Chp:4
The Server Tier

Prepared by: Er. Simanta Kasaju

Web Server Concept

- Web server is a computer where the web content is stored.
- Web server is a computer system which receives the HTTP request via TCP, which are used to distribute information on www.
- Commonly a web browser or web crawler initiates communication by making HTTP request for a specific resource and the server response with the content of that resource or an error message if unable to do so.
- Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.
- Web site is collection of web pages while web server is a software that respond to the request for web resources.

Cont.

- The main function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the HTTP.
- Pages delivered are most frequently HTML documents, which may include images, stylesheets and scripts in addition to text content.
- Web servers are not always used for serving the www. They can also be found embedded in devices such as printers ,routers, webcams and serving only a local network.

Web Server Working

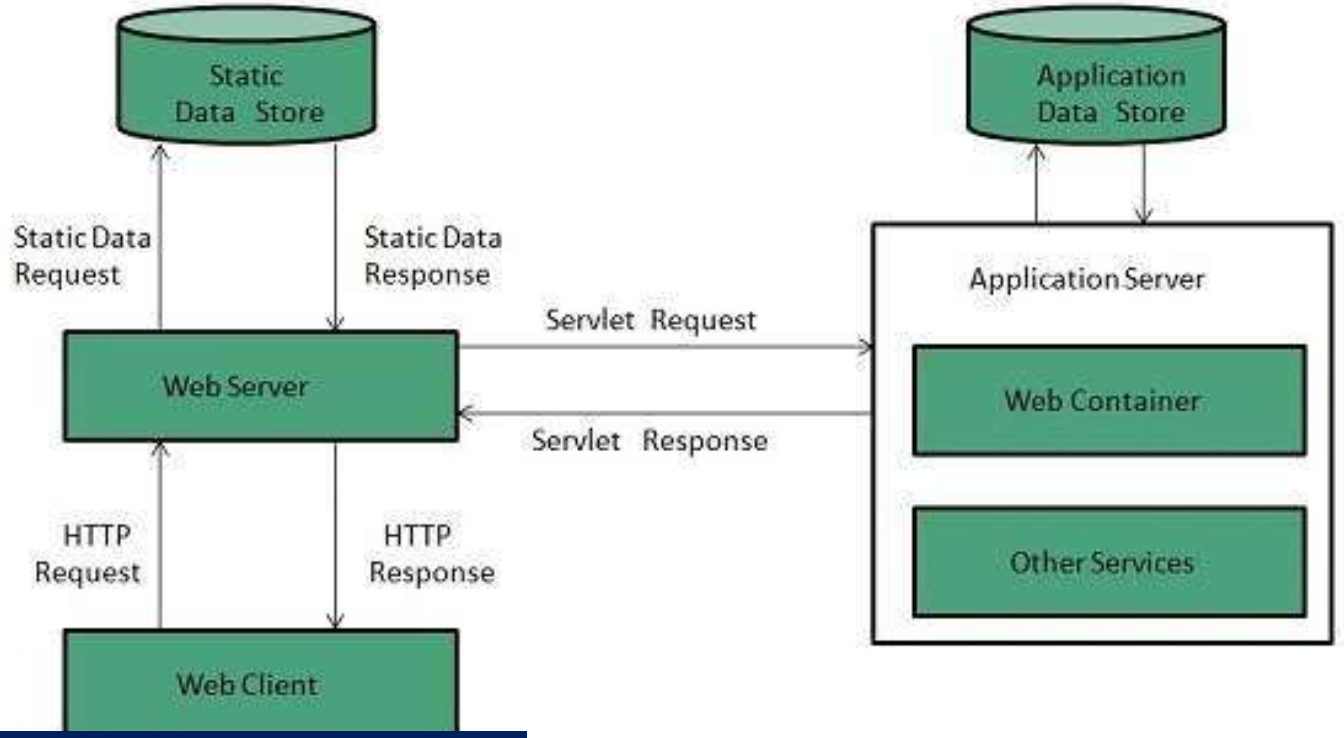
- Web server respond to the client request in either of the following two ways:
- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database

Key Points

•When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.

•If the requested web page is not found, web server will the send an HTTP response: Error 404 Not found.

•If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.



Examples: Following table describes the most leading web servers available today:

S.N.	Web Server Description
1	Apache HTTP Server This is the most popular web server in the world developed by the Apache Software Foundation. Apache web server is an open source software and can be installed on almost all operating systems including Linux, UNIX, Windows, FreeBSD, Mac OS X and more. About 60% of the web server machines run the Apache Web Server.
2.	Internet Information Services (IIS) The Internet Information Server (IIS) is a high performance Web Server from Microsoft. This web server runs on Windows NT/2000 and 2003 platforms (and may be on upcoming new Windows version also). IIS comes bundled with Windows NT/2000 and 2003; Because IIS is tightly integrated with the operating system so it is relatively easy to administer it.
3.	Lighttpd The lighttpd, pronounced lighty is also a free web server that is distributed with the FreeBSD operating system. This open source web server is fast, secure and consumes much less CPU power. Lighttpd can also run on Windows, Mac OS X, Linux and Solaris operating systems.
4.	Sun Java System Web Server This web server from Sun Microsystems is suited for medium and large web sites. Though the server is free it is not open source. It however, runs on Windows, Linux and UNIX platforms. The Sun Java System web server supports various languages, scripts and technologies required for Web 2.0 such as JSP, Java Servlets, PHP, Perl, Python, and Ruby on Rails, ASP and Coldfusion etc.
5.	Jigsaw Server Jigsaw (W3C's Server) comes from the World Wide Web Consortium. It is open source and free and can run on various platforms like Linux, UNIX, Wind written in Java and can run CGI scripts and PHP programs.

Load limits

- A web server has defined load limits. It can handle only a limited number of concurrent client connections and it can serve only a certain maximum number of requests per second depending on.
- Its own setting
- Hardware and software limitation of OS
- HTTP request type
- Content origin(static or dynamic) etc.

When a web server is near to or over its limits ,it becomes unresponsive.

Overload cause

At any time web servers can be overloaded because of

- Too much web traffic: when millions of clients connecting to the web site in a short interval overloaded can occur.
- Distributed Denial of service attacks
- Computers worms that sometimes cause abnormal traffic because of millions of infected computers
- Web servers partial unavailability. This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back end(eg database) failures etc, In these cases the remaining web servers get too much traffic and become overloaded.
- Internet connection slowdowns, so that client requests are served more slowly and the number of connections increases so much that server limits are reached

Anti-overloaded technique

To partially overcome load limits and to prevent overload we can use techniques like:

- Managing network traffic by using:
  - Firewalls to Block unwanted traffic coming from bad IP sources,
  - HTTP traffic managers to drop , redirect or rewrite requests having bad HTTP patterns
  - Bandwidth management and traffic shaping
- Deploying web cache techniques
  - Use different domains to serve different content (static and dynamic) by separate Web servers
  - Add more hardware resources
  - Use many web servers

Using control flow to control dynamic content generation

- Dynamic content in the context of HTML and the World Wide Web refers to website content that constantly or regularly changes based on user interactions, timing and other parameters that determine what content is delivered to the user.
- This means that the content of the site may differ for every user because of different parameters.
- Facebook is an excellent example of a site that delivers dynamic content, as every user gets different content based on their friends and social interactions, although the layout stays generally the same.
- This could be different text, video, advertisements or even an entirely different layout and color.
- Any element in a page which contains movement and can change over time may be considered as dynamic content.

There are two ways to provide dynamic content:

1. Using client-side scripting and frameworks such as JavaScript, AJAX and Bootstrap to change the UI behavior within a specific Web page in response to specific user actions and timings. This gives dynamic behavior to the UI presentation. This is normally used in Web applications and interactive websites.

Cont..

- 2. Using server-side scripting and processing to dynamically form both the layout and content to be delivered to the user based on parameters such as the user's location, time of day, browser being used or user preferences.
- Some good examples of this are social networking sites and content delivery sites.
- Social networking sites such as Facebook and Twitter provide entirely different content per user because of the difference of their connections and subscribed services, while sites like YouTube and Amazon provide dynamic content based on user-specific preferences based on past purchases or views; the server gives suggested items or content that the user may like based on historical data.

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:
- <?php
  // PHP code goes here
  ?>
- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

•Example

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

<h1>My first PHP
page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

sample

```
<html>
<body>
<?php
echo "I m studying php";
?>
</body>
</html>
```

```
<?php
$txt = "PHP";
echo "I m studying $txt";
?>
```

I m studying PHP

Comments in PHP

- A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.
- Comments can be used to:
- Let others understand your code
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code
- PHP supports several ways of commenting:

```
// This is a single-line comment

# This is also a single-line comment
```

```
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
```

PHP Variables

- Variables are "containers" for storing information
- In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php
$txt = "Hello
world!";
$x = 5;
$y = 10.5;
?>
```

PHP Variables
A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Output Variables

- The PHP echo statement is often used to output data to the screen.
- The following example will show how to output text and a variable:

```php
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

I love W3Schools.com!

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

9

```php
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
- local
- global
- static

Global and Local Scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

```php
<?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will
generate an error
  echo "<p>Variable x inside function is:
$x</p>";
}
myTest();

echo "<p>Variable x outside function is:
$x</p>";
?>
```

Variable x inside function is:

Variable x outside function is: 5

LOCAL SCOPE

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```
<?php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside
function is: $x</p>";
}
myTest();

// using x outside the function will
generate an error
echo "<p>Variable x outside function
is: $x</p>";
?>
```

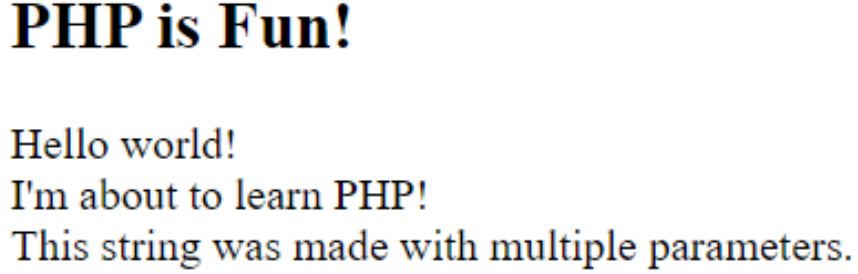
Variable x inside function is: 5

Variable x outside function is:

PHP echo and print Statements

- `echo` and `print` are more or less the same. They are both used to output data to the screen.
- The differences are small: `echo` has no return value while `print` has a return value of 1 so it can be used in expressions. `echo` can take multiple parameters (although such usage is rare) while `print` can take one argument. `echo` is marginally faster than `print`.

```
<?php
echo "<h2>PHP is
Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn
PHP!<br>";
echo "This ", "string ", "was
", "made ", "with multiple
parameters.";
?>
```



PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

```
<?php
$A="are u learning";
echo var_dump($A);
?>
```

```
string(14) "are u learning"
```

PHP Integer

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- Rules for integers:
- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

```php
<?php
$x = 5985;
var_dump($x);
?>
```

```
int(5985)
```

PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

```
float(10.365)
```

PHP Boolean

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

```
$A=true;
echo var_dump($A);
```

PHP Array

- An array stores multiple values in one single variable.
- In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<?php
$cars =
array("Volvo","BMW","Toy
ota");
var_dump($cars);
?>
```

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }
```

Array Creation

- Assigning a value to an array that does not yet exist, creates the array

```
$list[0] =17;

$list[2]="This is a nice day";

$list[]=33;  #this element key will be 3
```

the array construct

```
$list2 = array(12, 34, 45, 56);
   /* traditional array with 4 elements
      with the keys 0, 1, 2, 3 */

$list3 = array( 1 => 12, 3 => 34, 4 => 45, 7 => 56);
   /* an array with four elements
      with the keys 1, 3, 4, 7 */

$list4 = array();  # an empty array

$list5 = array("Tom" => 13, "Dick" => 18, "Jerry" =>
24);
   /* an array with three elements
      with the keys "Tom", "Dick", "Jerry" */
$list6 = array( "make" => "volvo", "model" => "245",
          "year" => 1974, 3 => "Sold" );
   /* PHP arrays can be mixtures of
      traditional arrays and hashes */
```

Accessing Array Elements

- Array elements are accessed by subscripting

```
$val = $list3[3];  # $val has a value of 34
$val = $list5['Tom']; # $val is 13
$val = $list6[3]; # $val is "Sold"
```

- The `list` construct works just like in `Perl`
- The `array_keys` function takes an array and returns the array keys

```
$keys6 = array_keys($list6);
/* $keys6 holds the keys of $list6 */
$valu6 = array_values($list6);
   /* $valu6 holds the values of $list6 */
```

```php
<?php
//with array keyword
   $color = array("red", "yellow", "green");
   echo "<pre>";
   print_r($color);
   // echo "<br>";
   echo "</pre>";

   //without array keyword
   echo "<pre>";
   $vehicle = ["car", "truck", "bus"];
   print_r($vehicle);
   echo "</pre>";

   //with loop
   $hobby=array("music","cricket","dance","travel");
   for($i=0;$i<4;$i++)
   {
      echo $hobby[$i] . "<br>";
   }
?>
```

```
Array
(
    [0] => red
    [1] => yellow
    [2] => green
)

Array
(
    [0] => car
    [1] => truck
    [2] => bus
)
```

music
cricket
dance
travel

Type of Array in PHP:

In PHP, there are three types of arrays:

i. **Indexed arrays** - Arrays with a numeric index
ii. **Associative arrays** - Arrays with named keys
iii. **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

To loop through and print all the values of an associative array, you could use a foreach loop as below:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

```
foreach ($array as $value) {
  code to be executed;
}
```

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
 echo "$value <br>";
}
?>
```

red
green
blue
yellow

```
$sub = array("dbms", "Wt", "LC", "NM", "Java");
 foreach($sub as  $value)
 {
    echo $value ."<br>";
 }
```

dbms
Wt
LC
NM
Java

```
echo "<ul>";
  $sub = array("A"=>"dbms",
"B"=>"Wt","C"=> "LC","D"=> "NM","E"=>
"Java");
  foreach($sub as $x=> $value)
  {
     echo  "<li>$x= $value </li>";
  }
```

- A= dbms
- B= Wt
- C= LC
- D= NM
- E= Java

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays.

For example:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array
 (
 array("Volvo",22,18),
 array("BMW",15,13),
 array("Saab",5,2),
 array("Land Rover",17,15)
 );
```

We can also put a `for` loop inside another `for` loop to get the elements of the $cars array as below:

```
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

```php
<?php
   $Family = [
[1,"simanta","kasaju",28],
[2,"Prashant","kasaju",32],
[3,"shashi","kasaju",45],
[4,"salina","kasaju",34]
   ];
   echo "<table border='2px'>";

for($i=0;$i<4;$i++){
      echo "<tr>";
   for($j=0;$j<4;$j++)
   {
         echo "<td>" .$Family[$i][$j]. "</td>";
   }
      echo "</tr>";
}
?>
```

1	simanta	kasaju	28
2	Prashant	kasaju	32
3	shashi	kasaju	45
4	salina	kasaju	34

Using foreach

```php
foreach($Family as $value1){
      echo "<tr>";
   foreach($value1 as $value2)
   {
         echo "<td>" .$value2. "</td>";
   }
      echo "</tr>";
}
```

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

In_array()

```php
<?php
$name=[1,2,'sim',5,'ram','shyam',9];
if(in_array('sim',$name)){
    echo "yes sim is present in the array";
}
else
{
    echo "not present";
}
?>
```

```php
<?php
$name=[1,2,'sim',5,'ram','shyam',9];
if(in_array('9',$name,true)){
    echo "yes sim is present in the array";
}
else
{
    echo "not present";
}
?>
```

strrev()

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

<?php
echo strrev("Hello world!");
?>

</body>
</html>
```

array_search()

It will return index value in case of indexed array and return key value in case of associative array

```php
<?php
$name=[1,2,'sim',5,'ram','shyam',9];
echo array_search('sim',$name);
echo"<br>";
if(array_search('sim',$name)){
    echo "yes sim is present in the array";
}
else
{
    echo "not present";
}
?>
```

array_replace()

```php
<?php
$name=['ram','shyam','sim','ham','shyam'];
$b=['apple','ball','cat'];
$c=array_replace($name,$b);
print_r ($c);
?>
```

array_push()

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

<?php
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);
?>

</body>
</html>
```

array_pop()

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

<?php
$a=array("red","green","blue");
array_pop($a);
print_r($a);
?>

</body>
</html>
```

array_reverse()

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

<?php
$a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota");
print_r(array_reverse($a));
?>

</body>
</html>
```

array_keys()

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

<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");
print_r(array_keys($a));
?>

</body>
</html>
```

array_count_values()

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

<?php
$a=array("A","Cat","Dog","A","Dog");
print_r(array_count_values($a));
?>

</body>
</html>
```

array_combine()

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

<?php
$fname=array("Peter","Ben","Joe");
$age=array("35","37","43");
$c=array_combine($fname,$age);
print_r($c);
?>

</body>
</html>
```

array_shift()

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

<?php
$a=array("a"=>"red","b"=>"green","c"=
>"blue");
echo array_shift($a)."<br>";
print_r ($a);
?>

</body>
</html>
```

array_slice()

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

<?php
$a=array("red","green","blue","yellow","brown");
print_r(array_slice($a,2));
?>

</body>
</html>
```

array_splice()

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

<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow")
;
$a2=array("a"=>"purple","b"=>"orange");
array_splice($a1,0,2,$a2);
print_r($a1);
?>

</body>
</html>
```

array_sum()

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

<?php
$a=array(5,15,25);
echo array_sum($a);
?>

</body>
</html>
```

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

<?php
$a=array("a"=>52.2,"b"=>13.7,"c"=
>0.9);
echo array_sum($a);
?>

</body>
</html>
```

array_unshift()

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

<?php
$a=array("a"=>"red","b"=>"green");
array_unshift($a,"blue");
print_r($a);
?>

</body>
</html>
```

Sizeof()

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

<?php
$cars=array("Volvo","BMW","Toyota");
echo sizeof($cars);
?>

</body>
</html>
```

sort()

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

<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

foreach ($cars as $key => $val) {
   echo "cars[" . $key . "] = " . $val . "<br>";
}
?>

</body>
</html>
```

Delete element from an array

```php
<?php

$A=[1,2,4,6,7,8,9];
array_splice($A,5,1);
print_r($A);
?>
```

Delete element from an array

```php
<?php

$A=[1,2,3,4,5,6,7,8];
unset($A[3]);
print_r($A);
?>
```

Multiple element delete from Associative array

```php
<?php
$array = [0 => "a", 1 => "b", 2 => "c", 3 =>
"D"];
$array = array_diff($array, ["a", "c"]);
print_r($array);
?>
```

```php
<?php

$A=[1,2,3,4,5,6,7,8];
$b=array_pop($A);
print_r($A);
print($b);
?>
```

PHP - Sort Functions For Arrays

In this chapter, we will go through the following PHP array sort functions:

- `sort()` - sort arrays in ascending order
- `rsort()` - sort arrays in descending order
- `asort()` - sort associative arrays in ascending order, according to the value
- `ksort()` - sort associative arrays in ascending order, according to the key
- `arsort()` - sort associative arrays in descending order, according to the value
- `krsort()` - sort associative arrays in descending order, according to the key

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x < $clength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>
```

BMW
Toyota
Volvo

```
<?php
$original = array("Simanta"=> 29, "Zunet" =>
20, "Prince" =>
"Charley","Apple" => 40, "Elaphent" =>
"hobby" );
?>
<h4>Original Array</h4>
<?php
foreach ($original as $key => $value )
print("[$key] => $value <br />");
$new = $original;
sort($new);
?>
<h4>Array Sorted by <tt>sort</tt></h4>
<?php
foreach ($new as $key => $value )
print("[$key] => $value <br />");
$new = $original;
sort($new, SORT_NUMERIC);
?>
<h4>Array Sorted by <tt>sort,
SORT_NUMERIC</tt></h4>
```

```
<?php
foreach ($new as $key => $value )
print("[$key] => $value <br />");
$new = $original;
rsort($new);
?>
<h4>Array Sorted by <tt>rsort</tt></h4>
<?php
foreach ($new as $key => $value )
print("[$key] => $value <br />");
$new = $original;
asort($new);
?>
<h4>Array Sorted by <tt>asort</tt></h4>
<?php
foreach ($new as $key => $value )
print("[$key] => $value <br />");
$new = $original;
arsort($new);
?>
<h4>Array Sorted by <tt>arsort</tt></h4>
<?php
foreach ($new as $key => $value )
print("[$key] => $value <br />");
?>
```

Original Array

[Simanta] => 29
[Zunet] => 20
[Prince] => Charley
[Apple] => 40
[Elaphent] => hobby

Array Sorted by `sort`

[0] => Charley
[1] => hobby
[2] => 20
[3] => 29
[4] => 40

Array Sorted by `sort, SORT_NUMERIC`

[0] => Charley
[1] => hobby
[2] => 20
[3] => 29
[4] => 40

Array Sorted by `rsort`

[0] => 40
[1] => 29
[2] => 20
[3] => hobby
[4] => Charley

Array Sorted by `asort`

[Prince] => Charley
[Elaphent] => hobby
[Zunet] => 20
[Simanta] => 29
[Apple] => 40

Array Sorted by `arsort`

[Apple] => 40
[Simanta] => 29
[Zunet] => 20
[Elaphent] => hobby
[Prince] => Charley

PHP Object

- Classes and objects are the two main aspects of object-oriented programming.
- A class is a template for objects, and an object is an instance of a class.
- When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

```php
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

My car is a black Volvo!

My car is a red Toyota!

PHP NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- **Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = null;
var_dump($x);
?>
</body>
</html>
```

NULL

PHP Resource

- The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.
- A common example of using the resource data type is a database call.
- We will not talk about the resource type here, since it is an advanced topic.

Functions

General Characteristics of Functions:

- The general form of a PHP function definition is as follows:
  function *name*([*parameters*]) {
  ...
  }
- Function definitions can be nested, as they can in JavaScript.
- Function names are not case sensitive.
- A document cannot have a function named `sum` and another named `Sum`. The PHP interpreter will see
  them as the same function and issue an error message stating that the document has two definitions for the same function.
- The `return` statement is used in a function to specify the value to be returned to the caller. Function execution ends when a `return` statement is encountered or the last statement in the function has been executed. In either case, control returns to the caller. If no `return` statement was executed, no value is returned.
- If one or more related functions are used by more than one document, it is convenient to store their definitions in a separate file and copy that file into those documents when they are requested by a client (browser). This is done with the `include` function.

Parameters:

An actual parameter can be any expression. A formal parameter must be a variable name.

- The number of actual parameters in a call to a function need not match the number of formal parameters defined in that function.
- If there are too few actual parameters in a call, the corresponding formal parameters will be unbound variables.
- If there are too many actual parameters, the excess actual parameters will be ignored. The absence of a requirement for matching numbers of parameters allows the language to support functions with a variable number of parameters.
- The default parameter-passing mechanism of PHP is pass by value.
- This means that, in effect, the values of actual parameters are copied into the memory locations associated with the corresponding formal parameters in the called function. The values of the formal parameters are never copied back to the caller, so passing by value implements one-way communication to the function. This is the most commonly needed mechanism for parameter passing.
- For example:

```php
<?php
function marks($score)
{
    $sum = 0;
    foreach($score as $value)
    {
        $sum += $value;
    }
    return $sum;
}
$sim=["Nepali"=>40,"English"=>70,"math"=>100,"Opt"=>100];
$total = marks($sim);
echo "you have scored $total marks out of 400";
?>
```

you have scored 310 marks out of 400

PHP type cast

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators(The PHP array operators are used to compare arrays.)

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

PHP Constants

```
Case insensitive
define("GREETING", "Welcome to
W3Schools.com!", true);
```

- PHP Constants
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- **Note:** Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

- To create a constant, use the `define()` function.
- Syntax
- define(*name*, *value*, *case-insensitive*)
- Parameters:
- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

```
Case sensitive
<?php
define("GREETING", "
Welcome to
W3Schools.com!");
echo GREETING;
?>
```

Welcome to W3Schools.com!

PHP Constant Arrays

- In PHP7, you can create an Array constant using the `define()` function.

```
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[2];
?>
```

Toyota

PHP Conditional Statements

- Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.
- In PHP we have the following conditional statements:
- `if` statement - executes some code if one condition is true
- `if...else` statement - executes some code if a condition is true and another code if that condition is false
- `if...elseif...else` statement - executes different codes for more than two conditions
- `switch` statement - selects one of many blocks of code to be executed

PHP - The if Statement

- The `if` statement executes some code if one condition is true.
- Syntax
- if (*condition*) {
  code to be executed if condition is true;
  }

```
<?php
$t = 40;

if ($t <"45") {
  echo "congratulation";
?>
```

congratulation

PHP - The if...else Statement

- The `if...else` statement executes some code if a condition is true and another code if that condition is false.
- Syntax
- if (*condition*) {
  code to be executed if condition is true;
  } else {
  code to be executed if condition is false;
  }

```
<?php
$t = 40;

if ($t >"45") {
  echo "congratulation";
}
else
echo "sorry ";
?>
```

sorry

PHP - The if...elseif...else Statement

- The `if...elseif...else` statement executes different codes for more than two conditions.
- Syntax
- if (*condition*) {
  code to be executed if this condition is true;
  } elseif (condition) {
  code to be executed if first condition is false and this condition is true;
  } else {
  code to be executed if all conditions are false;
  }

```
<?php
$t = 40;

if ($t >"90") {
  echo "A+";
}
else if($t <"85" &&
$t>"80"){
echo "B+ ";}
else
echo "fail";
?>
```

```
fail
```

The PHP switch Statement

- Use the `switch` statement to **select one of many blocks of code to be executed**.
- Syntax
- switch (*n*) {
  case *label1:*
  code to be executed if n=label1;
  break;
  case label2:
  code to be executed if n=label2;
  break;
  case label3:
  code to be executed if n=label3;
  break;
  ...
  default:
  code to be executed if n is different from all labels;
  }
- This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use `break` to prevent the code from running into the next case automatically. The `default` statement is used if no match is found.

```php
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red,
blue, nor green!";
}
?>
```

Your favorite color is red!

PHP Loops

- Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.
- Loops are used to execute the same block of code again and again, as long as a certain condition is true.
- In PHP, we have the following loop types:
- `while` - loops through a block of code as long as the specified condition is true
- `do...while` - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- `for` - loops through a block of code a specified number of times
- `foreach` - loops through a block of code for each element in an array
- The following chapters will explain and give examples of each loop type.

The PHP while Loop

- The `while` loop executes a block of code as long as the specified condition is true.
- Syntax
- while (*condition is true*) {
  code to be executed;
  }

```
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

```
<?php
$x = 0;

while($x <= 100) {
  echo "The number is: $x
<br>";
  $x+=10;
}
?>
```

```
The number is: 0
The number is: 10
The number is: 20
The number is: 30
The number is: 40
The number is: 50
The number is: 60
The number is: 70
The number is: 80
The number is: 90
The number is: 100
```

The PHP do...while Loop

- The `do...while` loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.
- Syntax
- do {
  code to be executed;
  } while (*condition is true*);

```php
<?php
$x = 1;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

```php
<?php
$x = 6;

do {
 echo "The number is: $x <br>";
 $x++;
} while ($x <= 5);
?>
```

The number is: 6

The PHP for Loop

- The `for` loop is used when you know in advance how many times the script should run.
- Syntax
- for (*init counter; test counter; increment counter*) {
  code to be executed for each iteration;
  }
- Parameters:
- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

```php
<?php
for ($x = 0; $x <= 10; $x++)
{
  echo "The number is: $x
<br>";
}
?>
```

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

```php
<?php
for ($x = 0; $x <= 100;
$x+=10) {
  echo "The number is: $x
<br>";
}
?>
```

```
The number is: 0
The number is: 10
The number is: 20
The number is: 30
The number is: 40
The number is: 50
The number is: 60
The number is: 70
The number is: 80
The number is: 90
The number is: 100
```

Some Program

Counting no of digits

```
<?php
$a=12345;
$count=0;
do
{
$a=floor($a/10);
$count++;
}while($a!=0);
echo "$count";

?>
```

Sum of digits

```
<?php
$a=12342;
$sum=0;
do
{
    $rem=$a%10;
    $sum+=$rem;
$a=floor($a/10);
}while($a!=0);
echo $sum;
```

Write a PHP program for following

Factorial of number

```php
<?php
$num = 4;
$factorial = 1;
for ($x=$num; $x>=1; $x--)
{
$factorial = $factorial * $x;
}
echo "Factorial of $num is $factorial";
?>
```

```php
<?php
function palindrome($n){
$number = $n;
$sum = 0;
while(floor($number)) {
$rem = $number % 10;
$sum = $sum * 10 + $rem;
$number = $number/10;
}
return $sum;
}
$input = 1235321;
$num = palindrome($input);
if($input==$num){
echo "$input is a Palindrome number";
} else {
echo "$input is not a Palindrome";
}
?>
```

```php
<?php
$num = 0;
$n1 = 0;
$n2 = 1;
echo "<h3>Fibonacci series for first 12 numb
ers: </h3>";
echo "\n";
echo $n1.' '.$n2.' ';
while ($num < 10 )
{
$n3 = $n2 + $n1;
echo $n3.' ';
    $n1 = $n2;
    $n2 = $n3;
    $num = $num + 1;
?>
```

```php
<?php
$a = 45;
$b = 78;
// Swapping Logic
$third = $a;
$a = $b;
$b = $third;
echo "After swapping:<br><br>";
echo "a =".$a."  b=".$b;
?>
```

```php
<?php
 $base = 10;
 $height = 15;
 echo "area with base $base and height $height= " . ($bas
e * $height) / 2;
 ?>
```

```php
<?php
for($i=0;$i<=5;$i++){
for($j=5-$i;$j>=1;$j--){
echo "* ";
}
echo "<br>";
}
?>
```

```
* * * * *
* * * *
* * *
* *
*
```

```php
<?php
for($i=0;$i<=5;$i++){
for($j=1;$j<=$i;$j++)
echo "* ";
}
echo "<br>";
}
?>
```

```
*
* *
* * *
* * * *
* * * * *
```

```php
<?php
for ($i=1; $i<=5; $i++)
{
 for ($j=1; $j<=5; $j++)
 {
   echo '* ';
  }
   echo "</br>";
}
?>
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Session & State

- Web is stateless that means the web browser does not know who you are and what you do,because the HTTp address doesn’t maintain state.If user inserts some information and move to the next page that data will be lost and user would not able to retrive the information.so we need to store the information about user.
- Session provides that facility to store information on server memory.
- A session is a way to store information(in variables) to be used across multiple pages.
- Session variables hold information about one single user and are available to all pages in one application.
- By default session variables last until the user closes the browser.

Cont..

- Unlike a cookie, the information is not stored on the users computer.
- We all know that the web uses the HTTP protocol and the HTTP protocol is a stateless protocol; in other words, when a client sends a request to the server, an instance of the page is created and the page is converted to HTML format and then the server provides the response and then the instance of the page and the value of the control are destroyed.
- So if we have a requirement to store the values of the controls and pass them into another web form then a State Management Technique is used.

Cont..

- Session is a State Management Technique. A Session can store the value on the Server.
- It can support any type of object to be stored along with our own custom objects.
- A session is one of the best techniques for State Management because it stores the data as client-based, in other words the data is stored for every user separately and the data is secured also because it is on the server.

Cont..

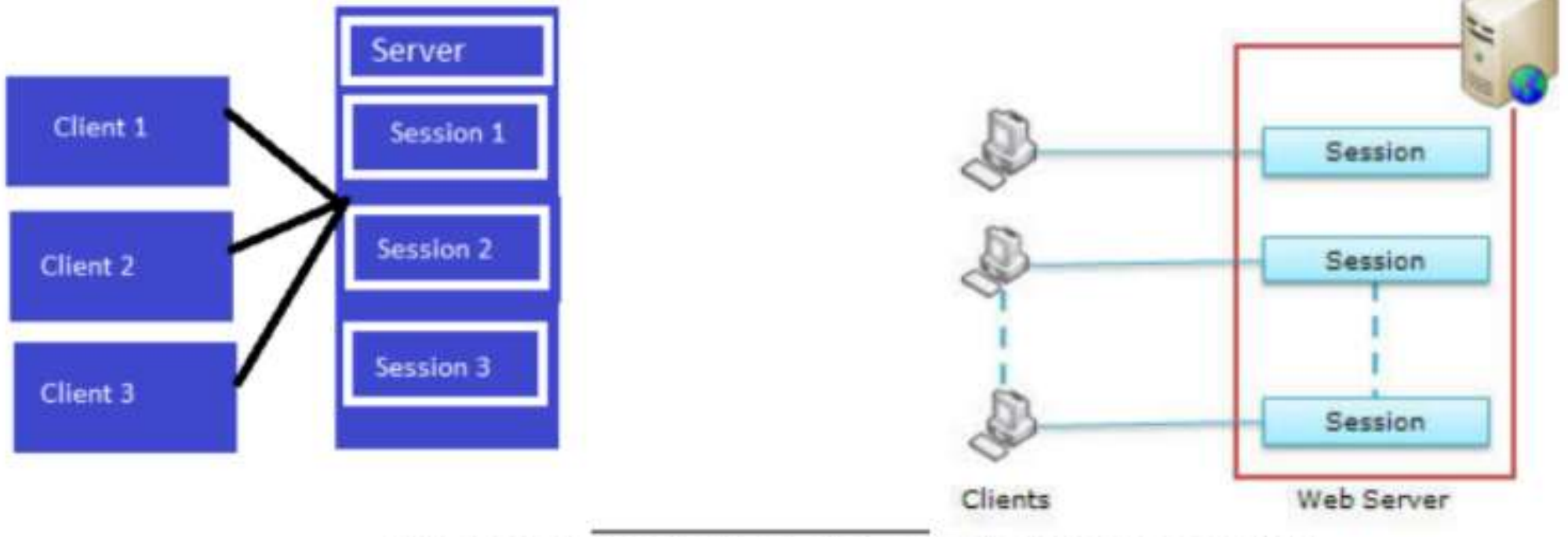


Fig: For every client, session data is stored separately

Cont..

- State management using session is one of the best features, because it is secure, transparent from users, and we can store any kind of object in it.
- Along with these advantages, some times session can cause performance issues in high traffic sites because it is stored in server memory and clients read data from the server.

Advantages

- It helps maintain user state and data all over the application.
- It is easy to implement and we can store any kind of object eg. Database, dataset etc.
- Stores client data separately.
- Session is secure and transparent from the user.

Disadvantages

- As session is stored on the server memory, it is not advisable to use session state when there is a large volume of data.
- With the use of session state, it will affect the performance of memory, because it is stored in server memory until you destroy the state
- The problem with sessions is that when you close your browser you also lose the session so, if you had a site requiring a login, this couldn't be saved as a session like it could as a cookie, and the user would be forced to re-login every time they visit.

Creating a session

- Simply by calling the session_start() function you can create a new session.Session variables are set with the PHP global: $_SESSION.

Syntax:

Session_start();

This is the initialization of session.Each and every page at the top script,we must initiate the session,if we need anhy session value.

Syntax:

$_SESSION['session_name]="value";

Example

- ```
  <?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>
  ```

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

Session variables are set.
  ```

Retrieving the value from session

- <?php
  session_start();
  ?>
  <!DOCTYPE html>
  <html>
  <body>
  <?php
  // Echo session variables that were set on previous page
  echo "Favorite color is " . $_SESSION["favcolor"] . ".
";
  echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
  ?>
  </body>
  </html>

We can retrive the value in any page if we store the value in session.Session carries the values drom one page to another

Favorite color is .
Favorite animal is .

To view variables

- <?php
  session_start();
  ?>
  <!DOCTYPE html>
  <html>
  <body>
  <?php
  print_r($_SESSION);
  ?>
  </body>
  </html>

Modify a PHP Session Variable

```php
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
```

Destroy a PHP Session

- <?php
  session_start();
  ?>
  <!DOCTYPE html>
  <html>
  <body>
  <?php
  // remove all session variables
  session_unset();
  // destroy the session
  session_destroy();
  ?>
  </body>
  </html>

To remove all global session variables and destroy the session,
use session_unset() and session_destroy():

Session ID

- When a client communicates with a server, only the session ID is transmitted between them. When the client requests for data, ASP.NET looks for the session ID and retrieves the corresponding data. This is done in the following steps:
- Client hits the web site and information is stored in the session. Server creates a unique session ID for that client and stores it in the Session State Provider.
- The client requests for some information with the unique session ID from the server.
- Server looks in the Session Providers and retrieves the serialized data from the state server and type casts the object

Client

-Request for Pages and want to store some information'
-Next time clients request again for some information

Web Server

-Creates secure session ID and store data in state provider
-Server takes the seesion ID to session State provider

State Provider

Store client data separately
Send data to server based on session ID

PHP Cookies

Only
the *name* parameter
is required. All other
parameters are
optional.

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.
- Create Cookies With PHP
- A cookie is created with the `setcookie()` function.
- Syntax
- setcookie(*name, value, expire, path, domain, secure, httponly*);
- Only the *name* parameter is required. All other parameters are optional.

Cretation of cookies

```
<?php
$cookie_name="Admin";
$cookie_value="Simanta Kasaju";
setcookie($cookie_name,$cookie_value,time()+(86400),"/");
?>
<html>
<body>
    <?php
    echo $_COOKIE[$cookie_name];
    ?>
</body>
</html>
```

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

```
<?php
$cookie_name="Admin";
$cookie_value="Simanta Kasaju";
setcookie($cookie_name,$cookie_value,time()+(86400*30),"
/");
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name]))
{
   echo "not set";
}
else {
 echo  $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

```
<?php
$cookie_name = "Admin";
$cookie_value = "Prashant Kasaju";
setcookie($cookie_name, $cookie_value,
time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo  "is not set!";
} else {
 $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

```php
<?php
$cookie_name="Admin";
$cookie_value="Simanta Kasaju";
setcookie($cookie_name,$cookie_value,time()-
(86400*30),"/");
?>
<html>

<body>
    <?php
    echo "cookie is deleted";
    ?>
</body>
</html>
```

Cookies	Sessions
Cookies are stored on the user's computer.	Sessions are stored on the server.
A cookie can store a limited amount of data, a maximum of 4KB.	The session can store an unlimited amount of data.
The cookie does not depend on the Session.	The session depends on the cookie.
The cookie expires according to the time of expiry set for it	The session expires after the user closes the web browser.
Cookie has many security issues as it can be accessed by anyone easily.	The session is secure as it cannot be accessed by anyone easily.
There is no function available to disable a Cookie.	A Session can be disabled by using the session_destroy() function.
The **setcookie**() function should always be used prior to tag.	The **session_start**() function should always appear prior to tag.

	Session		Cookies
1	Session expires when the browser is closed.	1	Cookies last longer than session
2	Session are stored on the server computer	2	Cookies are stored on the user's computer
3	Session are reliable and more secure as we cannot easily access the session value	3	Cookies are unreliable and less secure as we can easily access the cookie value
4	Session can store unlimited amount of data	4	Cookies can store only limited amount of data
5	Session is mainly used for login/logout purpose	5	cookies using for user activity tracking

Architecting Web Application

- Web application architecture defines the interactions between applications, middleware systems and databases to ensure multiple applications can work together.
- When a user types in a URL and taps “Go,” the browser will find the Internet-facing computer the website lives on and requests that particular page.
- The server then responds by sending files over to the browser. After that action, the browser executes those files to show the requested page to the user. Now, the user gets to interact with the website. Of course, all of these actions are executed within a matter of seconds. Otherwise, users wouldn’t bother with websites.
- What’s important here is the code, which has been parsed by the browser.
- This very code may or may not have specific instructions telling the browser how to react to a wide swath of inputs. As a result, web application architecture includes all sub-components and external applications interchanges for an entire software application.
- Of course, it is designed to function efficiently while meeting its specific needs and goals.
- Web application architecture is critical since the majority of global network traffic, and every single app and device uses web-based communication.
- It deals with scale, efficiency, robustness, and security.

Best Practices for Good Web Application Architecture

- You may have a working app, but it also needs to have good web architecture.
- Here are several attributes necessary for good web application architecture: Solves problems consistently and uniformly
- Make it as simple as possible
- Supports the latest standards include Offers fast response times
- Utilizes security standards to reduce the chance of malicious penetrations
- Does not crash
- Heals itself
- Does not have a single point of failure
- Scales out easily
- Allows for easy creation of known data
- Errors logged in a user-friendly way
- Automated deployments

PHP Error Handling

- Error handling is the process of catching errors raised by your program and then taking appropriate action. If you would handle errors properly then it may lead to many unforeseen consequences.
- Its very simple in PHP to handle an errors.
- An error message with filename, line number and a message describing the error is sent to the browser.
- When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.
- This tutorial contains some of the most common error checking methods in PHP.
- We will show different error handling methods:
- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Basic Error Handling: Using the die() function

- The first example shows a simple script that opens a text file:
- <?php
  $file=fopen("mytestfile.txt","r");
  ?>

If the file does not exist you might get an error like this:

```
Warning: fopen(mytestfile.txt)
[function.fopen]: failed to open stream:
No such file or directory
in C:\webfolder\test.php on line 2
```

```php
<?php
if(file_exists("mytestfile.txt")) {
  $file = fopen("mytestfile.txt", "r");
} else {
  die("Error: The file does not
exist.");
}
?>
```

Creating a Custom Error Handler

- Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.
- This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

error_function(error_level,error_message, error_file,error_line,error_context);

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Types:

Notice: These are small, non-critical errors that PHP encounters while executing a script.that dont stop PHP from executing a script. for example,:accessing a variable that has not yet been defined

Warnings:seriuos errors that require attention but still dont stop script to execute.eg: reading a file that doesnt exist in given path ,division by zero.

Fatal errors:syntax error or critical errors that stop execution .eg :calling a non-existent function, instantiating an object of a non-existent class.

Possible Error levels

These error report levels are the different types of error the user-defined error handler can be used for. These values cab used in combination using **|** operator

Sr.No	Constant & Description	Value
1	.E_ERROR Fatal run-time errors. Execution of the script is halted	1
2	E_WARNING Non-fatal run-time errors. Execution of the script is not halted	2
3	E_PARSE Compile-time parse errors. Parse errors should only be generated by the parser.	4
4	E_NOTICE Run-time notices. The script found something that might be an error, but could also happen when running a script normally	8
5	E_CORE_ERROR Fatal errors that occur during PHP's initial start-up.	16
6	E_CORE_WARNING Non-fatal run-time errors. This occurs during PHP's initial start-up.	32

7	**E_USER_ERROR** **Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()**	**256**
8	E_USER_WARNING Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()	512
9	E_USER_NOTICE User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()	1024
10	E_STRICT Run-time notices. Enable to have PHP suggest changes to your code which will ensure the best interoperability and forward compatibility of your code.	2048
11	E_RECOVERABLE_ERROR Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())	4096
12	E_ALL All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0)	8191

Error reporting

We can control which error are the displayed in script output with a build in function called error_reporting();

Eg:
<?php //only display fatal errors
error_reporting(E_ERROR);
Echo 1/0;
We can also use error_reporting() to turn off run time fatal errors.
Eg: <? Php //only display warning
Error_reporting(E_WARNING);
Echo somefunc();
It is important to realize that error_reporting doesn't automatically make our script error free.
all it does is hide error of a certain type. We can also pass error_reporting() a combination of error levels to customize errors.

```
<?php
error_reporting(E_WARNING);
echo $var;// notice
echo 1/0 ;//warning
echo myfun(); //fatal
?>
```

Now lets create a function to handle errors:

```
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}
```

Set Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

```
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr";
}
//set error handler
set_error_handler("customError");
//trigger error
echo($test);
?>
```

Error: [8] Undefined variable: test

Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.

```
<?php
$test=2;
if ($test>=1) {
  trigger_error("Val
ue must be 1 or
below");
}
?>
```

Notice: Value must be 1 or below
in **C:\webfolder\test.php** on line **6**

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
- E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

```php
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>=1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

Error: [512] Value must be 1 or below
Ending Script

What is an Exception

- With PHP 5 came a new object oriented way of dealing with errors.
- Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

  This is what normally happens when an exception is triggered:
- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code
- We will show different error handling methods:
- Basic use of Exceptions
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler
- **Note:** Exceptions should only be used with error conditions, and should not be used to jump to another place in the code at a specified point.

Why needs Exception Handling?

- PHP provides a powerful mechanism, exception handling. It allows you to handle runtime errors such as IOException, SQLException, ClassNotFoundException, and more. A most popular example of exception handling is - divide by zero exception, which is an arithmetic exception.

example

```php
<?php
//create function
with an exception
function checkNum($n
umber) {
  if($number>1) {
    throw new Except
ion("Value must be 1
or below");
  }
  return true;
}

//trigger exception
checkNum(2);
?>
```

```
Fatal error: Uncaught
exception 'Exception'
with message 'Value must be 1
or below' in
C:\webfolder\test.php:6
Stack trace: #0
C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown
in C:\webfolder\test.php on
line 6
```

The above error generates because try and catch is not defined to handle the error.

Try, throw and catch

- To avoid the error from the example above, we need to create the proper code to handle an exception.
- Proper exception code should include:
- `try` - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
- `throw` - This is how you trigger an exception. Each "throw" must have at least one "catch"
- `catch` - A "catch" block retrieves an exception and creates an object containing the exception information

- try
- The try block contains the code that may contain an exception. An exception raised in try block during runtime is caught by the catch block. Therefore, each try block must have at least one catch block. It consists of the block of code in which an exception can occur.
- Following points needs to be noted about the try:
- The try block must be followed by a catch or finally block.
- A try block must have at least one catch block.
- There can be multiple catch blocks with one try block.
- catch
- The catch block catches the exception raised in the try block. It contains the code to catch the exception, which is thrown by throw keyword in the try block. The catch block executes when a specific exception is thrown. PHP looks for the matching catch block and assigns the exception object to a variable.
- Following points to be noted about the catch:
- There can be more than one catch block with a try.
- The thrown exception is caught and resolved by one or more catch.
- The catch block is always used with a try block. It cannot be used alone.
- It comes just after the try block.

Cont..

- throw
- It is a keyword, which is used to throw an exception. Note that one throw at least has one "**catch** block" to catch the exception.
- It lists the exceptions thrown by function, which cannot be handled by the function itself.
- finally
- It is a block that contains the **essential code** of the program to execute. The finally block is also used for clean-up activity in PHP. It is similar to the catch block, which is used to handle exception. The only difference is that it always executes whether an exception is handled or not.
- The finally block can be specified after or in place of catch block. It always executes just after the try and catch block whether an exception has been thrown or not, and before the normal execution restarts. It is useful in the following scenarios - **Closing of database connection**, **stream**.

```php
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}

//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be
shown
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
finally { echo '</br> It is finally block, which always executes.';  }
?>
```

Example explained:
The code above throws an exception and catches it:
1.The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2.The checkNum() function is called in a "try" block
3.The exception within the checkNum() function is thrown
4.The "catch" block retrieves the exception and creates an object ($e) containing the exception information
5.The error message from the exception is echoed by calling $e->getMessage() from the exception object
However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

```php
<?php
class DivideByZeroException extends Exception {
 }
class DivideByNegativeNoException extends Exc
eption { }
function checkdivisor($dividend, $divisor){
    // Throw exception if divisor is zero
  try {
      if ($divisor == 0) {
        throw new DivideByZeroException;
      }
      else if ($divisor < 0) {
         throw new DivideByNegativeNoException;
      }
      else {
        $result = $dividend / $divisor;
        echo "Result of division = $result </br>";
      }
    }
    catch (DivideByZeroException $dze) {
        echo "Divide by Zero Exception! </br>";
    }
    catch (DivideByNegativeNoException $dnne)
{
        echo "Divide by Negative Number Exceptio
n </br>";
    }
    catch (Exception $ex) {
        echo "Unknown Exception";
    }
}

    checkdivisor(18, 3);
    checkdivisor(34, -6);
    checkdivisor(27, 0);
?>
```

classwork

- Write a php code for a number if it is even then throw even else throw odd.
- Use catch for that exception
- Use any test for try

```php
<?php
class evenException extends Exceptio
n { }
class oddException extends Exception
 { }
//user-
defined function with an exception
function test($num) {
    try{
   if($num%2==0) {
     //throw an exception
     throw new evenException;
   }
   else{
    throw new oddException;
   }
   return true;
}
//trigger an exception in a "try" block
catch (evenException $ee) {
    echo "even</br>";
}
catch (oddException $oe) {
    echo "odd </br>";
}
finally {
   echo "</br> It is finally block, which always
executes.<br>";
}
}
test(5);
test(24);
?>
```

File handling

Mode	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and truncates the file; or creates a new file if it doesn't exist. Place file pointer at the beginning of the file
w+	Read/Write. Opens and truncates the file; or creates a new file if it doesn't exist. Place file pointer at the beginning of the file
a	Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Write. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists
c	Write only. Opens the file; or creates a new file if it doesn't exist. Place file pointer at the beginning of the file
c+	Read/Write. Opens the file; or creates a new file if it doesn't exist. Place file pointer at the beginning of the file

Tag libraries

- Read yourself

End of chapter 4