form data is posted to cookies.php.

8 Cookies

Fig. 26.20 Gathering data to be written as a cookie.



Writing a cookie to the client computer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/ch26_PHP/cookies.html
Go Links
Done
Local intranet

Click Write Cookie to save your cookie data.

Name:
LunaTic

Height:
5 Feet 7 Inches

Favorite Color:
Blue

Write Cookie

cookies.php
(1 of 2)

```
<?php
   // Fig. 26.21: cookies.php
   // Program to write a cookie to a client's machine

   extract( $_POST );
   // write each form field’s value to a cookie and set the
   // cookie’s expiration date
   setcookie( "Name", $NAME, time() + 60 * 60 * 24 * 5 );
   setcookie( "Height", $HEIGHT, time() + 60 * 60 * 24 * 5 );
   setcookie( "Color", $COLOR, time() + 60 * 60 * 24 * 5 );
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transi
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transiti

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Cookie Saved</title>
   </head>

   <body style = "font-family: arial, sans-serif">
      <p>The cookie has been set with the following data:</p>

```

Function `setcookie` takes the name of the cookie to be set as the first argument, followed by the value to be stored in the cookie. The optional third argument specifies the expiration date of the cookie.

ne

cookies.php
(2 of 2)

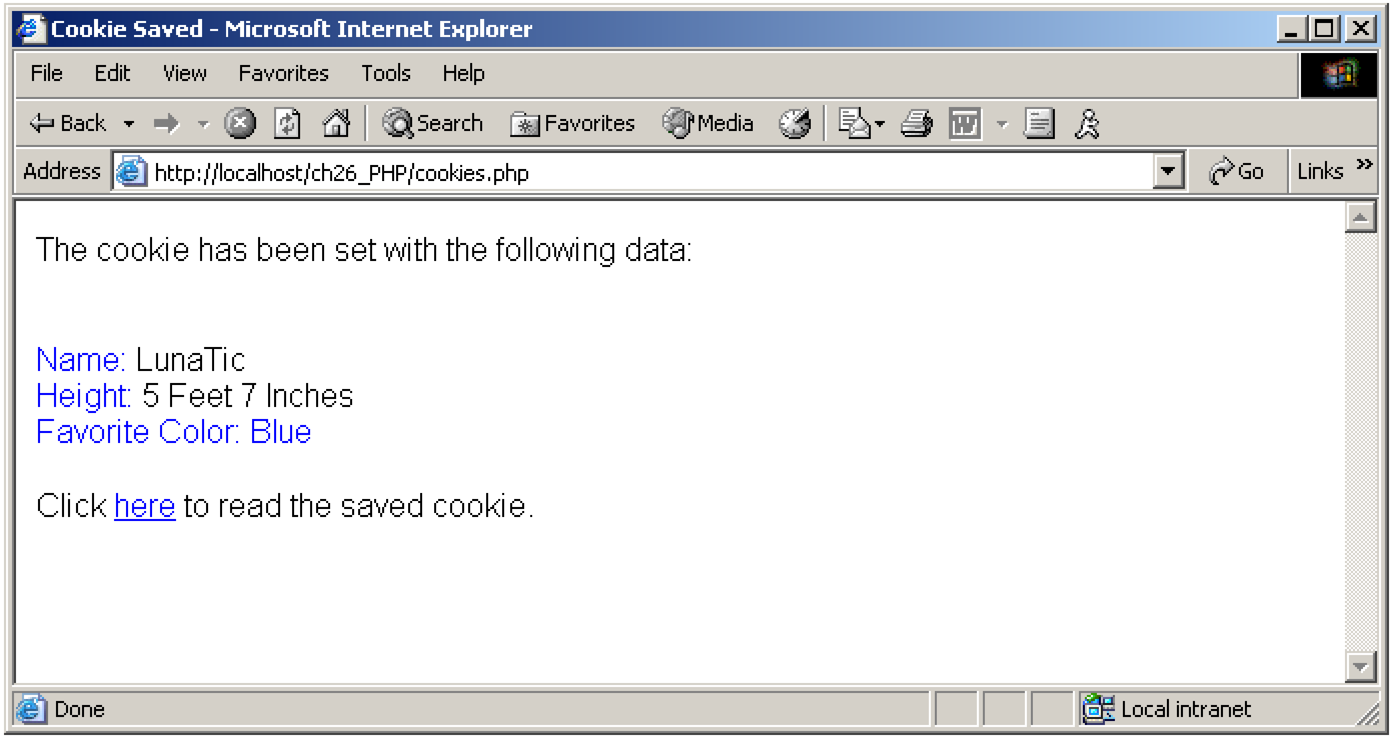
```
      <!-- print each form field’s value -->
      <br /><span style = "color: blue">Name:</spa
         <?php print( $NAME ) ?><br />

      <span style = "color: blue">Height:</span>
         <?php print( $HEIGHT ) ?><br />

      <span style = "color: blue">Favorite Color:</span>

      <span style = "color: <?php print( "$COLOR\">$COLOR" ) ?>
      </span><br />
      <p>Click <a href = "readCookies.php">here</a>
         to read the saved cookie.</p>
   </body>
</html>
```

Each form field’s `value` is printed to confirm the data that has been set as a cookie with the user.

Hyperlink to `readCookies.php`.

8 Cookies

Fig. 26.21 Writing a cookie to the client.



Cookie Saved - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/ch26_PHP/cookies.php
Go Links
Done
Local intranet

The cookie has been set with the following data:

Name: LunaTic
Height: 5 Feet 7 Inches
Favorite Color: Blue

Click here to read the saved cookie.

8 Cookies

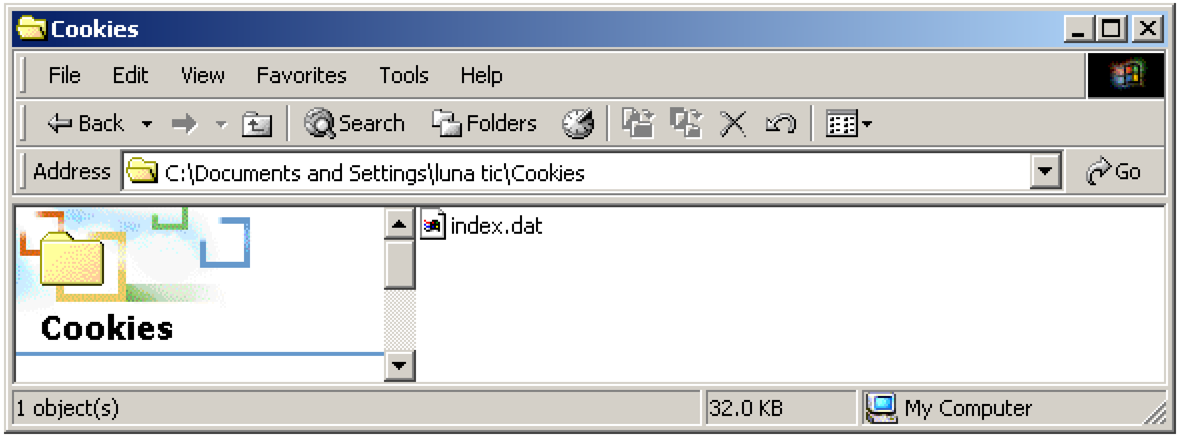
- Reading cookies
  - `$_COOKIE` environment variable
    - Array
  - `foreach` loop to access each element
    - Split into key and value

8 Cookies

- Cookie storage
  - Internet Explorer
    - Stores cookies in **Cookies** directory
    - Text file

8 Cookies

Fig. 26.22 **Cookies** directory before a cookie is written.

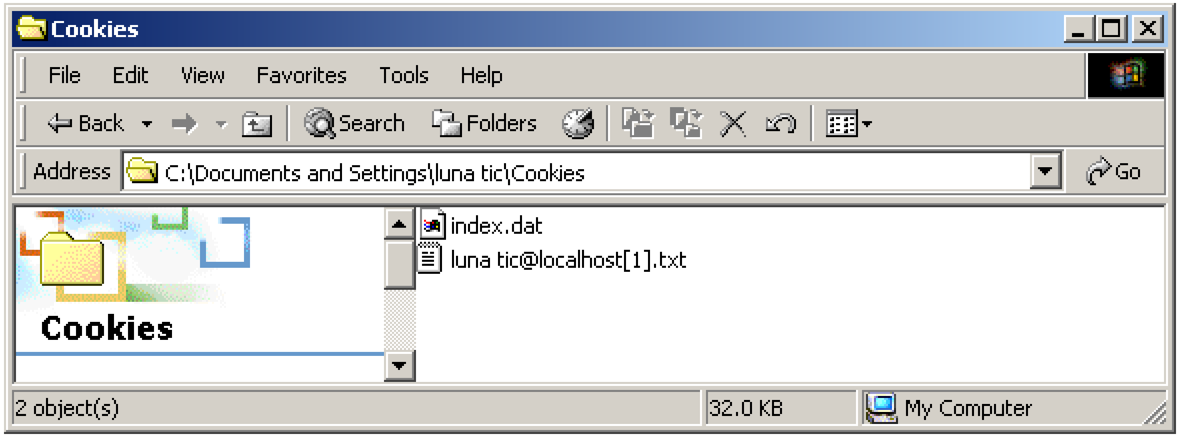


Address: C:\Documents and Settings\luna tic\Cookies

index.dat

1 object(s) | 32.0 KB | My Computer

Fig. 26.23 **Cookies** directory after a cookie is written.



Address: C:\Documents and Settings\luna tic\Cookies

index.dat
luna tic@localhost[1].txt

2 object(s) | 32.0 KB | My Computer

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.24: readCookies.php                       -->
<!-- Program to read cookies from the client's computer -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head><title>Read Cookies</title></head>

   <body style = "font-family: arial, sans-serif">

      <p>
         <strong>
            The following data is saved in a cookie on your
            computer.
         </strong>
      </p>

```

readCookies.php
(1 of 2)

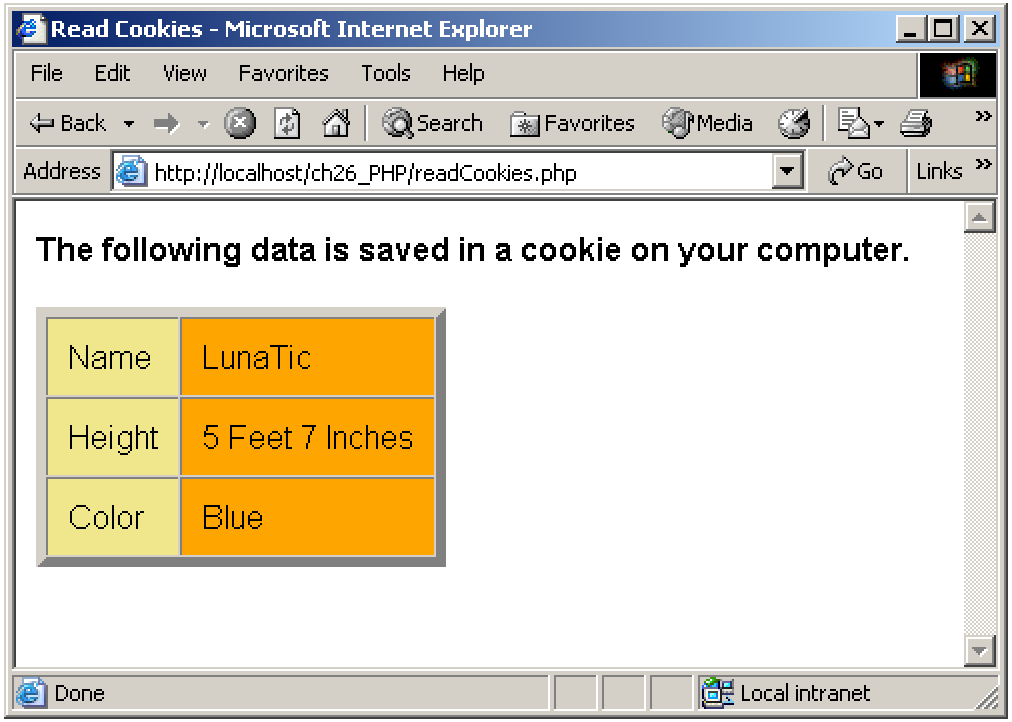
readCookies.php
(2 of 2)

```
      <table border = "5" cellspacing = "0" cellpadding = "10">
         <?php

            // iterate thro
            // name and value of each cookie
            foreach ( $_COOKIE as $key => $value )
               print( "<tr>
                  <td bgcolor=\"#F0E68C\">$key</td>
                  <td bgcolor=\"#FFA500\">$value</td>
                  </tr>" );
         ?>

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

The `foreach` loop iterates through the `$_COOKIE` array and prints the name and value of each cookie in an XHTML table.

8 Cookies

Fig. 26.24 Displaying the cookie's content.



Read Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/ch26_PHP/readCookies.php
Go Links
Done
Local intranet

The following data is saved in a cookie on your computer.

Name	LunaTic
Height	5 Feet 7 Inches
Color	Blue

9 Dynamic Content in PHP

- Dynamically alter XHTML content
  - Form’s `action` property set to same page that contains it
  - Perform different actions when page is loaded and form is submitted
    - `isset` variable
  - Check for errors
    - Write different XHTML when errors encountered
  - `$$`*variable* syntax
    - References variable whose name equals the value of `$`*variable*
  - If input is valid, make MySQL database calls

Outline

dynamicForm.php (1 of 9)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.25: dynamicForm.php             -->
<!-- Form for use with the form.php program -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sample form to take user input in XHTML</title>
</head>

<body>
   <?php
      extract ( $_POST );
      $iserror = false;

      // array of book titles
      $booklist = array( "Internet and WWW How to Program 3e",
         "C++ How to Program 4e",
         "Java How to Program 5e",
         "XML How to Program 1e" );

```

Build array of options for the form.

dynamicForm.php
(2 of 9)

```
      // array of possible operating systems
      $systemlist = array( "Windows XP",
         "Windows 2000",
         "Windows 98",
         "Linux",
         "Other");

      // array of name and alt values for the text input fields
      $inputlist = array( "fname" => "First Name",
         "lname" => "Last
         "email" => "Emai
         "phone" => "Phon

      if ( isset ( $submit ) ) {
         if ( $fname == "" ) {
            $formerrors[ "fnameerror" ] = true;
            $iserror = true;
         }

         if ( $lname == "" ) {
            $formerrors[ "lnameerror" ] = true;
            $iserror = true;
         }

```

If the page is being loaded as a result of a form submission, do error checking and then retrieve information from the database.

```
         if ( $email == "" ) {
            $formerrors[ "emailerror" ] = true;
            $iserror = true;
         }

         if ( !ereg( "^\([0-9]{3}\)[0-9]{3}-[0-9]{4}$", $phone ) ) {
            $formerrors[ "phoneerror" ] = true;
            $iserror = true;
         }

         if ( !$iserror ) {

            // build INSERT query
            $query = "INSERT INTO contacts " .
               "( LastName, FirstName, Email, Phone, Book, OS ) " .
               "VALUES ( '$lname', '$fname', '$email', " .
               "'" . quotemeta( $phone ) . "', '$book', '$os' )";

            // Connect to MySQL
            if ( !( $database = mysql_connect( "localhost",
               "httpd", "" ) ) )
               die( "Could not connect to database" );

            // open MailingList database
            if ( !mysql_select_db( "MailingList", $database ) )
               die( "Could not open MailingList database" );
```

If there were no errors, query the MySQL database.

dynamicForm.php
(3 of 9)

dynamicForm.php (4 of 9)

```
            // execute query in MailingList database
            if ( !( $result = mysql_query( $query, $database ) ) ) {
               print( "Could not execute query! <br />" );
               die( mysql_error() );
            }

            print( "<p>Hi
               <span style = 'color: blue'>
               <strong>$fname</strong></span>.
               Thank you for completing the survey.<br />

               You have been added to the
               <span style = 'color: blue'>
               <strong>$book</strong></span>
               mailing list.
               </p>
               <strong>The following information has been saved
               in our database:</strong><br />

               <table border = '0' cellpadding = '0' cellspacing = '10'>
               <tr>
               <td bgcolor = '#ffffaa'>Name</td>
               <td bgcolor = '#ffffbb'>Email</td>
               <td bgcolor = '#ffffcc'>Phone</td>
```

Outline

dynamicForm.php (5 of 9)

```
               <td bgcolor = '#ffffdd'>OS</td>
               </tr>
               <tr>

               <!-- print each form field's value -->
               <td>$fname $lname</td>
               <td>$email</td>
               <td>$phone</td>
               <td>$os</td>
               </tr></table>

               <br /><br /><br />
               <div style = 'font-size: 10pt; text-align: center'>
               <div style = 'font-size : 18pt'>
               <a href = 'formDatabase.php'>
               Click here 
               This is onl
               You have not been added to a mailing list.
               </div></body></html>" );
            die();
         }
      }

      print( "<h1>This is a sample registration form.</h1>
         Please fill in all fields and click Register." );
```

Halt the script so the form-generation code does not execute.

dynamicForm.php (6 of 9)

```
      if ( $iserror ) {
         print( "<br /><span style = 'color : red'>
            Fields with * need to be filled in properly.</span>" );
      }

      print( "<!-- post form data to form.php -->
         <form method = 'post' action = 'dynamicform.php'>
         <img src = 'images/user.gif' alt = 'User' /><br />
         <span style = 'color: blue'>
         Please fill out the fields below.<br />
         </span>

         <!-- create four
      foreach ( $inputlist as $inputname => $inputalt ) {
         $inputtext = $inputvalues[ $inputname ];

         print( "<img src = 'im
            alt = '$inputalt' /
            name = '$inputname'

         if ( $formerrors[ ( $inputname )."error" ] == true )
            print( "<span style = 'color : red'>*</span>" );

         print( "<br />" );
      }
```

Fill in the forms using $$*variable* syntax.

If the form input contained errors, place a red asterisk (*) next to the text field.

dynamicForm.php
(7 of 9)

```
      print( "<span style = 'font-size : 10pt" );

      if ( $formerrors[ "phoneerror" ] )
         print( "; color : red" );

      print( "'>Must be in the form (555)555-5555
         </span><br /><br />

         <img src = 'images/downloads.gif'
         alt = 'Publications' /><br />

         <span style = 'color: blue'>
         Which book would you like information about?
         </span><br />

         <!-- create drop-down list containing book names -->
         <select name = 'book'>" );

      foreach ( $booklist as $cu
         print( "<option" );

         if ( ( $currbook == $book ) )
            print( " selected = 'true'" );

```

Make sure the correct book is selected in the dropdown box.

dynamicForm.php
(8 of 9)

```
         print( ">$currbook</option>" );
      }

      print( "</select><br /><br />
         <img src = 'images/os.gif' alt = 'Operating System' />
         <br /><span style = 'color: blue'>
         Which operating system are you currently using?
         <br /></span>

         <!-- create five radio buttons -->" );

      $counter = 0;

      foreach ( $systemlist as $
         print( "<input type = '
            value = '$currsystem

         if ( $currsystem == $os ) print( "checked = 'checked'" );
         if ( $iserror && $counter == 0 ) print( "checked = 'checked'" );

         print( " />$currsystem" );

         if ( $counter == 2 ) print( "<br />" );
         $counter++;
      }

```

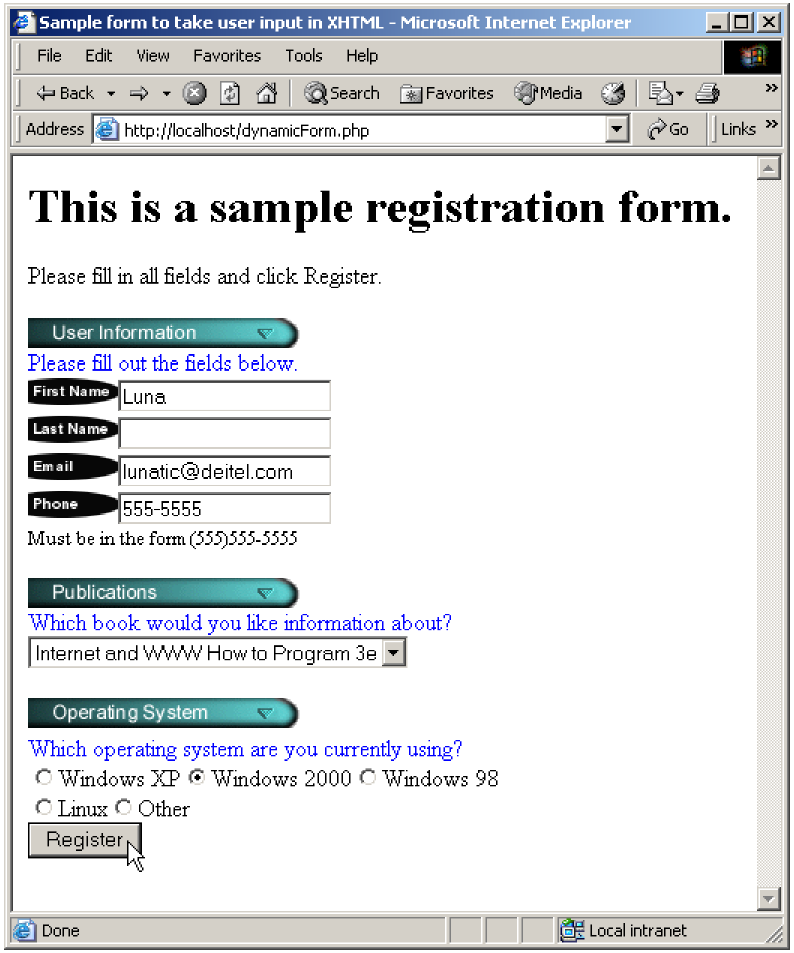
Make sure the correct OS is checked in the checkbox.

```
      print( "<!-- create a submit button -->
         <br />
         <input type = 'submit' name = 'submit' value = 'Register' />
         </form></body></html>" );
   ?>
```

dynamicForm.php (9 of 9)

9 Dynamic Content in PHP

Fig. 26.25 Dynamic form using PHP.


Sample form to take user input in XHTML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/dynamicForm.php
Go Links
Done
Local intranet

This is a sample registration form.

Please fill in all fields and click Register.

User Information

Please fill out the fields below.

First Name	Luna
Last Name	
Email	lunatic@deitel.com
Phone	555-5555

Must be in the form (555)555-5555

Publications

Which book would you like information about?

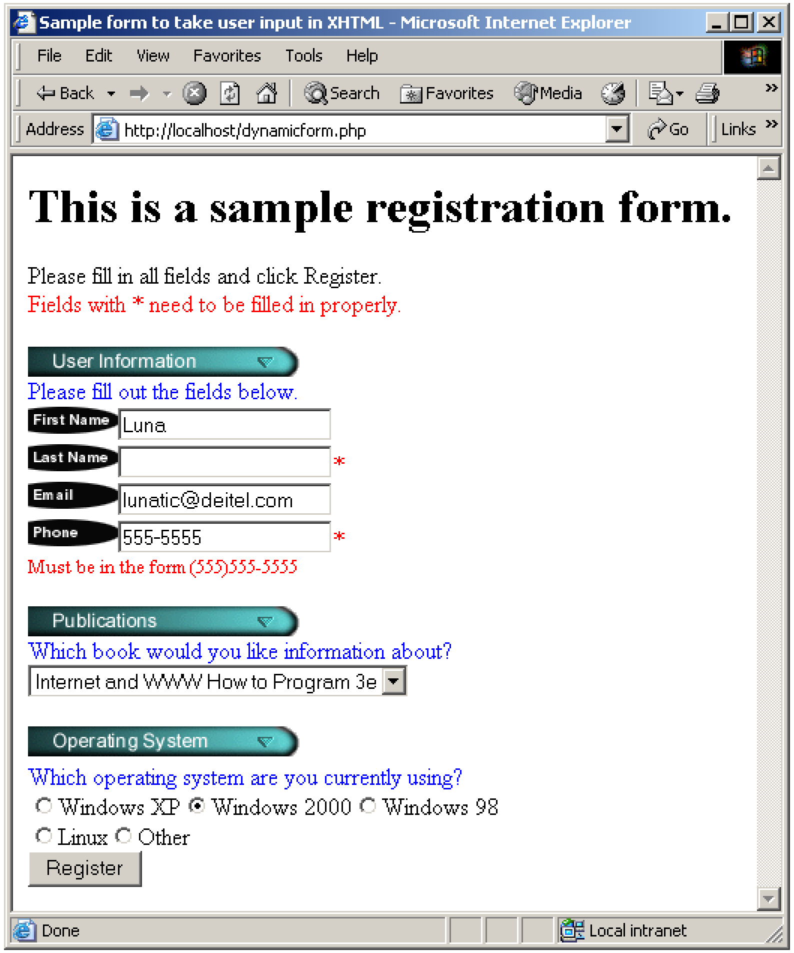
Internet and WWW How to Program 3e

Operating System

Which operating system are you currently using?

○ Windows XP ● Windows 2000 ○ Windows 98
○ Linux ○ Other

Register


Sample form to take user input in XHTML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/dynamicform.php
Go Links
Done
Local intranet

This is a sample registration form.

Please fill in all fields and click Register.
Fields with * need to be filled in properly.

User Information

Please fill out the fields below.

First Name	Luna	
Last Name		*
Email	lunatic@deitel.com	
Phone	555-5555	*

Must be in the form (555)555-5555

Publications

Which book would you like information about?

Internet and WWW How to Program 3e

Operating System

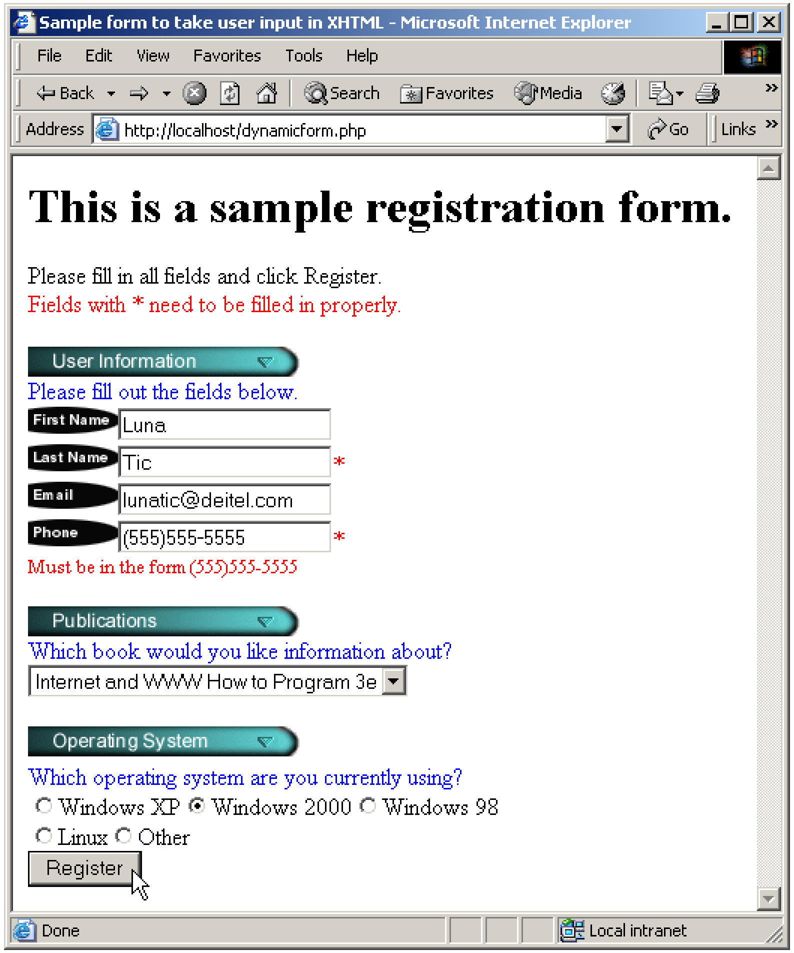
Which operating system are you currently using?

○ Windows XP ● Windows 2000 ○ Windows 98
○ Linux ○ Other

Register

9 Dynamic Content in PHP

Fig. 26.25 Dynamic form using PHP.


Sample form to take user input in XHTML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/dynamicform.php
Go Links
Done
Local intranet

This is a sample registration form.

Please fill in all fields and click Register.
Fields with * need to be filled in properly.

User Information

Please fill out the fields below.

First Name	Luna
Last Name	Tic *
Email	lunatic@deitel.com
Phone	(555)555-5555 *

Must be in the form (555)555-5555

Publications

Which book would you like information about?

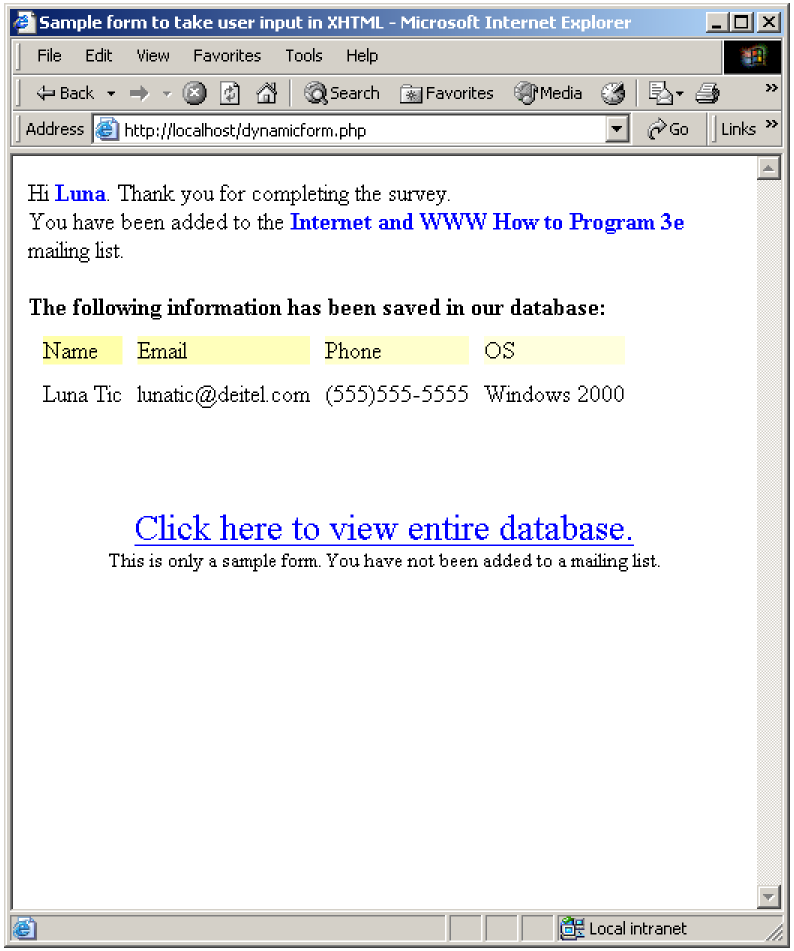
Internet and WWW How to Program 3e

Operating System

Which operating system are you currently using?

○ Windows XP ● Windows 2000 ○ Windows 98
○ Linux ○ Other

Register


Sample form to take user input in XHTML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/dynamicform.php
Go Links
Local intranet

Hi **Luna**. Thank you for completing the survey.
You have been added to the **Internet and WWW How to Program 3e** mailing list.

The following information has been saved in our database:

Name	Email	Phone	OS
Luna Tic	lunatic@deitel.com	(555)555-5555	Windows 2000

Click here to view entire database.

This is only a sample form. You have not been added to a mailing list.

formDatabase.php
(1 of 3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 26.26: formDatabase.php        -->
<!-- Program to query a database and -->
<!-- send results to the client.     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Search Results</title>
   </head>

   <body style = "font-family: arial, sans-serif"
      style = "background-color: #F0E68C">
      <?php

         extract( $_POST );

         // build SELECT query
         $query = "SELECT * FROM contacts";

         // Connect to MySQL
         if ( !( $database = mysql_connect( "localhost",
            "httpd", "" ) ) )
            die( "Could not connect to database" );
```

Build the query string.

```

         // open MailingList database
         if ( !mysql_select_db( "MailingList", $database ) )
            die( "Could not open MailingList database" );

         // query MailingList database
         if ( !( $result = mysql_query( $query, $database ) ) ) {
            print( "Could not execute query! <br />" );
            die( mysql_error() );
         }
      ?>

      <h3 style = "color: blue">
      Mailing List Contacts</h3>

      <table border = "1" cellpadding = "3" cellspacing = "2"
         style = "background-color: #ADD8E6">

         <tr>
            <td>ID</td>
            <td>Last Name</td>
            <td>First Name</td>
            <td>E-mail Address</td>
            <td>Phone Number</td>
            <td>Book</td>
```

formDatabase.php
(3 of 3)

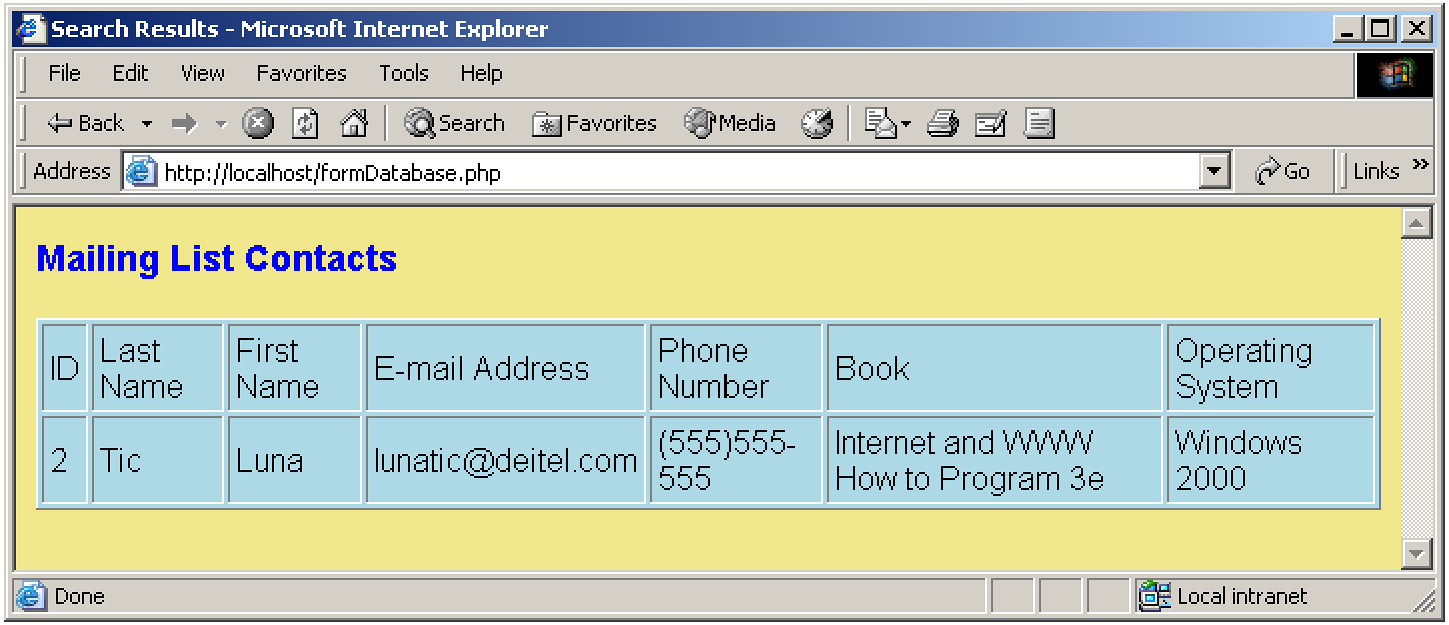
```
               <td>Operating System</td>
            </tr>
            <?php

               // fetch each record in resul
               for ( $counter = 0;
                  $row = mysql_fetch_row( $result );
                  $counter++ ){

                  // build table to dis
                  print( "<tr>" );

                  foreach ( $row as $key => $value )
                     print( "<td>$value</td>" );

                  print( "</tr>" );
               }

               mysql_close( $database );
            ?>

         </table>

      </body>
   </html>
```

Retrieve each mailing list member record from the database.

Dynamically create a table containing each mailing list member.

9 Dynamic Content in PHP

Fig. 26.26 Displaying the **MailingList** database.


Search Results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/formDatabase.php
Go Links
Done
Local intranet

Mailing List Contacts

ID	Last Name	First Name	E-mail Address	Phone Number	Book	Operating System
2	Tic	Luna	lunatic@deitel.com	(555)555-555	Internet and WWW How to Program 3e	Windows 2000

10 Operator Precedence

Operator	Type	Associativity
new	constructor	none
[]	subscript	right to left
~ ! ++ -- - @	bitwise not not increment decrement unary negative error control	right to left
* / %	multiplication division modulus	left to right
+ - .	addition subtraction concatenation	left to right
<< >>	bitwise shift left bitwise shift right	left to right
< > <= >=	less than greater than less than or equal greater than or equal	none
== != === !==	equal not equal identical not identical	none

Fig. 26.27 PHP operator precedence and associativity.

10 Operator Precedence

Operator	Type	Associativity
&	bitwise AND	left to right
^	bitwise XOR	left to right
|	bitwise OR	left to right
&&	logical AND	left to right
||	logical OR	left to right
= += -= *= /= &= |= ^= .= <<= >>=	assignment addition assignment subtraction assignment multiplication assignment division assignment bitwise AND assignment bitwise OR assignment bitwise exclusive OR assignment concatenation assignment bitwise shift left assignment bitwise shift right assignment	left to right
and	logical AND	left to right
xor	exclusive OR	left to right
or	logical OR	left to right
,	list	left to right

Fig. 26.27 PHP operator precedence and associativity.

Variables (4) – superglobal scope

- superglobal variables are available in all scopes throughout the script; no need to be declared global in a local function; were introduced in PHP 4
- the superglobal variables are:

$GLOBALS – contains references to all variables defined in the global scope of the script

$_SERVER - array containing information such as headers, paths, and script locations; built by the web server

$_GET - array of variables passed to the current script via the URL parameters

$_POST - array of variables passed to the current script via the HTTP POST method

$_FILES - array of items uploaded to the current script via the HTTP POST method

$_COOKIE - array of variables passed to the current script via HTTP Cookies

$_SESSION - array containing session variables available to the current script

$_REQUEST - array that by default contains the contents of $_GET, $_POST and $_COOKIE

$_ENV - array of variables passed to the current script via the environment method

Variables (5) – global vs. superglobal examples

```
function test_global()
    {
       // Most predefined variables aren't "super" and require
       // 'global' to be available to the functions local scope.
       global $HTTP_POST_VARS;

       echo $HTTP_POST_VARS['name'];

       // Superglobals are available in any scope and do
       // not require 'global'. Superglobals are available
       // as of PHP 4.1.0, and HTTP_POST_VARS is now
       // deemed deprecated.
       echo $_POST['name'];
    }
```

$GLOBALS

```
function test() {
        $foo = "local variable";

        echo '$foo in global scope: ' . $GLOBALS["foo"] . "\n";
        echo '$foo in current scope: ' . $foo . "\n";
}
$foo = "Example content";
test();
```

will print:

```
$foo in global scope: Example content
$foo in current scope: local variable
```

$_Server

- keys:

‘PHP_SELF’ – the filename currently executed

‘SERVER_ADDR’ – the IP address of the server

‘SERVER_PROTOCOL’ – name and version of the protocol via which the page is requested; HTTP/1.1

‘REQUEST_METHOD’ – the request method

‘QUERY_STRING’ – the query string

‘DOCUMENT_ROOT’ – the document root under which the current script is executed

‘REMOTE_ADDR’ – the client IP address

‘REMOTE_PORT’ – the client port

‘HTTP_ACCEPT’ – the HTTP accept field of the HTTP protocol

etc.

$_GET

- an html example

```
<form action="welcome.php" method="get">
    Name: <input type="text" name="fname" />
    Age: <input type="text" name="age" />
    <input type="submit" />
</form>
```

- after submit, the URL is:

```
    http://www.w3schools.com/welcome.php?fname=Peter&age=37
```

- the ‘welcome.php’ file:

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

$_POST

- an html example

```
<form action="welcome.php" method=“post">
    Name: <input type="text" name="fname" />
    Age: <input type="text" name="age" />
    <input type="submit" />
</form>
```

- after submit, the URL is:

```
    http://www.w3schools.com/welcome.php
```

- the ‘welcome.php’ file:

```
Welcome <?php echo $_POST["fname"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old!
```

Functions

- the syntax of defining a function is:

```
function functionName($param1, $param2,…,$paramn) {
        … statements…
        return …;
}
```

- ex.:

```
<?php
function add($x,$y) {
    $total=$x+$y;
    return $total;
}

echo "1 + 16 = " . add(1,16);
?>
```

Classes and Objects – simple example

```
class SimpleClass {
        // property declaration
        public $var = 'a default value';
        // method declaration
        public function displayVar() {
            echo $this->var;
        }
}
$instance = new SimpleClass();
class ExtendClass extends SimpleClass {
        // Redefine the parent method
        function displayVar() {
            echo "Extending class\n";
            parent::displayVar();
        }
}
$extended = new ExtendClass();
$extended->displayVar();
```

Types

- boolean: a non-zero numeric value or empty string or array, NULL are automatically converted to FALSE; other values are cast to TRUE
- integer, float, double: integers in decimal base, hexadecimal (prefixed by “0x”), and octal (prefixed by “0”)
- string
- array
- object: reference type to cast class instances to
- resource: a reference to an external resource(curl session, ftp session, database link, pdf document etc.) created and used by special functions
- NULL: a variable with no value (no value has been set or the variable has been unset())
- pseudo-types: mixed (e.g. the type parameter of gettype()), callback functions, void (e.g. function returning void)

The String type

- a character is a byte (native Unicode support expected in PHP 6)
- 4 ways of defining a string literal:
  - single quotes: $str = ‘this is a string’;
  - double quotes: $str = “this is a string”;
  - heredoc: (the closing identifier must be in the beginning of the line and can only be followed by ‘;’)

    ```
    $str = <<<FOO
    this is
    a string
    FOO;
    ```
  - nowdoc: (no parsing is done inside a nowdoc; usefull for embedding PHP code or large body of thext without escaping)

    ```
    $str = <<<‘FOO’
    this is
    a string
    FOO;
    ```

The String type (2)

- in a double quotes or heredoc string, variables are parsed within it, in a single quotes and nowdoc string, they are not
- there are 2 syntaxes for using variables in a string:
  - simple - variable is preceded by '$': echo "some text $var";
  - complex – complex expressions are enclosed in "{…}":

    echo "some text {$ob->vect['foo']->val}";
- a string can be indexed, e.g. $str[3] – 4th character of str
- in string context all other values are automatically converted to strings (e.g. 23->"23", TRUE->"1")
- in numeric context, strings are automatically converted to integer/float; e.g. $n=1+"2 zzz" => $n=3
- the "." operator is for string concatenation ('+' is not ok)

The String type (3) - functions

- echo(), print(), printf(), sprintf(), fprintf() – for displaying strings
- crypt(), md5(), sha1() – hashing function
- explode(), strtok() – string tokenizer
- ltrim(), rtrim(), str_replace(), str_shuffle(), str_split(), str_word_count(), strchr(), strcmp(), strlen(), strstr(), strpos(), strtolower(), strtoupper(), substr(), substr_compare(), substr_count(), substr_replace() – string manipulation functions
- sscanf() – parsing input

Arrays

- arrays in PHP are actually ordered maps (key-value pair sequences)
- keys can be only integer or string values
- in no key is specified for an element, the value of the previous key plus 1 is used (keys start at 0 if not specified)
- examples:

  $a = array("a"=>45, 2=>7, 36=>"zzz")

  $b = array(4=>40, 67, 87, "b"=>3) is the same as:

  $b = array(4=>40, 5=>67, 6=>87, "b"=>3)

  $c = array(2=>"zz", 45=>array("a"=>11, 23=>34)) – a multidimensional array

Arrays (2)

- accessing a component of the array by indexing it:

```
$v = array(1=>2, 2=>”zz”, vect=>array(2, 3, 4));
$v[2] = 45;
$v[‘vect’][1]=4;
```

- defining an array can be done by setting a value for a specific component:

```
$v[2]=3;
```

- removing a key/pair value or the whole array:

```
unset($v[2]);
unset($v);
```

- a primary value (i.e. integer, float, string, boolean) can be converted automatically to an array with one component having at index 0 that value
- count($v) counts the elements of $v and sort($v) sorts the elements of $v
- parsing a vector: foreach($persons as $p) { echo $p; }

Functions useful with types

- gettype($var) – return the type of $var
- settype($var,”newtype”) – for explicit conversion
- boolean is_array($var)
- boolean is_binary($var)
- boolean is_bool($var)
- boolean is_buffer($var)
- boolean is_callable($var)
- boolean is_double($var)
- boolean is_float($var)
- boolean is_int($var)
- boolean is_integer($var)
- boolean is_long($var)
- boolean is_null($var)
- boolean is_numeric($var)
- boolean is_object($var)
- boolean is_real($var)
- boolean is_resource($var)
- boolean is_scalar($var)
- boolean is_string($var)
- boolean is_unicode($var)

Operators

- arithmetic operators:

  + - * / % ++ --
- assignment operators:

  = += -= *= /= .= %=
- comparison operators:

  == != <> > >= < <=

  === (identical) !== (not identical)
- bitwise operators:

  & | ^ ~ << >>
- logical operators: && || ! and or xor
- string operators: . (concatenation)
- ternary operator: (expr) ? (exprTrue) : (exprFalse)

Other operators

- error control operator (@) : when '@' is placed in front of an expression, if that expression generates an error message, that error message will be ignored
- execution operator (`…`) – like in Unix shells:

  $output = `ls –l `
- cast operators: ex.: (string) $a; (float) $b;
- array operators:

  $a + $b : union of arrays $a and $b (duplicate keys are not overwritten)

  $a == $b : true if $a and $b have the same key/value pairs

  $a === $b : true if $a and $b have the same key/value pairs in the same order and of the same type

  $a!=$b and $a<>$b : true if $a and $b don't have the same key/value pairs

  $a !== $b : true if $a and $b are not identical

Constants

- their scope is global
- are declared using the function define() or using const:

```
define(“const1”, “something”);
```

- the constant name is not prepend with ‘$’ when referenced:

```
echo const1;
```

- there are some predefined constants PHP offers:

__LINE__ : the current line number of the file
__FILE__ : the full path and name of current file
__DIR__ : the directory of the file
__FUNCTION__ : the name of the current function
__CLASS__ : the class name
__METHOD__ : the class method name
__NAMESPACE__ : the current namespace

Instructions

- if (cond) {…} elseif (cond) {…} … else {…}
- while (cond) { … }
- switch($var) { case val1: statements; case val2: statements; … ; default: statements; }
- do { … } while(cond)
- break can exit a do-while/while/for/foreach/switch structure
- continue skips the rest of the current iteration and begins a new iteration (if the condition is true) in a do-while/while/for/foreach loop
- for(init ; continue_cond; next) { … }
- foreach($vector as $val) { … }
- foreach($vector as $key=>$val) { … }

Other instructions

- PHP offers an alternative syntax for if, switch, while, for, foreach where the opening brace '{' is changed to ':' and the closing brace '}' is changed to endif;, endswitch;, endwhile;, endfor;, endforeach;. ex.:

```
while($n<4):
        $i++;
        echo $i;
endwhile;
```

- return – ends execution of current function
- goto:

```
label:
        $i++;
…
goto label;
```

include() and require()

- include() and require() include in the current context another PHP file
- ex.: include “settings.php”;

  require “global.php”;
- the code included inherits the variable scope of the line on which the include occurs
- parsing drops out of PHP mode and into HTML mode at the beginning of the included file and resumes again at the end
- if “allow_url_fopen” is enabled, the file to be included can be specified using an URL

Predefined Variables (superglobals)

- Superglobals — Superglobals are built-in variables that are always available in all scopes
- $GLOBALS — References all variables available in global scope
- $_SERVER — Server and execution environment information
- $_GET — HTTP GET variables
- $_POST — HTTP POST variables
- $_FILES — HTTP File Upload variables
- $_REQUEST — HTTP Request variables
- $_SESSION — Session variables
- $_ENV — Environment variables
- $_COOKIE — HTTP Cookies
- $php_errormsg — The previous error message
- $HTTP_RAW_POST_DATA — Raw POST data
- $http_response_header — HTTP response headers
- $argc — The number of arguments passed to script
- $argv — Array of arguments passed to script

Cookies

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.
- creating a cookie:

```
<?php
    $expire=time()+60*60*24*30;
    setcookie("user", "Alex Porter", $expire);
?>

<html>
    .....
</html>
```

Cookies (2)

- retrieve a cookie value:

```
<html>
<body>

<?php
    if (isset($_COOKIE["user"]))
           echo "Welcome " . $_COOKIE["user"] . "!<br />";
    else
            echo "Welcome guest!<br />";
?>

</body>
</html>
```

Cookies (3)

- delete a cookie = assuring the expiration date is in the past

```
<?php
    // set the expiration date to one hour ago
    setcookie("user", "", time()-3600);
?>
```

PHP sessions

- A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.
- Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.
- starting a session:

```
<?php session_start(); ?>

<html>
    <body>


    </body>
</html>
```

PHP sessions (2)

- storing a session variable:

```
<?php
    session_start();

    if(isset($_SESSION['views']))
            $_SESSION['views']=$_SESSION['views']+1;
    else
            $_SESSION['views']=1;
    echo "Views=". $_SESSION['views'];
?>
```

PHP sessions (3)

- free a session variable:

```
<?php
    unset($_SESSION['views']);
?>
```

- destroy a session:

```
<?php
    session_destroy();
?>
```

PHP and MySQL

- opening and closing a connection:

```
<?php
    $con = mysql_connect("localhost",“user",“pass");
    if (!$con)
      {
      die('Could not connect: ' . mysql_error());
      }

    // some code

    mysql_close($con);
?>
```

PHP and MySQL (2)

- querying and displaying the result example:

```
<?php
	$con = mysql_connect("localhost","peter","abc123");
	if (!$con) {
		die('Could not connect: ' . mysql_error());
	}

	mysql_select_db("my_db", $con);
	$result = mysql_query("SELECT * FROM Persons");

	echo "<table border='1'>
	<tr>
	<th>Firstname</th>
	<th>Lastname</th>
	</tr>";

	while($row = mysql_fetch_array($result)) {
		echo "<tr>";
		echo "<td>" . $row['FirstName'] . "</td>";
		echo "<td>" . $row['LastName'] . "</td>";
		echo "</tr>";
	}
	echo "</table>";
	mysql_close($con);
?>
```

AJAX - Asynchronous JavaScript and XML

What is AJAX ?

- AJAX is not a new programming language, but a new technique for creating better, faster, and more interactive web applications.
- With AJAX, a JavaScript can communicate directly with the server, with the **XMLHttpRequest** object. With this object, a JavaScript can trade data with a web server, without reloading the page.
- AJAX uses asynchronous data transfer (HTTP requests) between the browser and the web server, allowing web pages to request small bits of information from the server instead of whole pages.
- The AJAX technique makes Internet applications smaller, faster and more user-friendly.

AJAX example

```
var xmlhttp
function showHint(str) {
        if (str.length==0)  {
         document.getElementById("txtHint").innerHTML="";
         return;
        }
        xmlhttp=GetXmlHttpObject();
        if (xmlhttp==null)  {
         alert ("Your browser does not support XMLHTTP!");
         return;
        }
        var url=“submit.php";
         url=url+"?q="+str;
         url=url+"&sid="+Math.random();
         xmlhttp.onreadystatechange=stateChanged;
         xmlhttp.open("GET",url,true);
         xmlhttp.send(null);
}
function stateChanged() {
       if (xmlhttp.readyState==4) {
            document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
       }
}
function GetXmlHttpObject() {
       if (window.XMLHttpRequest) {     // code for IE7+, Firefox, Chrome, Opera, Safari
            return new XMLHttpRequest();
          }
       if (window.ActiveXObject) {         // code for IE6, IE5
           return new ActiveXObject("Microsoft.XMLHTTP");
       }
       return null;
}
```