Web Programming

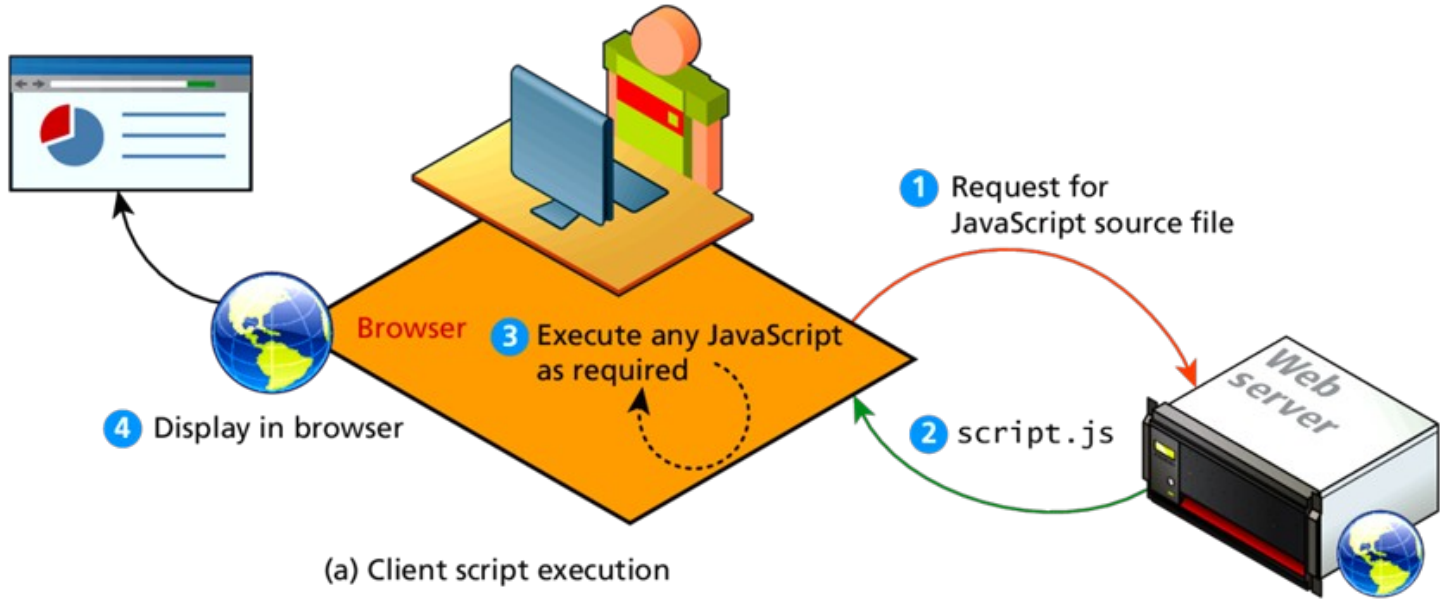
CS-406

Lecture # 07

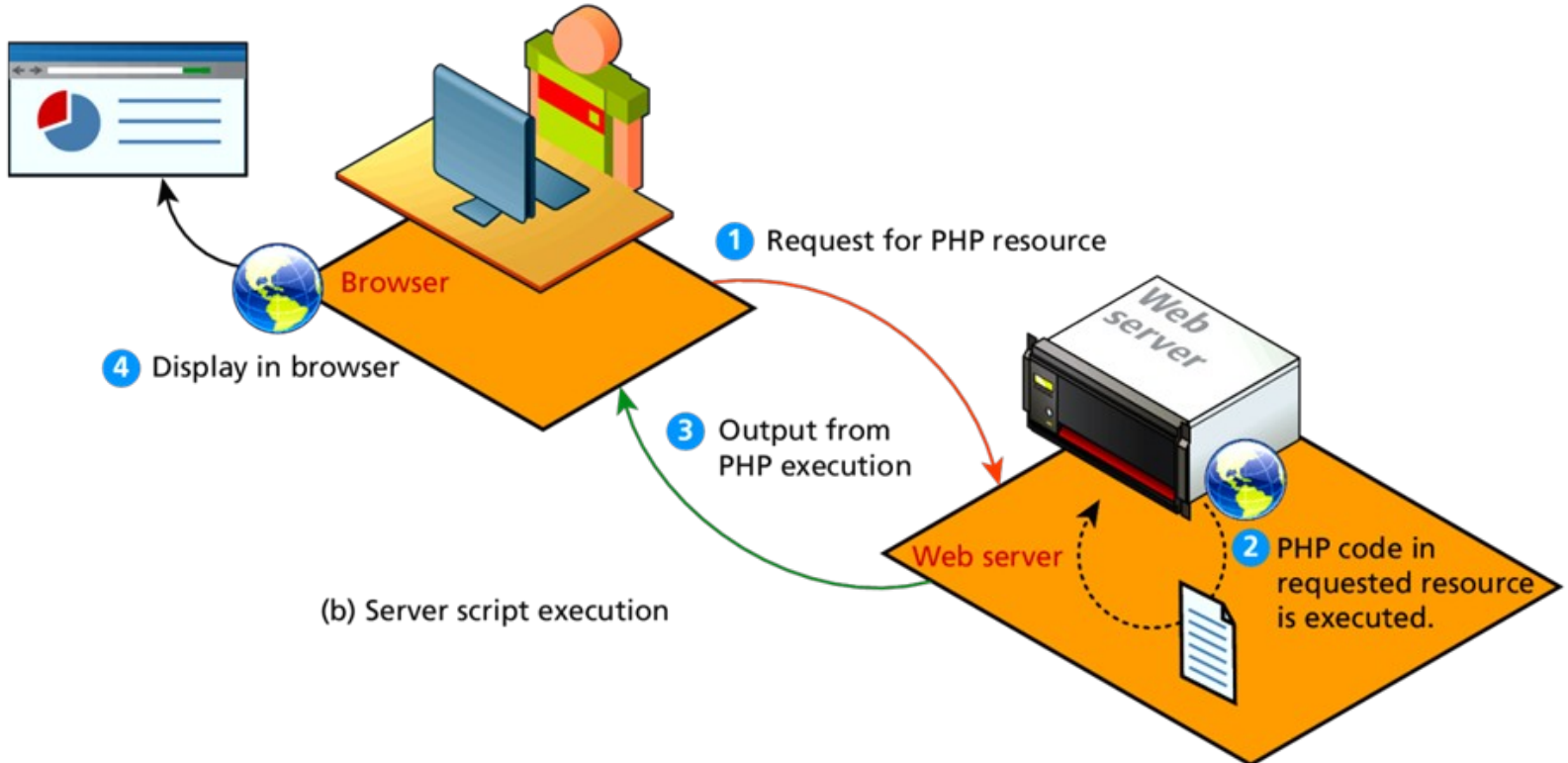
PHP

Server-side Scripting

- The basic hosting of web 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

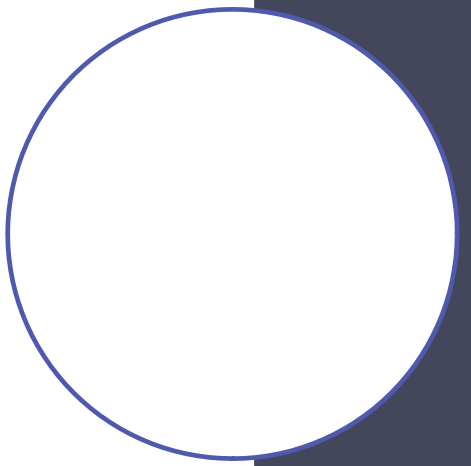


(a) Client script execution



(b) Server script execution

- **Server-side scripting** is a method of designing websites so that the process or user request is run on the originating server.
- Server-side scripts provide an interface to the user.
- Used to limit access to proprietary data.
- It also helps keep control of the script source code.



Client Side vs. Server Side

When a client (your computer) makes a request for a web page that information is processed by the web server. If the request is a server side script (e.g. Perl or PHP) before the information is returned to the client the script is executed on the server and the results of the script is returned to the client.

Once the client recieves the returned information from the server if it contains a client side script (e.g. JavaScript) your computer browser executes that script before displaying the web page.

- When you type in a URL, lots of code is at work to bring a page to your screen.
- What connects your site’s database to the browser, creating a smooth, user-friendly experience?
- That’s the software built by **server-side scripts,** languages that build your site behind the scenes.
- The goal of this software is to provide a seamless experience for the user that’s as close to a desktop application as possible.

A Web Server’s Responsibilities

A web server has many responsibilities:

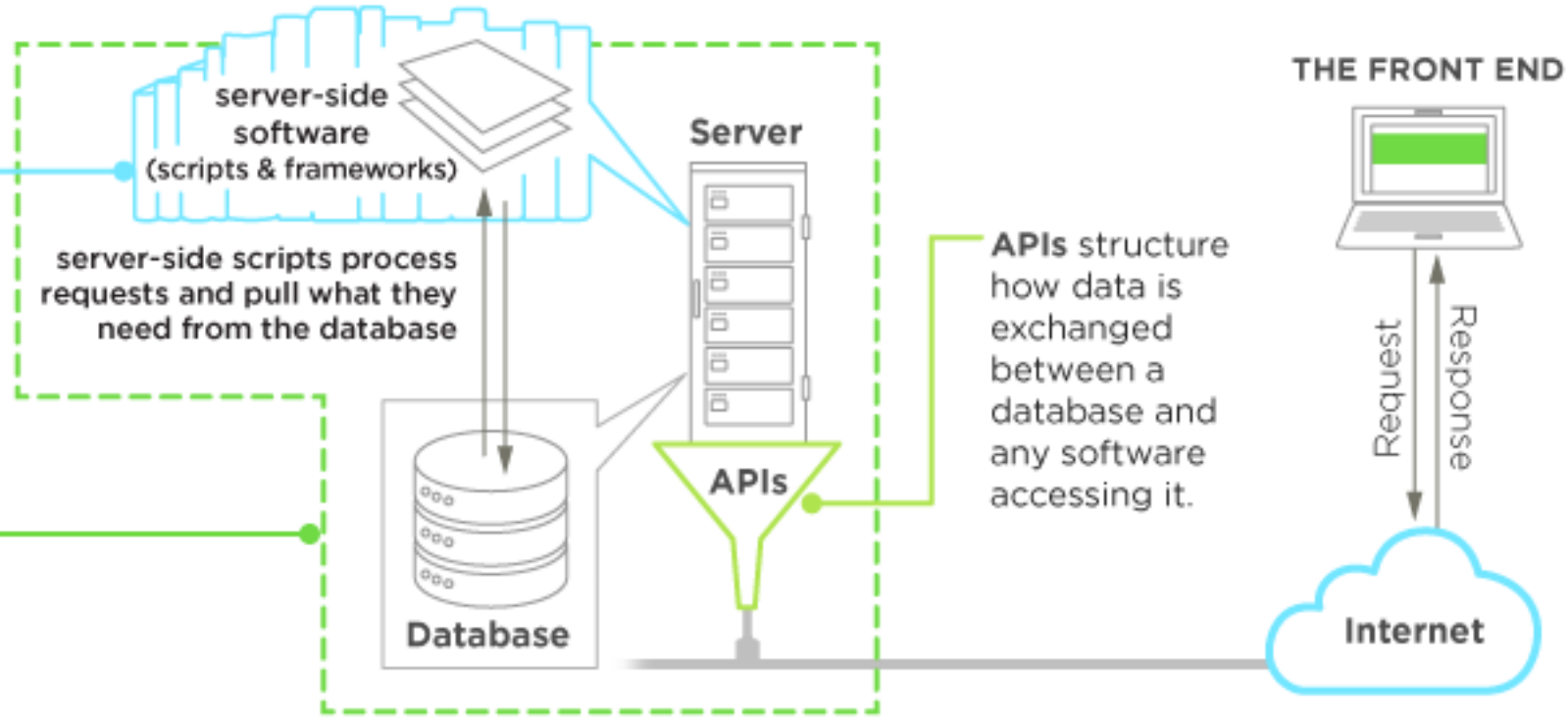
- handling HTTP connections
- responding to requests for static and dynamic resources
- managing permissions and access for certain resources
- encrypting and compressing data
- managing multiple domains and URLs
- managing database connections
- managing cookies and state
- uploading and managing files

BACK-END DEVELOPMENT & FRAMEWORKS IN SERVER SIDE SOFTWARE

Upwork™

FRAMEWORKS are libraries of server-side programming languages that construct the back-end structure of a site.

The **"STACK"** comprises the database, server-side framework, server, and operating system (OS).



Server side script Basics

- **Runs on a server**, embedded in the site's code
- Designed to **interact with back-end permanent storage**, like databases, and process information from the server to access the database—like a direct line from user to database
- Facilitates the **transfer of data** from server to browser, bringing pages to life in the browser, e.g., processing and then delivering a field that a user requests or submits in a form
- **Runs on-call**. When a webpage is "called up," or when parts of pages are "posted back" to the server with AJAX, server-side scripts process and return data
- Powers functions in dynamic web applications, such as user validation, saving and retrieving data, and navigating between other pages
- Plays a big role in how a database is built from the ground up and managed afterwards—an example of how roles often overlap in all aspects of development
- Build application programming interfaces (APIs), which control what data and software a site shares with other apps

Server-side Languages

- PHP
- Python
- Ruby
- Perl
- C++ & C#
- Java
- Erlang

Server-side Frameworks

- Ruby on rails
- Django
- ASP.NET
- Laravel
- Node.js
- Express.js and Koa

PHP

- The **PHP Hypertext Preprocessor (PHP)** is a programming language that allows web developers to create dynamic content that interacts with databases.
- PHP is basically used for developing web based software applications.
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.

- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Applications of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

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

Hello world

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php echo "Hello, World!";  ?>
  </body>
</html>
```

PHP Tags

```php
<?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

```html
<!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

Approach #1
Mixing HTML and PHP

isplay-artists.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>
```

Approach #2
Separating HTML and PHP

display-artists.php

```
<?php
  include "php/classes/artistCollection.php";
  include "php/classes/artist.php";
  ...
?>

<?php
  $artists = new ArtistCollection();
?>
<!DOCTYPE html>
<html>
...
<body>
...
<?php
   echo $artists->outputEachArtist();
?>
...
</body>
</html>
```

artistCollection.php

```
class ArtistCollection
{
    private $collection = array();

    function __construct()
    {
        $this->loadFromDatabase();
    }
    public function outputEachArtist()
    {
        foreach ($this->collection as $artist)
        {
            $artist->output();
        }
    }
    private function loadFromDatabase()
    {
        ...
    }
}
```

artist.php

```
class Artist
{
    var $Id;
    var $FirstName;
    var $lastName;
    ...
    public function output()
    {
       ...
       echo "<a href='artist.php'><img src='images/artists/" . $this->id . "'></a><br/>";
       echo $this->firstName . " " . $this->lastName;
    }
}
```

HTML and PHP

Two approaches

splay-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>
```

display-artists.php

```
<?php
  include "php/classes/artistCollection.php";
  include "php/classes/artist.php";
  ...
?>

<?php
  $artists = new ArtistCollection();
?>
<!DOCTYPE html>
<html>
...
<body>
...
<?php
   echo $artists->outputEachArtist();
?>
...
</body>
</html>
```

artistCollection.php

Approach #2
Separating HTML and PHP

```
class ArtistCollection
{
    private $collection = array();

    function __construct()
    {
        $this->loadFromDatabase();
    }
    public function outputEachArtist()
    {
        foreach ($this->collection as $artist)
        {
            $artist->output();
        }
    }
    private function loadFromDatabase()
    {
        ...
    }
}
```

artist.php

```
class Artist
{
    var $Id;
    var $FirstName;
    var $lastName;
    ...
    public function output()
    {
      ...
      echo "<a href='artist.php'><img src='images/artists/" . $this->id . "'></a><br/>";
      echo $this->firstName . " " . $this->lastName;
    }
}
```

PHP Comments

- 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

?>
```

PHP 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

# 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

Easy

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

```
$username = ”World";

echo "Hello". $username;

$a=1;

$b=2;

echo $a+$b;
```

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 "<b>" . $firstName . " ". $lastName. "</b>";

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

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\" >";
```

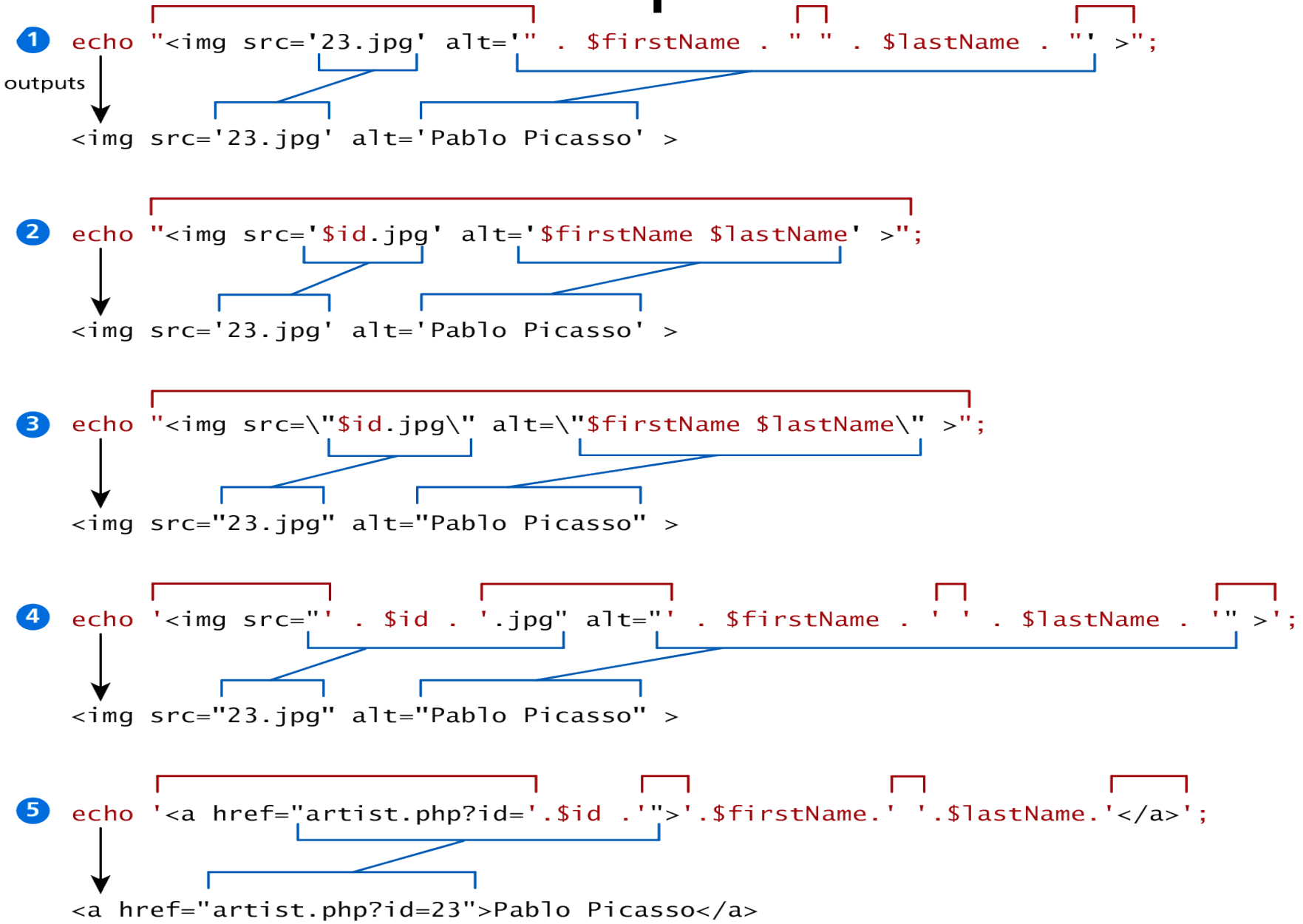
String escape Sequences

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

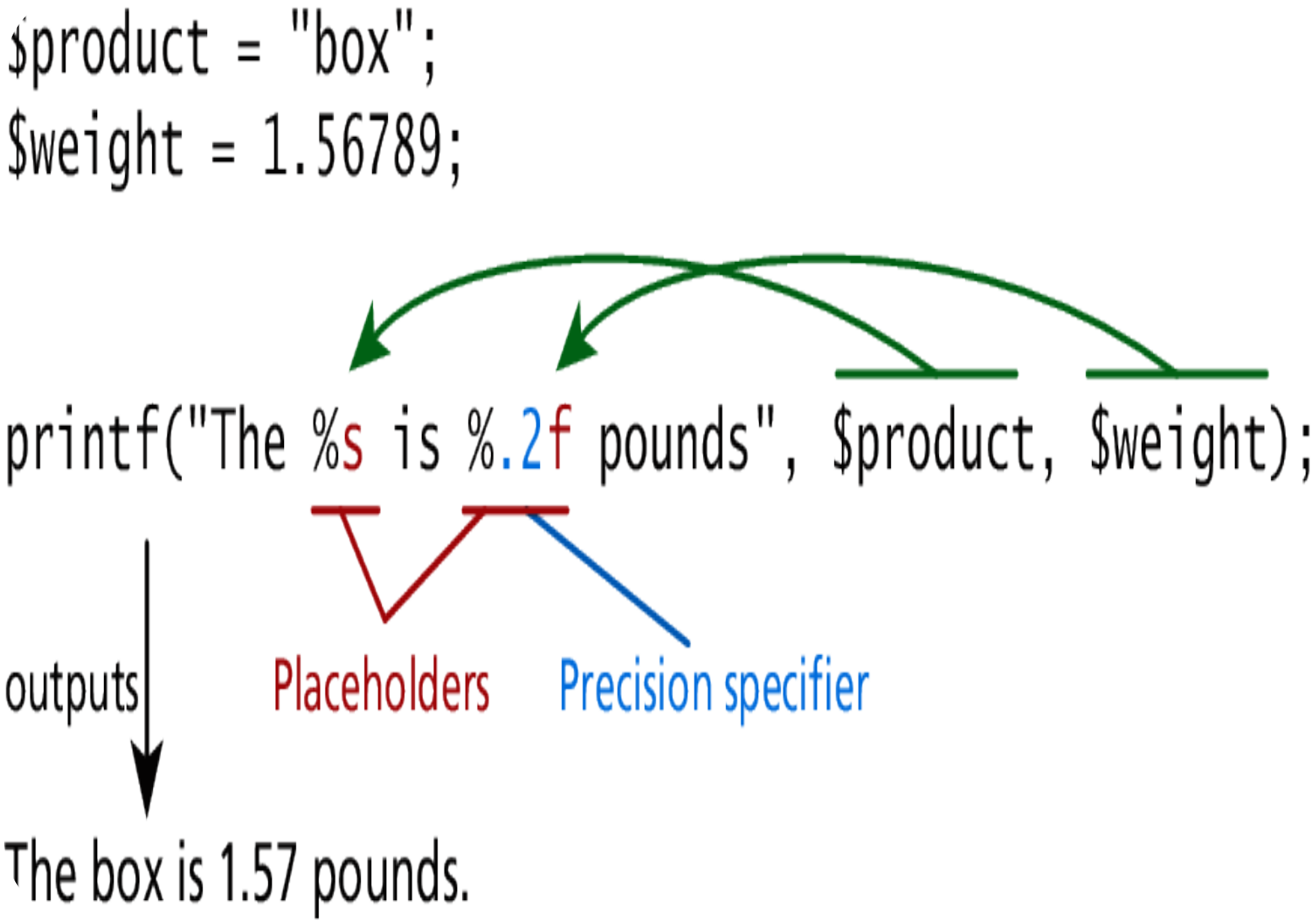
Good ol’ printf

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



PrintF

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

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

If…else

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

```php
// 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

Alternate syntax for Control Structures

PHP has an alternative syntax for most of its control structures. In this alternate syntax

- the colon (:) replaces the opening curly bracket,
- while the closing brace is replaced with endif;, endwhile;, endfor;, endforeach;, or endswitch;

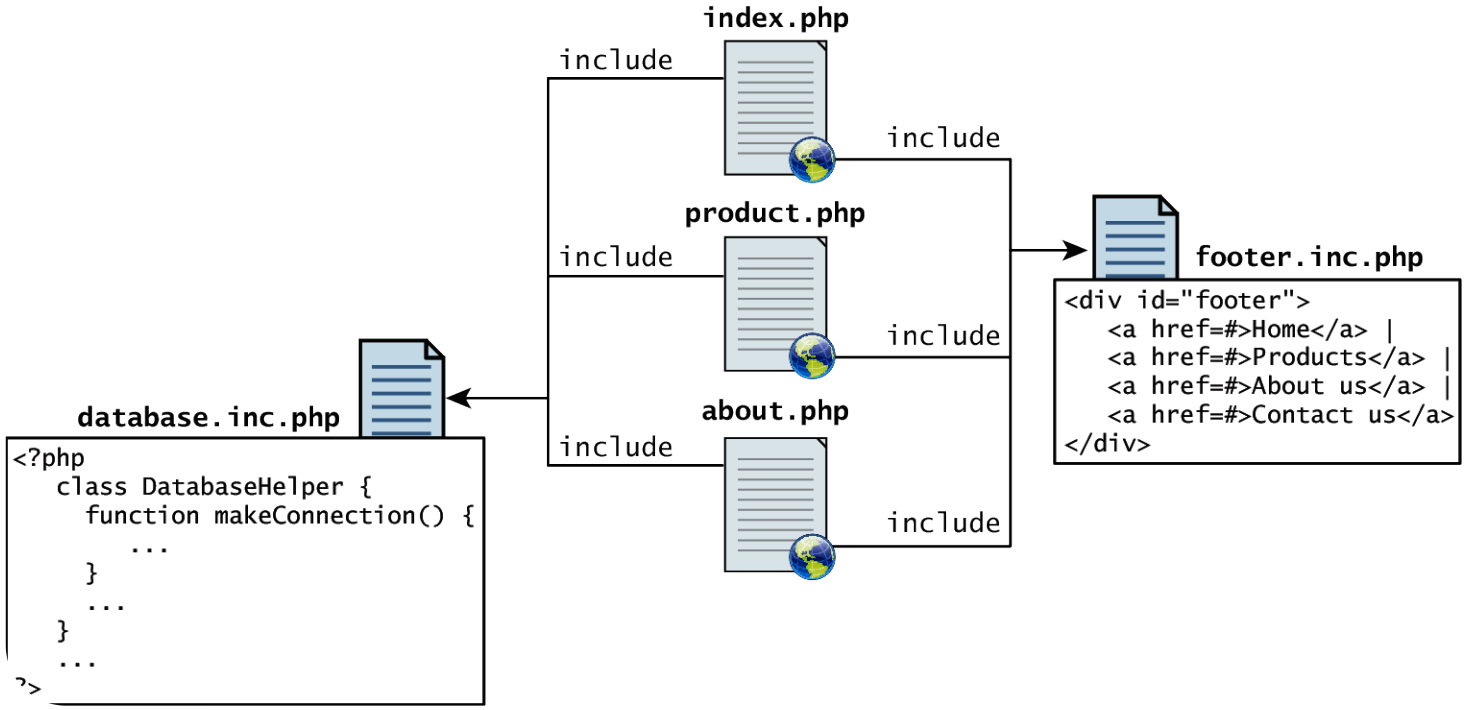
```
<?php if ($userStatus == "loggedin") :  ?>
   <a href="account.php">Account</a>
   <a href="logout.php">Logout</a>
<?php else : ?>
   <a href="login.php">Login</a>
   <a href="register.php">Register</a>
<?php endif; ?>
```

LISTING 8.12 Alternate syntax for control structures

Include Files

Organize your code

PHP does have one important facility that is generally unlike other non-web programming languages, namely the ability to include or 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";

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

Include Files

Scope

Include files are the equivalent of copying and pasting.

- Variables defined within an include file will have the scope of the line on which the include occurs
- Any variables available at that line in the calling file will be available within the called file
- If the include occurs inside a function, then all of the code contained in the called file will behave as though it had been defined inside that function

Functions

You mean we don’t write everything in main?

Just as with any language, writing code in the main function (which in PHP is equivalent to coding in the markup between <?php and ?> tags) is not a good habit to get into.

A **function** in PHP contains a small bit of code that accomplishes one thing. In PHP there are two types of function: user-defined functions and built-in functions.

1. A **user-defined function** is one that you the programmer define.

2. A **built-in function** is one of the functions that come with the PHP environment

Functions

syntax

```
/**
* 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

While the example function in Listing 8.13 returns a value, there is no requirement for this to be the case.

Functions

No return – no big deal.

```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

Call a function

Now that you have defined a function, you are able to use it whenever you want to. To call a function you must use its name with the () brackets.

Since getNiceTime() returns a string, you can assign that return value to a variable, or echo that return value directly, as shown below.

$output = getNiceTime();

echo getNiceTime();

If the function doesn’t return a value, you can just call the function:

outputFooterMenu();

Parameters

Parameters are the mechanism by which values are passed into functions.

To define a function with parameters, you must decide

- how many parameters you want to pass in,
- and in what order they will be passed
- Each parameter must be named

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

```php
/**
* 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 Value

By default, arguments passed to functions are **passed by value** in PHP. This means that PHP passes a copy of the variable so if the parameter is modified within the function, it does not change the original.

```
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=15
```

LISTING 8.17 Passing a parameter by value

Pass Parameters by Reference

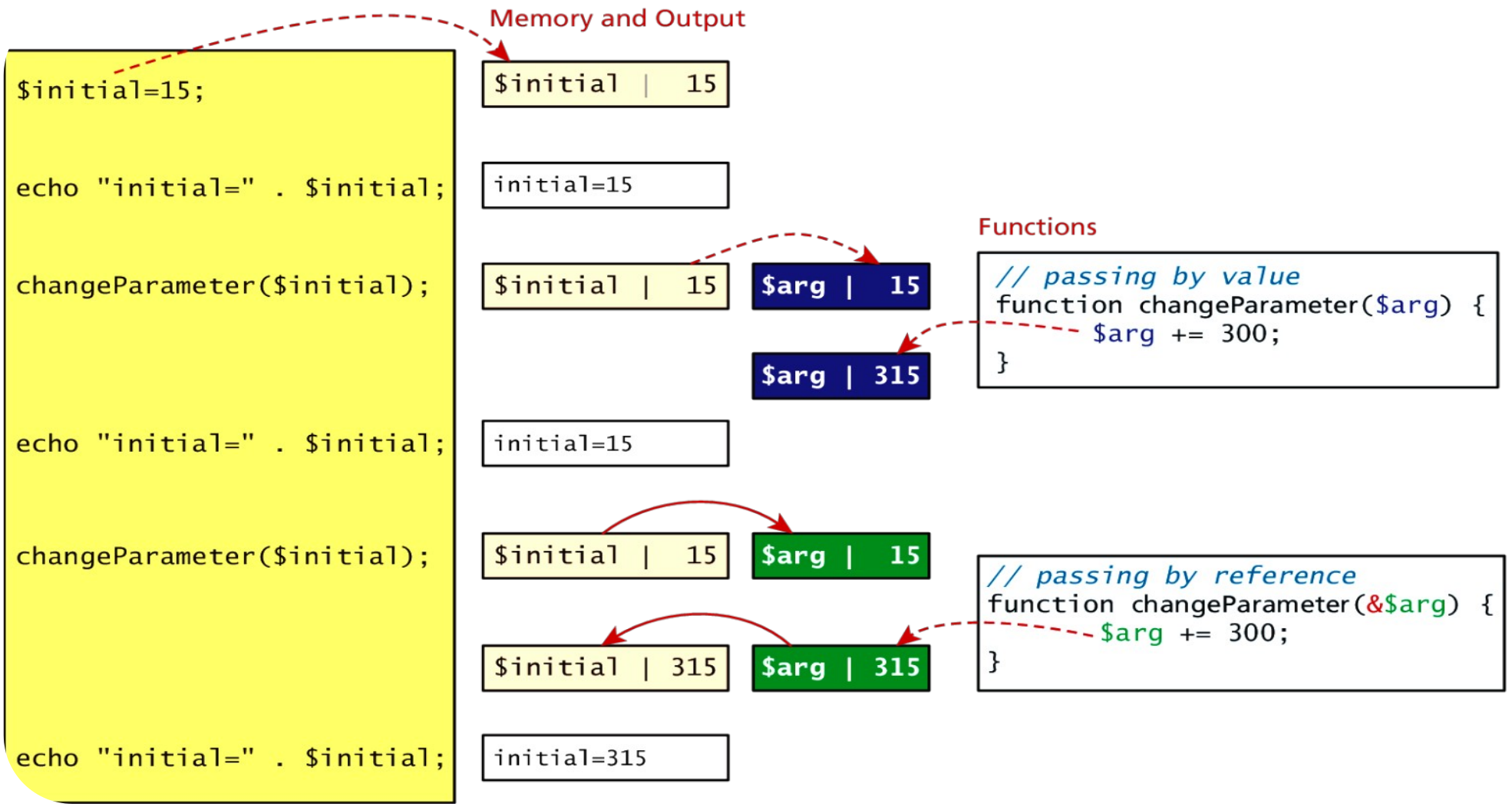
PHP also allows arguments to functions to be **passed by reference**, which will allow a function to change the contents of a passed variable.

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

Value vs 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

Forms

Form - <form>

```
<form action = ".."
      method = "...">


...
</form>
```

- action = URL
- method = get | post

Text field - <input>

For single line text input:

```
<input type = "text"
       name = "mytextfield"
       value = "initial value"
       size = "50"
       maxlength = "50"/>
```

Password - <input>

For single line *masked* text input

```
<input type = "password"
       name = "pwd"/>
```

Radio button - <input>

Single choice

```
Male: <input type = "radio" name =
 "gender" value="Male"/>
Female: <input type = "radio" name
 = "gender" value="Female"/>
```

Checkbox - <input>

Multiple choice

```
<input type = "checkbox"
       name = "sailing"/>
```

Labelling

- Old School (Strict DTD only):

```
<input type="checkbox" name="choice1"> Choice 1
</input>
```

- Strict DTD compliant:

```
Choice 1: <input type="checkbox" name="choice1/>
```

- Elegant:

```
<label for="choice1ID">Choice 1</label>
<input type="checkbox" name="choice1 id="choice1ID"/>
```

Drop-down list <select><option>

Single choice

```
<select name="country" size="1">
  <option value="UK">United Kingdom</option>
  <option value="USA">United States of America</option>
  <option value="NK">North Korea</option>
  <option value="BE">Belgium</option>
```

(multiple) selection list <select><option>

Multiple choice

```
<select name="country[]" size="3"
 multiple="multiple">
  <option value="UK">United
 Kingdom</option>
  <option value="USA">United States of
 America</option>
  <option value="NK">North
 Korea</option>
  <option value="BE">Belgium</option>
</select>
```

Text area - <textarea>

For multiple line text input

```
<textarea name="myTextArea"
          cols="30"
          rows="5">
</textarea>
```

http://www.w3schools.com/tags/tag_textarea.asp

Button - <input> or <button>

```
<input type="submit"
       name="submitButton"
       value="Submit Form"/>
```

How does one access the submitted data?

Superglobals

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

Using PHP

- Data stored in a variable
- Depends on submission method:
  - GET
  - POST
- If POST:
  - $_POST['varName'] (recommended)
- If GET, use:
  - $_GET['varName']

Two basic approaches

- Approach 1
  - HTML form
  - PHP script to process it
- Approach 2
  - PHP script containing the form and the processing script:

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

Examples

- Display content of $_POST
- Display value of text input field
- Display content of the text area
- Display value(s) of (multiple) selection list

Common operations

1. Check existence

- Has a variable been set?

```
if (isset($_POST['myCheckbox2']))
 echo "Option 2 was selected";
else
 echo "Option 2 wasn't selected.";
```

2. Check which button was pressed

- Same as above

```
if (isset($_POST['button1']))
 echo "Button 1 was pressed";
elseif (isset($_POST['button2']))
 echo "Button 2 was pressed";
else
 echo "no button pressed";
```

3. Email data

```
$to = "m.rutter@napier.ac.uk";
$subject = "form data";
$body = "Country chosen by the user:
 $_POST['country1'] ";

mail($to,$subject,$body);
```

Requires appropriate server configuration (this is not permitted on the school's server).

Example 1: mandatory text field

- Check that the value of the text field contains characters and redirect the user accordingly
- Two cases:
  1. Separate form and processing script
  2. Processing is done in the form

Example 1
Case 1

- In processing script:

```
<?php
if ($_POST['myTextField'] == ''){
  header("location: form.html");
}?>
```

- Problem:
  - No information regarding the error is passed back to the form
- Solution:
  - Make the form a PHP script
  - Send back some information about the error
  - Will be done in the tutorial

Example 2: directed study validate phone number

- Requirement: ensure that phone number has the following format: +44ddddddddddd, where d is a digit.
- Hint:
  - Check that the string has 13 characters
  - Check that the first three characters (substring) is equal to ‘+44’.

4. Error checking

- Ensure entered data is valid
- Examples
  - Mandatory fields
  - Postcode format
  - Phone number format
  - Date format
  - etc

References

- https://www.w3schools.com
- https://www.upwork.com/hiring/development/server-side-scripting-back-end-web-development-technology/
- https://en.wikipedia.org/wiki/Server-side_scripting
- https://www.w3schools.com/php/
- https://www.tutorialspoint.com/php/index.htm
- https://en.wikipedia.org/wiki/PHP