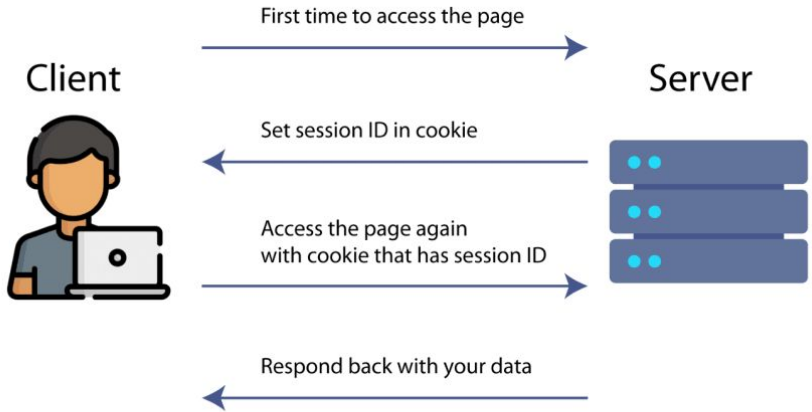
Web Development III

Block: II

Prepared By: Soniya Suthar

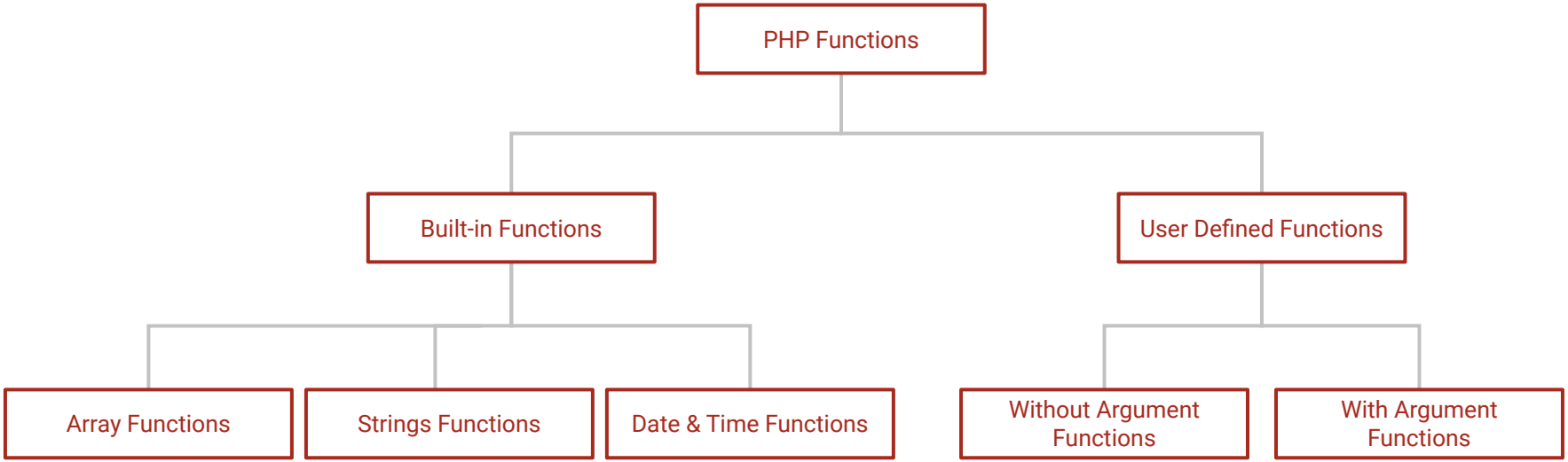
Block: II - Agenda

- Functions, Recursion
- Regular Expressions
- PHP include & Require
- PHP Superglobals
- Form Handling, File Handling
- Cookies, Sessions
- Date & Time, JSON



Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads, it will be executed by a call to the function.



UDF Without Arg

```php
<?php
	function writeMsg() {
		echo "Hello world!";
	}
	writeMsg(); //call the function
?>
```

UDF With Arg

```php
<?php
	function Name($fname) {
		echo "$fname Suthar<br>";
	}
	Name("Soniya"); //call the function with arg
	Name("Sona"); //call the function with arg
?>
```

Functions - Returning values

```php
<?php
    function sum(int $x, int $y) {
        $z = $x + $y;
        return $z;
    }
    echo "5 + 10 = " . sum(5, 10) . "<br>";
    echo "7 + 13 = " . sum(7, 13) . "<br>";
    echo "2 + 4 = " . sum(2, 4);
?>
```

Functions - Default values

```
<?php
      function hi(int $x='Soniya') {
              echo "Hello.. " + $x;
      }
      hi('Soniya Mam');
      hi();
?>
```

Recursive Function

```php
<?php
    function num($a) {
        if($a<=5){
            echo "$a </br>";
            num($a+1); //the function calls itself - Recursion
        }
        else{
            echo "End of Recursive Function";
        }

    }

    num(1);
?>
```

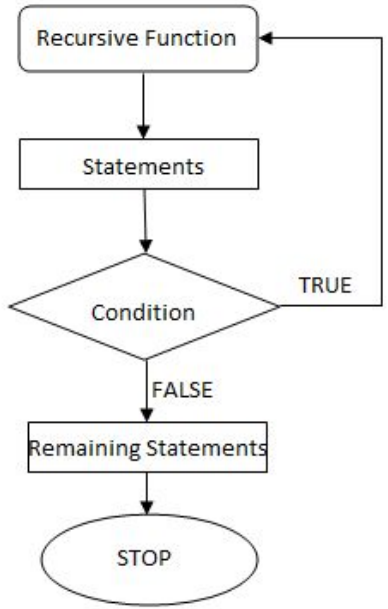
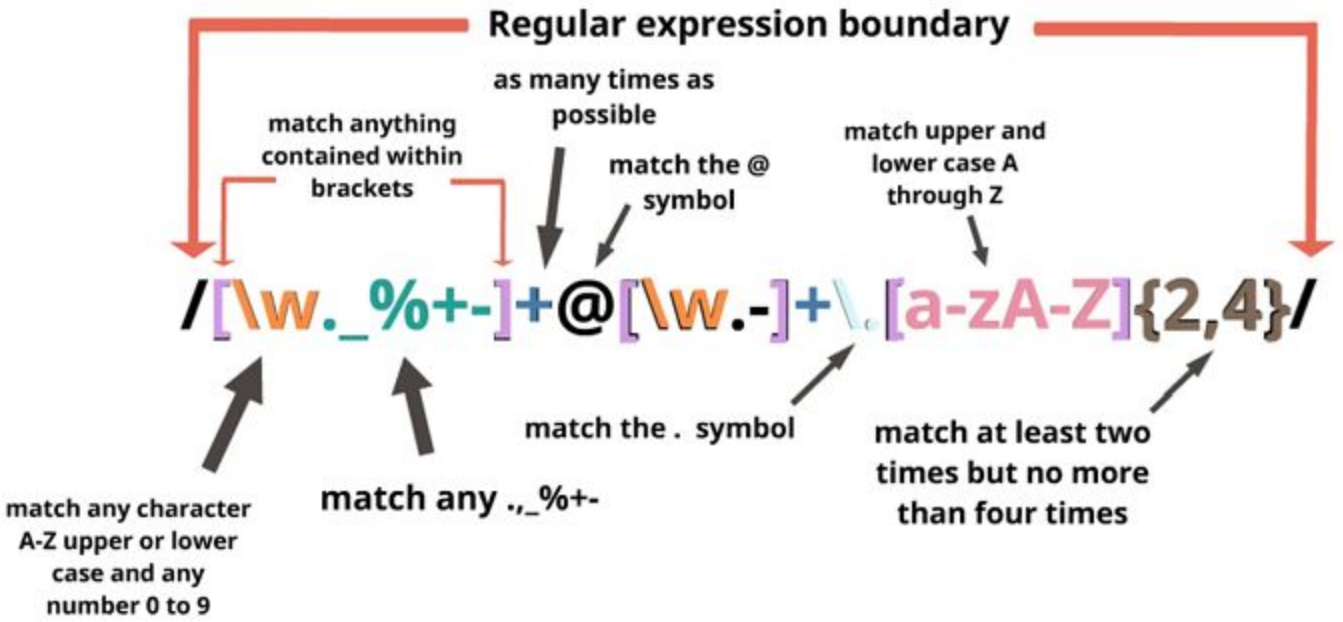


Fig: Flowchart showing recursion

Regular Expressions



Regular Expressions

- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern. It can be used to perform all types of text search and text replace operations.
- PHP provides a variety of functions that allow you to use regular expressions. The `preg_match()`, `preg_match_all()` and `preg_replace()` functions are some of the most commonly used ones.

Function	Description
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

Regular Expression Modifiers

Modifier	Description
i	Performs a case-insensitive search
m	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u	Enables correct matching of UTF-8 encoded patterns

Regular Expression Patterns

Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Using preg_match()

```php
<?php
    $str = "Soniya Suthar";
    $pattern = "/s/i";
    echo preg_match($pattern, $str);
    // Outputs 1
?>
```

Using preg_match_all()

```php
<?php
    $str = "Soniya Suthar";
    $pattern = "/s/i";
    echo preg_match_all($pattern, $str);
    // Outputs 2
?>
```

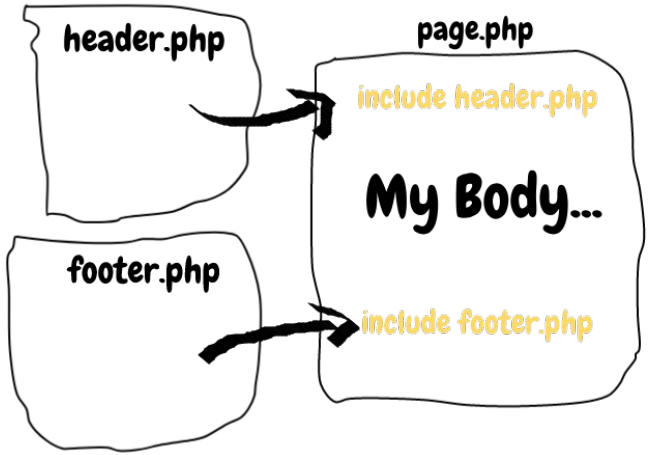
Using preg_replace()

```php
<?php
    $str = "Soniya Suthar";
    $pattern = "/ /i";
    echo preg_replace($pattern, "Mam ", $str); // Outputs "SoniyaMam Suthar"
?>
```

PHP Include & Require

- It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.
- The include and require statements are identical, except upon failure:
  - `require` will produce a fatal error (E_COMPILE_ERROR) and stop the script
  - `include` will only produce a warning (E_WARNING) and the script will continue
- Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.
- Syntax:

```
include 'filename';

      OR

require 'filename';
```



PHP Include

- Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php echo "<p>Copyright &copy; 2021-" . date("Y") . " soniyasuthar.com</p>";
?>
```

- To include the footer file in a page, use the `include` statement:

```
<html>

<body>

<h1>Welcome to my home page! </h1>

<p>Some text..</p>

<?php include 'footer.php';?>

</body>

</html>
```

PHP Include VS Require:

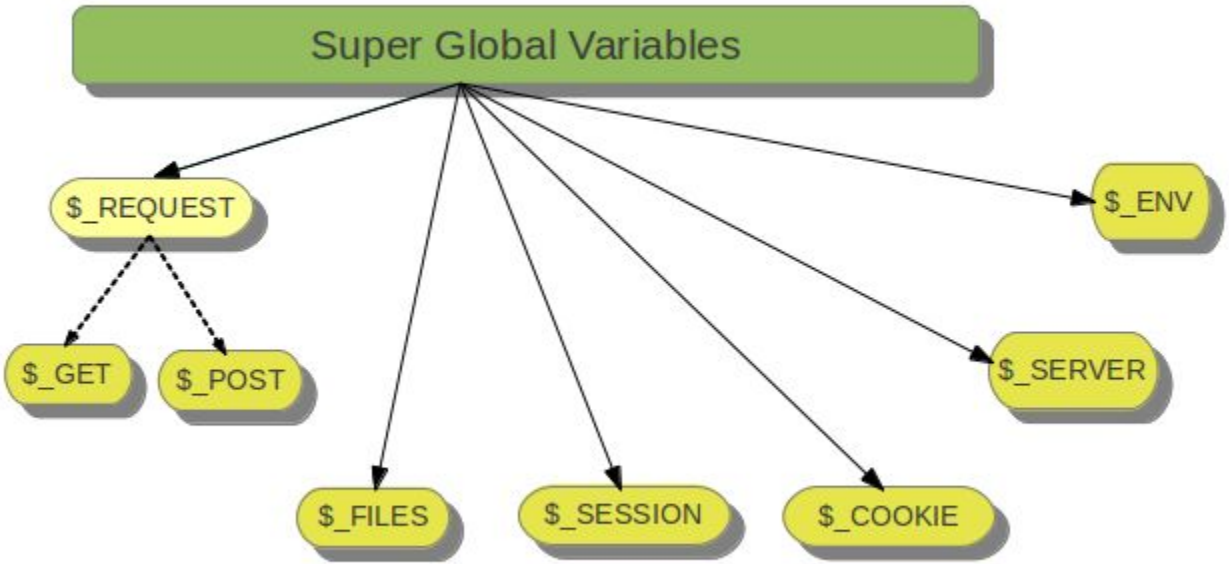
- The `require` statement is also used to include a file into the PHP code. There is one big difference between include and require; when a file is included with the `include` statement and PHP cannot find it, the script will continue to execute.

```
<html>

<body>

<h1>Welcome to my home page! </h1>

<?php include 'noFileExists.php' ;

echo "I have a $color $car." ;

?>

</body>

</html>
```

```
<html>

<body>

<h1>Welcome to my home page! </h1>

<?php require 'noFileExists.php' ;

echo "I have a $color $car." ;

?>

</body>

</html>
```

- If we do the same example using the `require` statement, the echo statement will not be executed because the script execution dies after the `require` statement returned a fatal error.

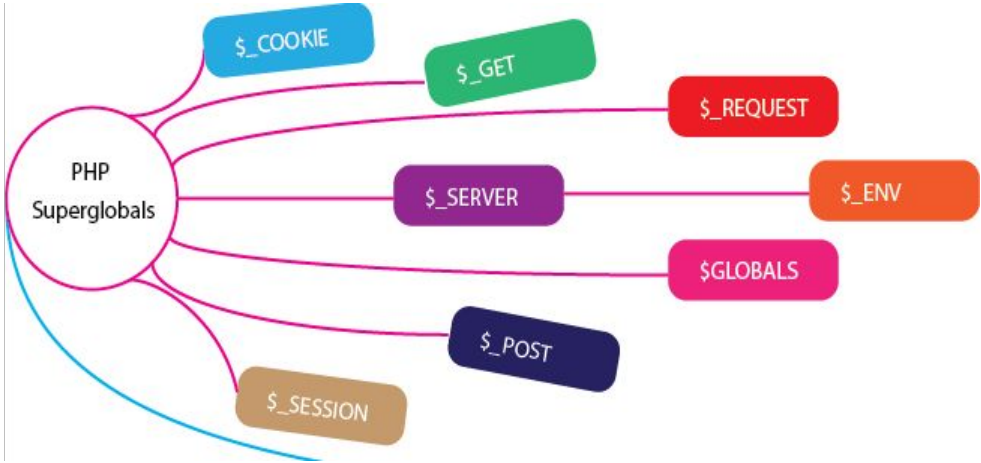
PHP Superglobals



PHP Superglobals

- Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- Super global variables are built-in variables that are always available in all scopes.

- The PHP superglobal variables are:
  - $GLOBALS
  - $_SERVER
  - $_REQUEST
  - $_POST
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION



PHP Superglobals: $GLOBALS

- $GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
- PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.

```php
<?php

    $x = 75;

    $y = 25;

    function addition() {

      $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];

    }

    addition();

    echo $z;

?>
```

PHP Superglobals: $_SERVER

- $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

```php
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

PHP Superglobals: $_REQUEST

- PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

```
<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") {

  // collect value of input field

  $name = $_REQUEST['fname'];

  if (empty($name)) {

    echo "Name is empty";

  } else {

    echo $name;

  }

}

?>

</body>

</html>
```

PHP Superglobals: $_POST

- PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

```
<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") {

  // collect value of input field

  $name = $_POST['fname'];

  if (empty($name)) {

    echo "Name is empty";

  } else {

    echo $name;

  }

}

?>

</body>

</html>
```

PHP Superglobals: $_GET

- PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".
- $_GET can also collect data sent in the URL.

```
<html>

<body>

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

  Name: <input type="text" name="fname">

  <input type="submit">

</form>

<?php

if ($_SERVER["REQUEST_METHOD"] == "GET") {

  // collect value of input field

  $name = $_GET['fname'];

  if (empty($name)) {

    echo "Name is empty";

  } else {

    echo $name;

  }

}

?>

</body>

</html>
```

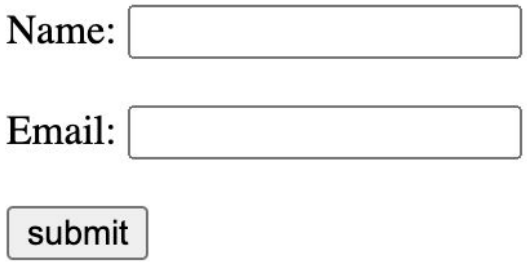
Form Handling

Main_file.php

```
<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>
```

welcome_get.php

```
<html>

<body>

Welcome <?php echo $_GET["name"]; ?><br>

Your email address is: <?php echo
$_GET["email"]; ?>

</body>

</html>
```



Form Handling: GET vs. POST

- Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- $_GET is an array of variables passed to the current script via the URL parameters.
- $_POST is an array of variables passed to the current script via the HTTP POST method.

Form Handling: GET

- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.

Form Handling: POST

- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Form Handling: Validation

- An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.
- There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things.
    - Empty String
    - Validate String
    - Validate Numbers
    - Validate Email
    - Validate URL
    - Input length

Form Handling: Validation

- Some of validation rules for the form are as follows:

Field	Validation Rules
Name	Should required letters and white-spaces
Email	Should required @ and .
Website	Should required a valid URL
Radio	Must be selectable at least once
Check Box	Must be checkable at least once
Drop Down menu	Must be selectable at least once

Empty String

```
if (emptyempty ($_POST["name"])) {
    $errMsg = "Error! You didn't enter the Name.";
        echo $errMsg;
} else {
    $name = $_POST["name"];
}
```

Validate String

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
    $ErrMsg = "Only alphabets and whitespace are allowed.";
        echo $ErrMsg;
} else {
    echo $name;
}
```

Validate Number

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
    $ErrMsg = "Only numeric value is allowed.";
    echo $ErrMsg;
} else {
    echo $mobileno;
}
```

Validate Email

```
$email = $_POST ["Email"];
$pattern =
"^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
    $ErrMsg = "Email is not valid.";
        echo $ErrMsg;
} else {
    echo "Your valid email address is: " .$email;
}
```

Input Length Validation

```php
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);

if ( $length < 10 && $length > 10) {
    $ErrMsg = "Mobile must have 10 digits.";
        echo $ErrMsg;
} else {
    echo "Your Mobile number is: " .$mobileno;
}
```

Button Click Validate

```php
if (isset ($_POST['submit']) {
    echo "Submit button is clicked.";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        echo "Data is sent using POST method ";
    }
} else {
    echo "Data is not submitted";
}
```

Validate URL

```
$websiteURL = $_POST["website"];
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "URL is not valid";
  echo $websiteErr;
} else {
    echo "Website URL is: " .$websiteURL;
}
```

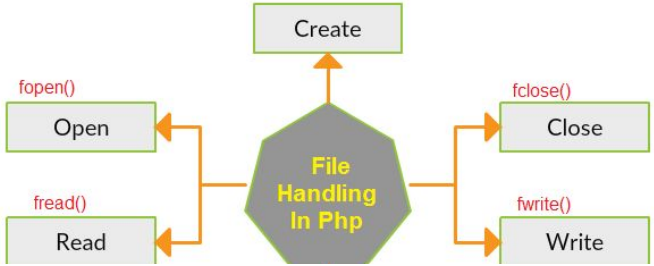
File Handling: Open, Close, Read & Write

- The PHP fopen() function is used to open a file.
- Syntax:

  fopen ($filename, string $mode)

- The PHP fclose() function is used to close an open file pointer.
- Syntax:

  fclose ($handle)



- The PHP fread(), fgets() and fgetc() function is used to read the content of the file.
- It accepts two arguments: resource and file size.
- Syntax:

  fread ($handle, $length), fgets($filename), fgetc($filename)

- The PHP fwrite() function is used to write content of the string into file.
- Syntax:

  fwrite ($handle , $string)

File Handling: Open Modes

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

File Handling: Open, Close, Read & Write

```php
<?php
	$myfile = fopen("myfile.txt", "r") or die("Unable to open file!");
	echo fread($myfile,filesize("myfile.txt"));
	fclose($myfile);
	$myfile = fopen("myfile.txt", "w") or die("Unable to open file!");
	$txt = "Soniya Mam";
	fwrite($myfile, $txt);
	fclose($myfile);
?>
```

File Upload

- PHP allows you to upload single and multiple files through few lines of code only.
- The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file as per following:

    - $_FILES['filename']['name']: returns file name.
    - $_FILES['filename']['type']: returns MIME type of the file.
    - $_FILES['filename']['size']: returns size of the file (in bytes).
    - $_FILES['filename']['tmp_name']: returns temporary file name of the file which was stored on the server.
    - $_FILES['filename']['error']: returns error code associated with this file.

- The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded through the POST request. It moves the file if it is uploaded through the POST request.

File Upload

Mainfile.html

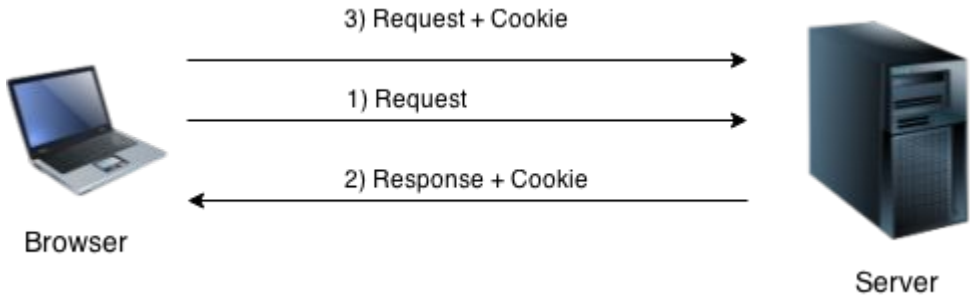
```
<form action="Upload.php" method="post"
 enctype="multipart/form-data">
    Select File:
    <input type="file" name="fileToUpload"/>
    <input type="submit" value="Upload Image" name="submit"/>
</form>
```

Upload.php

```
<?php
$target_path = "e:/";
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {
    echo "File uploaded successfully!";
} else{
    echo "Sorry, file not uploaded, please try again!";
}
?>
```

PHP Cookies

- PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.
- Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



- PHP provided setcookie() function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.
- The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).
- PHP $_COOKIE superglobal variable is used to get cookie.
- To modify a cookie, just set (again) the cookie using the setcookie() function, To delete a cookie, use the setcookie() function with an expiration date in the past.

PHP Cookies: Set

- Syntax:

  setcookie(name, value, expire, path, domain, security);

- Details of arguments:
  - Name – This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
  - Value – This sets the value of the named variable and is the content that you actually want to store.
  - Expiry – This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
  - Path – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
  - Domain – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
  - Security – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

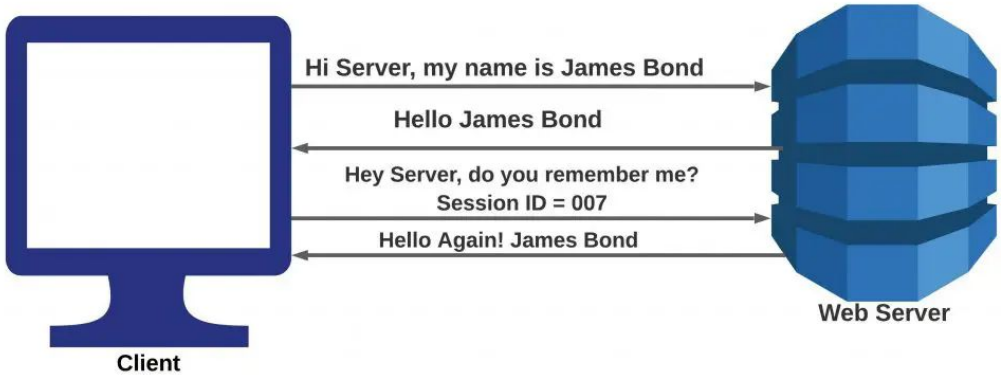
PHP Cookies: Create/Retrieve/Delete

```php
// Create/Retrieve

<?php
setcookie("user", "Student");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
   echo "Sorry, cookie is not found!";
} else {
   echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

```php
// Delete

<?php
setcookie ("CookieName", "", time() - 3600);// set the expiration date to one hour ago
?>
<html>
<body>
<?php
   echo "Cookie Deleted";
?>
</body>
</html>
```

PHP Session

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- The web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
- PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP Session: Start & Destroy

- PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.
- PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Session1.php

```
<?php
        session_start();
?>
<html>
<body>
<?php
        $_SESSION["user"] = "Soniya";
        echo "Session information are set successfully.<br/>";
?>
<a href="Session2.php">Visit next page</a>
</body>
</html>
```

Session2.php

```
<?php
        session_start();
?>
<html>
<body>
<?php
        echo "User is: ".$_SESSION["user"];
        session_destroy();
?>
</body>
</html>
```

PHP Session: Page View Counter

```php
<?php
  session_start();
  if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  } else {
    $_SESSION['counter']++;
  }
  echo ("Page Views: ".$_SESSION['counter']);
?>
```

PHP Date and Time

- Get a Date:
  - The required *format* parameter of the date() function specifies how to format the date (or time).
  - Here are some characters that are commonly used for dates:
    - d - Represents the day of the month (01 to 31)
    - m - Represents a month (01 to 12)
    - Y - Represents a year (in four digits)
    - l (lowercase 'L') - Represents the day of the week
  - Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

- Get a Time:
  - Here are some characters that are commonly used for times:
    - H - 24-hour format of an hour (00 to 23)
    - h - 12-hour format of an hour with leading zeros (01 to 12)
    - i - Minutes with leading zeros (00 to 59)
    - s - Seconds with leading zeros (00 to 59)
    - a - Lowercase Ante meridiem and Post meridiem (am or pm)

PHP Date and Time

- PHP date() function shows day, month and year altogether. Date and time are stored in computer in UNIX Timestamp format. It calculates time in number of seconds on GMT i.e started from January 1, 1970, 00:00:00 GMT.
- Syntax: `date(format,timestamp)`

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

- The PHP time() function is used to return the current time as a UNIX timestamp. This function would work by our server time and can be formatted easily.
- Syntax: `time()`

```
<?php
        $today=time();
        echo($today . "\n");
        echo(date("Y-m-d",$today));
?>
```

```
<?php
  print date("m/d/y G.i:s<br>", time());
  echo "<br>";
  print "Today is ";
  print date("j of F Y, \a\\t g.i a", time());
?>
```

PHP and JSON

- JSON stands for JavaScript Object Notation, and is a syntax for storing and exchanging data.
- Since the JSON format is a text-based format, it can easily be sent to and from a server, and used as a data format by any programming language.
- PHP has some built-in functions to handle JSON.
- The json_encode() function is used to encode a value to JSON format.
- The json_decode() function is used to decode a JSON object into a PHP object or an associative array.The json_decode() function returns an object by default. The json_decode() function has a second parameter, and when set to true, JSON objects are decoded into associative arrays.

```
<?php
$a = array('firstName' => ‘Soniya’, 'lastName' => ‘Suthar’, 'email' => 'soniya@gmail.com');
echo json_encode($a);
?>
```


```
<?php
$j = '{"firstName" : "‘Soniya’", "lastName" : "‘Suthar’", "email" : "soniya@gmail.com";
$arr = json_decode($j, true);
foreach($arr as $key => $value) {
  echo $key . " => " . $value . "<br>";
?>
```

Prepared By: Soniya Suthar