Introduction to Server-Side Development with PHP

Chapters 11 and 15

Randy Connolly and Ricardo Hoar

Fundamentals of Web Development

© 2015 Pearson

http://www.funwebdev.com

Objectives

1. Server-Side Development
2. Web Server's Responsibilities
3. Quick Tour of PHP
4. Program Control
5. Functions
6. PHP Error Reporting

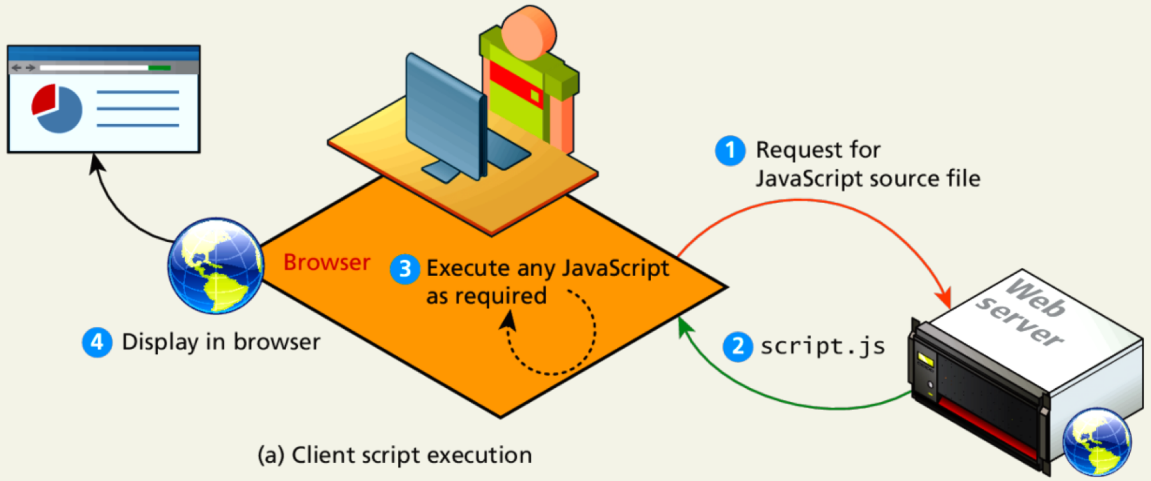
Section 1 of 6

WHAT IS SERVER-SIDE DEVELOPMENT

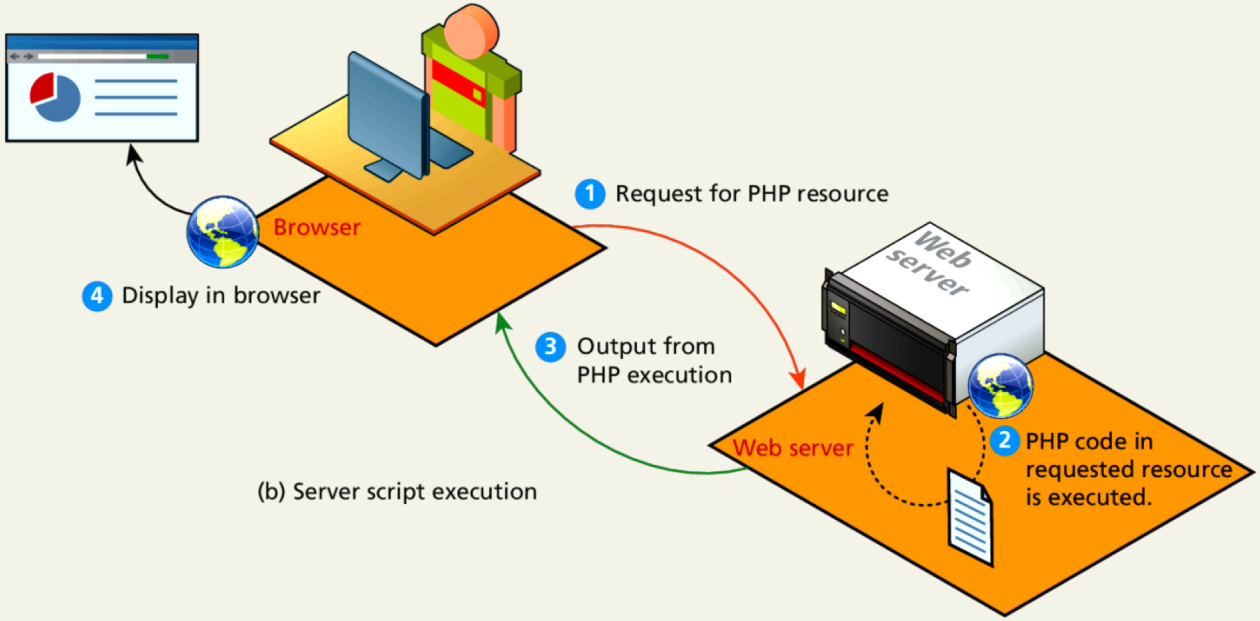
What is Server-Side Development

- The basic hosting of your files is achieved through a **web server**.
- **Server-side development** is much more than web hosting: it involves the use of a programming technology like PHP or ASP.NET to create scripts that **dynamically generate content**
- What is the difference between **server-side** and **client-side** scripting?

Comparing Client and Server Scripts



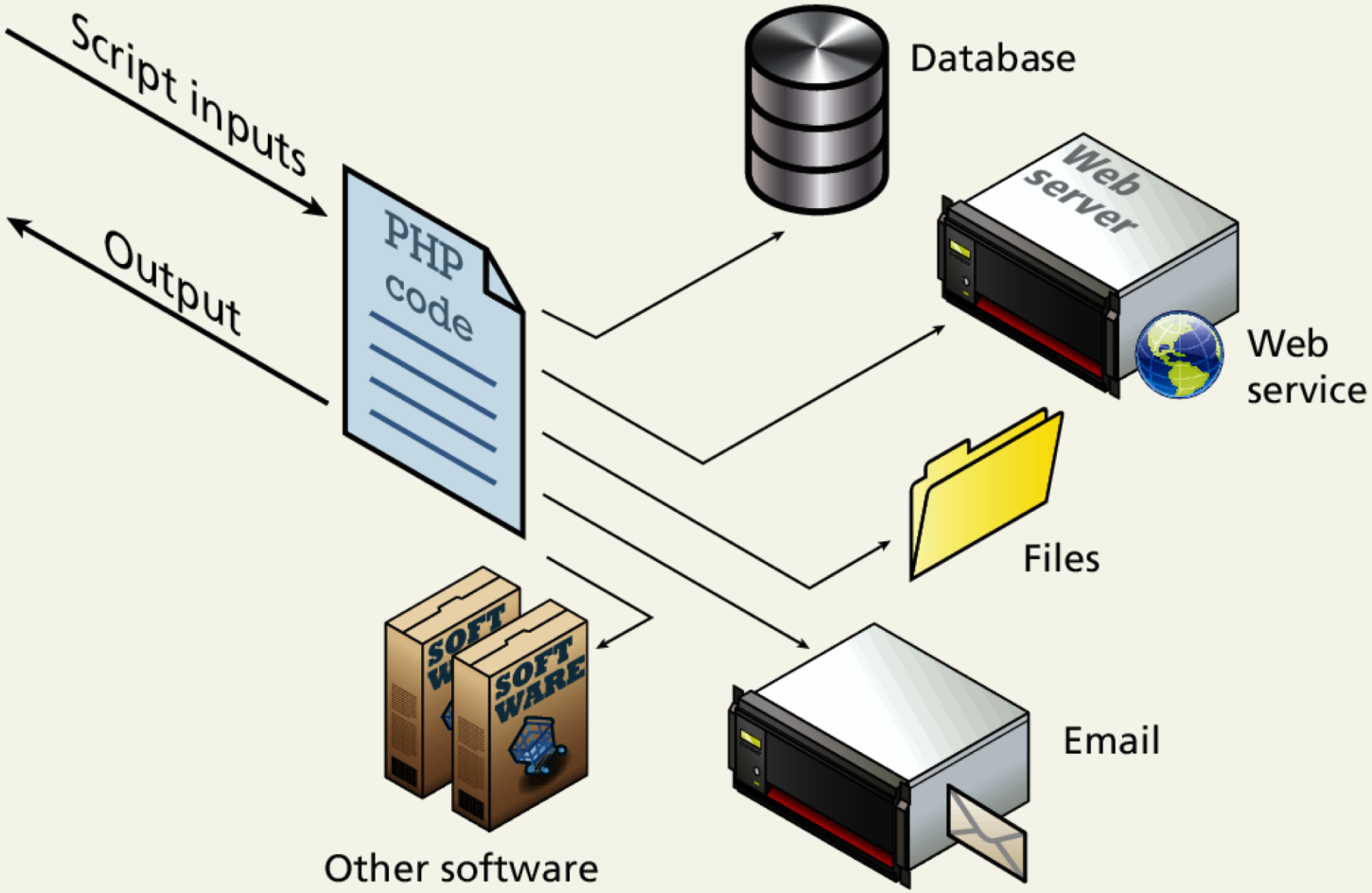
(a) Client script execution



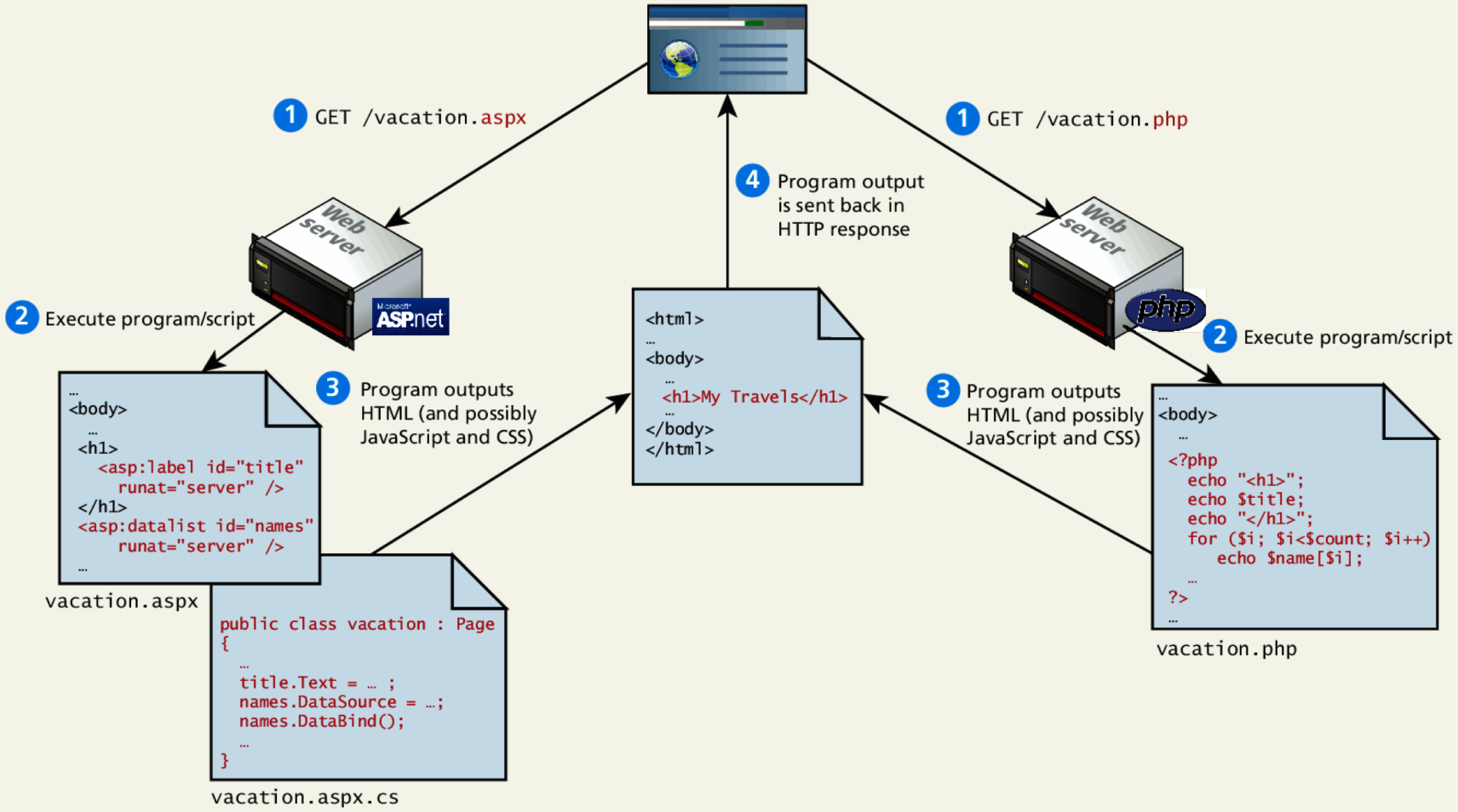
(b) Server script execution

Server-Side Script Resources

So many tools in your kit



Web Development Technologies



Comparing Server-Side Technologies

- **ASP.NET**. ASP.NET is part of Microsoft's .NET Framework and can use any .NET programming language (though C# is the most commonly used). ASP.NET uses an explicitly object-oriented approach.
- **JSP (Java Server Pages)**. JSP uses Java as its programming language and like ASP.NET it uses an explicit object-oriented approach.
- **Node.js**. This is a more recent server environment that uses JavaScript on the server side, thus allowing developers already familiar with JavaScript to use just a single language for both client-side and server-side development. Unlike the other development technologies listed here, node.js also is its own web server software, thus eliminating the need for Apache, IIS, or some other web server software.

Comparing Server-Side Technologies

- **Python**. This terse, object-oriented programming language has many uses, including being used to create web applications. It is also used in a variety of web development frameworks such as Django and Pyramid.
- **Ruby on Rails**. This is a web development framework that uses the Ruby programming language. Ruby on Rails emphasizes using an MVC design pattern.

Comparing Server-Side Technologies

- **PHP**:
  - Dynamically typed language that can be embedded directly within the HTML,
  - It now supports most common object-oriented features, such as classes and inheritance.
  - By default, PHP pages are compiled into an intermediary representation called **opcodes** that are analogous to Java's byte-code.
  - Originally, PHP stood for *personal home pages*, although it now is a recursive acronym that means *PHP: Hypertext Processor*.

Market Share

Websites are built with what?

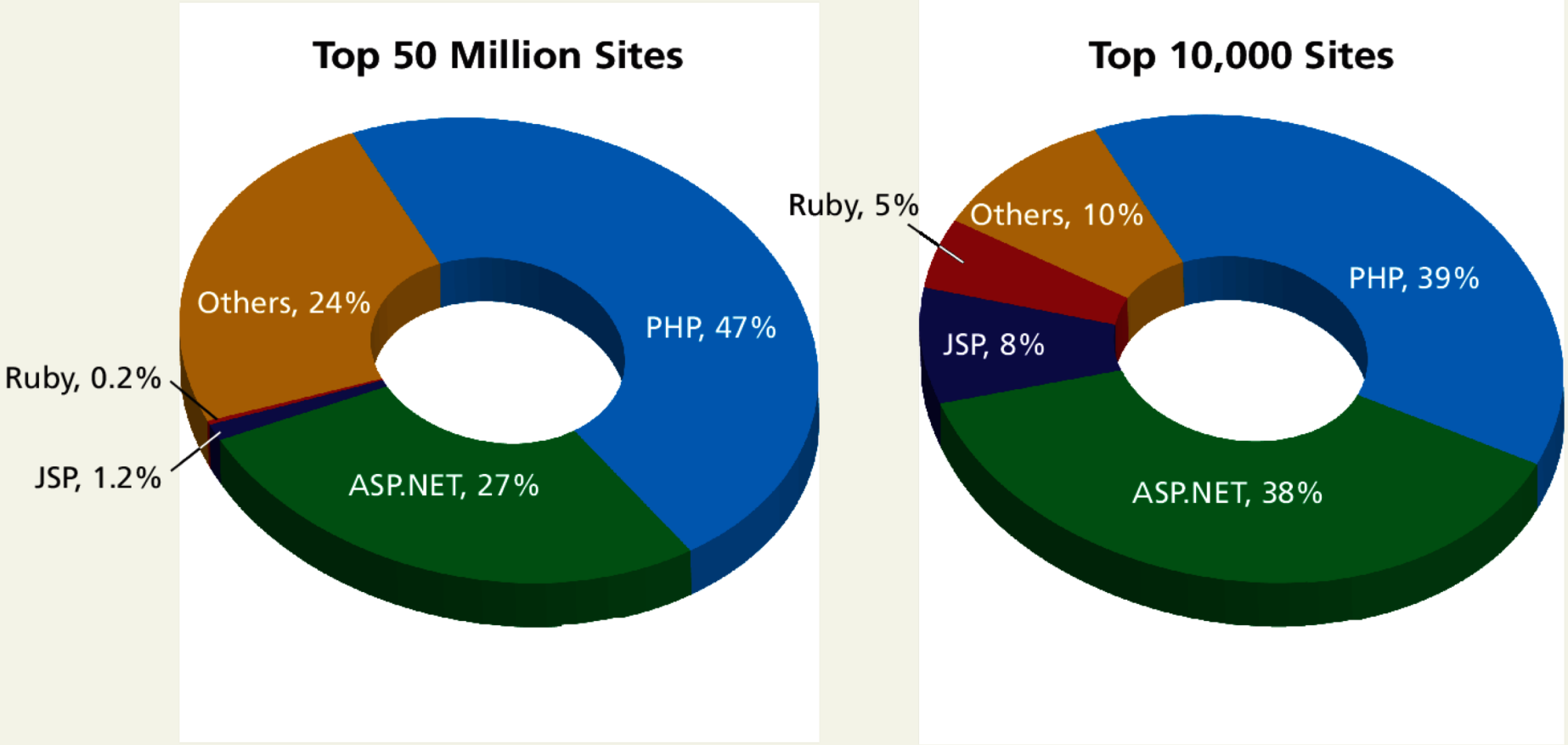
Have you wondered what technologies are used to built a certain web application?

Check: https://builtwith.com/KSU.edu.sa

Market Share

Of web development environments

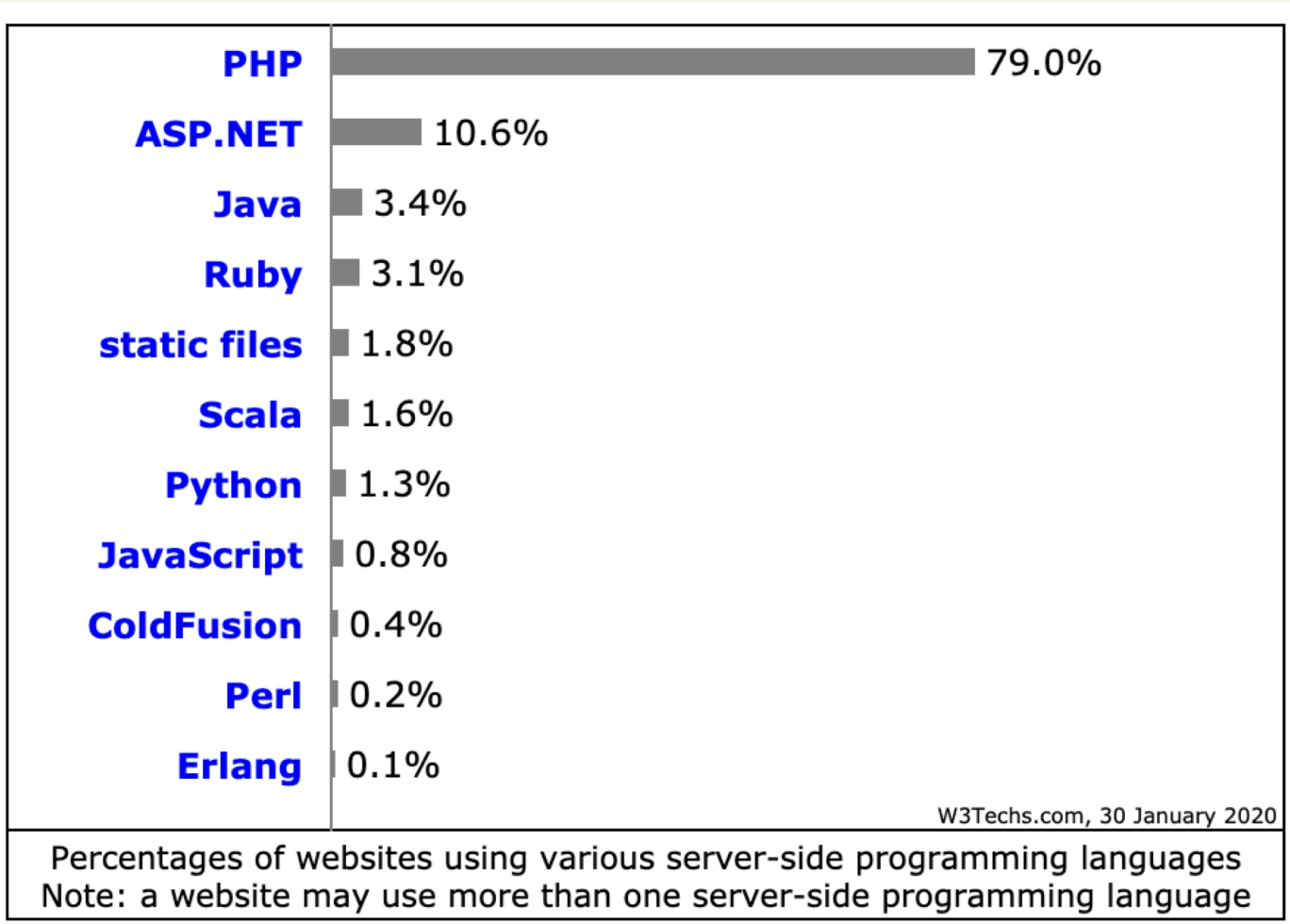
2015



Market Share

Of web development environments

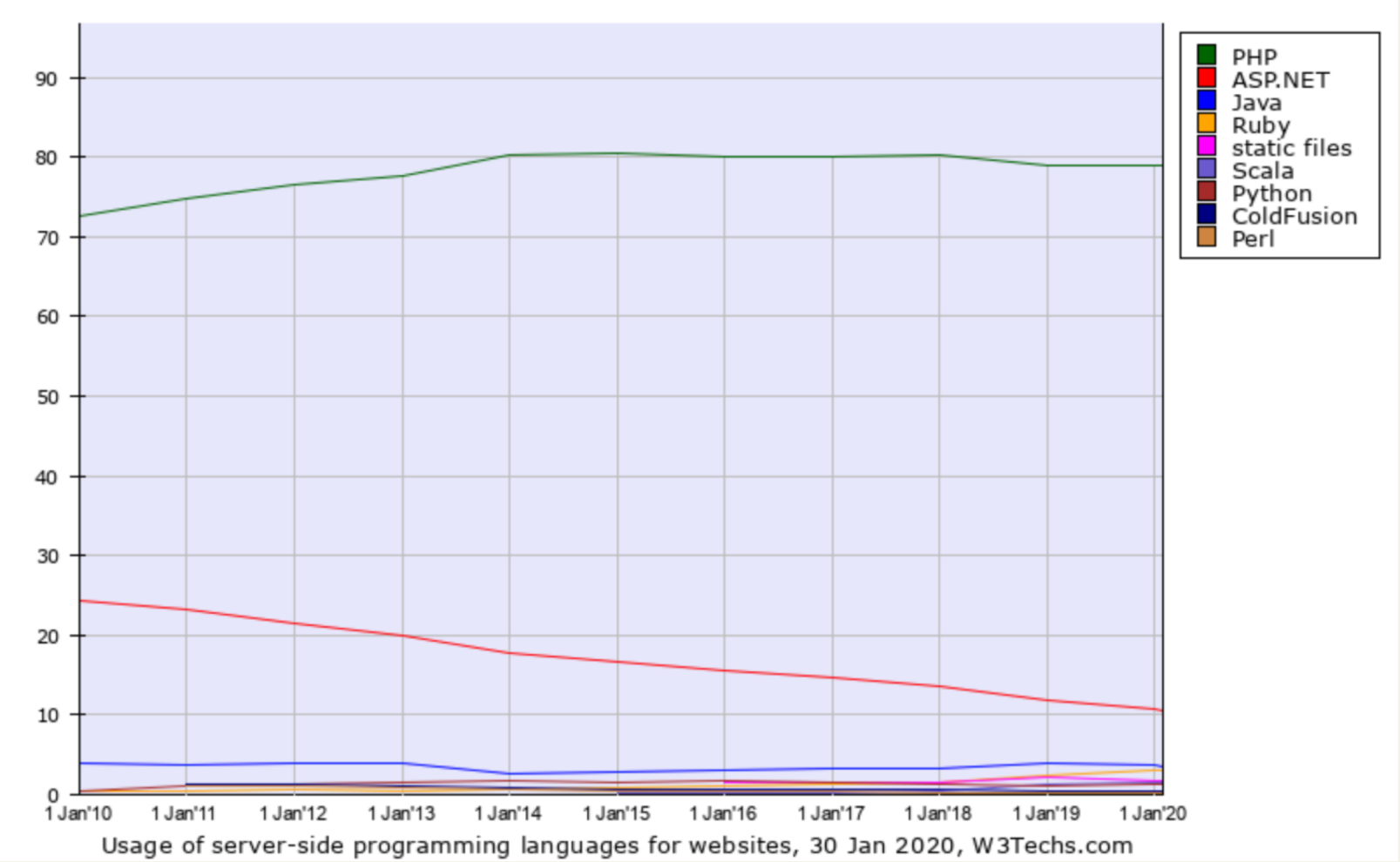
2020



Market Share

Of web development environments

Trend over time (2010-2020) https://w3techs.com/



Section 2 of 6

WEB SERVER'S RESPONSIBILITIES

A Web Server's Responsibilities

A web server has many responsibilities:

- handling HTTP connections
- responding to requests for static and dynamic resources
- managing permissions and access for certain resources
- encrypting and compressing data
- managing multiple domains and URLs
- managing database connections
- managing cookies and state
- uploading and managing files

LAMP stack

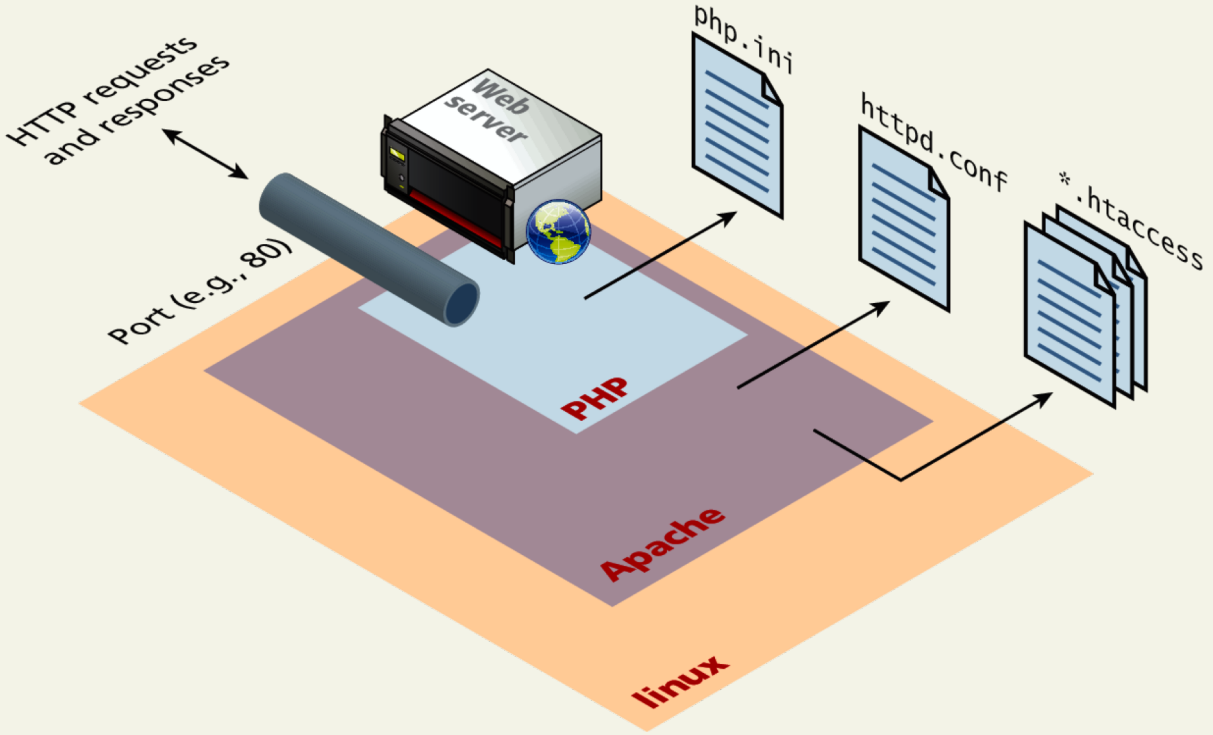
WAMP, MAMP, …

You will be using the LAMP software stack

- **L**inux operating system
- **A**pache web server
- **M**ySQL DBMS
- **P**HP scripting language

Apache and Linux

Consider the **Apache** web server as the intermediary that interprets HTTP requests that arrive through a network port and decides how to handle the request, which often requires working in conjunction with PHP.



PHP Internals

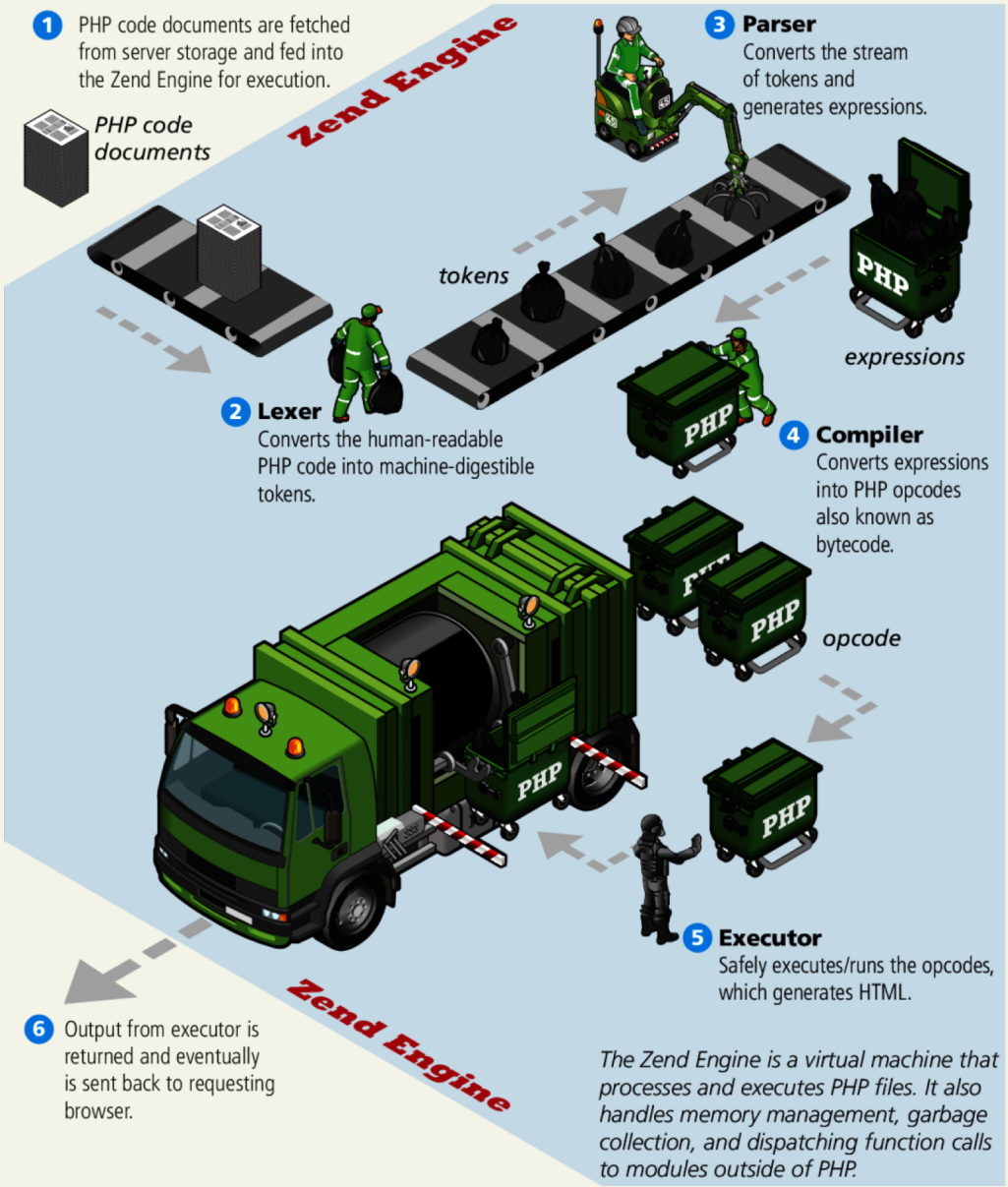
PHP itself is written in C

There are 3 main modules

1. **PHP core**. The Core module defines the main features of the PHP environment, including essential functions for variable handling, arrays, strings, classes, math, and other core features.
2. **Extension layer**. This module defines functions for interacting with services outside of PHP. This includes libraries for MySQL, FTP, SOAP web services, and XML processing, among others.
3. **Zend Engine**. This module handles the reading in of a requested PHP file, compiling it, and executing it.

Zend Engine

No, your code is not garbage.



Installing LAMP locally

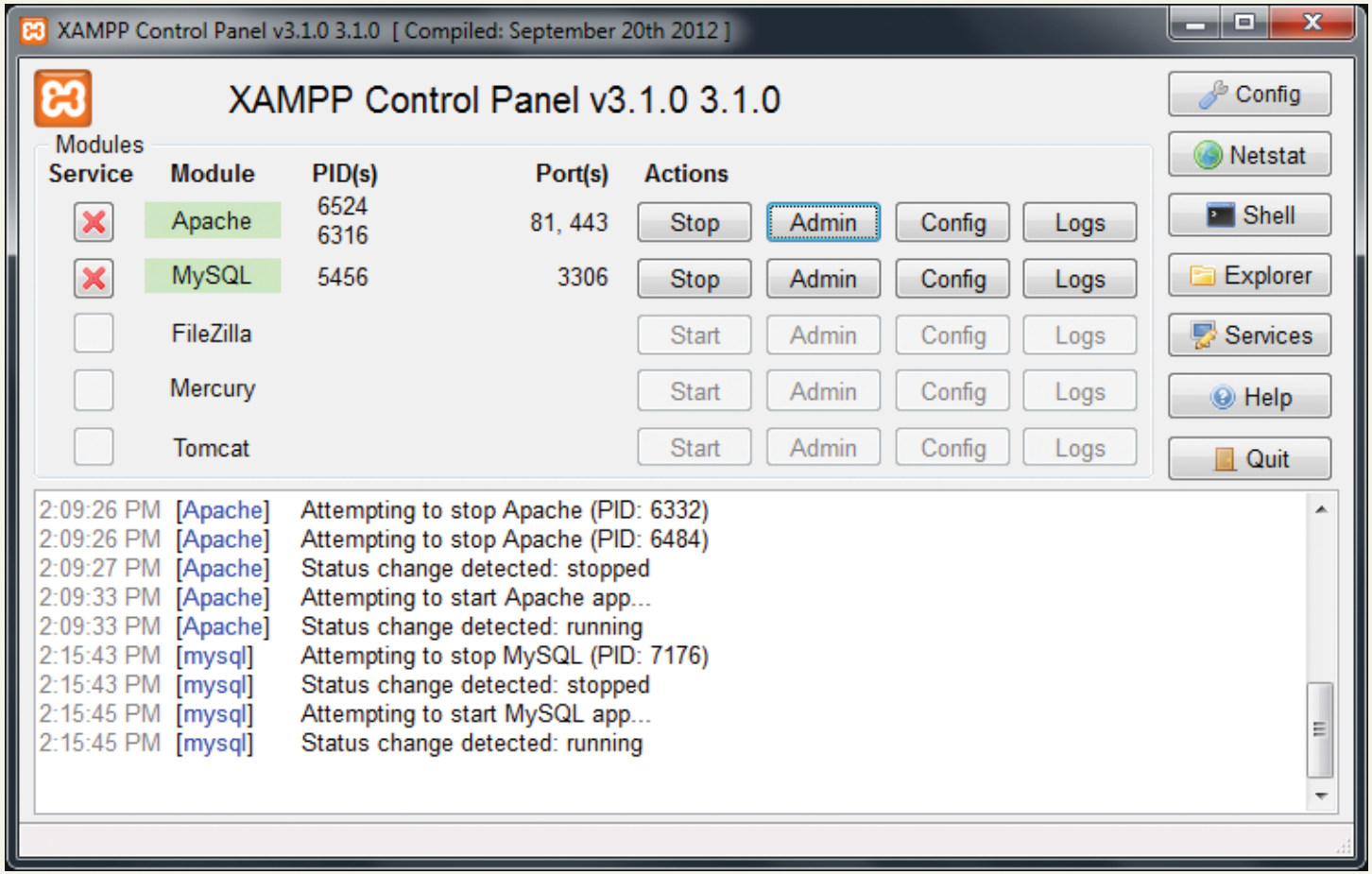
Turn your laptop into a server..

The easiest and quickest way to do so is to use the

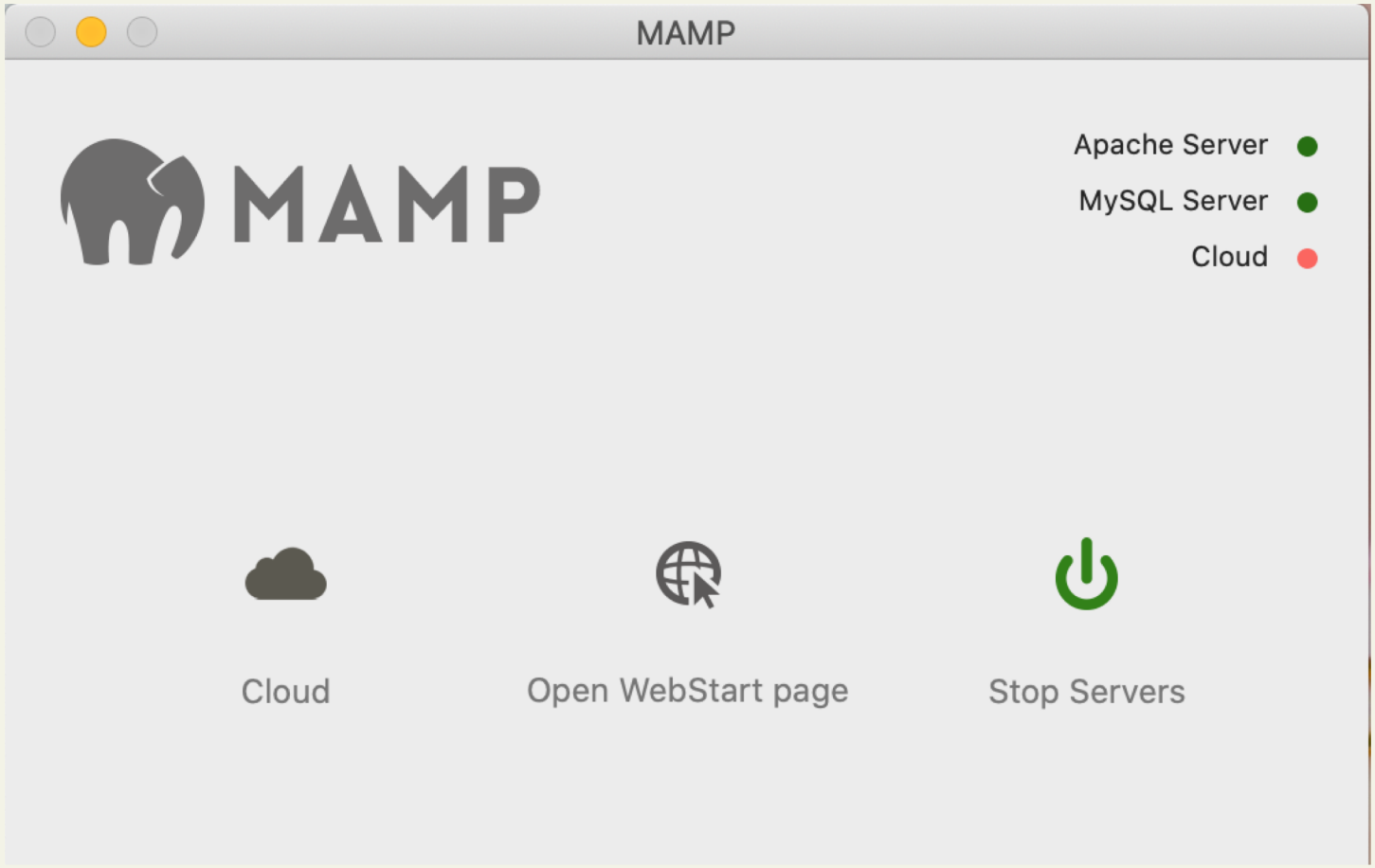
- **XAMPP** For Windows installation package
- **MAMP** for Mac installation package

Both of these installation packages install and configure Apache, PHP, and MySQL.

XAMPP Control Panel

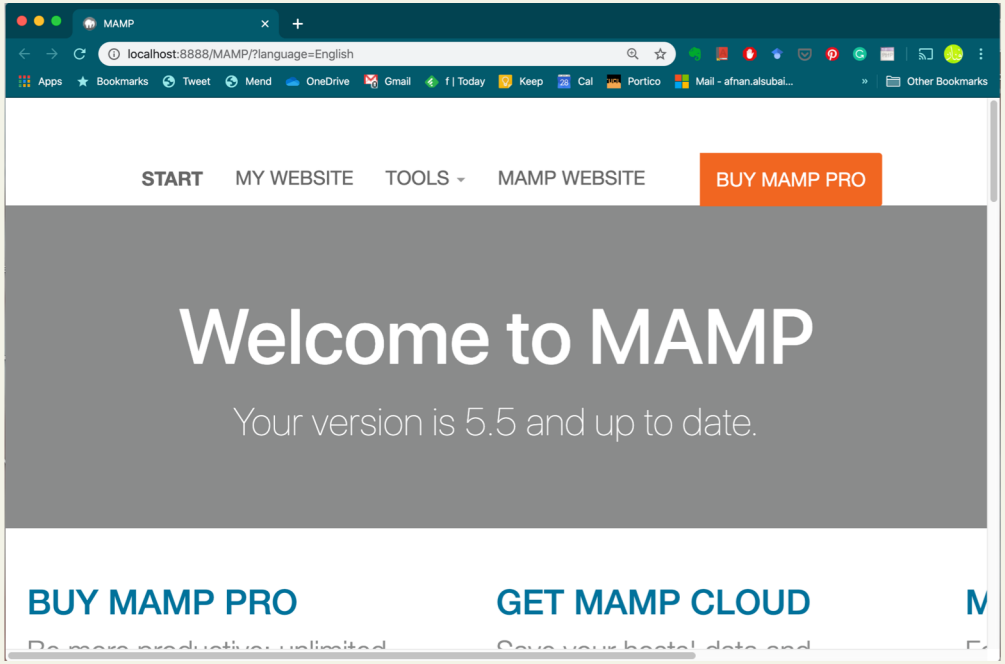


MAMP Control Panel



XAMPP/MAMP Settings

- PHP requests in your browser will need to use the **localhost** domain (127.0.0.1)
- PHP files will have to be saved somewhere within:
  - Windows: **`C:\xampp\htdocs`**
  - Mac **`/Applications/MAMP/htdocs`**



Section 3 of 6

QUICK TOUR OF PHP

Quick Tour

- PHP, like JavaScript, is a dynamically typed language.
- it uses classes and functions in a way consistent with other object-oriented languages such as C++, C#, and Java
- The syntax for loops, conditionals, and assignment is identical to JavaScript
- Differs when you get to functions, classes, and in how you define variables

PHP Tags

The most important fact about PHP is that the programming code can be embedded directly within an HTML file.

- A PHP file will usually have the extension **.php**
- programming code must be contained within an opening **<?php** tag and a matching closing **?>** tag
- any code outside the tags is echoed directly out to the client

PHP Tags

```
<?php
$user = "Randy";
?>
<!DOCTYPE html>
<html>
<body>
<h1>Welcome <?php echo $user; ?></h1>
<p>
The server time is
<?php
echo "<strong>";
echo date("H:i:s");
echo "</strong>";
?>
</p>
</body>
</html>
```

LISTING 8.1 PHP tags

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome Randy</h1>
<p>
The server time is <strong>02:59:09</strong>
</p>
</body>
</html>
```

LISTING 8.2 Listing 8.1 in the browser

HTML and PHP

Two approaches

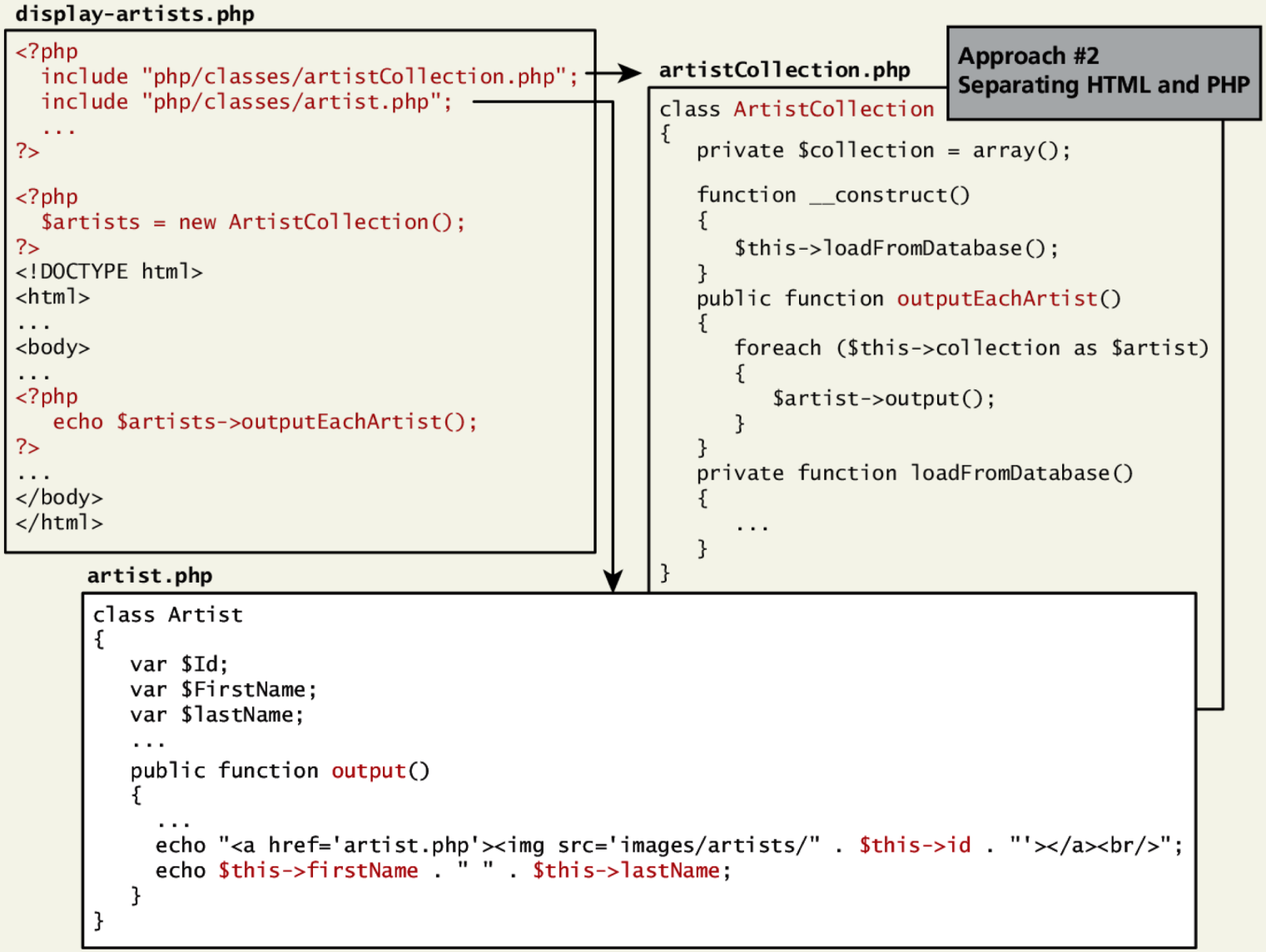
display-artists.php

Approach #1
Mixing HTML and PHP

```
<?php
  $db = new mysqli('localhost', 'dbuser', 'dbpassword', 'dbname');
  $sql = "SELECT * FROM Artists ORDER BY lastName";
  $result = $db->query($sql);
?>
...
<body>
…
<ul>
<?php
while( $row = $result->fetch_assoc() ) {
  echo "<li>";
?>
<img src="images/add.png" /> <img src="images/remove.png" />
<?php
  echo "<a href='artist.php'><img src='images/artists/" . $row['id'] . "'></a><br/>";
  echo $row['firstName'] . " " . $row['lastName'];
  echo "</li>";
}
?>
</ul>
…
<?php
$result->close();
$db->close ();
?>
</body>
</html>
```

HTML and PHP

Two approaches



PHP Comments

3 kinds

The types of comment styles in PHP are:

- **Single-line comments**. Lines that begin with a # are comment lines and will not be executed.
- **Multiline (block) comments**. These comments begin with a /* and encompass everything that is encountered until a closing */ tag is found.
- **End-of-line comments**. Whenever // is encountered in code, everything up to the end of the line is considered a comment.

PHP Comments

3 kinds

```
<?php

# single-line comment

/*

This is a multiline comment.

They are a good way to document functions or
complicated blocks of code

*/

$artist = readDatabase(); // end-of-line comment

?>
```

Variables

Variables in PHP are **dynamically typed.**

Variables are also **loosely typed** in that a variable can be assigned different data types over time

To declare a variable you must preface the variable name with the dollar ($) symbol.

$count = 42;

Data Types

Data Type	Description
Boolean	A logical true or false value
Integer	Whole numbers
Float	Decimal numbers
String	Letters
Array	A collection of data of any type (covered in the next chapter)
Object	Instances of classes

Constants

A **constant** is somewhat similar to a variable, except a constant's value never changes . . . in other words it stays constant.

- Typically defined near the top of a PHP file via the **define()** function
- once it is defined, it can be referenced without using the $ symbol

Constants

```php
<?php

# Uppercase for constants is a programming convention
define("DATABASE_LOCAL", "localhost");
define("DATABASE_NAME", "ArtStore");
define("DATABASE_USER", "Fred");
define("DATABASE_PASSWD", "F5^7%ad");
...
# notice that no $ prefaces constant names
$db = new mysqli(DATABASE_LOCAL, DATABASE_NAME, DATABASE_USER,
  DATABASE_NAME);

?>
```

LISTING 8.4 PHP constants

Writing to Output

Hello World

To output something that will be seen by the browser, you can use the echo() function.

```
echo ("hello"); //long form

echo "hello";  //shortcut
```

String Concatenation

Strings can easily be appended together using the concatenate operator, which is the period (.) symbol.

```
$username = "World";

echo "Hello". $username;
```

Will Output **Hello World**

String Concatenation

Example

```
$firstName = "Pablo";

$lastName = "Picasso";

/*

Example one:

These two lines are equivalent. Notice that you can reference
PHP variables within a string literal defined with double quotes.
The resulting output for both lines is: <em>Pablo Picasso</em>

*/

echo "<em>" . $firstName . " ". $lastName. "</em>";

echo "<em> $firstName $lastName </em>";

// Both output into html: <em> Pablo Picasso </em>
```

String Concatenation

Example

```
/*

Example two:

These two lines are also equivalent. Notice that you can use either the single
quote symbol or double quote symbol for string literals.

*/

echo "<h1>";

echo '<h1>';

/*

However, variables inside a  ' '  string are not interpolated (they are not
replaced with their value):

*/

echo "<em> $firstName $lastName </em>";

// output into html file: <em> Pablo Picasso </em>

echo ' <em> $firstName $lastName </em> ' ;

// output into html file: <em> $firstName $lastName </em>
```

String Concatenation

Example

```
/*

Example three:

These two lines are also equivalent. In the second
example, the escape character (the backslash) is used
to embed a double quote within a string literal defined
within double quotes.

*/

echo '<img src="23.jpg" >';

echo "<img src=\"23.jpg\" >";
```

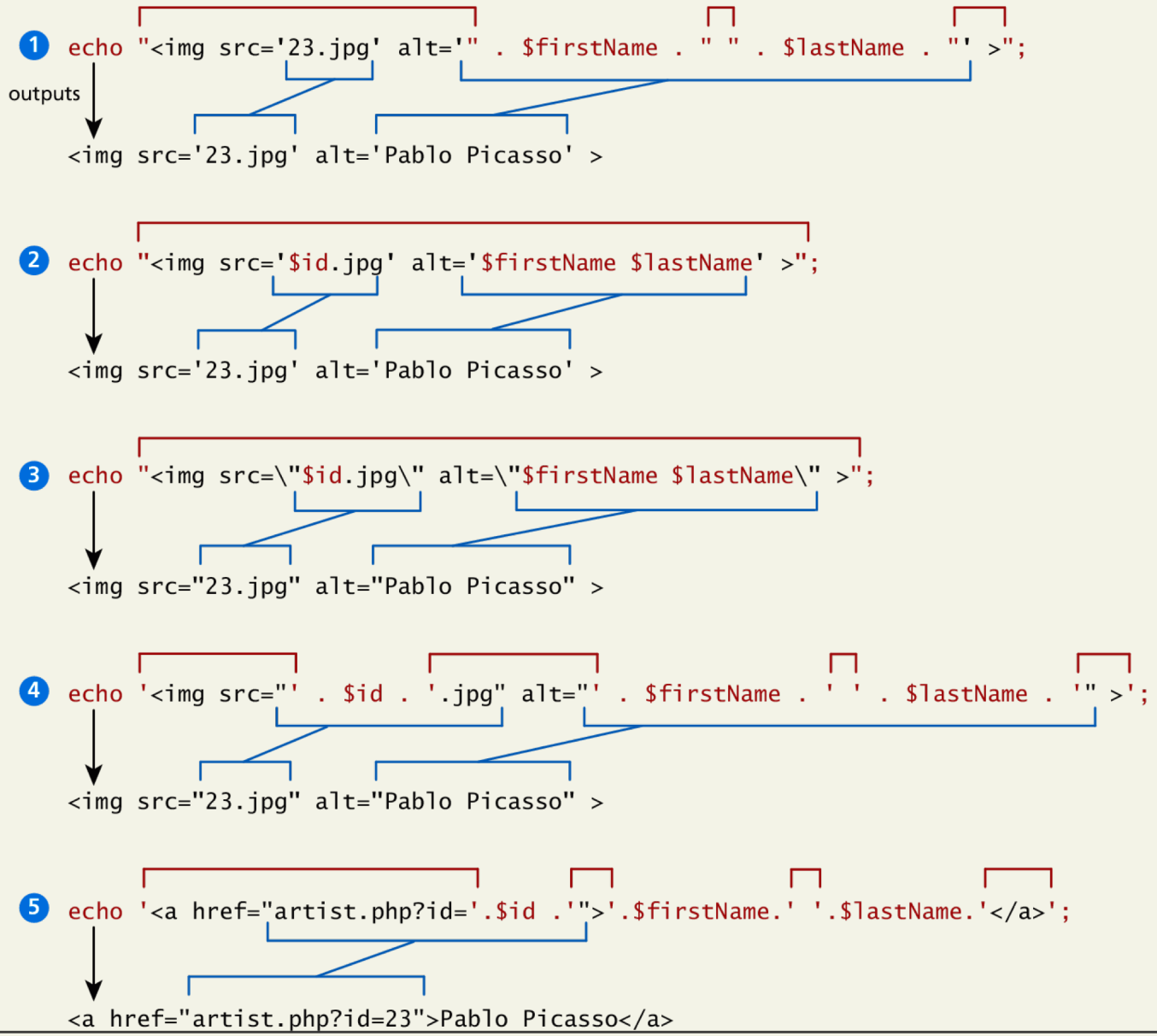
String escape Sequences

Sequence	Description
\n	Line feed
\t	Horizontal tab
\\	Backslash
\$	Dollar sign
\"	Double quote

Complicated Concatenation

```
echo "<img src='23.jpg' alt='". $firstName . " ". $lastName . "' >";

echo "<img src='$id.jpg' alt='$firstName $lastName' >";

echo "<img src=\"$id.jpg\" alt=\"$firstName $lastName\" >";

echo '<img src="' . $id. '.jpg" alt="' . $firstName . ' ' . $lastName . '" >';

echo '<a href="artist.php?id=' .$id .'">' .$firstName .' ' . $lastName .'</a>';
```

Illustrated Example



PrintF

Good ol' printf

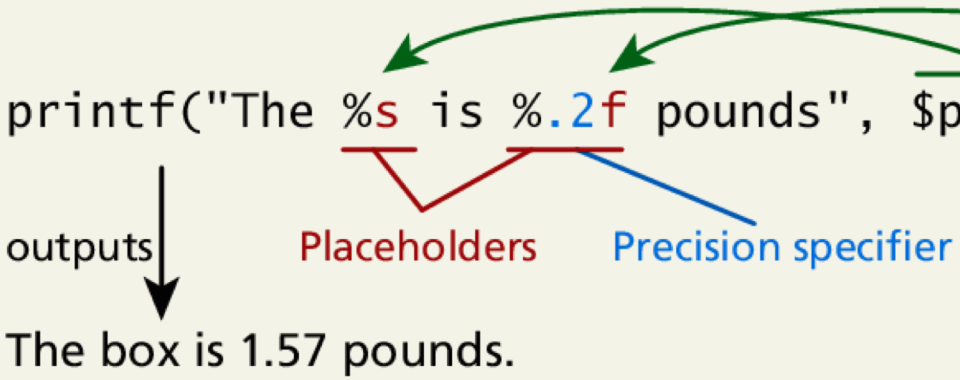
As an alternative, you can use the **printf()** function.

- derived from the same-named function in the C programming language
- includes variations to print to string and files (sprintf, fprintf)
- takes at least one parameter, which is a string, and that string optionally references parameters, which are then integrated into the first string by placeholder substitution
- Can also apply special formatting, for instance, specific date/time formats or number of decimal places

PrintF

Illustrated example

```
$product = "box";
$weight = 1.56789;

printf("The %s is %.2f pounds", $product, $weight);
```



The box is 1.57 pounds.

PrintF

Type specifiers

Each placeholder requires the percent (%) symbol in the first parameter string followed by a type specifier.

- b for binary
- d for signed integer
- f for float
- o for octal
- x for hexadecimal

PrintF

Precision

Precision allows for control over how many decimal places are shown. Important for displaying calculated numbers to the user in a “pretty” way.

Precision is achieved in the string with a period (.) followed by a number specifying how many digits should be displayed for floating-point numbers.

Section 4 of 6

PROGRAM CONTROL

If…else

The syntax for conditionals in PHP is almost identical to that of JavaScript

```
// if statement with condition
if ( $hourOfDay > 6 && $hourOfDay < 12 ) {
   $greeting = "Good Morning";
}
else if ($hourOfDay == 12) {   // optional else if
   $greeting = "Good Noon Time";
}
else {                         // optional else branch
   $greeting = "Good Afternoon or Evening";
}
```

LISTING 8.7 Conditional statement using if . . . else

If...else

Alternate syntax

```
<?php if ($userStatus == "loggedin") {  ?>
   <a href="account.php">Account</a>
   <a href="logout.php">Logout</a>
<?php }  else { ?>
   <a href="login.php">Login</a>
   <a href="register.php">Register</a>
<?php } ?>


<?php
   // equivalent to the above conditional
   if ($userStatus == "loggedin") {
      echo '<a href="account.php">Account</a> ';
      echo '<a href="logout.php">Logout</a>';
   }
   else {
      echo '<a href="login.php">Login</a> ';
      echo '<a href="register.php">Register</a>';
   }
?>
```

LISTING 8.8 Combining PHP and HTML in the same script

Switch…case

Nearly identical

```
switch ($artType) {
   case "PT":
   $output = "Painting";
     break;
     case "SC":
   $output = "Sculpture";
     break;
     default:
   $output = "Other";
}

// equivalent
if ($artType == "PT")
   $output = "Painting";
else if ($artType == "SC")
   $output = "Sculpture";
else
   $output = "Other";
```

LISTING 8.9 Conditional statement using switch

While and Do..while

Identical to other languages

```php
$count = 0;
while ($count < 10)
{
    echo $count;
    $count++;
}

$count = 0;
do
{
    echo $count;
    $count++;
} while ($count < 10);
```

LISTING 8.10 while loops

For

Identical to other languages

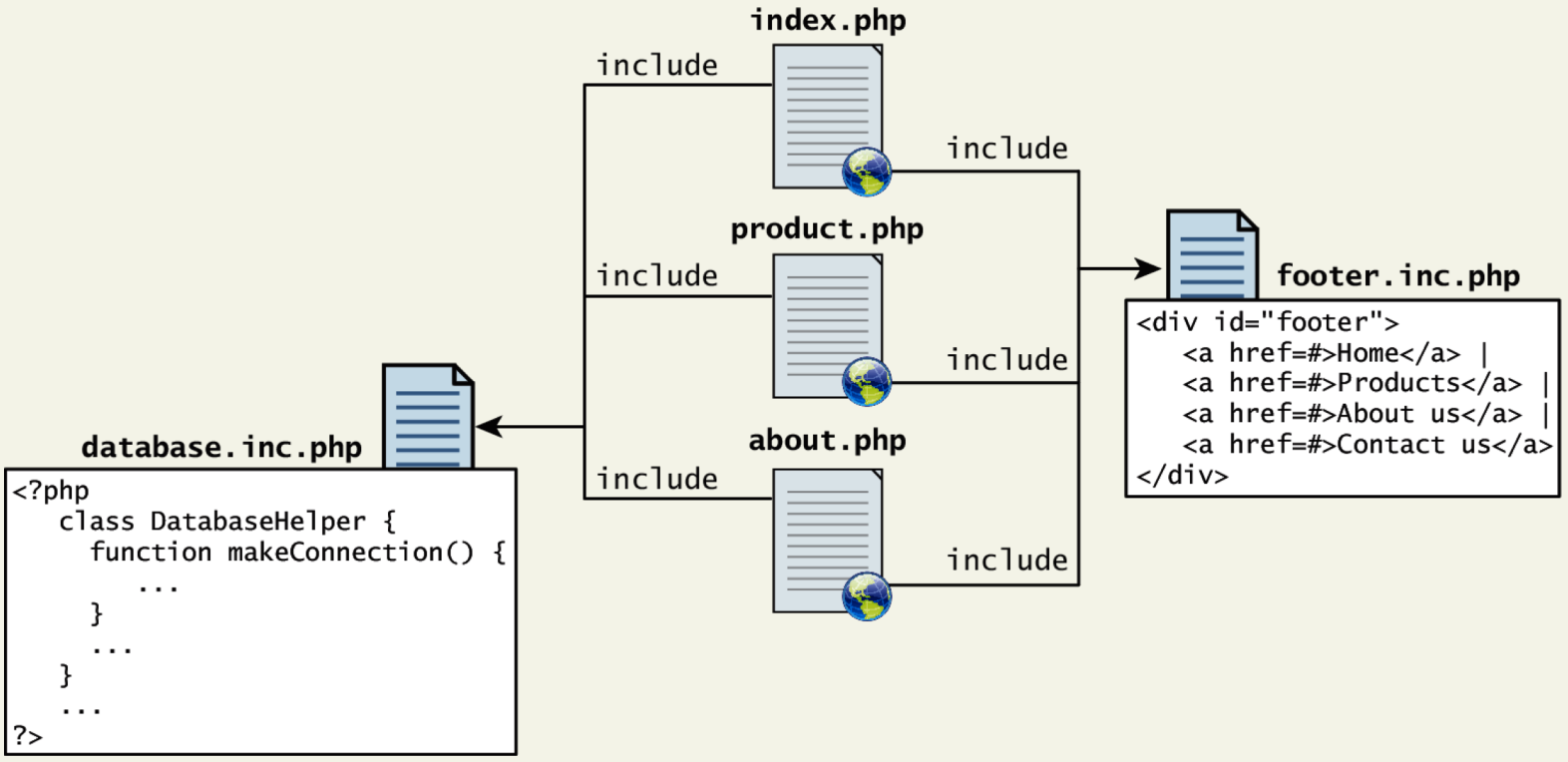
```
for ($count=0; $count < 10; $count++)
{
   echo $count;
}
```

LISTING 8.11 for loops

Include Files

Organize your code

PHP does have one important facility that is generally unlike other non-web programming languages, namely the ability to **insert** content from one file into another.



Include Files

Organize your code

PHP provides four different statements for including files, as shown below.

include "somefile.php";

include_once "somefile.php";

require "somefile.php";

require_once "somefile.php";

What happens when the file cannot be included (ie. Not found, or permission denied):

- With include, a warning is displayed and then execution continues.
- With require, an error is displayed and execution stops.

Section 5 of 6

FUNCTIONS

Functions

You mean we don't write everything in main?

Just as with any language, it's a good practice to separate your code into functions (depending on their logic)

There are two types of Functions in PHP:

1. **User-defined functions:** are ones that you the programmer define.
2. **Built-in functions:** are functions that come with the PHP environment

Functions

When defining functions in PHP, you don't specify a type for the function (type of returned value), nor the types of the parameters.

```php
/**
* This function returns a nicely formatted string using the current
* system time.
*/
function getNiceTime() {
   return date("H:i:s");
}
```

LISTING 8.13 The definition of a function to return the current time as a string

A function that returns a value

```php
/**
* This function outputs the footer menu
*/
function outputFooterMenu() {
   echo '<div id="footer">';
   echo '<a href=#>Home</a> | <a href=#>Products</a> | ';
   echo '<a href=#>About us</a> | <a href=#>Contact us</a>';
   echo '</div>';
}
```

LISTING 8.14 The definition of a function without a return value

A function that returns void

Parameters

When defining functions in PHP, you don't specify a type for the function (type of returned value), nor the types of the parameters.

```php
/**
* This function returns a nicely formatted string using the current
* system time. The showSeconds parameter controls whether or not to
* include the seconds in the returned string.
*/
function getNiceTime($showSeconds) {
  if ($showSeconds==true)
    return date("H:i:s");
  else
    return date("H:i");
}
```

LISTING 8.15 A function to return the current time as a string with an integer parameter

Thus to call our function, you can now do it in two ways:

```
echo getNiceTime(1);  // this will print seconds
echo getNiceTime(0);  // will not print seconds
```

Parameter Default Values

```
/**
* This function returns a nicely formatted string using the current
* system time. The showSeconds parameter controls whether or not
* to show the seconds.
*/
function getNiceTime($showSeconds=1){
   if ($showSeconds==true)
      return date("H:i:s");
   else
      return date("H:i");
}
```

LISTING 8.16 A function to return the current time with a parameter that includes a default

Now if you were to call the function with no values, the $showSeconds parameter would take on the default value, which we have set to 1, and return the string with seconds.

Pass Parameters by Reference

By default, arguments passed to functions are **passed by value** in PHP.

PHP also allows arguments to functions to be **passed by reference**, which will allow a function to change the contents of a passed variable.

The mechanism in PHP to specify that a parameter is passed by reference is to add an ampersand (&) symbol next to the parameter name in the function declaration

```
function changeParameter(&$arg) {
   $arg += 300;
   echo "<br/>arg=". $arg;
}

$initial = 15;
echo "<br/>initial=" . $initial;   // output: initial=15
changeParameter($initial);         // output: arg=315
echo "<br/>initial=" . $initial;   // output: initial=315
```

LISTING 8.18 Passing a parameter by reference

Variable Scope in functions

All variables defined within a function (such as parameter variables) have **function scope**, meaning that they are only accessible within the function.

Any variables created outside of the function in the main script are unavailable within a function.

```
$count= 56;

function testScope() {
        echo $count;      // outputs 0 or generates run-time
                          //warning/error

}

testScope();
echo $count;              // outputs 56
```

Global variables

Sometimes unavoidable

Variables defined in the main script are said to have **global scope.**

Unlike in other programming languages, a global variable is not, by default, available within functions.

PHP does allow variables with global scope to be accessed within a function using the **global** keyword

```
$count= 56;

function testScope() {
   global $count;
   echo $count;    // outputs 56
}

testScope();
echo $count;       // outputs 56
```

LISTING 8.19 Using the global keyword

Section 6 of 6

PHP ERROR REPORTING

Types of Errors

- **Expected errors**

  Things that you expect to go wrong. Bad user input, database connection, etc…

- **Warnings**

  Problems that generate a PHP warning message but will not halt the execution of the page

- **Fatal errors**

  Are serious in that the execution of the page will terminate unless handled in some way

Checking user input

Checking for a number

```
$id = $_GET['id'];
if (!empty($id) && is_numeric($id) ) {
   // use the query string since it exists and is a numeric value
   ...
}
```

LISTING 12.1 Testing a query string to see if it exists and is numeric

Exceptions vs Errors

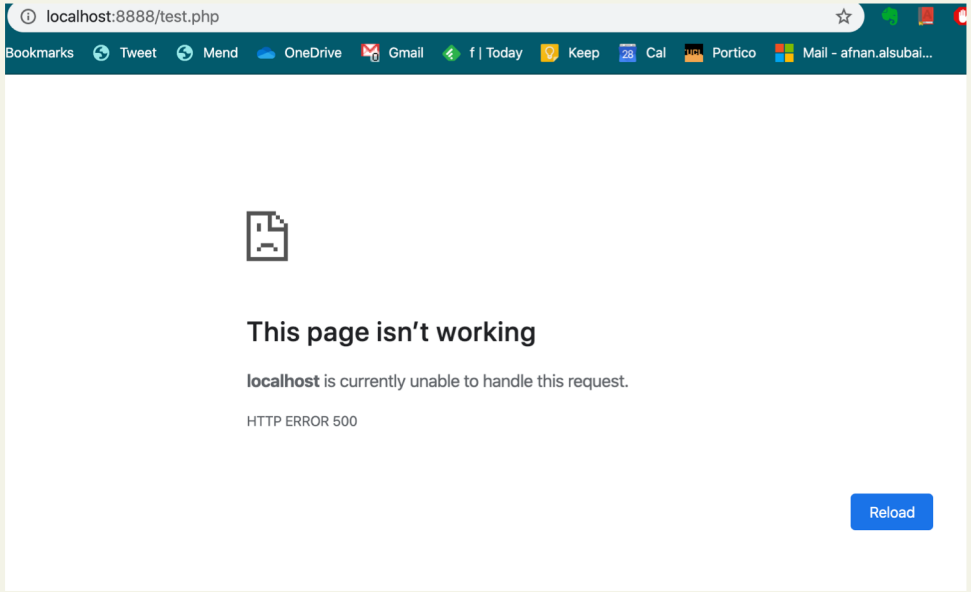
Not the same thing

- An **error** is some type of problem that generates a nonfatal warning message or that generates an error message that terminates the program's execution.
- An **exception** refers to objects that are of type Exception and which are used in conjunction with the object-oriented try . . . catch language construct for dealing with runtime errors.

Error Detection

- As you test your code using a browser, the browser does not always display if something went wrong:

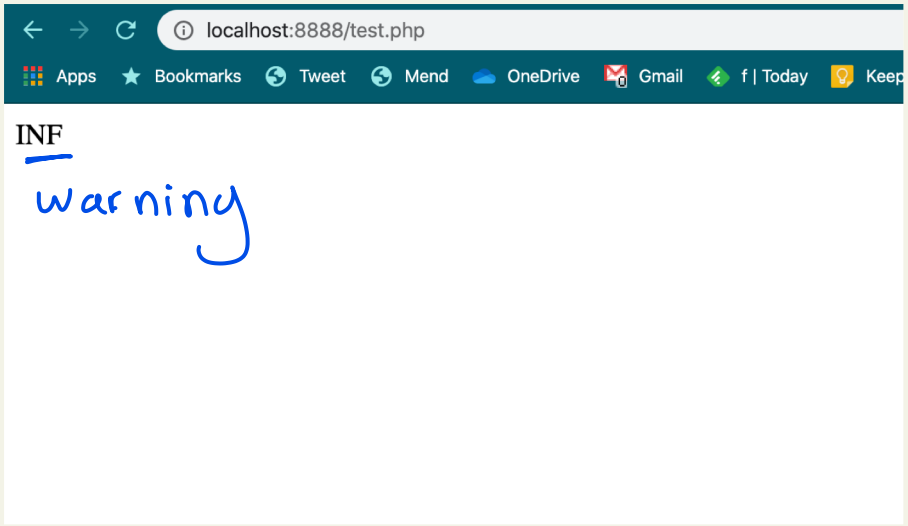
```
<!DOCTYPE html>
<html>
  <body>
<?php
echo "Hello! " // ;
echo "hi" // ;
?>
  </body>
</html>
```



Error Detection

- As you test your code using a browser, the browser does not always display if something went wrong, even if it is non-fatal:

```
<!DOCTYPE html>
<html>
  <body>
<?php
echo 4/0;
?>
  </body>
</html>
```



PHP error reporting

Lots of control

PHP has a flexible and customizable system for reporting warnings and errors that can be set programmatically at runtime or declaratively at design-time within the **php.ini** file. There are three main error reporting flags:

- error_reporting
- display_errors
- log_errors

The error_reporting setting

What is an error?

The **error_reporting** setting specifies which type of errors are to be reported.

It can be set programmatically inside **any** PHP file:

```
error_reporting(E_ALL);
```

It can also be set within the **php.ini** file:

```
error_reporting = E_ALL
```

The error_reporting setting

Some error reporting constants

Constant Name	Value	Description
E_ALL	8191	Report all errors and warnings
E_ERROR	1	Report all fatal runtime errors
E_WARNING	2	Report all nonfatal runtime errors (that is, warnings)
	0	No reporting

The display_errors setting

To show or not to show

The **display_error** setting specifies whether error messages should or should not be displayed in the browser.

It can be set programmatically via the ini_set() function:

```
ini_set('display_errors','0');
```

It can also be set within the **php.ini** file:

```
display_errors = Off
```

The log_error setting

To record or not to record

The **log_error** setting specifies whether error messages should or should not be sent to the server error log.

It can be set programmatically via the ini_set() function:

```
ini_set('log_errors','1');
```

It can also be set within the **php.ini** file:

```
log_errors = On
```

The log_error setting

Where to store.

The location to store logs in can be set programmatically:

```
ini_set('error_log', '/restricted/my-errors.log');
```

It can also be set within the **php.ini** file:

```
error_log = /restricted/my-errors.log
```

The log_error setting

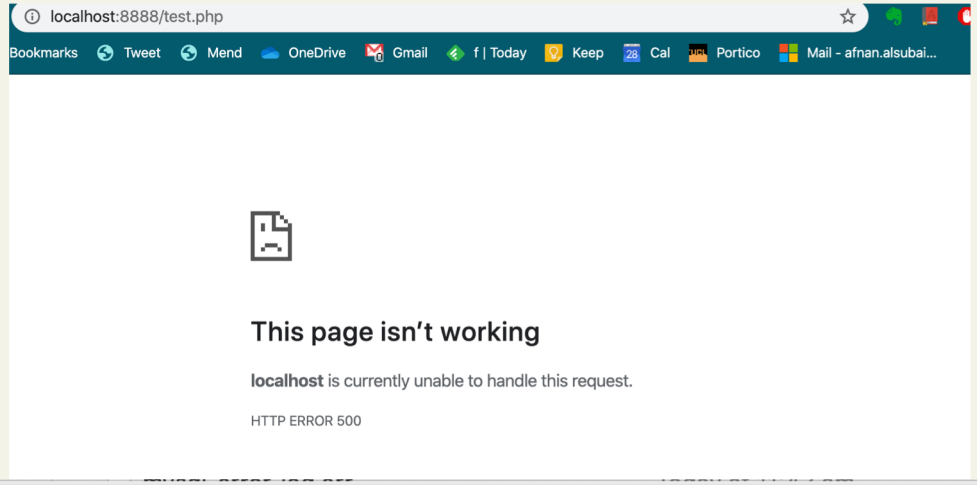
Error_log()

You can also programmatically send messages to the error log at any time via the error_log() function

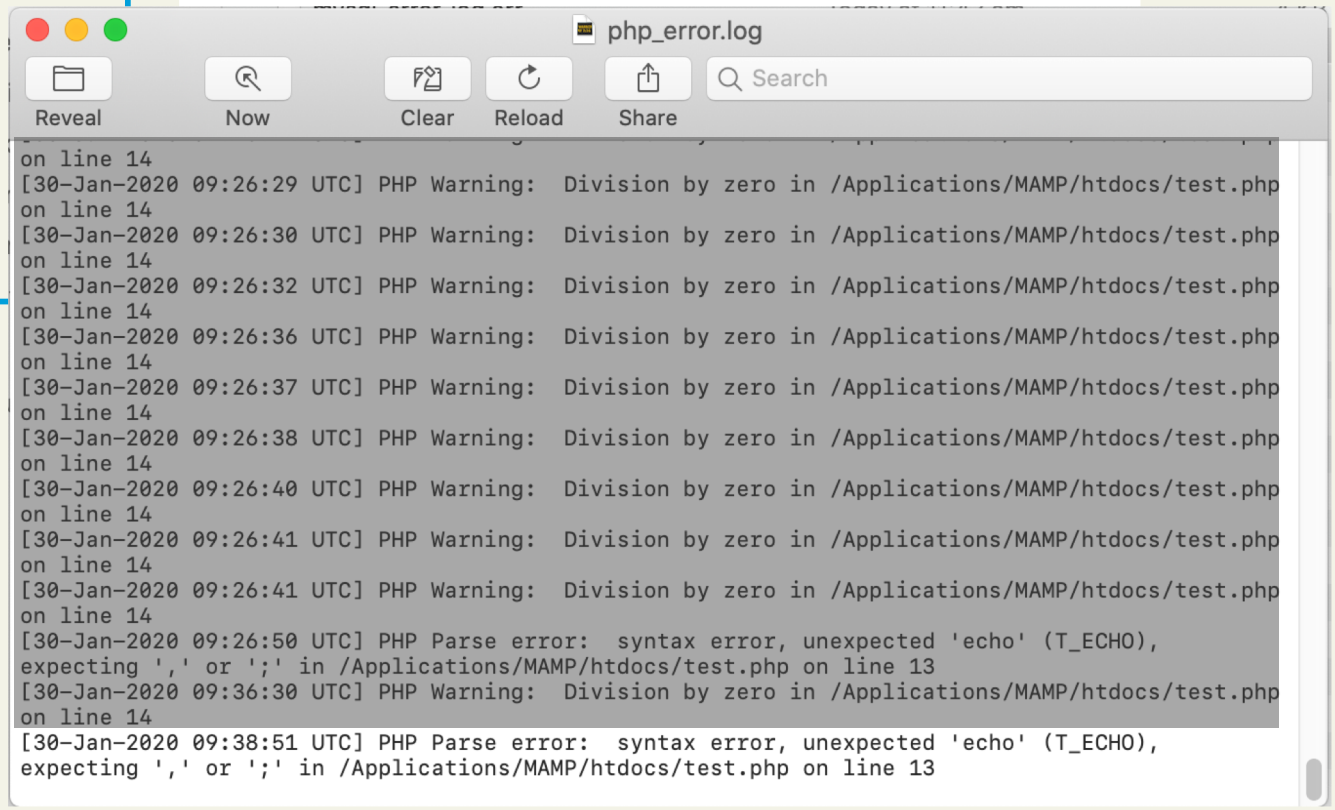
```
$msg = 'Some horrible error has occurred!';

// send message to system error log (default)
error_log($msg,0);

// email message
error_log($msg,1,'support@abc.com','From: somepage.php@abc.com');

// send message to file
error_log($msg,3, '/folder/somefile.log');
```

Examples

```
<?php
error_reporting(E_ALL);
?>
<!DOCTYPE html>
<html>
  <body>
<?php
echo "Hello! " // ;
echo "hi" // ;
?>
  </body>
</html>
```

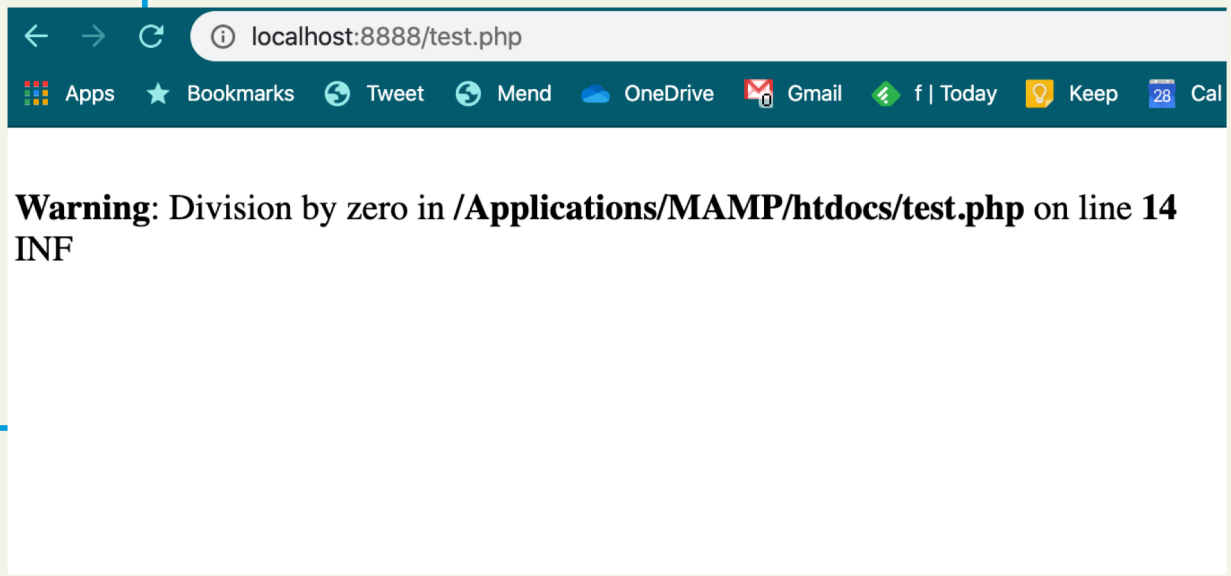


In the log:



Examples

```
<?php
ini_set('display_errors', 1);
?>
<!DOCTYPE html>
<html>
  <body>
<?php
echo 4/0;
?>
  </body>
</html>
```



What You've Learned

1. Server-Side Development
2. Web Server's Responsabilities
3. Quick Tour of PHP
4. Program Control
5. Functions
6. PHP Error Reporting