CSY2028
Web Programming

Ganesh Khatri
ganesh.khatri@nami.edu.np

Topic 2 – Server Setup

- Client Server Architecture
- Introduction to XAMPP
- Introduction to PHP
- Running PHP Script in XAMPP
- Your First PHP Script
- Using PHP includes to reduce redundancy when creating web sites

Client-Server Architecture

- Web Sites are hosted on Web Servers
- These are computers, usually accessed over the internet, which store the web pages that make up the web site
- When you use a browser to view a web site, you connect to the server and download the files that make up the page (The HTML, CSS and images)

Client-Server Architecture

- The HTML and CSS files are processed by the client. In most cases, that client is a web browser (Firefox, Chrome, Internet Explorer, Safari, etc)
- It is also possible to do some processing on the server prior to sending the HTML to the client
- This is known as server-side programming
- The person viewing the website can view the HTML and CSS (Right click, view source) however they can never see any processing that was done on the server

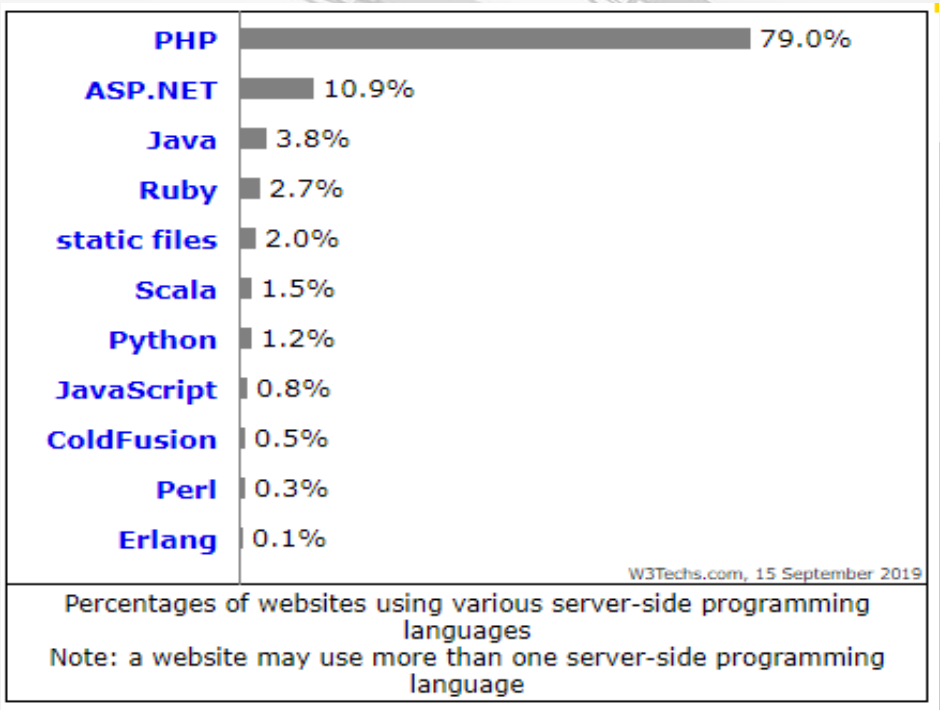
Server-Side Programming

- There are several languages used for Server-Side Programming including
  - PHP
  - Python
  - Ruby
  - Java
  - C#
  - C
  - Perl
  - Javascript*

PHP Programming

- This module will be using PHP because
  - It is widely used in the industry

(Source : https://w3techs.com)
(Sep, 2019)



PHP Programming

- It's easy to find help
  - If you have a problem, someone else would have had it before and found a solution. Googling your problem will usually give you the solution!
- The documentation is very good
- It's free and open source + works on all operating systems.
- It's easy to learn compared with others

PHP Programming

- PHP is most often used to generate the HTML that is finally sent to the browser
- This can be things like:
  - Including information from a database
  - Performing calculations and inserting them into the HTML
  - Getting data from elsewhere (e.g. a database) and formatting the result as HTML

Running PHP Scripts

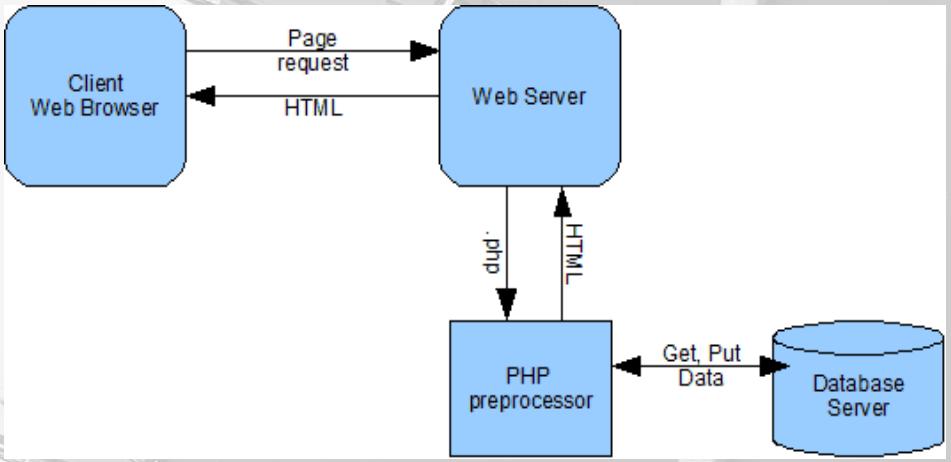
- PHP Scripts are generally run on a server and accessed via a web browser
- You can install PHP on your desktop/laptop machine and run PHP scripts via the command line
- For CSY2028 we want to view the result of PHP scripts in a browser
- This requires a web server

Web Server

- Your browser (e.g. Chrome, Firefox) will connect to the web server
- The server sends the HTML code for the requested file
- The browser displays the HTML it receives
- The HTML does not have to be a simple .html file, it can be generated by a program on the server

PHP Scripts

- You cannot run PHP scripts in a web browser!
- Web browsers do not understand PHP code
- You must connect the browser to a web server
- The server then runs the PHP code and sends the resulting HTML to the browser



Your Own Web Server

- It's possible to set up a web server on your desktop/laptop
- This involves installing and configuring PHP
- As well as installing a piece of software to serve the pages, listen to connections on HTTP and return the requested files
- The server we will use is Apache although others are available

Web Server

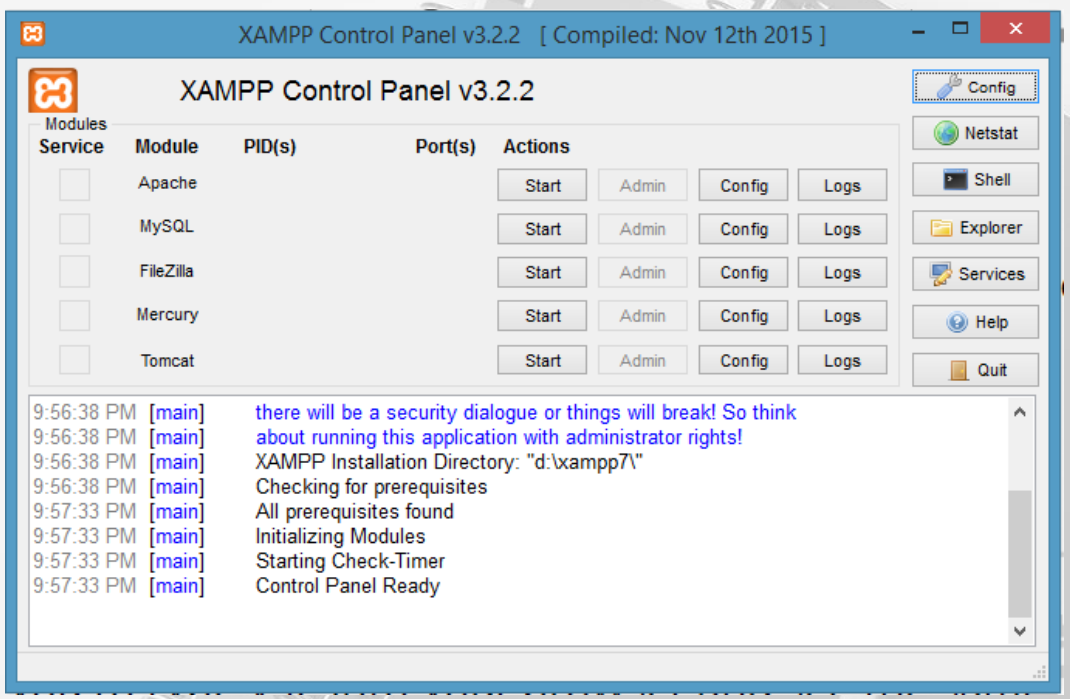
- There are several ways of getting a server installed on your machine
- Install PHP, Apache and MySQL manually and configure them yourself
    - Unless you know what you're doing, this can be difficult
    - There are a lot of different configuration options for both PHP and Apache, knowing how to set them up is art in itself

Getting Started

- Download latest version of XAMPP from https://www.apachefriends.org
- Install it on any drive (like C, D, E etc)
- Open xampp-control.exe file located inside installed “xampp” folder.
- Start Apache and MYSQL servers
- Create a project folder(any name : csy2028) inside htdocs folder
- csy2028 is the website root directory and create a file “index.php”
- Now run the project in browser by typing “localhost/csy2028”

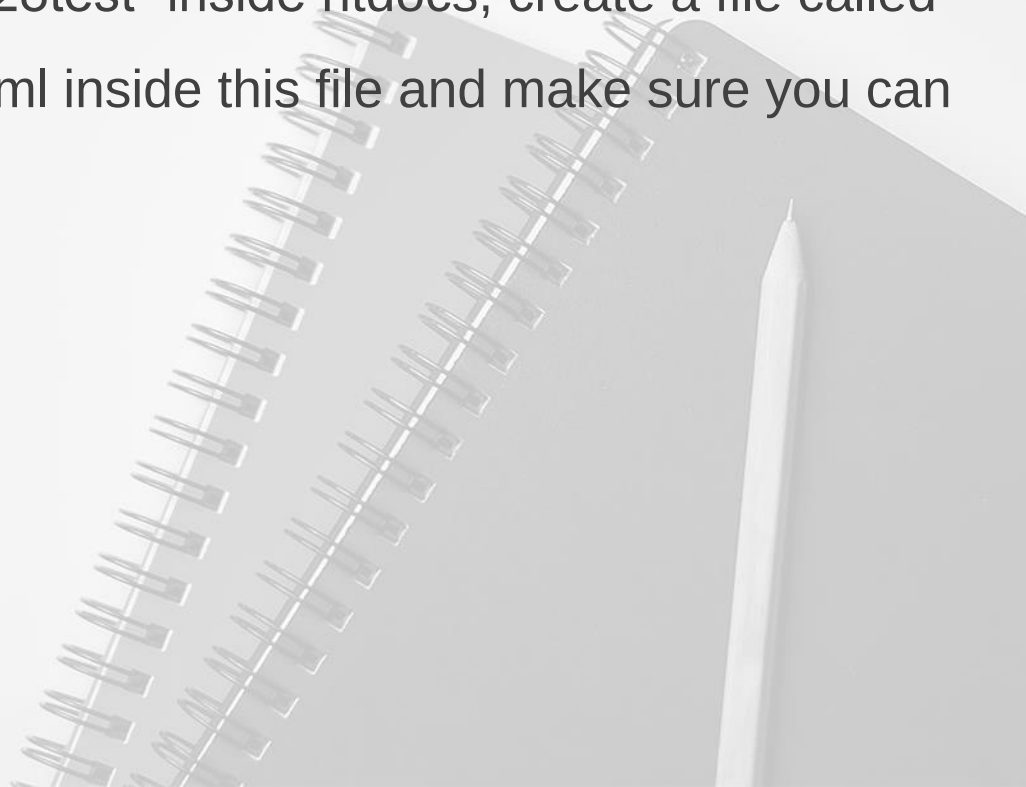
Getting Started

- XAMPP Control looks like this :



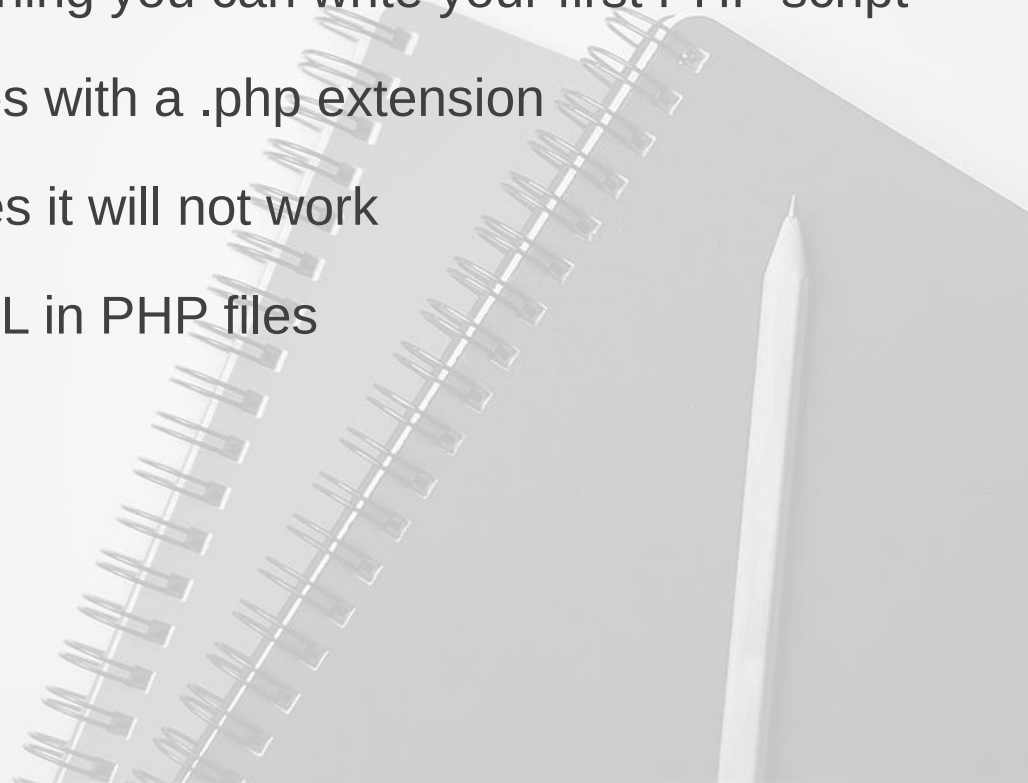
Exercise 1

- Create a project named “csy2028test” inside htdocs, create a file called test.php inside it, write some html inside this file and make sure you can access this file from the server.



PHP

- Once you have your server running you can write your first PHP script
- PHP code must be saved in files with a .php extension
- If you put PHP code in .html files it will not work
- However, you can include HTML in PHP files



PHP

- All PHP code is written between PHP tags
- PHP has start and end tags <?php and ?>

```
<?php
// your code here
?>
```

PHP Print

- To print to the screen use the echo command followed by your text in single quotes
- All statements must end with a semicolon (;)

```php
<?php
echo 'hello world';
?>
```

Single Vs Double Quotes

- PHP supports the use of single and double quotes
- They are mostly interchangeable
- However, by convention single quotes (apostrophes) are used in PHP
- Single quotes make it easier to work with HTML

```php
<?php

echo '<div class="content">My content</div>';

echo "<div class=\"content\">My Content</div>";

?>
```

Frist PHP File

- By saving a file with a .php extension e.g. test.php inside the project directory, it will be accessible on http://localhost/project/test.php
- When you connect to the page, the server processes the code and anything output using echo is displayed in the browser
- You can use echo to print HTML
- Anything inside PHP tags (<?php … ?>) is processed on the server before being sent to the browser
- If you view the source of the page in the browser, you will not see PHP code, only the HTML that has been printed

Mixing HTML and PHP

- It's possible to mix HTML and PHP code in the same file. You can start and end PHP code as many times as you like in a file

```
<!DOCTYPE html>
<html>
<head>
        <title>My Web Site</title>


</head>
<body>
<?php

echo 'Hello World';

?>
</body>
</html>
```

Mixing HTML and PHP

- The PHP code is never sent to the browser. If you right click and view the source code in your browser you will see the HTML but not the PHP

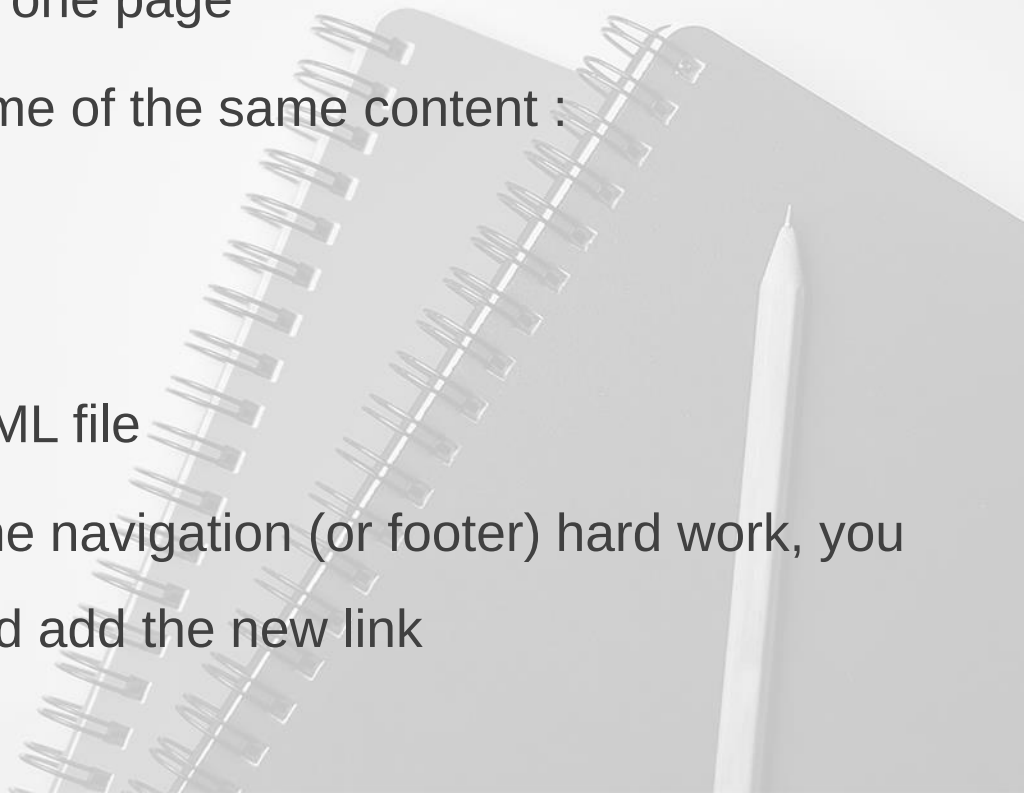
```
<!DOCTYPE html>
<html>
<head>
        <title>My Web Site</title>


</head>
<body>
Hello World
</body>
</html>
```

Exercise 2

1. Create a PHP script that prints “Hello world” to the screen in a file called hello.php inside folder csy2028 and make sure it’s visible on http://localhost/csy2028/hello.php
2. Create a PHP script that prints the HTML for a link to google:
   - Click here to visit google.com
   - Open the page in your browser and check it works. You should not see PHP code on the page, only the link!
   - You must use the PHP echo command to print the text and the code must be inside PHP tags!
   - You cannot open the PHP files directly in the browser, you must go through the web server!

Using PHP to Reduce Repetition

- Most web sites have more than one page
- A lot of those pages include some of the same content :
  - Navigation
  - Page Header
  - Page Footer
- You can repeat this in each HTML file
- But doing so makes updating the navigation (or footer) hard work, you have to open up every page and add the new link



Using PHP to Reduce Repetition

- Using PHP you can store the navigation, or footer, in one place and include it on each page
- Last week you built this web page :

Using PHP to Reduce Repetition

- It's possible to put the navigation it its own file. Instead of :

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <header>
      <h1>Heading</h1>
    </header>
    <div class="content">
    <nav>
      <ul>
        <li>
          <a href="#">Link 1</a>
        </li>
        <li>
          <a href="#">Link 2</a>
        </li>
        <li>
          <a href="#">Link 3</a>
        </li>
      </ul>
```

```
    </nav>
    <main>
      <p>Lorem ipsum....</p>
    </main>

    <div>
      Right hand side
    </div>
    </div>

    <footer>
      &copy; Your Name 2015
    </footer>
  </body>
</html>
```

Using PHP to Reduce Repetition

- You can store the navigation in nav.php and include it using PHP's require function
- Nav.php – index.php

```
<ul>
  <li>
    <a href="#">Link 1</a>
  </li>
  <li>
    <a href="#">Link 2</a>
  </li>
  <li>
    <a href="#">Link 3</a>
  </li>
</ul>
```

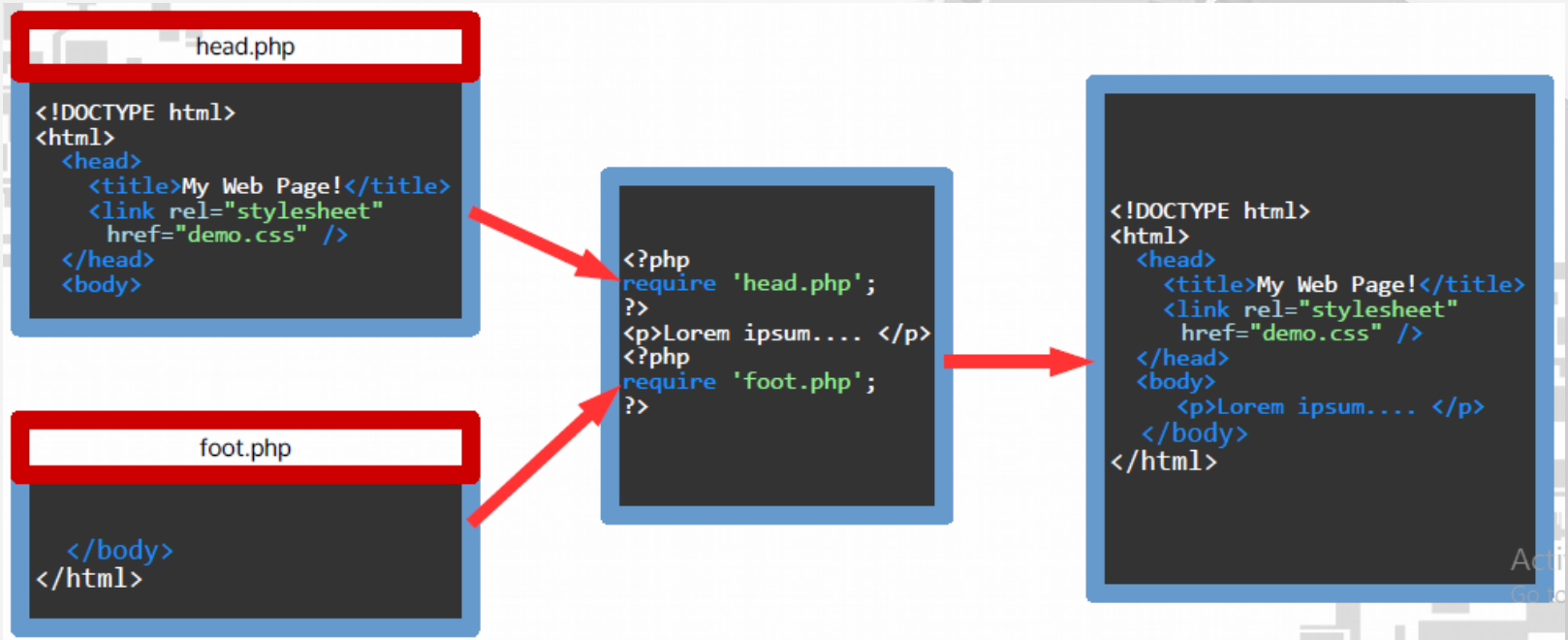
```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <header>
      <h1>Heading</h1>
    </header>
    <div class="content">
    <nav>
        <?php
        require 'nav.php';
        ?>
    </nav>
    <main>
      <p>Lorem ipsum....</p>
    </main>

….
```

Using PHP to Reduce Repetition

- A lot of the time, the only thing that will differ between pages is the content in <main>
- One simple way of handling this is head.php which includes everything before the content and foot.php which includes everything after the content:

require

- You can think of `require` as being a programmed copy/paste. The contents of the required file fill be copied into the file running the require line



head.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet"
     href="demo.css" />
  </head>
  <body>
    <header>
      <h1>Heading</h1>
    </header>
    <div class="content">
    <nav>
      <ul>
        <li>
          <a href="#">Link 1</a>
        </li>
        <li>
          <a href="#">Link 2</a>
        </li>
        <li>
          <a href="#">Link 3</a>
        </li>
      </ul>

    </nav>
    <main>
```

foot.php

```
    </main>

    <div>
      Right hand side
    </div>
    </div>

    <footer>
      &copy; Your Name 2015
    </footer>
  </body>
</html>
```

index..php

```
<?php
require 'head.php';
?>
<p>Lorem ipsum.... </p>
<?php
require 'foot.php';
?>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet"
     href="demo.css" />
  </head>
  <body>
    <header>
      <h1>Heading</h1>
    </header>
    <div class="content">
    <nav>
      <ul>
        <li>
          <a href="#">Link 1</a>
        </li>
        <li>
          <a href="#">Link 2</a>
        </li>
        <li>
          <a href="#">Link 3</a>
        </li>
      </ul>

    </nav>
    <main>
```

```
    <p>Lorem ipsum.... </p>
    </main>

    <div>
      Right hand side
    </div>
    </div>

    <footer>
      &copy; Your Name 2015
    </footer>
  </body>
</html>
```

Using PHP to Reduce Repetition

- This allows the creation of multiple pages that sit within the same content

about.php

```php
<?php
require 'head.php';
?>
<p>I'm a student at the
University of Northampton</p>
<?php
require 'foot.php';
?>
```

contact.php

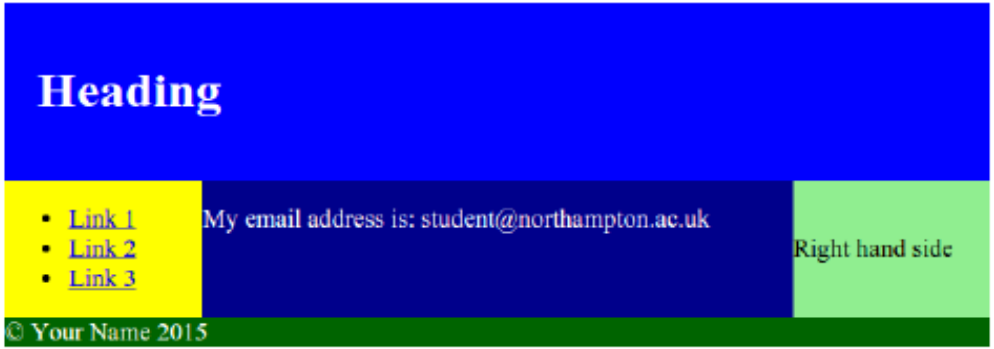
```php
<?php
require 'head.php';
?>
<p>My email address is:
student@northampton.ac.uk</p>
<?php
require 'foot.php';
?>
```

hobbies.php

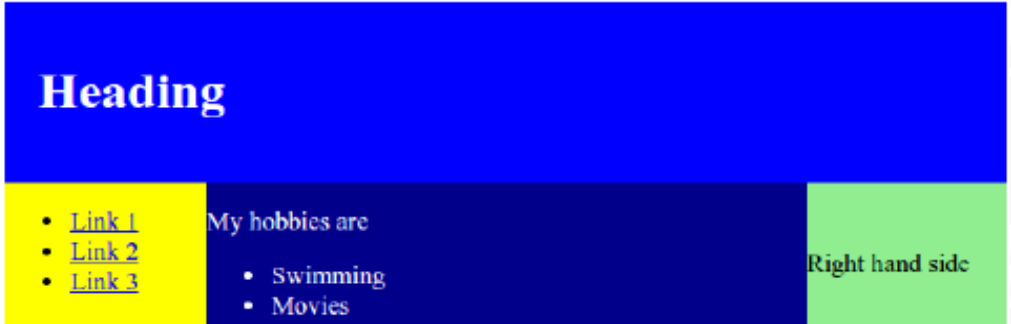
```php
<?php
require 'head.php';
?>
<p>My hobbies are</p>
<ul>
 <li>Swimming</li>
 <li>Movies</li>
</ul>
<?php
require 'foot.php';
?>
```

about.php

contact.php



hobbies.php

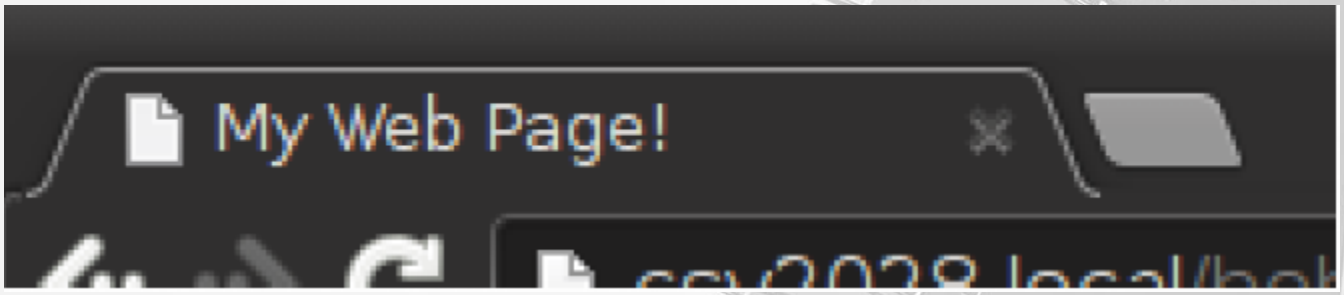


Exercise 3

1. Split up your web page from last week into 3 files (or download ex3.zip which contains the completed exercise from last week)
    - head.php – Everything up to and including <main>
    - foot.php – Everything from </main> to the bottom of the file
    - index.php – Everything between <main> and </main>
2. Use `require` to include the header/footer on the index.php page
    - If you open the page in a browser you should see the full page with the header/footer included!
    - You cannot open the PHP file directly in the browser, you must go through the web server!
3. Add two more files with different content, e.g. `about.php` and `contact.php` . Use the same head.php and foot.php, only the content should differ between the the new files and `index.php`
4. Change the navigation links to link to each of the 3 pages and add relevant titles

Separate Files

- By storing the navigation/header in its own file, adding a link to the navigation will add a link to every page
- However, what if you want something unique per file? For example, the page title shows like this on each page:



Unique Titles

- It would be better if each page had it's own title:
  - About
  - Hobbies
  - Contact
- To do this we can use a variable

Variables

- Variables can be used to store information that can be retrieved at a later time during the script
- Each variable has a name
- In PHP variables start with the dollar sign ($) and their name must start with a letter for example:
- $myVariable or $var
- $123 is not valid

Variables

- Variable names can contain numbers but must not start with them
  - $variable1 is valid but $1variable is not
- Variable names cannot contain spaces, dashes or any other character that has meaning to the language e.g. {, [,], “, ., +
- Variable names should be descriptive so you shouldn't usually need anything other than A-Z, 0-9 and the _ character

Variables

- To declare a variable, assign it a value in quotes (for strings):

```php
$myVariable = 'my variable value';
```

```php
$myVariable = 'my variable value';
echo $myVariable;
```

Variables

- Variables are shared across files that are included
- PHP scripts are executed in order
- As long as you declare a variable before you use it, you can use it in another file

one.php

```php
<?php
$myVariable = 'Hello World!';
require 'two.php';
?>
```

two.php

```php
<?php
echo $myVariable;
?>
```

Output of one.php

```
Hello World!
```

Variables

- This will cause an error because two.php is included before the variable is defined

```php
<?php

require 'two.php';

$myVariable = 'Hello World!';


?>
```

Exercise 4

1. Amend head.php to use a variable for the page title

2. Specify a unique title for each page e.g. “About” and “Contact”. As you move between pages you should see a different title on each page

3. Create the layout as a single file layout.php with variables for $content and $title. Use this layout instead of head.php and foot.php with the rest of your pages