PHP Arrays and Superglobals

Abdallah Karakra & Sobhi Ahmed

Chapter 9

Objectives

1 Arrays

2 $_GET and $_POST Superglobal arrays

3 $_SERVER Array

4 $_FILES Array

5 Reading/Writing Files

Uploaded By: Jibreel Bornat

Fundamentals of Web Development

ARRAYS

Arrays

Background

An array is a data structure that

- **Collects a number of related elements together in a single variable.**

- Allows the set to be Iterated
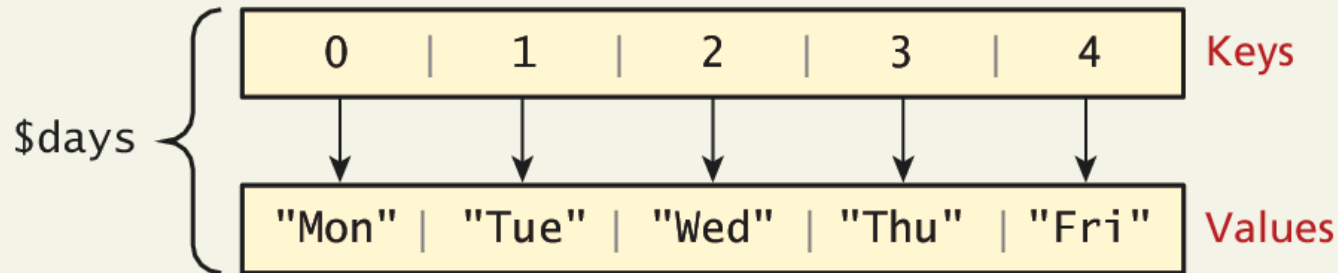
- Allows access of any element

Since PHP implements an array as a dynamic structure:

- **Add** to the array

- **Remov**e from the array

Arrays

Key Value

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

Arrays

Keys

Array keys are the means by which you refer to single elements 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.**

- Don't mix key types i.e. "1" vs 1

- If you don't explicitly define them they are 0,1,…

Arrays

Values

Array values, unlike keys, are not restricted to integers and strings.

They can be any object, type, or primitive supported in PHP.

You can even have objects of your own types, so long as the keys in the array are integers and strings.

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:

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

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

```php
1   <?php
2       /*Days of the week*/
3       echo "Have a look to the below array <br>";
4   //    $days = array("Mon","Tue","Wed","Thu","Fri");
5       $days = ["Mon","Tue","Wed","Thu","Fri"];
6       $i=0;
7 ▼    while ($i<count($days)){
8           echo "days[$i]= ".$days[$i]."<br>";
9           $i++;
10      }
11  ?>
```

Have a look to the below array
days[0]= Mon
days[1]= Tue
days[2]= Wed
days[3]= Thu
days[4]= Fri

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";
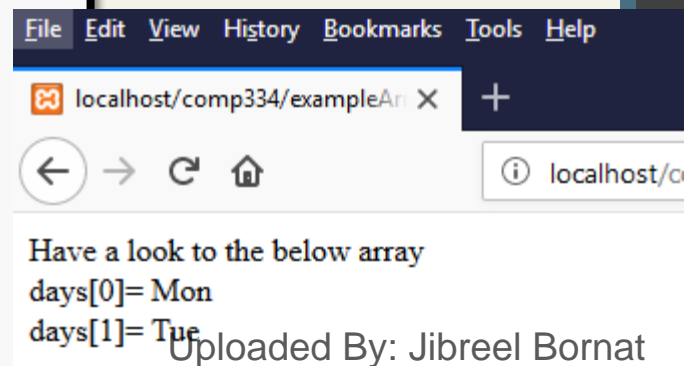
Arrays

```php
<?php
   /*Days of the week*/
   echo "Have a look to the below array <br>";

   $days=array();
   $days[0]="Mon";
   $days[1]="Tue";
   $days[2]="Wed";
   $days[3]="Thu";
   $days[4]="Fri";

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

Have a look to the below array
days[0]= Mon
days[1]= Tue
days[2]= Wed
days[3]= Thu
days[4]= Fri

```php
<?php
   /*Days of the week*/
   echo "Have a look to the below array <br>";

   $days = array();
   $days[] = "Mon"; //set the next sequential value to "Mon"
   $days[] = "Tue";


   $i=0;
   while ($i<count($days)){
       echo "days[$i]= ".$days[$i]."<br>";
       $i++;
   }
?>
```
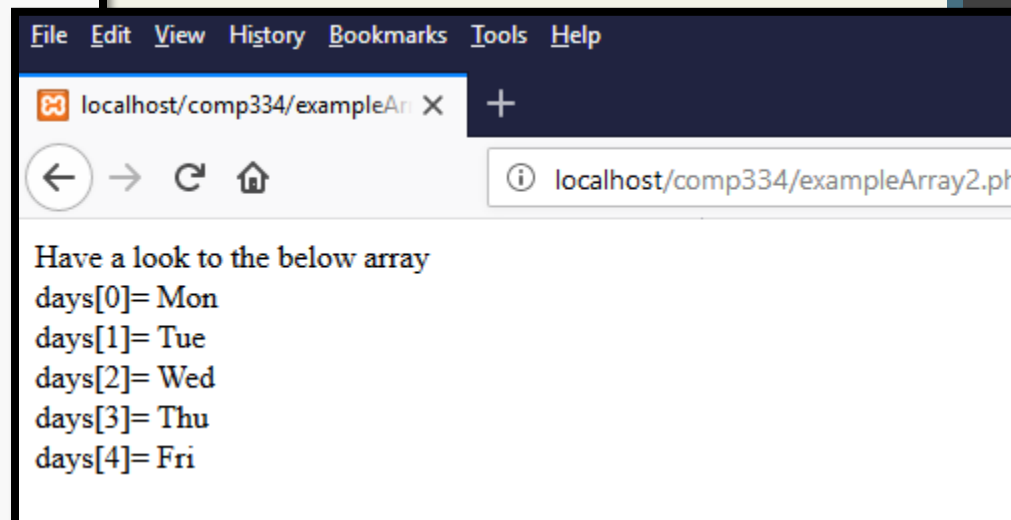
Have a look to the below array
days[0]= Mon
days[1]= Tue

Arrays

Access values

To access values in an array you refer to their key using the square bracket notation.

echo "Value at index 1 is ". $days[1];

Uploaded By: Jibreel Bornat

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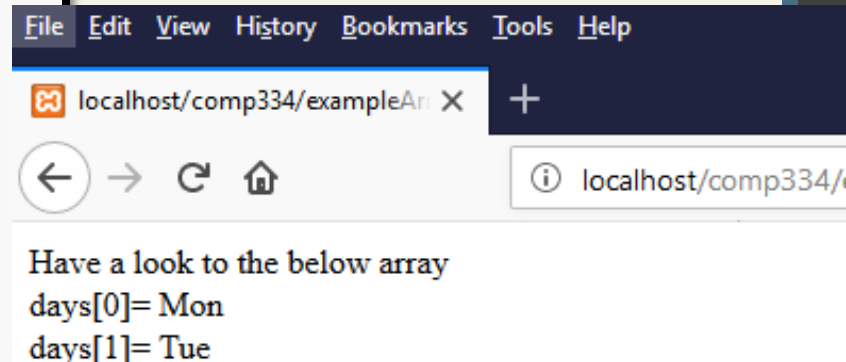
In PHP, you are also able to explicitly define the keys in addition to the values.

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

```
                 key
                  ⊥
$days = array(0 => "Mon", 1 => "Tue", 2 => "Wed", 3 => "Thu", 4=> "Fri");
                          ⊤
                        value
```

```php
1   <?php
2       /*Days of the week*/
3       echo "Have a look to the below array <br>";
4
5       $days = array(0=>"Mon",1=>"Tue");
6
7
8       $i=0;
9 ▼     while ($i<count($days)){
10          echo "days[$i]= ".$days[$i]."<br>";
11          $i++;
12      }
13  ?>
```

File Edit View History Bookmarks Tools Help

localhost/comp334/exampleAr ✕   +

← → C ⌂   ⓘ localhost/comp334/

Have a look to the below array
days[0]= Mon
days[1]= Tue

Uploaded By: Jibreel Bornat

Fundamentals of Web Development
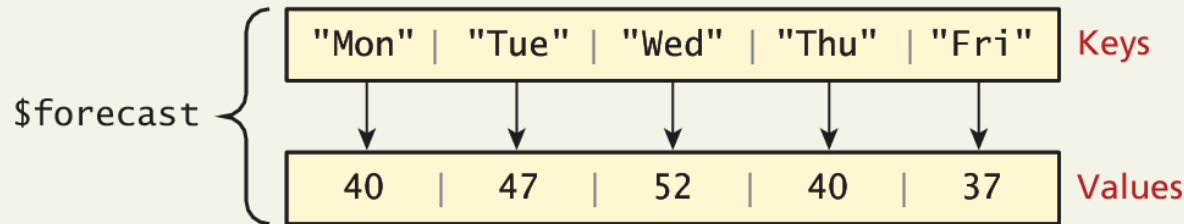
Super Explicit

Array declaration with string keys, integer values

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

```
            "Mon" | "Tue" | "Wed" | "Thu" | "Fri"     Keys
$forecast
             40   |  47   |  52   |  40   |  37        Values
```

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

```php
1    <?php
2       /*Days of the week*/
3       echo "Have a look to the below array <br>";
4
5        $days = array('A'=>"Mon",'B'=>"Tue");
6
7
8       $i='A';
9 ▼     while ($i<'C'){
10          echo "days[$i]= ".$days[$i]."<br>";
11          $i++;
12      }
13   ?>
```
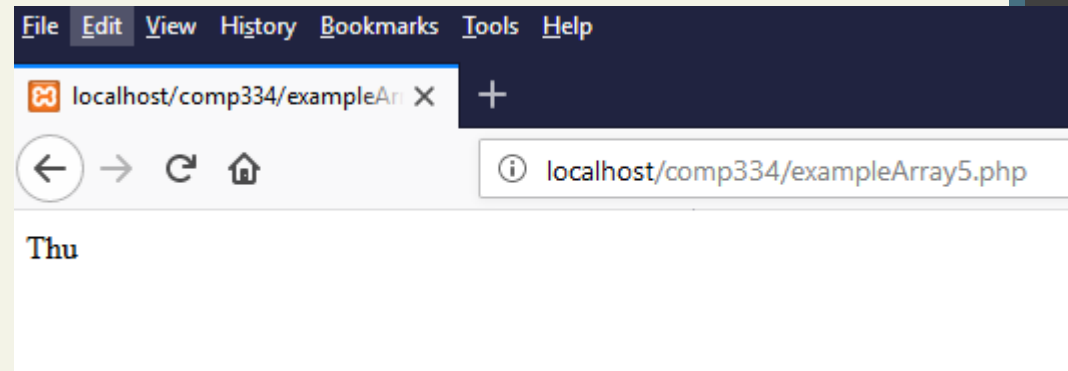
File Edit View History Bookmarks Tools Help

localhost/comp334/exampleAr ✕   +

← → C �   ⓘ localhost/comp3

Have a look to the below array
days[A]= Mon
days[B]= Tue

Uploaded By: Jibreel Bornat
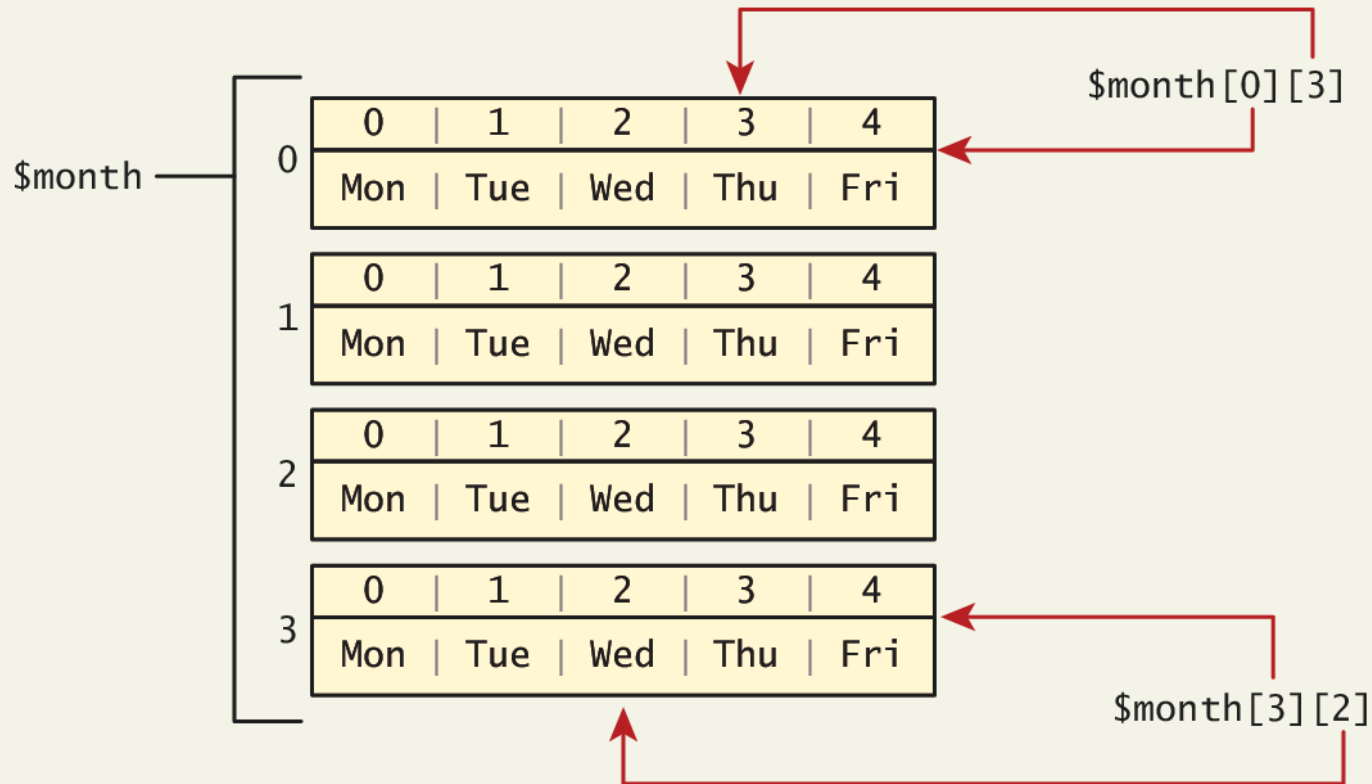
Multidimensional Arrays

Creation

```php
1    <?php
2    $month = array(
3        array("Mon","Tue","Wed","Thu","Fri"),
4        array("Mon","Tue","Wed","Thu","Fri"),
5        array("Mon","Tue","Wed","Thu","Fri"),
6        array("Mon","Tue","Wed","Thu","Fri")
7    );
8    echo $month[0][3]; // outputs Thu
9
10   ?>
```

File Edit View History Bookmarks Tools Help

localhost/comp334/exampleAr ✕   +

← → C ⌂   ⓘ localhost/comp334/exampleArray5.php

Thu

Randy Connolly and Ricardo Hoar   Fundamentals of Web Development

Uploaded By: Jibreel Bornat

Multidimensional Arrays

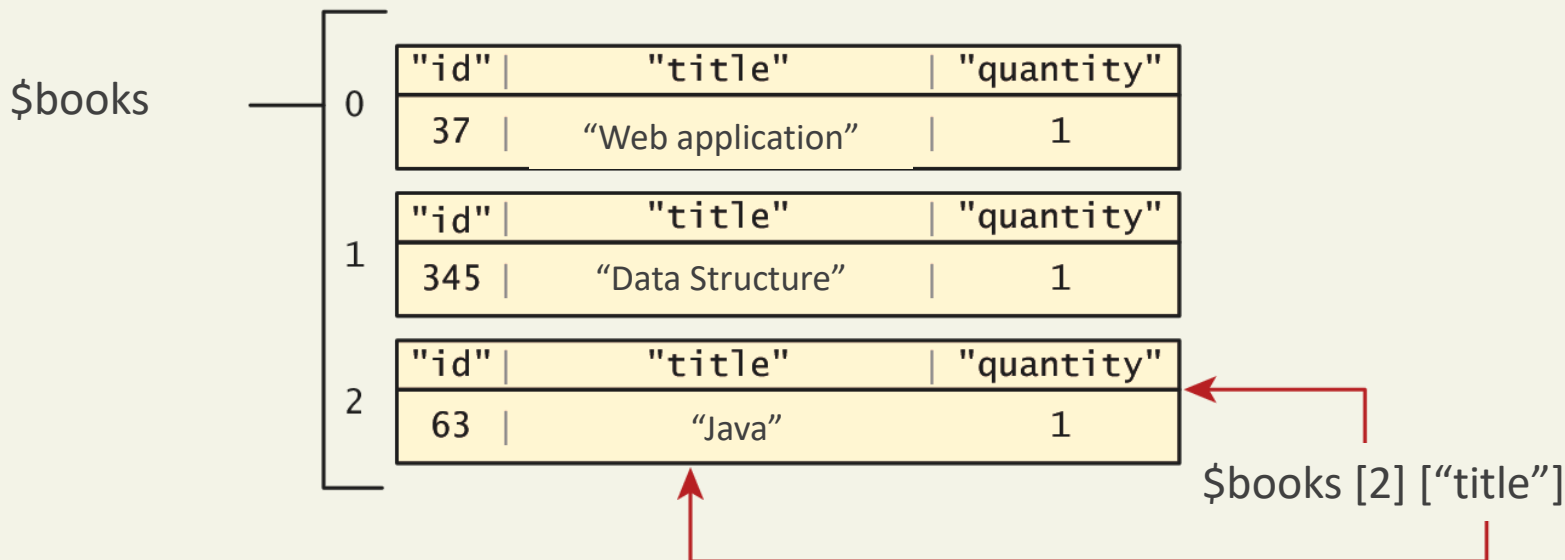Access

Multidimensional Arrays

Another example

```php
1  <?php
2  $books = array();
3  $books[] = array("id" => 37, "title" => "Web application", "quantity" => 1);
4  $books[] = array("id" => 345, "title" => "Data Structure", "quantity" => 1);
5  $books[] = array("id" => 63, "title" => "Java", "quantity" => 1);
6
7  echo $books[2]["title"]; // outputs java
8
9  ?>
```

$books

"id"	"title"	"quantity"
37	"Web application"	1

0

"id"	"title"	"quantity"
345	"Data Structure"	1

1

"id"	"title"	"quantity"
63	"Java"	1

2

$books [2] ["title"]

Multidimensional Arrays

Another example

```php
$products = array(
'pens' => array(
'ball' => "Ball Point",
'hilite' => "Highlighters",
'marker' => "Markers"),
'misc' => array(
'tape' => "Sticky Tape",
'glue' => "Adhesives",
'clips' => "Paperclips") );
echo $products['pens']['ball'];
```

Output: **Ball Point**

Uploaded By: Jibreel Bornat

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>";
}
```
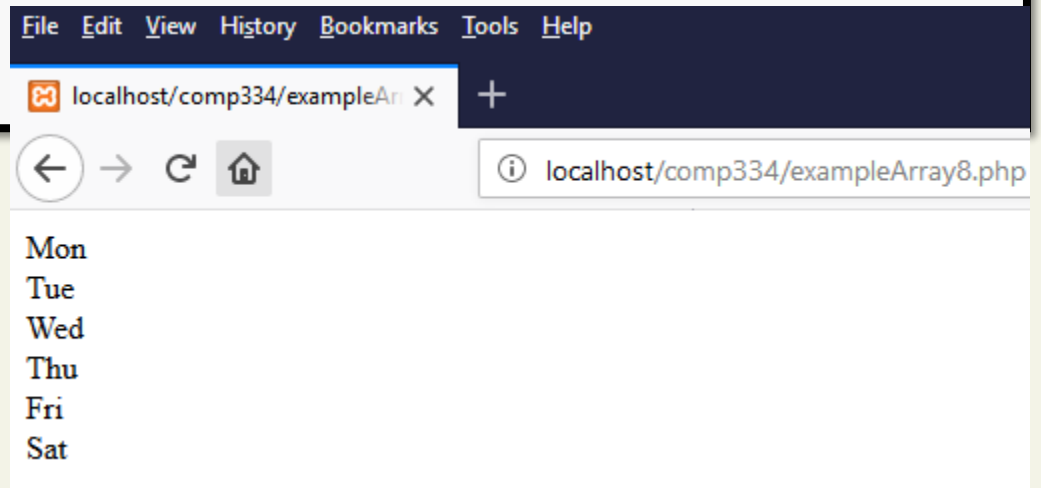
LISTING 9.2 Iterating through an array using while, do while, and for loops

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.

```php
1  <?php
2  $days = array("one"=>"Mon","two"=>"Tue","three"=>"Wed","four"=>"Thu","five"=>"Fri");
3  $days["six"] = "Sat";
4  // foreach: iterating through the values
5  foreach ($days as $value)
6   echo $value."<br>"
7  ?>
```
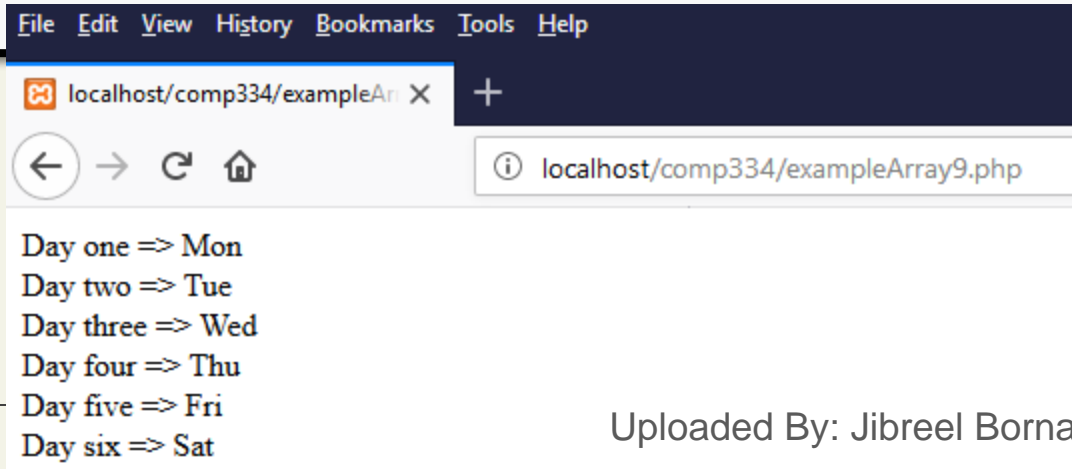
File   Edit   View   History   Bookmarks   Tools   Help

localhost/comp334/exampleAr ×   +

localhost/comp334/exampleArray8.php

Mon
Tue
Wed
Thu
Fri
Sat

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.

```php
1  <?php
2  $days = array("one"=>"Mon","two"=>"Tue","three"=>"Wed","four"=>"Thu","five"=>"Fri");
3  $days["six"] = "Sat";
4    // foreach: iterating through the values AND the keys
5  foreach ($days as $key=> $value)
6   echo "Day ".$key." => ".$value."<br>"
7  ?>
```

File Edit View History Bookmarks Tools Help

localhost/comp334/exampleAr ×   +

← → C ⌂   ⓘ localhost/comp334/exampleArray9.php

Day one => Mon
Day two => Tue
Day three => Wed
Day four => Thu
Day five => Fri
Day six => Sat

Uploaded By: Jibreel Bornat

Iterating through an array

Foreach loop to print multidimensional associative array

```
$myArray = array( 'PersonalInfo'=>array('Name'=>'AJ',
'Age'=>14,
'Sex'=>'M'),
'StudentInfo'=>array('school'=>'CUFE',
'city'=>'Bangalore',
'country'=>'India'
)
);
```

```
foreach($myArray as $a=>$b){
echo "My ". $a ." :";
foreach($b as $c=>$d){
echo "My ".$c. " is " .$d.". <br>";
}
}
```

Output/Result:

....

My PersonalInfo :

My Name is AJ.

My Age is 14.

My Sex is M.

My StudentInfo :

My school is CUFE.

My city is Bangalore.

My country is India.

Uploaded By: Jibreel Bornat

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
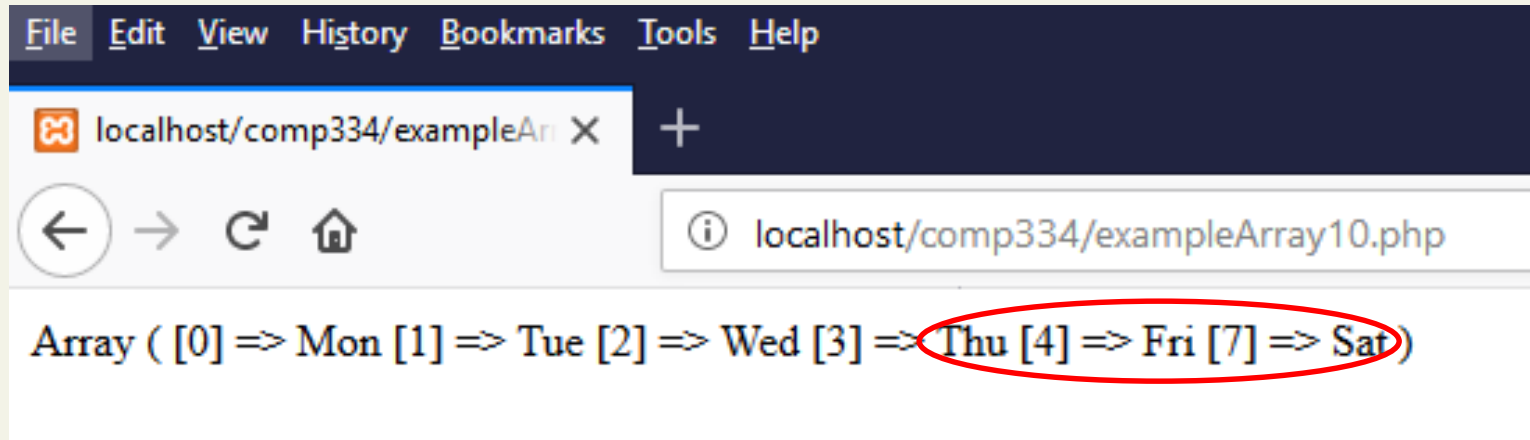
$days[ ] = "Sun";

Adding to an array

And quickly printing

PHP is more than happy to let you "skip" an index

```php
1    <?php
2    $days = array("Mon","Tue","Wed","Thu","Fri");
3    $days[7] = "Sat";
4    print_r($days);
5
6    ?>
```

File Edit View History Bookmarks Tools Help

localhost/comp334/exampleAr ✕   +

← → C ⌂   ⓘ localhost/comp334/exampleArray10.php

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

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**

```php
$oddKeys = array (1 => "hello", 3 => "world", 5 => "!");
if (isset($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)

https://www.w3schools.com/php/php_ref_array.asp

Uploaded By: Jibreel Bornat

Section **2** of 5

$_GET AND $_POST SUPERGLOBAL ARRAYS

Randy Connolly and Ricardo Hoar

Fundamentals of Web Development

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.**

```
protocol        status code
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 28 Nov 2009 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
X-Pingback: http://net.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent
```
HTTP headers as Name: Value

Superglobal Arrays

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

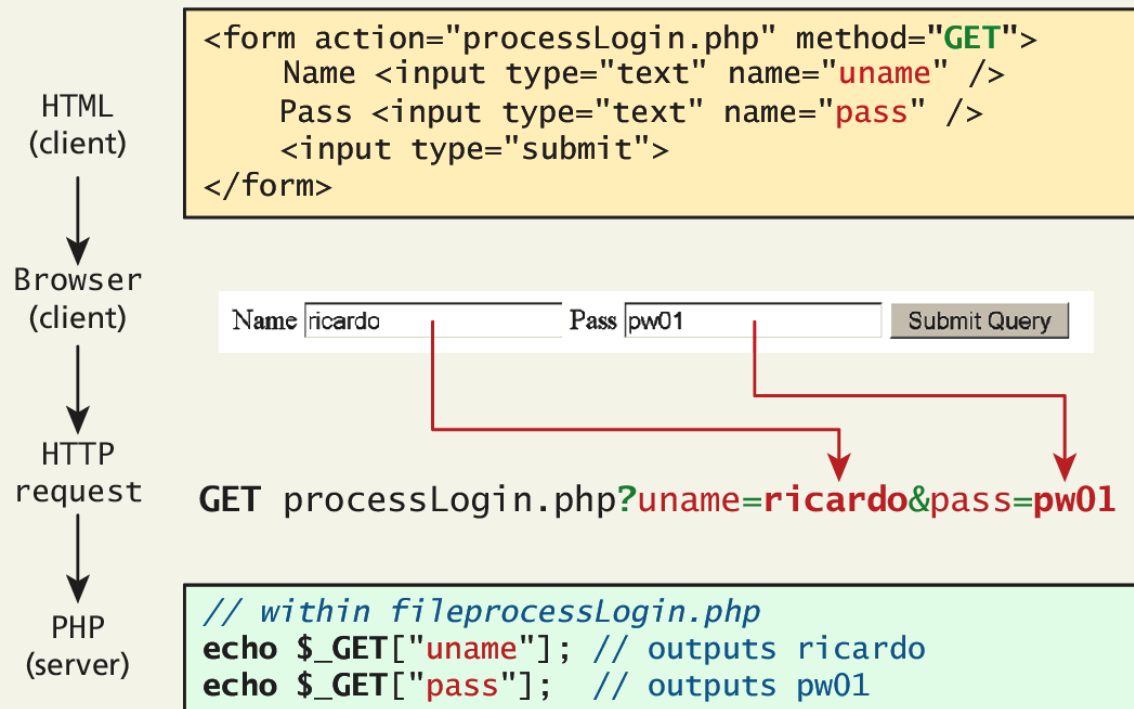The PHP superglobal variables are:

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

Superglobal Arrays

Predefined arrays of variables	Descriptions
$GLOBALS	Contains a reference to all variables which are currently available.
$_SERVER	Contains the predefined variables which are set by the web server.
$_GET	Contains the predefined variables which are provided to the script by a URL query string.
$_POST	Contains the predefined variables which are provided to the script via HTTP POST.
$_COOKIE	Contains the predefined variables which are provided by cookies.
$_FILES	Contains the predefined variables which are provided via HTTP post file uploads.
$_ENV	Contains the predefined variables which are provided to the script via the environment.
$_REQUEST	Contains the predefined variables which are provided to the script via the GET, POST and COOKIE mechanisms.
$_SESSION	Contains the predefined variables which are currently registered to a scripts session.

$_GET and $_POST

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.

HTML
(client)

```
<form action="processLogin.php" method="GET">
    Name <input type="text" name="uname" />
    Pass <input type="text" name="pass" />
    <input type="submit">
</form>
```

Browser
(client)

Name ricardo   Pass pw01   Submit Query

HTTP
request

GET processLogin.php?uname=ricardo&pass=pw01

PHP
(server)

```
// within fileprocessLogin.php
echo $_GET["uname"]; // outputs ricardo
echo $_GET["pass"];  // outputs pw01
```

$_GET and $_POST

Illustrative example

```
<!DOCTYPE HTML>
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Name: SobhiAhmed
E-mail: sahmed@birzeit.edu
Submit Query

welcome_get.php:

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```

Welcome SobhiAhmed
Your email address is: sahmed@birzeit.edu

The same result could also be achieved using the HTTP POST method

Don't forget to do form validation

```
1    <!DOCTYPE html>
2  ▼ <html>
3  ▼ <body>
4
5    <h2>Say Hello for the user</h2>
6
7  ▼ <form method="get" action="SayHello.php">
8        <label for="inputName">Name:</label>
9        <input type="text" name="name" id="inputName">
10       <input type="submit" value="Submit">
11   </form>
12
13   <p>If you click the "Submit" button, the form-data will be sent to a page called "SayHello.php".</p>
14
15   </body>
16   </html>
```
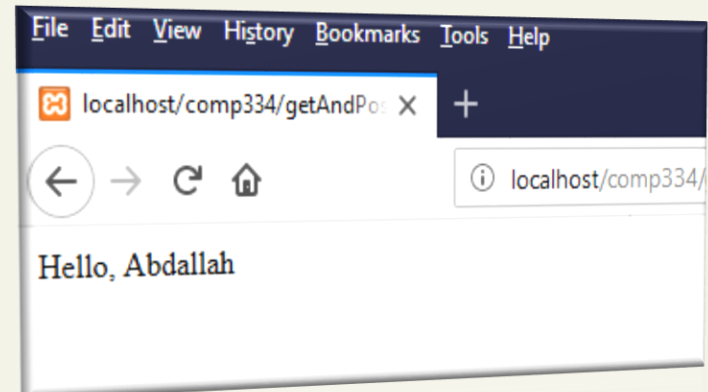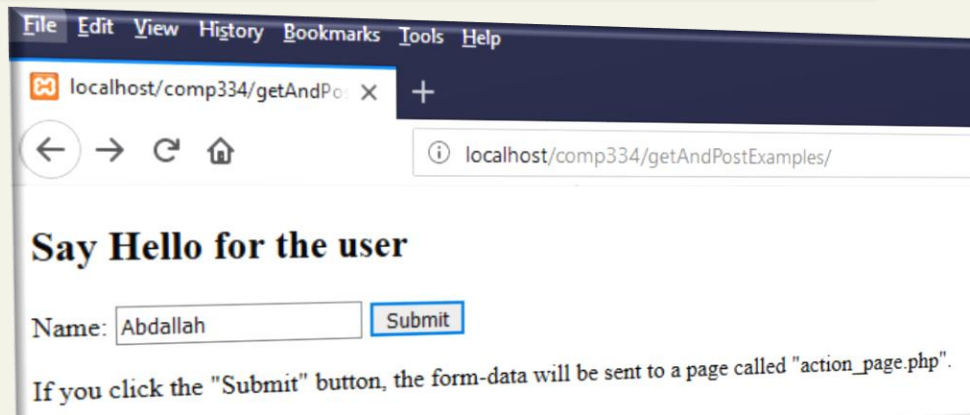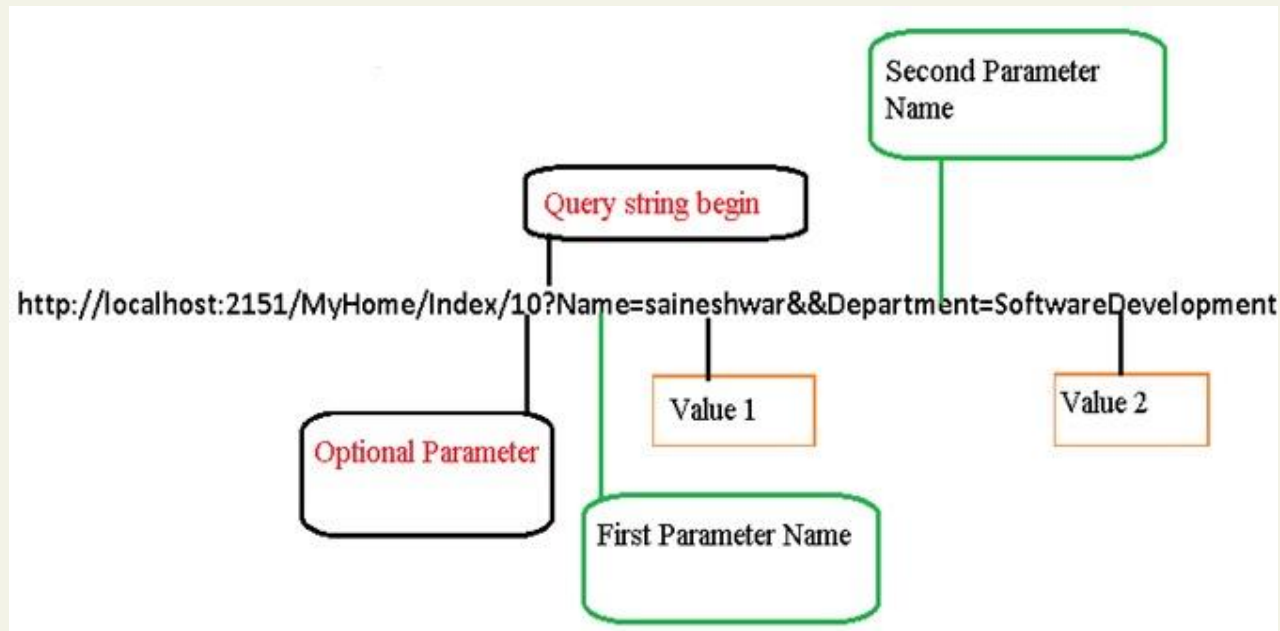
index.php

localhost/comp334/getAndPostExamples/example1/SayHello.php?name=Abdallah

SayHello.php

```php
<?php
if(isset($_GET["name"])){
    echo "<p>Hello, " . $_GET["name"] . "</p>";
}
?>
```

File Edit View History Bookmarks Tools Help

localhost/comp334/getAndPos ✕ +

localhost/comp334/getAndPostExamples/

Say Hello for the user

Name: Abdallah   Submit

If you click the "Submit" button, the form-data will be sent to a page called "action_page.php".

File Edit View History Bookmarks Tools Help

localhost/comp334/getAndPos ✕ +

localhost/comp334/

Hello, Abdallah

Randy Connolly and Ricardo Hoar

Uploaded By: Jibreel Bornat

Fundamentals of Web Development

Query String

Uploaded By: Jibreel Bornat

$_GET and $_POST

Sound familiar?

- **Get requests** parse query strings into the **$_GET array**

- **Post requests** are parsed into the **$POST array**

Determine if any data sent

```html
1   <!DOCTYPE html>
2 ▼ <html>
3 ▼ <body>
4
5   <h1>Some page that has a login form</h1>
6 ▼ <form action="samplePage.php" method="POST">
7   Name <input type="text" name="uname"/>
8   Pass <input type="password" name="pass"/><input type="submit">
9   </form>
10  </body>
11  </html>
```

```php
1   <?php
2 ▼ if ($_SERVER["REQUEST_METHOD"] == "POST") {
3 ▼     if ( isset($_POST["uname"]) && isset($_POST["pass"]) ) {
4           // handle the posted data.
5           echo "handling user login now ...<br/>";
6           echo "... here we could redirect or authenticate<br/> ";
7           echo " and hide login form or something else<br/>";
8       }
9   }
10  ?>
```

Determine if any data sent
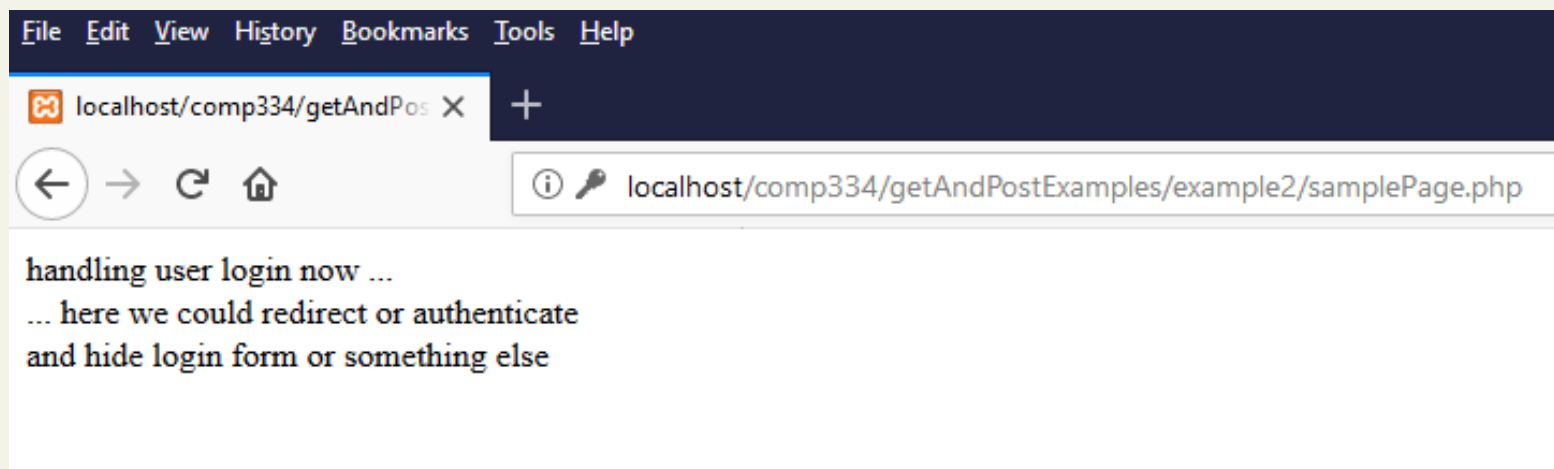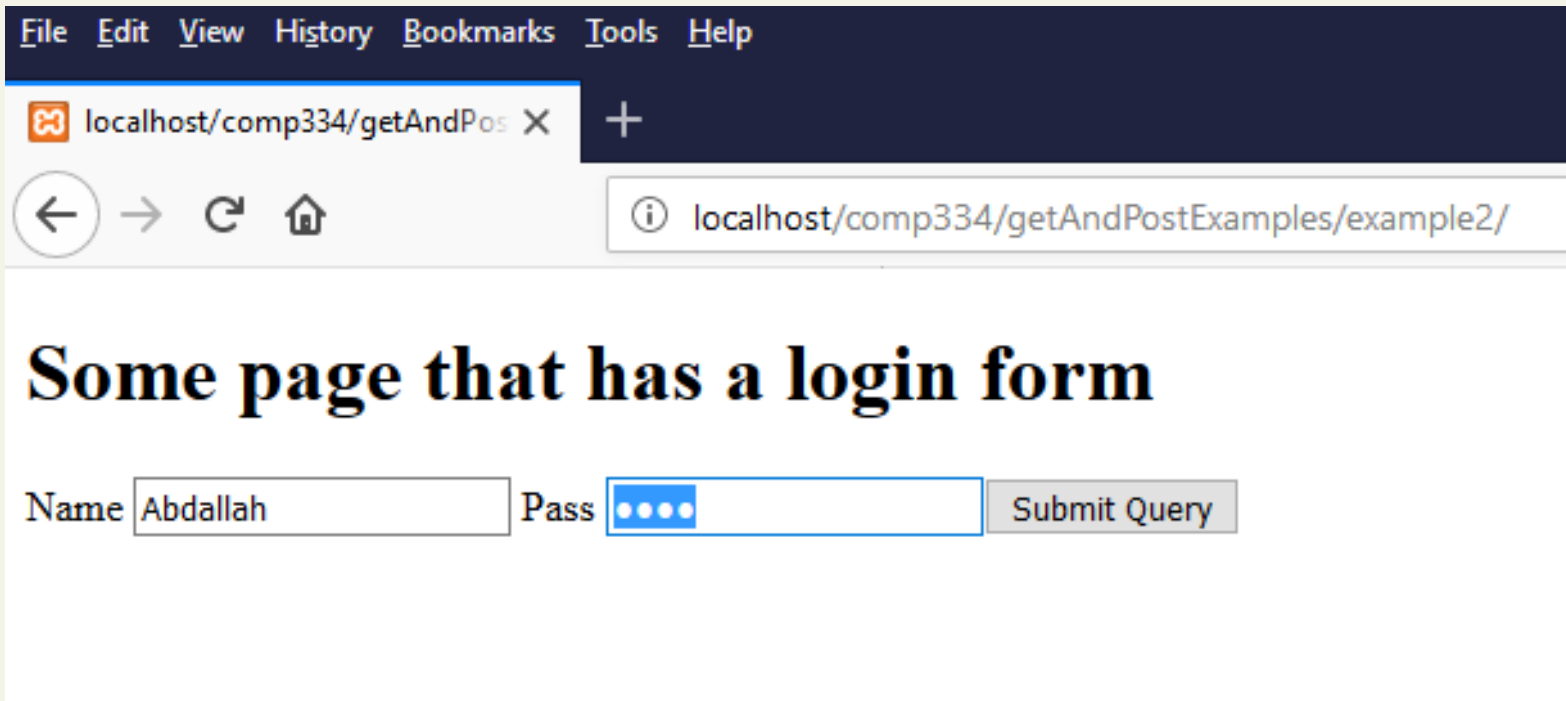
$_GET and $_POST

Sound familiar?

GET method :
- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data

`localhost/comp334/getAndPostExamples/example1/SayHello.php?name=Abdallah`

using POST method:
- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

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

Uploaded By: Jibreel Bornat

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

$_SERVER

$_SERVER['REQUEST_METHOD'] $_SERVER['SERVER_PROTOCOL']

$_SERVER['REQUEST_TIME'] —

$_SERVER['HTTP_USER_AGENT'] —

$_SERVER['HTTP_CONNECTION'] —

```
POST /page.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Accept-Encoding: gzip
Connection: Keep-Alive
…
<html> …
```

— $_SERVER['HTTP_HOST']

— $_SERVER['HTTP_ACCEPT_ENCODING']

HTTP Request Header

Web server

Server configuration files
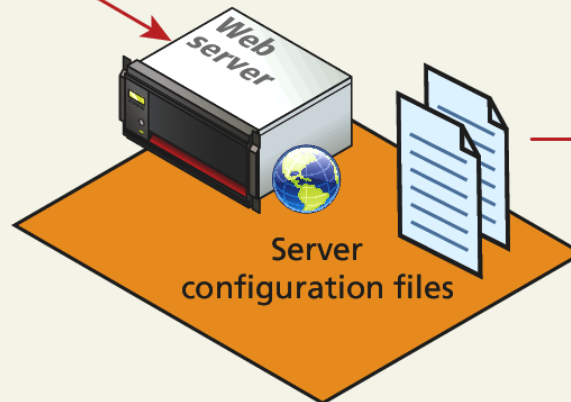
$_SERVER['SERVER_NAME']
$_SERVER['SERVER_ADDR']
$_SERVER['SERVER_PORT']
...

SERVER INFORMATION KEYS

- 'PHP_SELF' The filename of the currently executing script, relative to the document root. For instance, $_SERVER['PHP_SELF'] in a script at the address http://example.com/foo/bar.php would be /foo/bar.php.

```php
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name ;
    }
}
?>

</body>
</html>
```
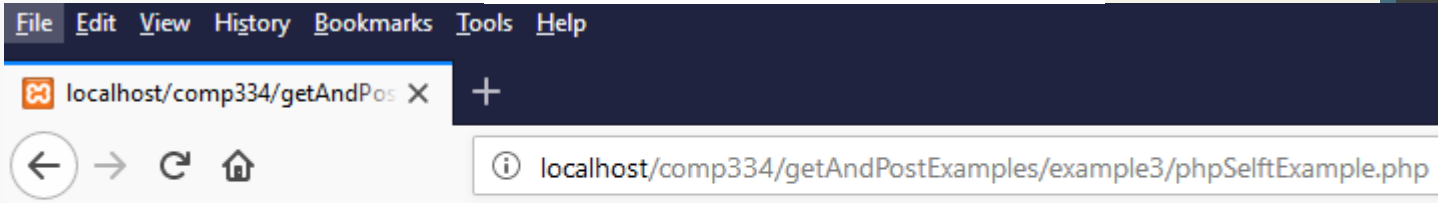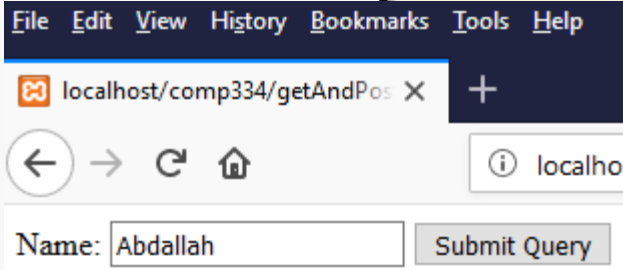
phpSelftExample.php

File Edit View History Bookmarks Tools Help

localhost/comp334/getAndPos X +

← → C ⌂   ⓘ localho

Name: Abdallah   Submit Query

File Edit View History Bookmarks Tools Help

localhost/comp334/getAndPos X +

← → C ⌂   ⓘ localhost/comp334/getAndPostExamples/example3/phpSelftExample.php

Name: [   ]   Submit Query

Abdallah

Cookies

Cookies are a client-side approach for persisting state information.

Cookies are a mechanism for **storing data on the client computer** by the remote browser.

Because the cookie will be **available the next time the web page is visite**d, cookies can be used to **track or identify return users to a web page**.

They are name=value pairs that are saved within one or more text files that are managed by the browser.

Cookies

How do they Work?

While cookie information is stored and retrieved by the browser, the information in a cookie travels within the HTTP header.

- The user can delete cookies or tamper with them

Creating a Cookie

- To create a cookie we need to use the **setcookie()** function:

```
bool setcookie (name , value , expire , path , domain, secure )
```

- With the setcookie() function all the arguments except the name argument are optional.

Creating a Cookie

Name	Description
Name	Name of the cookie file.
Value	Data to be stored into cookie file.
Expire	Date string that defines the valid life time of that cookie.
Path	Subset of URLs in a domain for which the cookie is valid.
Domain	The domain that the cookie is available on.
Secure	If set to "1" it will only be transmitted if the communications channel with the host is secure. (https).

Uploaded By: Jibreel Bornat

Using Cookies

Writing a cookie

```php
<?php
    // add 1 day to the current time for expiry time
    $expiryTime = time()+60*60*24;

    // create a persistent cookie
    $name = "Username";
    $value = "Ricardo";
    setcookie($name, $value, $expiryTime);
?>
```

LISTING 13.1 Writing a cookie

It is important to note that cookies must be written before any other page output.

Using Cookies

Reading a cookie

```php
<?php
    if( !isset($_COOKIE['Username']) ) {
        //no valid cookie found
    }
    else {
        echo "The username retrieved from the cookie is:";
        echo $_COOKIE['Username'];
    }
?>
```

LISTING 13.2 Reading a cookie
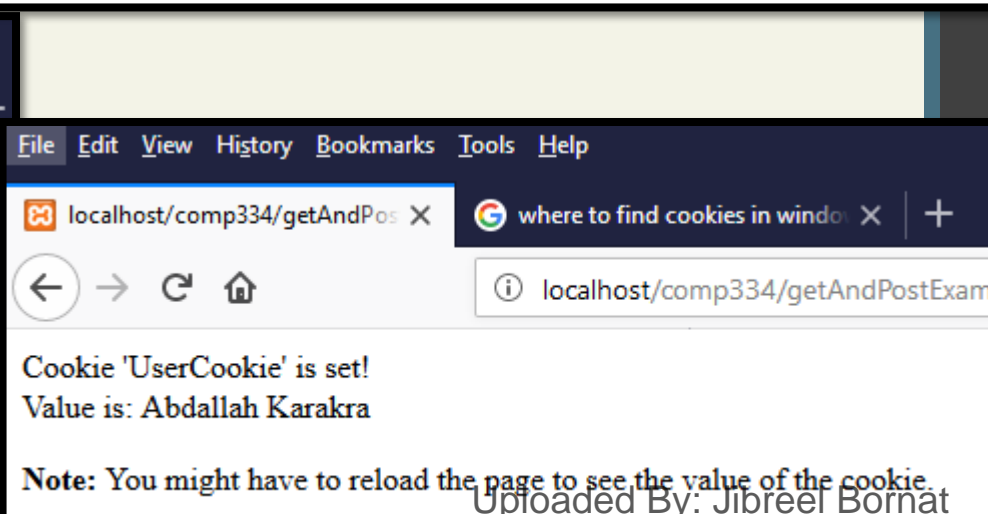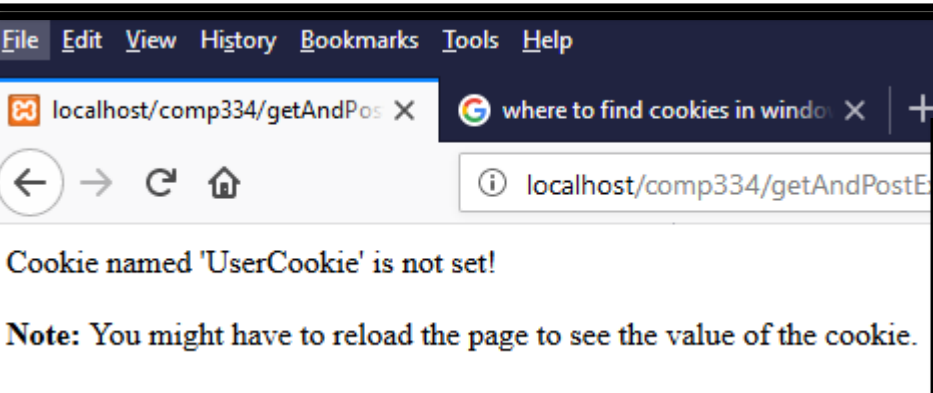
```php
<!DOCTYPE html>
<?php
$cookie_name = "UserCookie";
$cookie_value = "Abdallah Karakra";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>

</body>
</html>
```

File   Edit   View   History   Bookmarks   Tools   Help

localhost/comp334/getAndPos ✕   where to find cookies in windo ✕   +

← → C ⌂   ⓘ localhost/comp334/getAndPostE

Cookie named 'UserCookie' is not set!

Note: You might have to reload the page to see the value of the cookie.

File   Edit   View   History   Bookmarks   Tools   Help

localhost/comp334/getAndPos ✕   where to find cookies in windo ✕   +

← → C ⌂   ⓘ localhost/comp334/getAndPostExam

Cookie 'UserCookie' is set!
Value is: Abdallah Karakra

Note: You might have to reload the page to see the value of the cookie.

Randy Connolly and Ricardo Hoar

Deleting a Cookie

- Cookies can be deleted by simply using the **setcookie()** function with only the name of the cookie, for example:

```
setcookie ("TestCookie");
```

Sessions

- Sessions are similar to cookies in that they serve basically the same purpose; to **preserve some data between pages on a web site.**

- However sessions differ from cookie in that they are **stored on the server**.

- More secure

- Allow variables and their values to be stored **for each and every user**

Session State

Session state is ideal for storing more complex objects or data structures that are associated with a user session.

- In PHP, session state is available to the via the **$_SESSION** variable

- Must use session_start() to enable sessions.

Uploaded By: Jibreel Bornat

```php
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>


</body>
</html>
```

demo_session1.php

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>


</body>
</html>
```

demo_session2.php

Favorite color is green.
Favorite animal is cat.

Session State

Accessing State

```php
<?php

session_start();

if ( isset($_SESSION['user']) ) {
    // User is logged in
}
else {
    // No one is logged in (guest)
}
?>
```

LISTING 13.5 Accessing session state

The session_start() function must be the very first thing in your document. Before any HTML tags.

Session State

Checking Session existence

```php
<?php
include_once("ShoppingCart.class.php");

session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
    //session variables can be strings, arrays, or objects, but
    // smaller is better
    $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

LISTING 13.6 Checking session existence

Uploaded By: Jibreel Bornat