PHP Arrays, Superglobals, and Classes

Chapters 12 and 13

array

Randy Connolly and Ricardo Hoar

Fundamentals of Web Development

http://www.funwebdev.com

Objectives

1. Arrays
2. $_GET and $_POST Superglobal arrays
3. $_SERVER Array
4. Classes and Objects in PHP

Section 1 of 4

ARRAYS

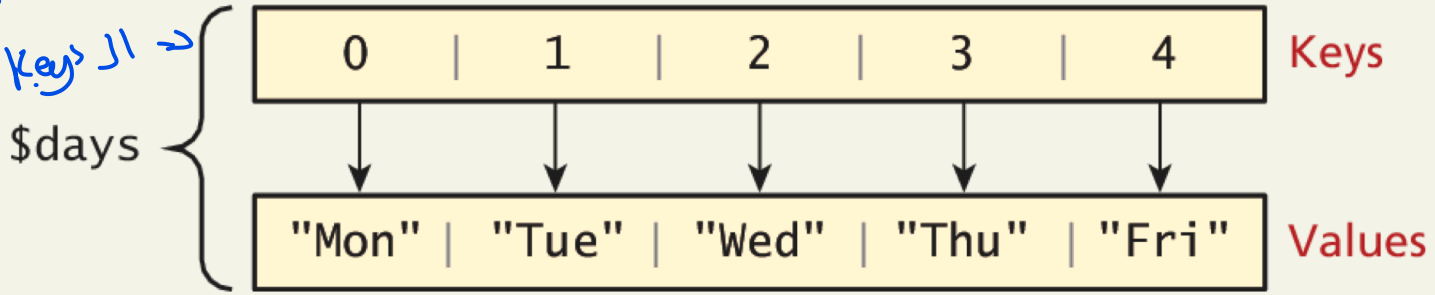
Arrays

Key Value

In PHP an array is actually an **ordered map**, which associates each value in the array with a key.

تختلف عن البرامج الثانيه

اذا ما حددت الـ Key بيعطيك أوتوماتيك



Arrays

Keys

Array keys are the means by which you refer to single element in the array.

In most programming languages array keys are limited to integers, start at 0, and go up by 1.

In PHP, array **keys** *must* be either **integers** or **strings** and **need not be sequential**.

- Keys can be of different types, try not to confuse them i.e. “1” vs 1
- If you don’t explicitly define them they are 0,1,…

Array **values** can be of any PHP type, and can be inter-mixed.

Arrays

Defining an array

The following declares an empty array named days:

$days = array();

You can also initialize it with a comma-delimited list of values inside the ( ) braces using either of two following syntaxes:

0 1 2 3 4

```
$days = array("Mon","Tue","Wed","Thu","Fri");
```

```
$days = ["Mon","Tue","Wed","Thu","Fri"]; // alternate
```

Arrays

Defining an array

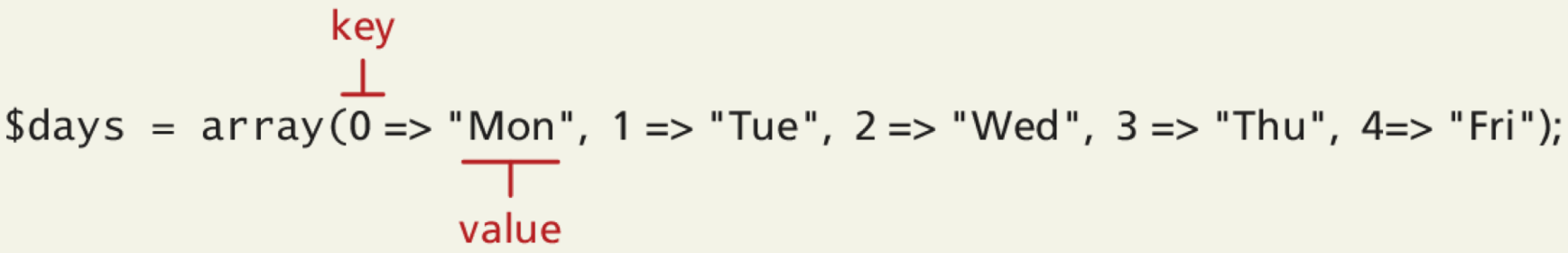
You can also declare each subsequent element in the array individually:

```
$days = array();

$days[0] = "Mon"; //set 0th key's value to "Mon"

$days[1] = "Tue";

// also alternate approach

$daysB = array();

$daysB[] = "Mon"; //set the next sequential value to "Mon"

$daysB[] = "Tue";
```

Keys and Values

In PHP, you are also able to explicitly define the keys in addition to the values.

This allows you to use keys other than the classic 0, 1, 2, . . . , n to define the indexes of an array.



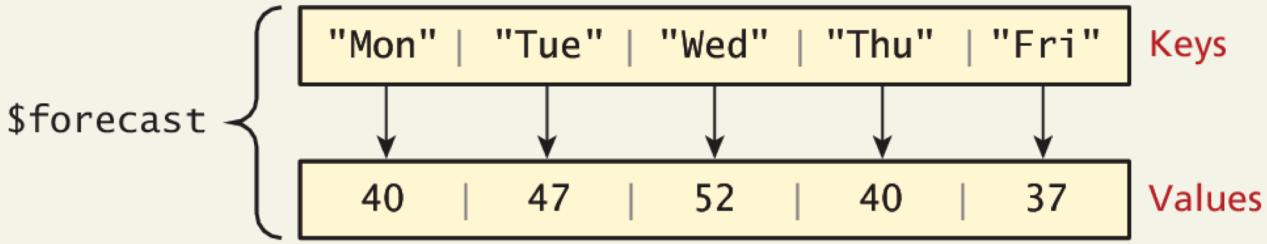
Super Explicit

Array declaration with string keys, integer values

```
$forecast = array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
```

key

value




```
echo $forecast["Tue"];  // outputs 47
echo $forecast["Thu"];  // outputs 40
```

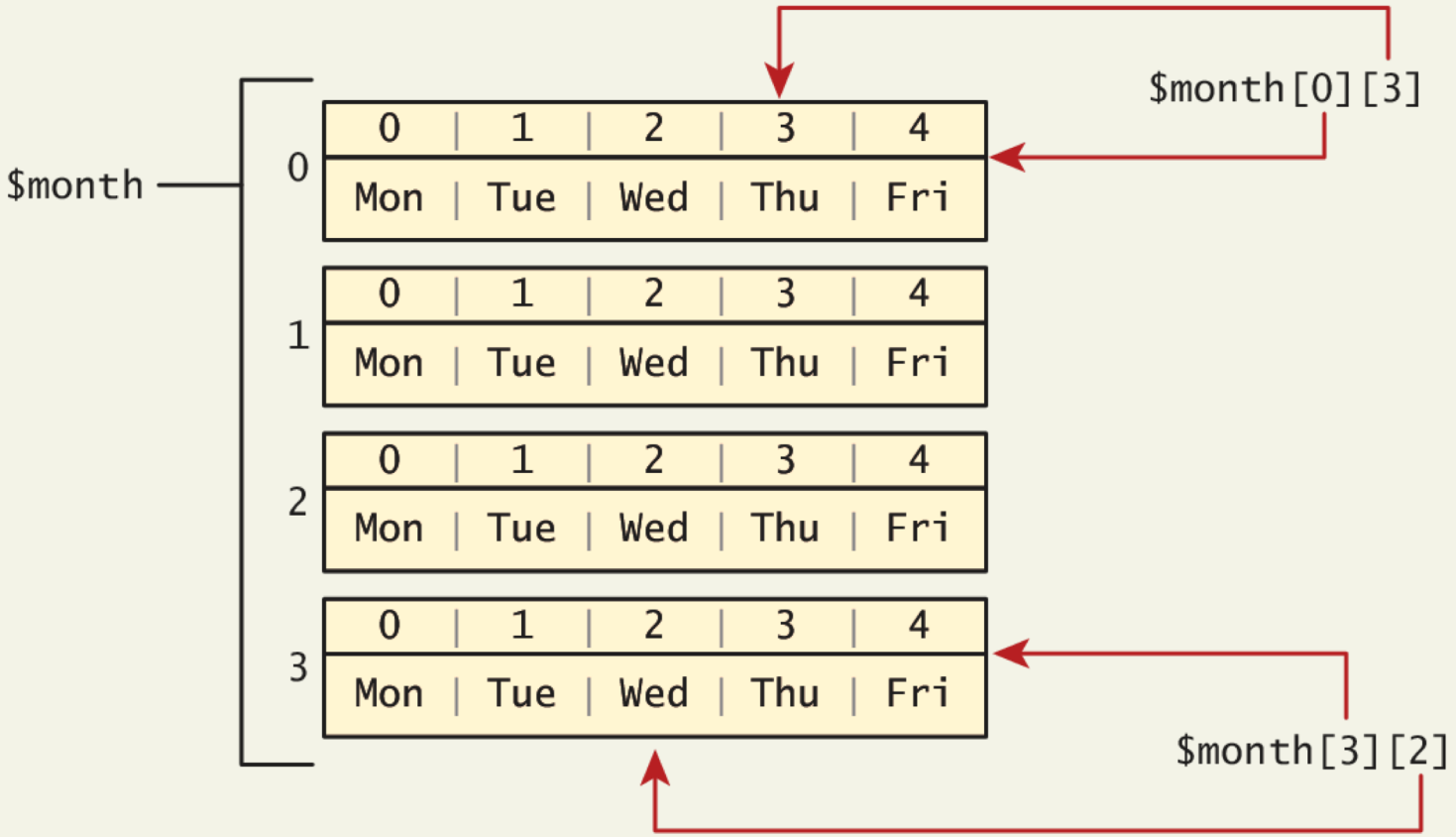
Multidimensional Arrays

Creation

```
$month = array(
        array("Mon","Tue","Wed","Thu","Fri"),
        array("Mon","Tue","Wed","Thu","Fri"),
        array("Mon","Tue","Wed","Thu","Fri"),
        array("Mon","Tue","Wed","Thu","Fri")
);
echo $month[0][3]; // outputs Thu
```

Multidimensional Arrays

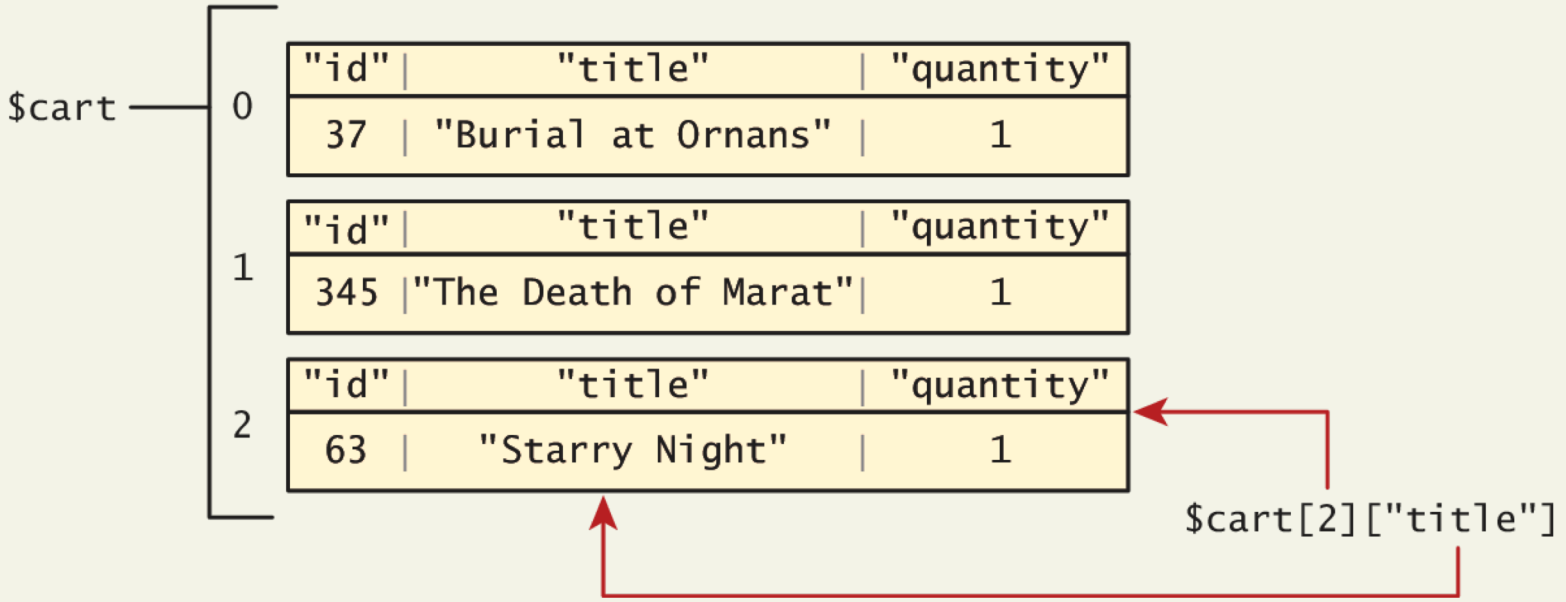
Access



Multidimensional Arrays

Another example

```
$cart = array();

$cart[] = array("id" => 37, "title" => "Burial at Ornans", "quantity" => 1);

$cart[] = array("id" => 345, "title" => "The Death of Marat", "quantity" => 1);

$cart[] = array("id" => 63, "title" => "Starry Night", "quantity" => 1);
```



Iterating through an array

```php
// while loop
$i=0;
while ($i < count($days)) {
   echo $days[$i] . "<br>";
   $i++;
}


// do While loop
$i=0;
do {
   echo $days[$i] . "<br>";
   $i++;
} while ($i < count($days));


// for loop
for ($i=0; $i<count($days); $i++) {
   echo $days[$i] . "<br>";
}
```

LISTING 9.2 Iterating through an array using while, do while, and for loops

Iterating through an array

Foreach loop is pretty nice

The challenge of using the classic loop structures is that when you have nonsequential integer keys (i.e., an associative array), you can't write a simple loop that uses the $i++ construct. To address the dynamic nature of such arrays, you have to use iterators to move through such an array.

```
// foreach: iterating through the values
foreach ($forecast as $value) {
   echo $value . "<br>";
}

// foreach: iterating through the values AND the keys
foreach ($forecast as $key => $value) {
   echo "day" . $key . "=" . $value;
}
```

LISTING 9.3 Iterating through an associative array using a foreach loop

Adding to an array

To an array

An element can be added to an array simply by using a key/index that hasn't been used

```
$days[5] = "Sat";
```

A new element can be added to the end of any array

```
$days[ ] = "Sun";
```

Adds the value "Sun" to the largest existing index + 1

Adding to an array

And quickly printing

PHP is more than happy to let you “skip” an index

```
$days = array("Mon","Tue","Wed","Thu","Fri");

$days[7] = "Sat";

print_r($days);
```

Array ([0] => Mon [1] => Tue [2] => Wed [3] => Thu [**4**] => Fri [**7**] => Sat)’

If we try referencing $days[6], it will return a **NULL** value

Deleting from an array

You can explicitly delete array elements using the unset() function

```php
$days = array("Mon","Tue","Wed","Thu","Fri");

unset($days[2]);
unset($days[3]);

print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [4] => Fri )

$days = array_values($days);
print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [2] => Fri )
```

LISTING 9.4 Deleting elements

Deleting from an array

You can explicitly delete array elements using the unset() function.

array_values() reindexes the array numerically

```php
$days = array("Mon","Tue","Wed","Thu","Fri");

unset($days[2]);
unset($days[3]);

print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [4] => Fri )

$days = array_values($days);
print_r($days); // outputs: Array ( [0] => Mon [1] => Tue [2] => Fri )
```

LISTING 9.4 Deleting elements

Checking for a value

Since array keys need not be sequential, and need not be integers, you may run into a scenario where you want to check if a value has been set for a particular key.

To check if a value exists for a key, you can therefore use the isset() function, which returns true if a value has been set, and false otherwise

```
$oddKeys = array (1 => "hello", 3 => "world", 5 => "!");
if (isset($oddKeys[0])) {
   // The code below will never be reached since $oddKeys[0] is not set!
   echo "there is something set for key 0";
}
if (isset($oddKeys[1])) {
   // This code will run since a key/value pair was defined for key 1
   echo "there is something set for key 1, namely ". $oddKeys[1];
}
```

LISTING 9.5 Illustrating nonsequential keys and usage of isset( )

Array Sorting

Sort it out

There are many built-in sort functions, which sort by key or by value. To sort the $days array by its values you would simply use:

```
sort($days);
```

As the values are all strings, the resulting array would be:

Array ([0] => Fri [1] => Mon [2] => Sat [3] => Sun [4] => Thu [5] => Tue [6] => Wed)

A better sort, one that would have kept keys and values associated together, is:

```
asort($days);
```

Array ([4] => Fri [0] => Mon [5] => Sat [6] => Sun [3] => Thu [1] => Tue [2] => Wed)

More array operations

Too many to go over in depth here…

- array_keys($someArray)
- array_values($someArray)
- array_rand($someArray, $num=1)
- array_reverse($someArray)
- array_walk($someArray, $callback, optionalParam)
- in_array($needle, $haystack)
- shuffle($someArray)
- …

Section 2 of 4

$_GET AND $_POST SUPERGLOBAL ARRAYS

Superglobal Arrays

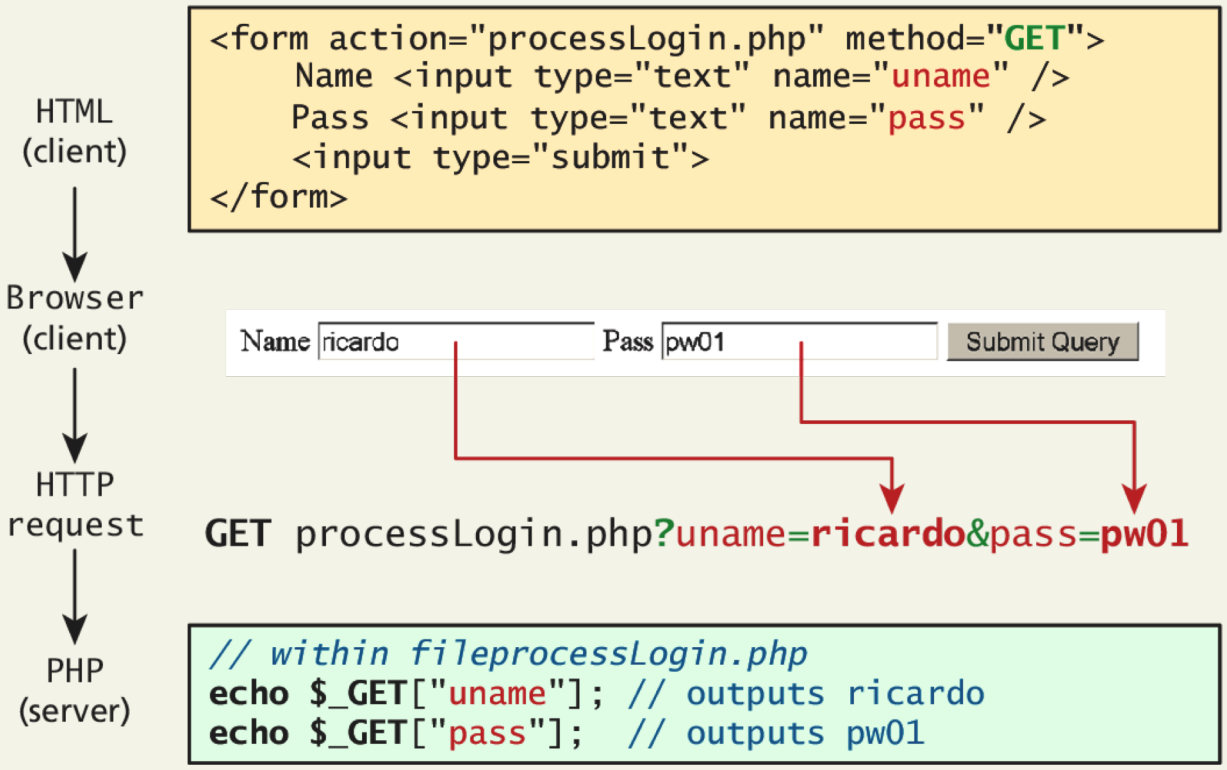
PHP uses special predefined associative arrays called **superglobal variables** that allow the programmer to easily access HTTP headers, query string parameters, and other commonly needed information.

They are called superglobal because they are always in scope, and always defined.

$_GET and $_POST

Sound familiar?

The $_GET and $_POST arrays are the most important superglobal variables in PHP since they allow the programmer to access data sent by the client in a query string.



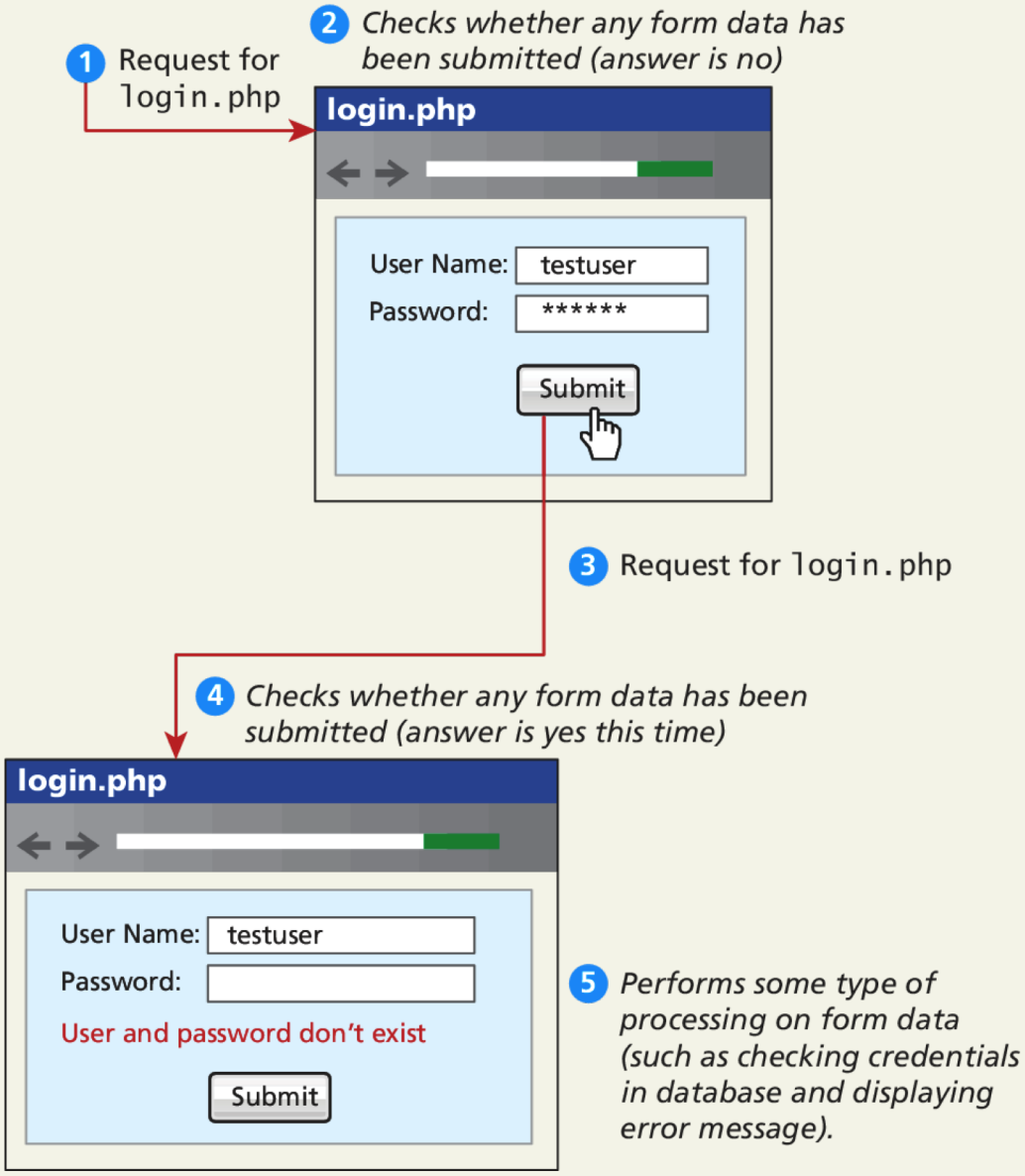
$_GET and $_POST

Sound familiar?

- **GET** requests parse query strings into the $_GET array
- **POST** requests are parsed into the $_POST array

This mechanism greatly simplifies accessing the data posted by the user, since you need not parse the query string or the POST request headers!

Determine if any data sent



Determine if any data sent

Isset()

```
<!DOCTYPE html>
<html>
<body>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   if ( isset($_POST["uname"]) && isset($_POST["pass"]) ) {
      // handle the posted data.
      echo "handling user login now ...";
      echo "... here we could redirect or authenticate ";
      echo " and hide login form or something else";
   }
}
?>
<h1>Some page that has a login form</h1>
<form action="samplePage.php" method="POST">
   Name <input type="text" name="uname"/><br/>
   Pass <input type="password" name="pass"/><br/>
   <input type="submit">
</form>
</body>
</html>
```

LISTING 9.6 Using isset() to check query string data

Accessing Form Array Data

Sometimes in HTML forms you might have multiple values associated with a single name;

```
<form method="get">
   Please select days of the week you are free.<br />
   Monday <input type="checkbox" name="day" value="Monday" /> <br />
   Tuesday <input type="checkbox" name="day" value="Tuesday" /> <br />
   Wednesday <input type="checkbox" name="day" value="Wednesday" /> <br />
   Thursday <input type="checkbox" name="day" value="Thursday" /> <br />
   Friday <input type="checkbox" name="day" value="Friday" /> <br />
   <input type="submit" value="Submit">
</form>
```

LISTING 9.7 HTML that enables multiple values for one name

Accessing Form Array Data

HTML tweaks for arrays of data

Unfortunately, if the user selects more than one day and submits the form, the $_GET['day'] value in the superglobal array *will only contain the last value from the list* that was selected.

To overcome this limitation, you must change the name attribute for each checkbox from day to day[].

```
Monday <input type="checkbox" name="day[]" value="Monday" />

Tuesday <input type="checkbox" name="day[]" value="Tuesday" />
```

Accessing Form Array Data

Meanwhile on the server

After making this change in the HTML, the corresponding variable $_GET['day'] will now have a value that is of type array.

```php
<?php

echo "You submitted " . count($_GET['day']) . "values";
foreach ($_GET['day'] as $d) {
   echo $d . ", ";
}

?>
```

LISTING 9.8 PHP code to display an array of checkbox variables

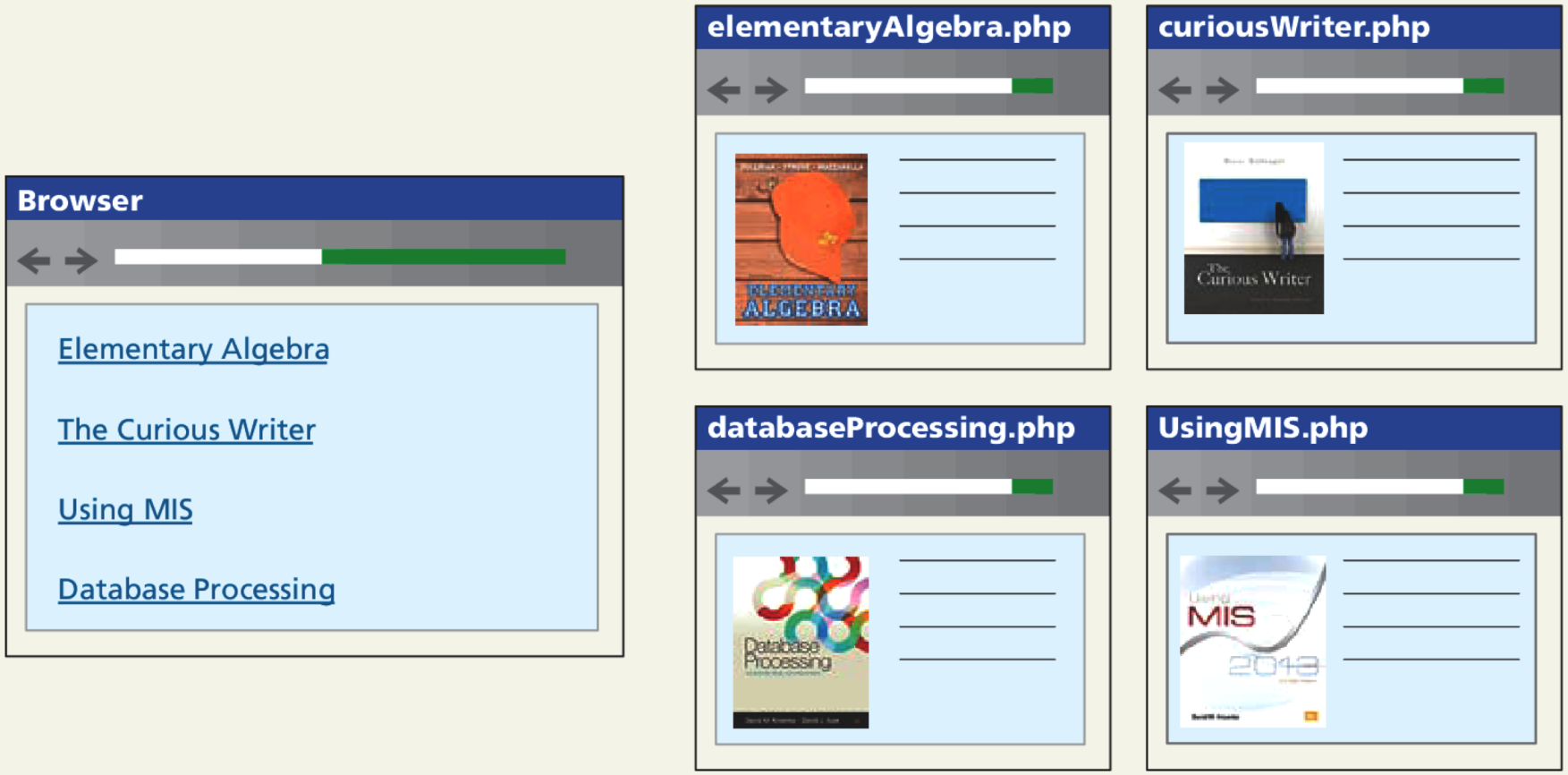
Using Query String in Links

Design idea

Imagine a web page in which we are displaying a list of book links. One approach would be to have a separate page for each book.

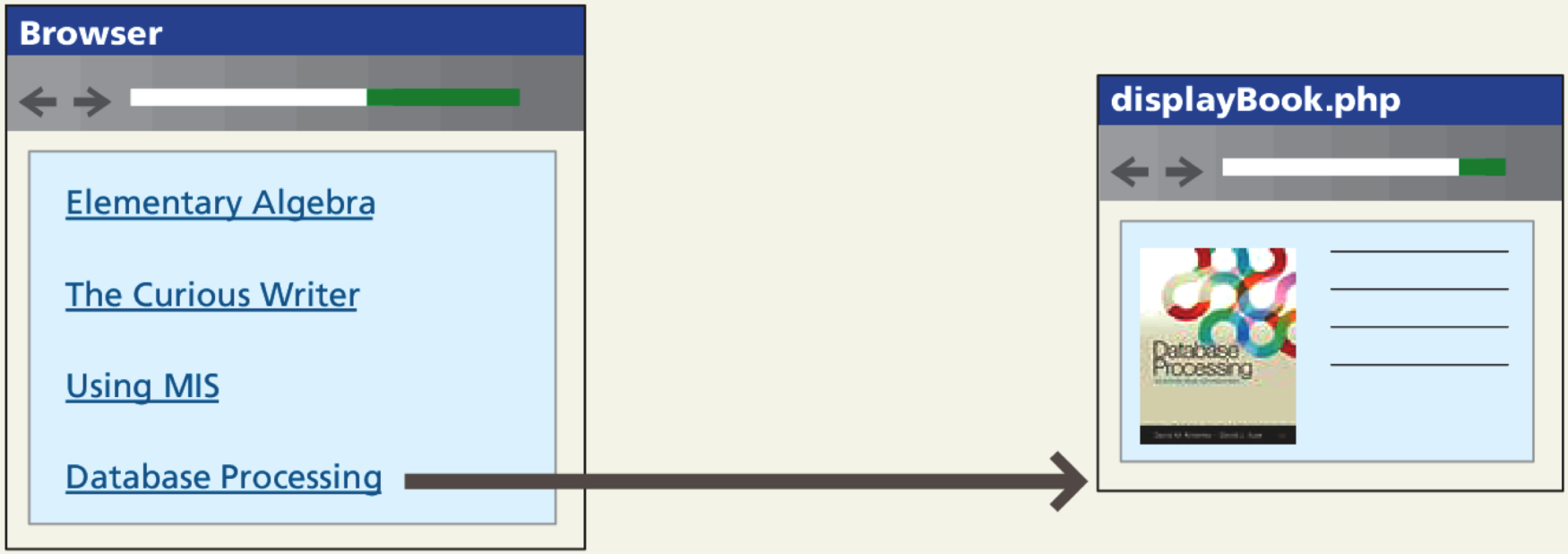
Using Query Strings in links

Not a great setup



Using Query Strings in links

Use the query string to reduce code duplication




```
<a href="displayBook.php?isbn=0132145375">Database Processing</a>
```

Query string

Sanitizing Query Strings

Just because you are expecting a proper query string, doesn't mean that you are going to get a properly constructed query string.

- **distrust all user input**

The process of checking user input for incorrect or missing information is sometimes referred to as the process of **sanitizing user inputs.**

Sanitation

Don't forget trim()

```
// This uses a database API . . . we will learn about it in Chapter 11
$pid = mysqli_real_escape_string($link, $_GET['id']);

if ( is_int($pid) ) {
     // Continue processing as normal
}
else {
     // Error detected. Possibly a malicious user
}
```

LISTING 9.9 Simple sanitization of query string values

Section 3 of 4

$_SERVER ARRAY

$_SERVER

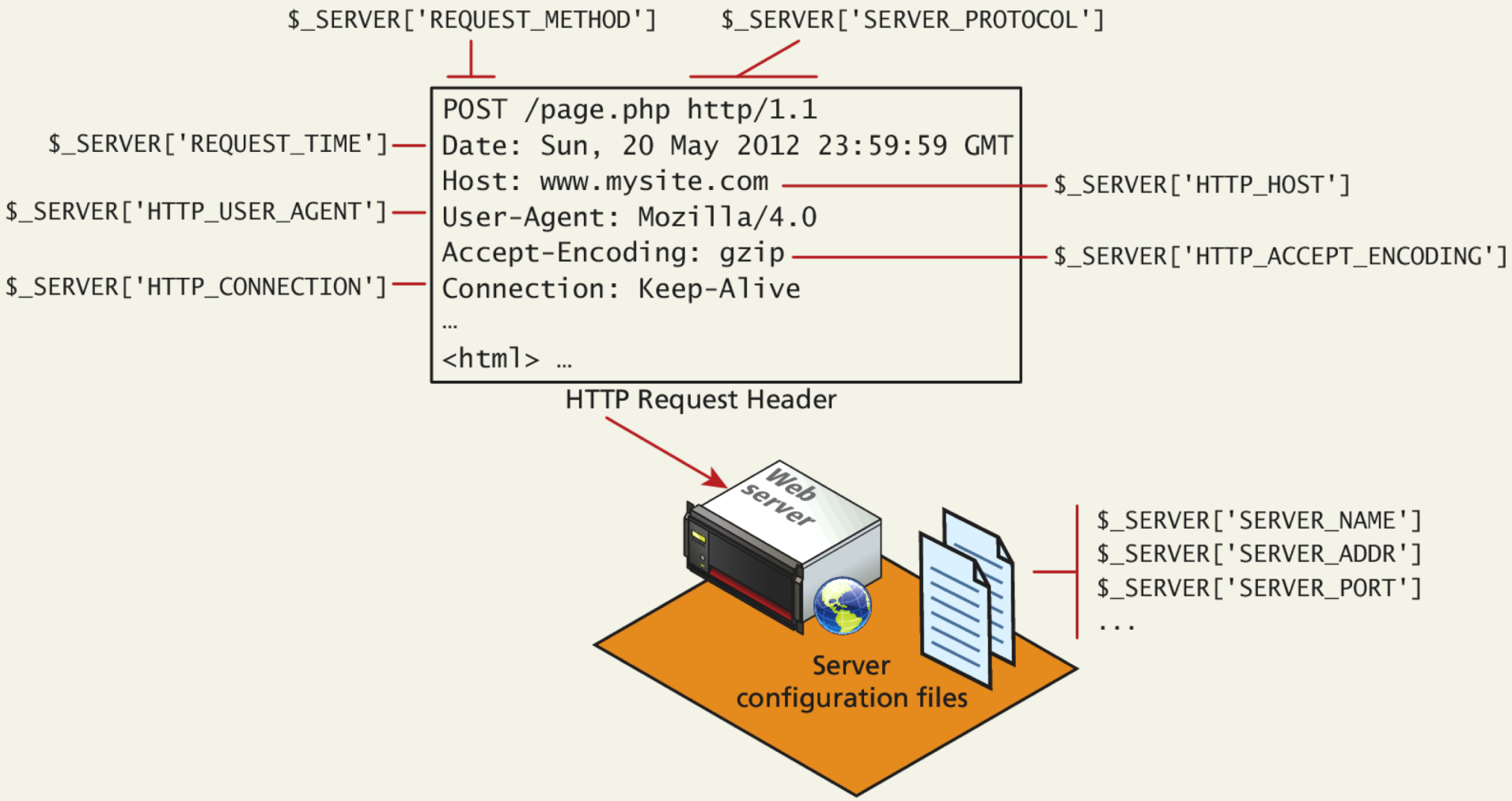
The $_SERVER associative array contains

- HTTP request headers (send by client)
- configuration options for PHP

To use the $_SERVER array, you simply refer to the relevant case-sensitive keyname:

```
echo $_SERVER["SERVER_NAME"] . "<br/>";

echo $_SERVER["SERVER_SOFTWARE"] . "<br/>";

echo $_SERVER["REMOTE_ADDR"] . "<br/>";
```

$_SERVER


$_SERVER['REQUEST_METHOD']
$_SERVER['SERVER_PROTOCOL']
POST /page.php http/1.1
$_SERVER['REQUEST_TIME']
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
$_SERVER['HTTP_HOST']
$_SERVER['HTTP_USER_AGENT']
User-Agent: Mozilla/4.0
Accept-Encoding: gzip
$_SERVER['HTTP_ACCEPT_ENCODING']
$_SERVER['HTTP_CONNECTION']
Connection: Keep-Alive
…
<html> …
HTTP Request Header
Web server
$_SERVER['SERVER_NAME']
$_SERVER['SERVER_ADDR']
$_SERVER['SERVER_PORT']
...
Server configuration files

SERVER INFORMATION KEYS

- SERVER_NAME contains the name of the site that was requested
- SERVER_ADDR tells us the IP of the server
- DOCUMENT_ROOT tells us the location from which you are currently running your script
- SCRIPT_NAME key that identifies the actual script being executed

Request Header Keys

- REQUEST_METHOD returns the request method that was used to access the page: that is, GET, HEAD, POST, PUT
- REMOTE_ADDR key returns the IP address of the requestor
- HTTP_USER_AGENT contains the operating system and browser that the client is using
- HTTP_REFERER contains the address of the page that referred us to this one (if any) through a link

Security

Headers can be forged

All headers can be forged!

- The HTTP_REFERER header can lie about where the referral came from
- The USER_AGENT can lie about the operating system and browser the client is using.

Section 4 of 4

OBJECTS AND CLASSES IN PHP

Defining Classes

In PHP

The PHP syntax for defining a class uses the class keyword followed by the class name and { } braces

```
class Artist {
   public   $firstName;
   public   $lastName;
   public   $birthDate;
   public   $birthCity;
   public   $deathDate;
}
```

LISTING 10.1 A simple Artist class

Instantiating Objects

In PHP

Defining a class is not the same as using it. To make use of a class, one must **instantiate** (create) objects from its definition using the *new* keyword.

```
$picasso = new Artist();

$dali = new Artist();
```

Properties

The things in the objects

Once you have instances of an object, you can access and modify the properties of each one separately using the variable name and an arrow (->).

```
$picasso = new Artist();
$dali = new Artist();
$picasso->firstName = "Pablo";
$picasso->lastName = "Picasso";
$picasso->birthCity = "Malaga";
$picasso->birthDate = "October 25 1881";
$picasso->deathDate = "April 8 1973";
```

LISTING 10.2 Instantiating two Artist objects and setting one of those object's properties

Constructors

A Better way to build

Constructors let you specify parameters during instantiation to initialize the properties within a class right away.

In PHP, constructors are defined as functions (as you shall see, all methods use the function keyword) with the name **__construct().**

Notice that in the constructor each parameter is assigned to an internal class variable using the $this-> syntax. you **must** always use the $this syntax to reference all properties and methods associated with this particular instance of a class.

Constructors

An Example

```
class Artist {
   // variables from previous listing still go here
   ...

   function __construct($firstName, $lastName, $city, $birth,
                        $death=null) {
      $this->firstName = $firstName;
      $this->lastName = $lastName;
      $this->birthCity = $city;
      $this->birthDate = $birth;
      $this->deathDate = $death;
   }
}
```

LISTING 10.3 A constructor added to the class definition

Constructors

Using the constructor

```
$picasso = new Artist("Pablo","Picasso","Malaga","Oct 25,1881","Apr 8,1973");

$dali = new Artist("Salvador","Dali","Figures","May 11 1904", "Jan 23 1989");
```

Methods

Functions In a class

Methods are like functions, except they are associated with a class.

They define the tasks each instance of a class can perform and are useful since they associate behavior with objects.

```
$picasso = new Artist( . . . )

echo $picasso->outputAsTable();
```

Methods

The example definition

```
class Artist {
    ...
    public function outputAsTable() {
      $table = "<table>";
      $table .= "<tr><th colspan='2'>";
      $table .= $this->firstName . " " . $this->lastName;
      $table .= "</th></tr>";
      $table .= "<tr><td>Birth:</td>";
      $table .= "<td>" . $this->birthDate;
      $table .= "(" . $this->birthCity . ")</td></tr>";
      $table .= "<tr><td>Death:</td>";
      $table .= "<td>" . $this->deathDate . "</td></tr>";
      $table .= "</table>";
      return $table;
    }
}
```

LISTING 10.4 Method definition

Class constants

Never changes

Constant values can be stored more efficiently as class constants so long as they are not calculated or updated

They are added to a class using the **const** keyword.

```
const EARLIEST_DATE = 'January 1, 1200';
```

Unlike all other variables, constants don't use the $ symbol when declaring or using them.

Accessed both inside and outside the class using

- **self::EARLIEST_DATE** in the class and
- **classReference::EARLIEST_DATE** outside.

Server and Desktop Objects

Not the same

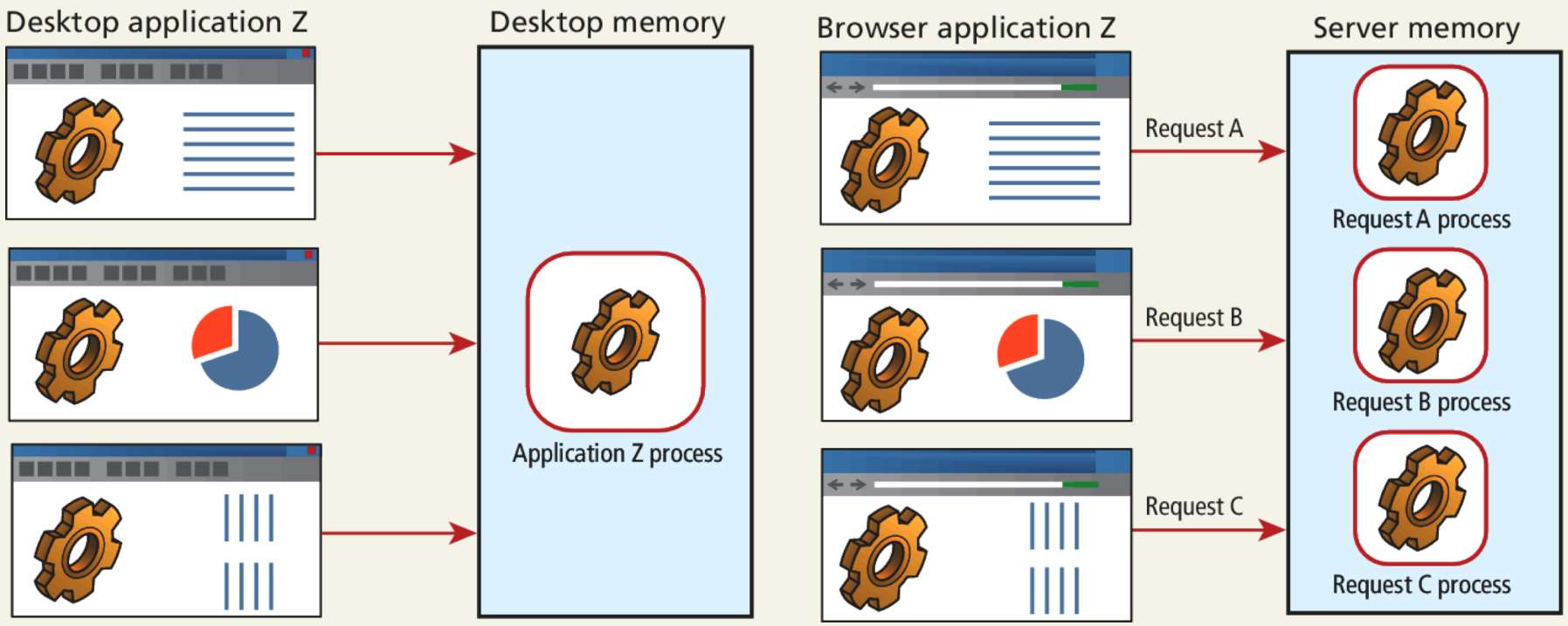
While desktop software can load an object into memory and make use of it for several user interactions, **a PHP object is loaded into memory only for the life of that HTTP request**.

We must use classes differently than in the desktop world, since the object must be recreated and loaded into memory for every request.

Unlike a desktop, there are potentially many thousands of users making requests at once, so not only are objects destroyed upon responding to each request, but memory must be shared between many simultaneous requests, each of which may load objects into memory or each request that requires it.

Server and Desktop Objects

Not the same



What You've Learned

1. Arrays
2. $_GET and $_POST Superglobal arrays
3. $_SERVER Array
4. Classes and Objects in PHP