

Internet Programming and E-commerce

C_ITEC311

Toolbox

Compiled by Zivanai Taruvinga

Edited by

Version 1.0

© November 2015 CTI Education Group

Contents

Section A: Individual Exercises

Individual Exercise 1: Internet and the World Wide Web (WWW)	2
Individual Exercise 2: Hypertext Markup Language (HTML) Basics	2
Individual Exercise 3: Hypertext Markup Language (HTML) Basics	2
Individual Exercise 4: Migration from HTML 4 to XHTML and HTML 5	3
Individual Exercise 5: Cascading Style Sheet (CSS)	3
Individual Exercise 6: JavaScript (JS) Programming	3
Individual Exercise 7: Hypertext Preprocessor (PHP)	4

Section B: Lab Exercises

Lab Exercise 1: Hypertext Markup Language (HTML) Basics	5
Lab Exercise 2: Hypertext Markup Language (HTML) Forms	6
Lab Exercise 3: HTML and Cascading Style Sheet (CSS)	6
Lab Exercise 4: JavaScript (JS) Basics	7
Lab Exercise 5: JavaScript (JS) and HTML Forms	8
Lab Exercise 6: Hypertext Preprocessor (PHP) Basics	9

Section C: Tutorials

Tutorial 1: CSS Horizontal Navigation Bar	10
Tutorial 2: PHP Sessions	20

Section A: Individual Exercises

Individual Exercise 1: Internet and the World Wide Web (WWW)

- 1.1 Briefly explain the relationship between the Internet and the World Wide Web (WWW).
- 1.2 Identify and explain the major protocols used on the internet.
- 1.3 Discuss the importance of Domain Name Systems.

Individual Exercise 2: Hypertext Markup Language (HTML) Basics

- 2.1 Provide the basic HTML tags for every HTML document. Briefly explain the purpose of each tag.
- 2.2 Discuss the usage and importance of using the meta tag on e-commerce websites.
- 2.3 Provide the syntax of creating any of the six headings supported by HTML. Why is it important to use different headings in our web documents?
- 2.4 Use an example to explain how you can align text within a paragraph. What alignment values can you use for different paragraph alignments?
- 2.5 List and explain the effect of any eight HTML formatting tags.
- 2.6 Explain the effects of each of the five attributes of HTML rules.
- 2.7 By means of an example, explain how you could link different pages of a website.
- 2.8 What image types are supported by HTML? How can you insert an image in a web page with text aligned to the right?
- 2.9 List and explain the purpose of the tags used within an HTML Table.
- 2.10 Explain why it is important to use lists on a website. What are the different types of lists that we can use in HTML?
- 2.11 Using examples, explain the different ways that can be used to define color values in HTML.

Individual Exercise 3: Hypertext Markup Language (HTML) Basics

- 3.1 Explain the importance of HTML forms on our websites.
- 3.2 What is the purpose of the commonly used `<form>` tag attributes?
- 3.3 Explain the purpose of any six values of the type attribute in HTML4.
- 3.4 Identify and explain the different objects that can be used on HTML forms. Provide the syntax of these objects in your explanation.

Individual Exercise 4: Migration from HTML 4 to XHTML and HTML 5

- 4.1 Identify and explain the thirteen specific guidelines published by the W3C for developers who wish to quickly understand the differences between HTML4 and XHTML.
- 4.2 What do you understand by the term **Form 2.0** as introduced in HTML5? Discuss the new type attribute values introduced in HTML5.
- 4.3 Explain the difference between HTML4 and HTML5 document structures.

Individual Exercise 5: Cascading Style Sheet (CSS)

- 5.1 Explain the three different ways that can be used to add CSS to an HTML document.
- 5.2 Explain the importance of the CSS Box Model.
- 5.3 Illustrate how you can use the class and id selectors in CSS and HTML.
- 5.4 Identify and explain the effect of the values that can be assigned to the CSS **position** property.

Individual Exercise 6: JavaScript (JS) Programming

- 6.1 Explain the major advantages of using client-side JavaScript.
- 6.2 Provide the different ways (for both HTML4 and HTML5) of adding JavaScript to an HTML document.
- 6.3 Find the error(s) in each of the following code segments and explain how to solve it:
 - a.

```
x=1;
while(x<=5) ;
{
    ++x;
}
```
 - b.

```
switch(n)
{
    case 1
        document.write("The number is 1");
    case 2
        document.write("The number is 2");
    default
        document.write("Number not found");
    break;
}
```
- 6.4 Given the following code segment, identify and explain the scope of x, y, output() and cube().

```
var x;
function output()
{
    for(x=1; x<=5; x++)
    {
        document.write(cube(x)+"<br>");
    }
}
function cube(y)
```

```
{  
    return y*y*y;  
}
```

- 6.5 Explain the different possible ways that can be used to retrieve the contents of a text box called name, from a JavaScript
- 6.6 Using an example, explain how you can redirect a user from one page to another using JavaScript.
- 6.7 What is a JavaScript cookie?
- 6.8 Using examples, explain how you can create, read and delete a cookie.

Individual Exercise 7: Hypertext Preprocessor (PHP)

- 7.1 Briefly explain each of the two ways used to display data on a browser in PHP.
- 7.2 Use an example to explain the usage of the PHP_SELF variable.
- 7.3 What are the differences between the \$_GET and the \$_POST superglobals.

Section B: Lab Exercises

Lab Exercise 1: Hypertext Markup Language (HTML) Basics

1.1 Use HTML tables to design the following web page layout:

<p align="center">BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY</p>	
<p><u>Internet Programming</u></p> <p><u>Operating Systems</u></p> <p><u>Distributed Systems</u></p> <p><u>Information Systems</u></p> <p><u>Software Development</u></p>	<p>Internet Programming</p> <p>Module Description</p> <p>Technologies including the Internet and the World Wide Web and more specifically, their use in electronic-commerce are reshaping the way that business leaders think about management, strategy and business design. This module represents a combination and application of the skills and knowledge that managers encounter in their online activities.</p> <p>Students will have to develop, implement and maintain a complete interactive e-commerce web site using appropriate design techniques. In addition, students will be prepared as current and future executives, managers and strategists to create value in the networked economy.</p>
<p align="center">Student Number - CTI Education Group® 2016</p>	

Insert appropriate text or image for the logo and centre aligned links as shown above. Insert your student number as footer in the format shown above. Duplicate the web page you have created until you have the following web pages:

```
internetProgramming.html
operatingSystems.html
distributedSystems.html
informationSystems.html
softwareDevelopment.html
```

Insert the module description for each module as defined in the CTI 2016 prospectus or download the CTI BSc IT Module Description (https://www.cti.ac.za/wp-content/uploads/2013/12/2015_CTI_BSc-IT_Module-Description_Final1.pdf). Make sure to include appropriate page names for the links on your web pages.

You can use any other HTML component (headings, rules, marquees, images, lists, etc), as learned in unit 2.

Save everything in the folder structure **lab/01** (root folder **lab** containing a folder called **01**)

Lab Exercise 2: Hypertext Markup Language (HTML) Forms

Copy and paste folder **01** (from lab exercise 1) and rename the copy to **02**. Both **01** and **02** will have the root directory **lab**.

Edit the pages in folder **02** as follows:

- 2.1 Insert another link on the navigation and call it "**Contact Us**".
- 2.2 Create another page called `contacts.html`. This page must resemble the one shown below:

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY		
<p><u>Internet Programming</u></p> <p><u>Operating Systems</u></p> <p><u>Distributed Systems</u></p> <p><u>Information Systems</u></p> <p><u>Software Development</u></p> <p style="text-align: center;"><u>Contact Us</u></p>	<p>Name: <input style="width: 100%;" type="text"/></p> <p>E-Mail: <input style="width: 100%;" type="text"/></p> <p>Cell: <input style="width: 100%;" type="text"/></p> <p>Select Modules:</p> <div style="border: 1px solid black; padding: 5px;"> <div style="display: flex; justify-content: space-between;"> <div> <input type="checkbox"/> Programming in Java <input type="checkbox"/> Procedural Programming <input type="checkbox"/> Database Systems </div> <div> <input type="checkbox"/> Project Management <input type="checkbox"/> Network Technologies <input type="checkbox"/> Computer Systems </div> </div> </div> <p style="text-align: center; margin-top: 10px;"> <input type="button" value="SUBMIT"/> </p>	<div style="border: 1px solid black; padding: 5px;"> Address Telephone Fax e-mail </div>
Student Number - CTI Education Group® 2016		

`contacts.html`

Lab Exercise 3: HTML and Cascading Style Sheet (CSS)

Copy and paste folder **02** and rename it to **03**. Edit the pages as follows:

- 3.1 Change the layout of the pages in folder **03** from using HTML table to CSS. The CSS file must be saved in a folder called **css**.
- 3.2 Apply CSS to the navigation and change its appearance to the following:

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY	
Internet Programming	
Operating Systems	
Distributed Systems	
Information Systems	
Software Development	
Contact Us	
Student Number - CTI Education Group® 2016	

- 3.3 Apply appropriate hover effects to the navigation and make sure that all the HTML formatting tags on the pages have been replaced with CSS.

Hint: Use unordered list for the navigation and apply CSS to the list accordingly.

Lab Exercise 4: JavaScript (JS) Basics

- 4.1 Using JavaScript loops, write a script that outputs the following:

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

- 4.2 Write JavaScript code that will display all the even numbers between 1 and 100. The script must start a new line for every multiple of 20.
- 4.3 Write a JavaScript function that takes an integer value and returns the number with its digits reversed. For example, given 1234, the function must return 4321. The function must use prompt box to get user input.

Apply appropriate exception handling and customize a message to inform the users not to enter non-numeric values. The result of this script must be displayed on the status bar.

- 4.4 Write a JavaScript that inputs a telephone number as a string in the form (000) 000-0000. The script should use string method split to extract the area code as a token, the first three digits of the phone number as a token and the last four digits of the phone number as a token. Display the area code and the seven digit phone number on an alert in the following format:

```
Area code: 000
Phone Number: 000 0000
```

- 4.5 Write a JavaScript to display all the objects within the window object on a browser.
- 4.6 Given that a year will be a leap year if it is divisible by 4 but not by 100 and if a year is divisible by 4 and by 100, it is not a leap year unless it is also divisible by 400. Use a JavaScript prompt box to accept a year (integer e.g. 2002) and decide whether the given year is a leap year or not. Display the output using a JavaScript alert.

Lab Exercise 5: JavaScript (JS) and HTML Forms

Copy and paste folder **03** you created in lab exercise 3. Rename the copy to **04**. Create a sub-folder in folder **04** and rename it to **js**.

Apply the following:

- 5.1 Use external JavaScript to validate all the objects in the contact form of the `contacts.html` page. You must use regular expressions to validate the email and telephone field. The form can only submit if at least one module is selected.
- 5.2 If the form passes the validation, welcome the user and display the form data on a web browser in any user friendly format.
- 5.3 Add another page called "`home.html`" to your work and update the navigation as required. The `home.html` page must have the following:

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY	
Home	Include a series of rotating IT related images here. Must have at least three images
Internet Programming	
Operating Systems	
Distributed Systems	
Information Systems	
Software Development	Include a description of the BSc in IT here.
Contact Us	
Student Number - CTI Education Group® 2016	

All the images must be saved in a sub-folder called **images**.

- 5.4 Insert a JavaScript cookie on the `home.html` page. The cookie must alert the user to enable cookies on the client computer if the cookies are disabled. Otherwise, set a cookie called **modules** on the client computer. The cookie must expire after 30 days from the last visit.

Lab Exercise 6: Hypertext Preprocessor (PHP) Basics

- 6.1 Create a PHP Script that contains five different variables. Assign values to the variables and use the `is_*` family of functions to test and print the type of the variables.
- 6.2 Create a PHP function that accepts four string variables and return a string that contains an HTML table element, enclosing each of the variables in separate table cells.
- 6.3 Create an array of doubles and integers. Loop through the array, converting each element to a floating point number with precision of 2. Right align the output within a field of 20 characters.

Section C: Tutorials

Tutorial 1: CSS Horizontal Navigation Bar

In the following tutorial, we are going to learn how we can use CSS and HTML to create a horizontal navigation bar that resembles the one shown below:



Figure 1.1 Horizontal Navigation Bar

Source: Taruvinga 2016

To start with, we need to create a folder (home directory) and save it for example as **mysite**. Within the **mysite** home directory, create a sub folder and name it **css**. We are going to use the **css** sub folder to store our **external css** file(s).

Create a text file called **index.html** and save it within the **mysite** home directory. Create another text file and save it as **myStyle.css** within the **css** sub folder.

Open the **index.html** file and type the following:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Horizontal Navigation Bar</title>
5      <link rel="stylesheet" href="css/myStyle.css">
6  </head>
7  <body>
8      <nav>
9          <ul>
10             <li><a class="active" href="#home">Home</a></li>
11             <li><a href="#blog">Our Blog</a></li>
12             <li><a href="#contact">Contact Us</a></li>
13             <li><a href="#about">About Us</a></li>
14          </ul>
15      </nav>
16 </body>
17 </html>
```

Figure 1.2 HTML 5 index.html

Source: Taruvinga 2016

This is an HTML 5 document with the title "Horizontal Navigation Bar" and a link to the external css file called **myStyle.css** saved in the **css** sub folder.

Within the body tags, we have unordered list of links. These are enclosed within the nav element (HTML 5). Before HTML 5 and the nav element, we could use the following:

```
<ul class="nav">
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#blog">Our Blog</a></li>
  <li><a href="#contact">Contact Us</a></li>
  <li><a href="#about">About Us</a></li>
</ul>
```

Figure 1.3: Pre-HTML 5 and the nav tags.

Source: Taruvinga 2016

We must always use unordered lists as shown above for CSS navigation bars. This allows screen readers to pause between each link instead of reading all the links as a sentence.

The output for the HTML 5 code in Figure 1.2 will be as follows:

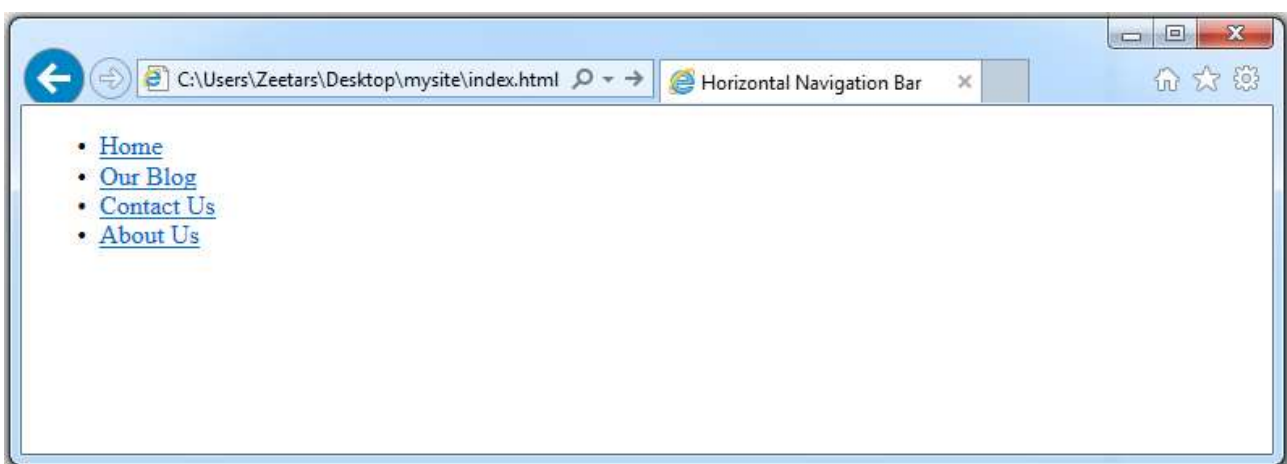


Figure 1.4: HTML 5 navigation bar without CSS.

Source: Taruvinga 2016

The next step is to open our **myStyle.css** file and add some CSS rules to format our navigation bar. Firstly, let us remove the bullets on the lists by adding the following to our CSS file:

```
1 ul{
2     list-style-type:none;
3 }
```

Figure 1.5: CSS to remove bullets on navigation bar

Source: Taruvinga 2016

If we save our file and refresh **index.html**, we get the following:

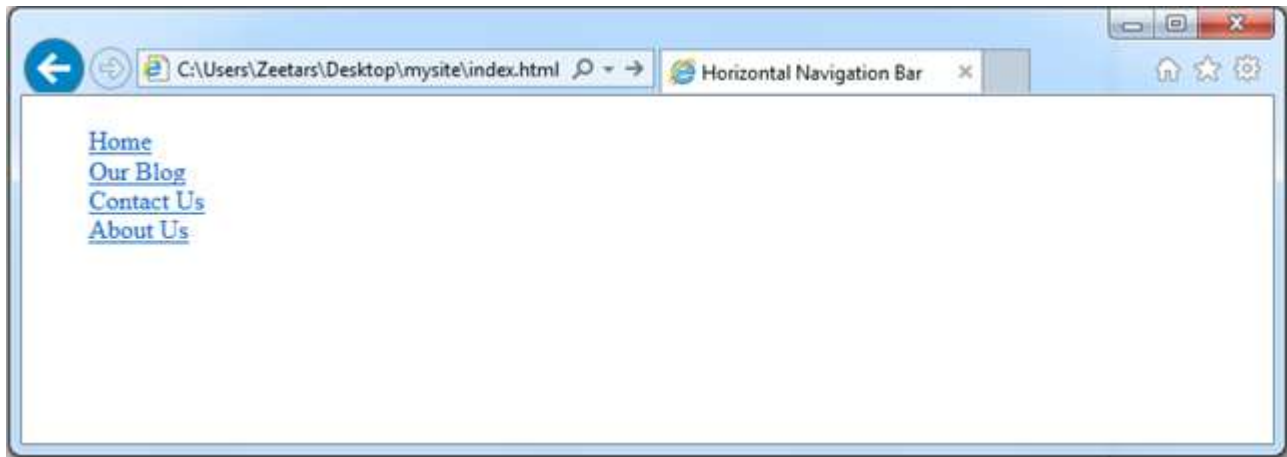


Figure 1.6: Navigation bar without bullets.

Source: Taruvinga 2016

We managed to remove the bullets, however, we still have some underlining. This is caused by the anchor (<a>) tags. To remove the underlining, let us insert the following in our CSS file:

```
li a{  
    text-decoration:none;  
}
```

Figure 1.7: CSS to remove the anchor (<a>) underlining.

Source: Taruvinga 2016

After saving our CSS file and refreshing the **index.html**, the output will be as follows:

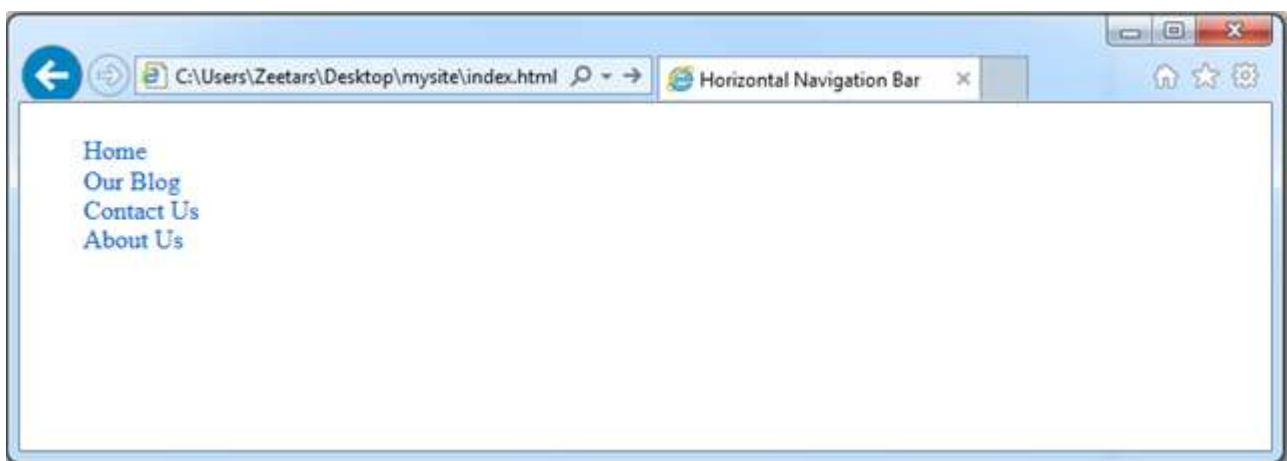


Figure 1.8: HTML 5 navigation bar without underlining.

Source: Taruvinga 2016

Our lists are sitting within the nav tags. Now we need to define the width, height and background color of the nav element. This can be achieved by adding the following CSS rule:

```

nav{
    width:960px;
    height:40px;
    background-color:#333333;
}

```

Figure 1.9: Width, Height and Background Color for the Navigation Bar.

Source: Taruvinga 2016

The output will be as follows:

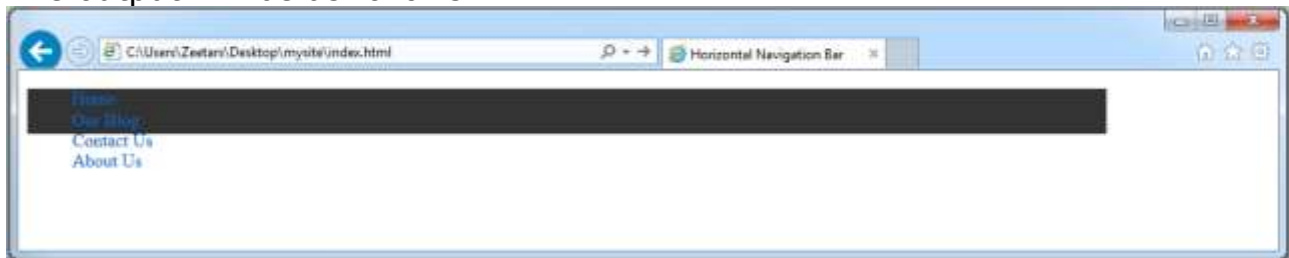


Figure 1.10: Horizontal Navigation Bar

Source: Taruvinga 2016

We need the ul element to have the same width and height of the nav element. This can be achieved by changing the ul selector in the CSS file to the following:

```

ul{
    list-style-type:none;
    width:100%;
    height:40px;
}

```

Figure 1.11: Adding width and height of the ul element.

Source: Taruvinga 2016

So far our CSS file is as follows:

```

1  nav{
2      width:960px;
3      height:40px;
4      background-color:#333333;
5  }
6  ul{
7      list-style-type:none;
8      width:100%;
9      height:40px;
10 }
11 li a{
12     text-decoration:none;
13 }

```

Figure 1.12: CSS file

Source: Taruvinga 2016

Our links are displayed from the top going towards the bottom of the page. We need them to be displayed from left to right. To achieve this, we add the following CSS rule:

```
li{  
    float:left;  
}
```

Figure 1.13: Using float to change the arrangement of list elements.

Source: Taruvinga 2016

The output will be as follows:

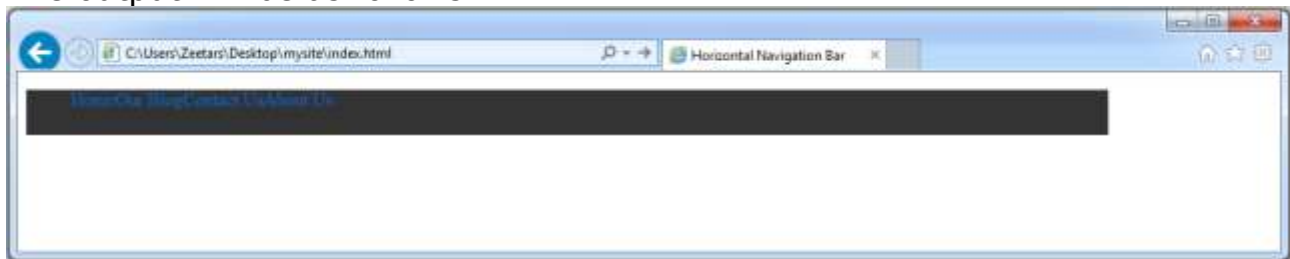


Figure 1.14: Using the float property.

Source: Taruvinga 2016

We need to align the links more to the left by adding the padding property to the ul selector in the CSS file as follows:

```
ul{  
    list-style-type:none;  
    width:100%;  
    height:40px;  
    padding:0;  
}
```

Figure 1.15: Adding the padding property

Source: Taruvinga 2016

The output will be as follows:



Figure 1.16: After adding the padding property

Source: Taruvinga 2016

We have to define the padding for each anchor so that there will be some space between anchor texts. This can be achieved by editing our anchor selector to the following:

```
li a{  
    text-decoration:none;  
    padding-left:10px;  
    padding-right:10px;  
}
```

Figure 1.17: Using padding to add space between anchor texts

Source: Taruvinga 2016

The output with space between anchor texts will be as follows:

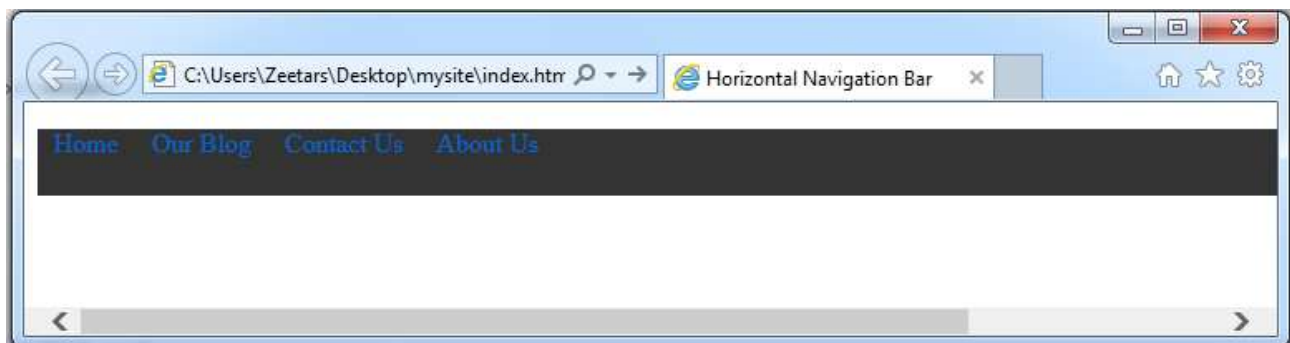


Figure 1.18: Using padding to add space between anchor texts

Source: Taruvinga 2016

We need to align (vertical alignment) the anchor text to the middle of the navigation bar. To achieve this, we define the **line-height** of the list item (li) as follows:

```
li{  
    float:left;  
    height:40px;  
    line-height:40px;  
}
```

Figure 1.19: Using line-height property to define the vertical alignment of text

Source: Taruvinga 2016

The output will be as follows:

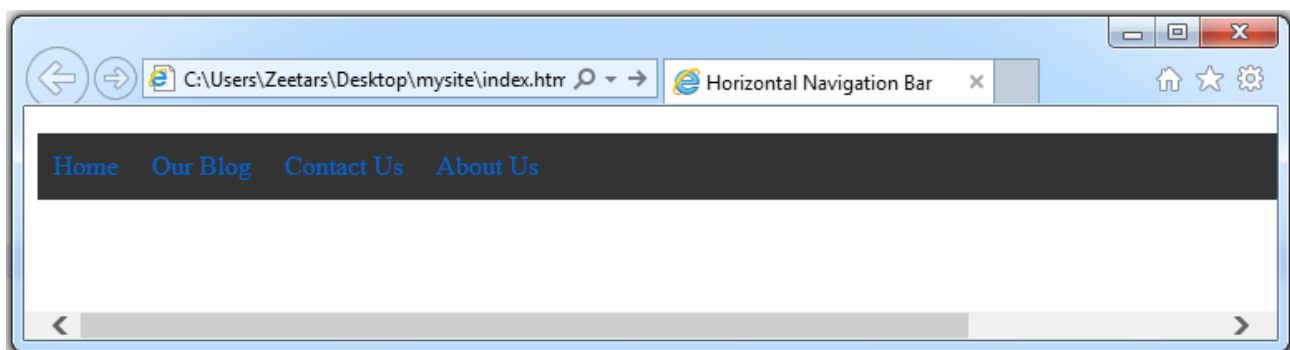


Figure 1.20: Using line-height property to define the vertical alignment of text

Source: Taruvinga 2016

The next step is for us to change the anchor text color to white by editing the anchor (**a**) selector in the CSS file to the following:

```
li a{  
    text-decoration:none;  
    padding-left:10px;  
    padding-right:10px;  
    color:#FFFFFF;  
}
```

Figure 1.21: Defining the color for the anchor text

Source: Taruvinga 2016

Here is the output:

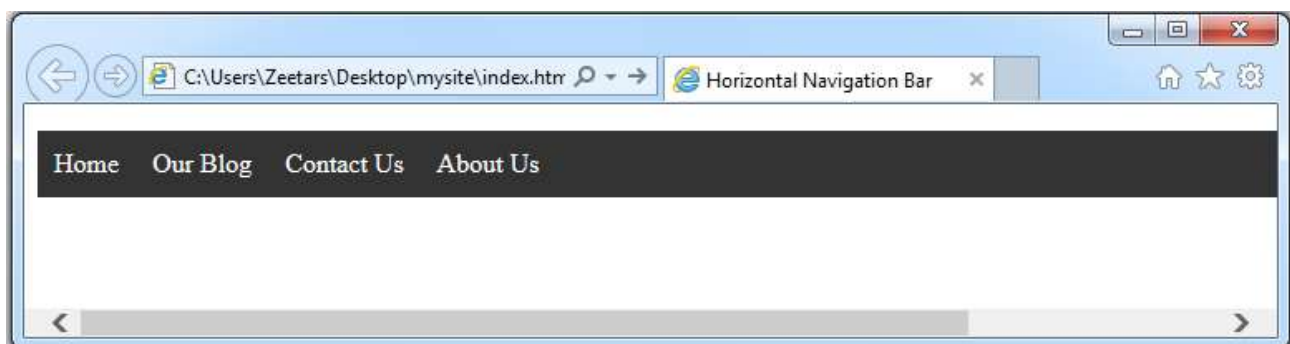


Figure 1.22: Defining the color for the anchor text

Source: Taruvinga 2016

We need to apply the **hover effect** to our anchor. This is used to change the text color, background color, or add other effects to the anchor when we put the cursor (mouse pointer) on top of the anchor. In this case, we need to change the background color to white and text color to blue. To achieve this we add the following to the CSS file:

```
li a:hover{  
    background-color: #FFFFFF;  
    color:#0000FF;  
}
```

Figure 1.23: Adding a hover effect

Source: Taruvinga 2016

Output will be as follows:

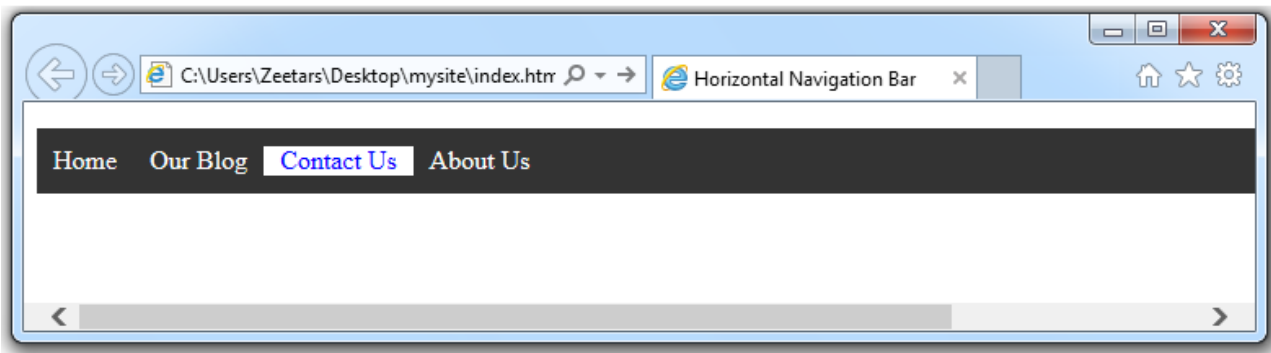


Figure 1.24: The hover effect

Source: Taruvinga 2016

In Figure 1.24 above, the anchor is covering only the section with a white background on the "Contact Us" link. We want it to cover the entire height of the navigation bar. To achieve this, we display the anchor as "block" by editing the anchor selector in the CSS file to the following:

```
li a{  
    text-decoration:none;  
    padding-left:10px;  
    padding-right:10px;  
    color:#FFFFFF;  
    display:block;  
}
```

Figure 1.25: The display property

Source: Taruvinga 2016

The output will be as follows:

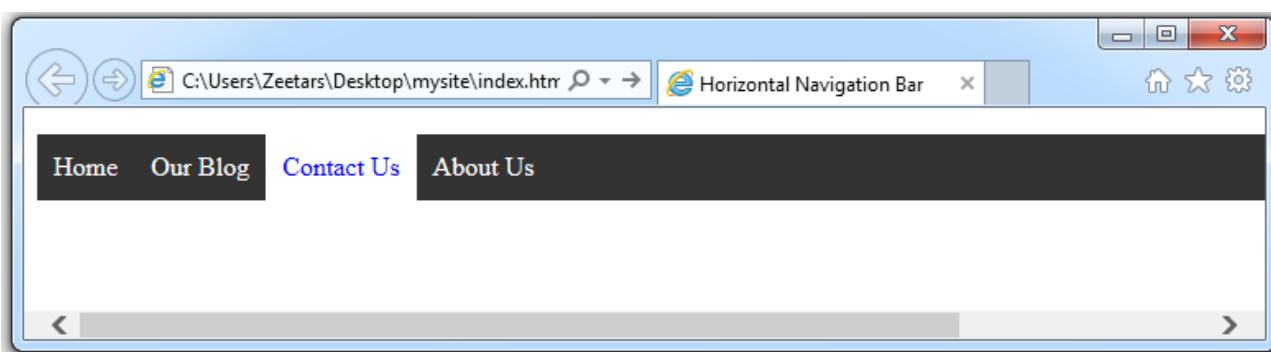


Figure 1.26: The display property

Source: Taruvinga 2016

Lastly, we need to center align the navigation bar on our web browser. To achieve this, we need to edit the nav selector in the CSS file as follows:

```
nav{  
    width:960px;  
    height:40px;  
    background-color:#333333;  
    margin:0 auto;  
}
```

Figure 1.27: Using "margin:0 auto" to center align web page content

Source: Taruvinga 2016

The final navigation bar will be as follows:



Figure 1.28: CSS Horizontal Navigation Bar

Source: Taruvinga 2016

Our final CSS file is as follows:

```
1  nav{
2      width:960px;
3      height:40px;
4      background-color:#333333;
5      margin:0 auto;
6  }
7  ul{
8      list-style-type:none;
9      width:100%;
10     height:40px;
11     padding:0;
12 }
13 li{
14     float:left;
15     height:40px;
16     line-height:40px;
17 }
18 li a{
19     text-decoration:none;
20     padding-left:10px;
21     padding-right:10px;
22     color:#FFFFFF;
23     display:block;
24 }
25 li a:hover{
26     background-color: #FFFFFF;
27     color:#0000FF;
28 }
```

Figure 1.29: The final CSS file

Source: Taruvinga 2016

Tutorial 2: PHP Sessions

In the following tutorial, we are going to learn how we can use PHP Sessions to store information that can be used across multiple pages. We need two pages:

content.php: This will be a page with any restricted information. Users can gain access to this page only after logging in.

index.php: A page containing the login form used to authenticate people before viewing the restricted information on **content.php** page.

The **index.php** page must start a session, store the username and redirect a user to **content.php** after successful login. The stored username will then be used by the **content.php** page to determine whether someone is trying to access the restricted content after logging in or by visiting the page directly. If someone is trying to access the restricted content directly (without logging in), the **content.php** page must redirect the user to the **index.php** page.

Let us create the home directory (call it mysite) for our web page and save it in our **local server**. I am using **XAMPP** as my local server. Create and save two pages (index.php and content.php) in the **mysite** directory.

Open **index.php** and insert the following code:

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>PHP Sessions</title>
6  </head>
7  <body>
8      <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
9          <table align="center">
10             <tr><th colspan="2">::Login::</th></tr>
11             <tr><td colspan="2"><?php ?></td></tr>
12             <tr><td>Username:</td><td><input type="text" name="username"></td></tr>
13             <tr><td>Password:</td><td><input type="password" name="password"></td></tr>
14             <tr><td align="right"><input type="submit" value="Login" name="login"></td>
15             <td><input type="reset" value="Clear"></td></tr>
16          </table>
17      </form>
18  </body>
19 </html>

```

Figure 2.1: **index.php**

Source: Taruvinga 2016

This is an HTML 5 document with a form starting from line 8 to line 17. We used the PHP **\$_SERVER['PHP_SELF']** as the value for the **action attribute** of the opening form tag in line 8 so that when we click the login

button in line14, the form content will be submitted to the same page (index.php). We used HTML tables to align the objects within the form. We will be using line 11 to display an error message if the login details are incorrect. We have text boxes in Line 12 and 13 for Username and Password respectively.

The output of the code in Figure 2.1 is as follows:

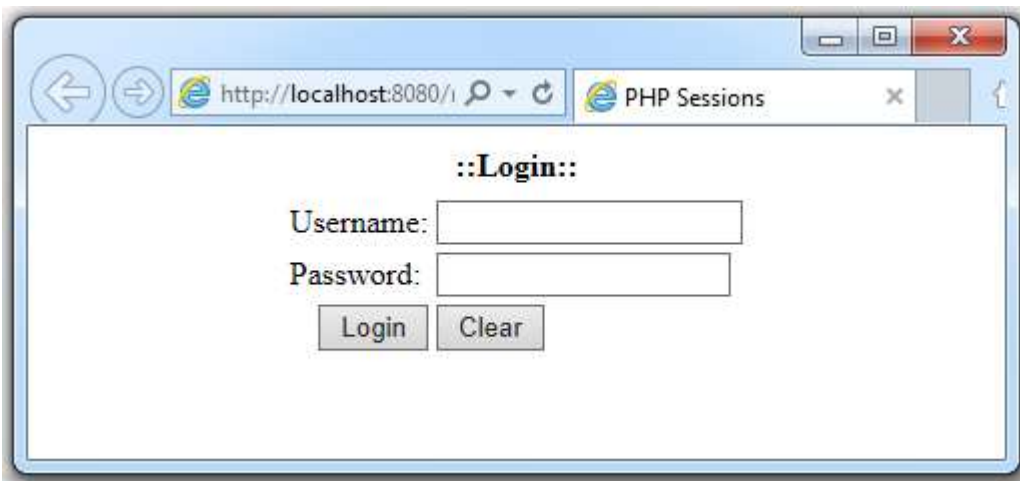


Figure 2.2: Login Screen

Source: Taruvinga 2016

Now, we need to insert a PHP script that is going to process the details posted from the form. This script is going to be on the same page (index.php) just before any HTML code:

```
1 <?php
2 $errorMessage="";
3 if(isset($_POST['login']))
4 {
5     $username=$_POST['username'];
6     $password=$_POST['password'];
7     if($username=="zeetars" && $password=="12345")
8     {
9         header('Location:content.php');
10    }
11    else
12    {
13        $errorMessage="Incorrect Login Details!!";
14    }
15 }
16 ?>
```

Figure 2.3: PHP Script to process the form details

Source: Taruvinga 2016

In Line 2, we setting a variable **\$errorMessage** to any empty string. This variable will be used to customise an error message if the login details are incorrect. Line 3 is as follows:

```
if(isset($_POST['login']))
```

We are using the **isset()** function to test if a form element with a name called **login** (our submit button) is activated (clicked). If this button is clicked, then we collect the username and password from the form into PHP variable called \$username and \$password in Line 5 and 6. Line 7 is used to test if the username and password posted is the same as "zeetars" and "12345" respectively. In this case, the username and password is hardcoded in our script. When you are working with database driven websites, these values must be stored in a database.

If the username and password matches, we redefine the header to redirect the user to the **content.php** page as shown in Line 9. If the login details are incorrect, we set the **\$errorMessage** variable to "Incorrect Login Details!!" as shown in Line 13. The error message will be displayed in Line 11 of the login form in Figure 2.1, hence we need to echo the error message in that line as follows:

```
<tr><td colspan="2"><?php echo $errorMessage; ?></td></tr>
```

The combined PHP script and HTML code will be as follows:

```

1  <?php
2      $errorMessage="";
3      if(isset($_POST['login']))
4      {
5          $username=$_POST['username'];
6          $password=$_POST['password'];
7          if($username=="zeetars" && $password=="12345")
8          {
9              header('Location:content.php');
10             }
11             else
12             {
13                 $errorMessage="Incorrect Login Details!!";
14             }
15         }
16     ?>
17     <!DOCTYPE html>
18     <html lang="en">
19     <head>
20         <meta charset="utf-8">
21         <title>PHP Sessions</title>
22     </head>
23     <body>
24         <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
25             <table align="center">
26                 <tr><th colspan="2">::Login::</th></tr>
27                 <tr><td colspan="2"><?php echo $errorMessage; ?></td></tr>
28                 <tr><td>Username:</td><td><input type="text" name="username"></td></tr>
29                 <tr><td>Password:</td><td><input type="password" name="password"></td></tr>
30                 <tr><td align="right"><input type="submit" value="Login" name="login"></td>
31                 <td><input type="reset" value="Clear"></td></tr>
32             </table>

```

Figure 2.4: Combined PHP and HTML code

Source: Taruvinga 2016

After successful login, we are redirecting the user to **content.php**. We want the **content.php** page to welcome the user by his username. Since we are redirecting the user, we cannot be able to use the post or even the get method to send values from our login form to the **content.php** page. To send any values, we need to store them in **PHP Sessions**.

To start a PHP Session, we use:

```
session_start();
```

After starting a session, we need to set the session variables. In this case, we only need to set the username as follows:

```
$_SESSION['username']=$username;
```

We can edit our PHP script on the **index.php** page to the follows:


```

1  <?php
2      session_start();
3      $errorMessage="";
4      if(isset($_POST['login']))
5      {
6          $username=$_POST['username'];
7          $password=$_POST['password'];
8          if($username=="zeetars" && $password=="12345")
9          {
10             $_SESSION['username']=$username;
11             header('Location:content.php');
12         }
13         else
14         {
15             $errorMessage="Incorrect Login Details!!";
16         }
17     }
18  ?>

```

Figure 2.5: Starting a PHP Session

Source: Taruvinga 2016

Line 2 is starting a session and Line 10 is setting the session variable "username" to the contents of the \$username variable.

This means that we can access the username from any page within our site as long as the session is not destroyed.

Open the **content.php** page and type the following:

```

1  <?php
2      session_start();
3  ?>
4  <!DOCTYPE html>
5  <html lang="en">
6  <head>
7      <meta charset="utf-8">
8      <title>PHP Sessions</title>
9  </head>
10 <body>
11     <h2>Welcome <?php echo $_SESSION['username']; ?></h2>
12     <?php session_destroy(); ?>
13 </body>
14 </html>

```

Figure 2.6: content.php page with session

Source: Taruvinga 2016

We must start the session in every page that we want to access the values stored in a session. In our case, Line 2 is starting the session on the **contact.php** page and Line 11 is accessing and displaying the contents of the session variable username. Line 12 is destroying the session.

Let us test our pages. If we visit our site, the login form (index.php) will be displayed as follows:

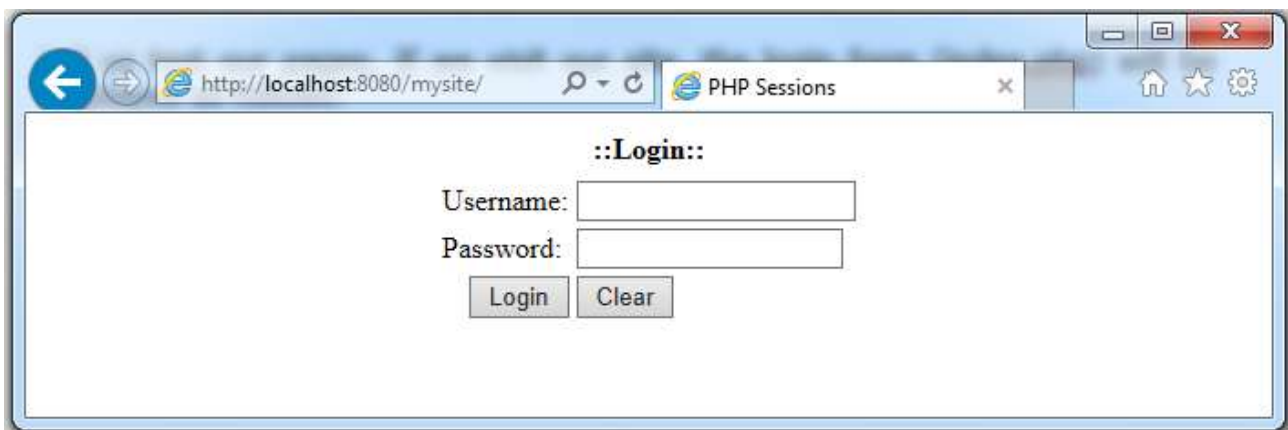
A screenshot of a web browser window. The address bar shows 'http://localhost:8080/mysite/'. The page title is 'PHP Sessions'. The main content area displays a login form titled '::Login:'. The form has two input fields: 'Username:' and 'Password:'. Below the fields are two buttons: 'Login' and 'Clear'.

Figure 2.7: Login Form

Source: Taruvinga 2016

If we enter wrong login details or click the "Login" button without any login details, the error message will be set and appear on the form as follows:

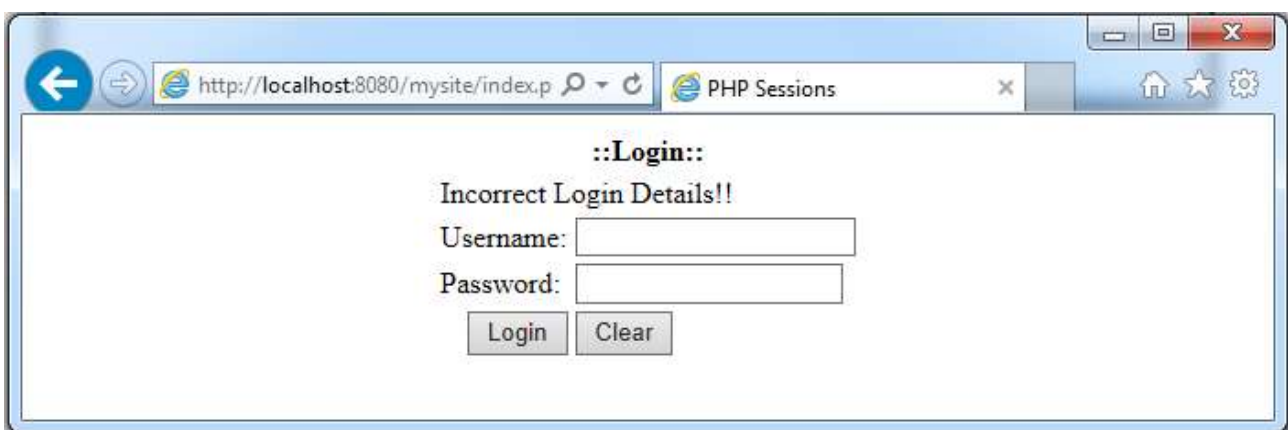
A screenshot of a web browser window. The address bar shows 'http://localhost:8080/mysite/index.p'. The page title is 'PHP Sessions'. The main content area displays the same login form as in Figure 2.7, but with an error message 'Incorrect Login Details!!' displayed above the 'Username:' field. The 'Login' and 'Clear' buttons are still present.

Figure 2.8: Incorrect Login Details

Source: Taruvinga 2016

If we enter the correct login details, we will be redirected to the **content.php** page and the following will be displayed:

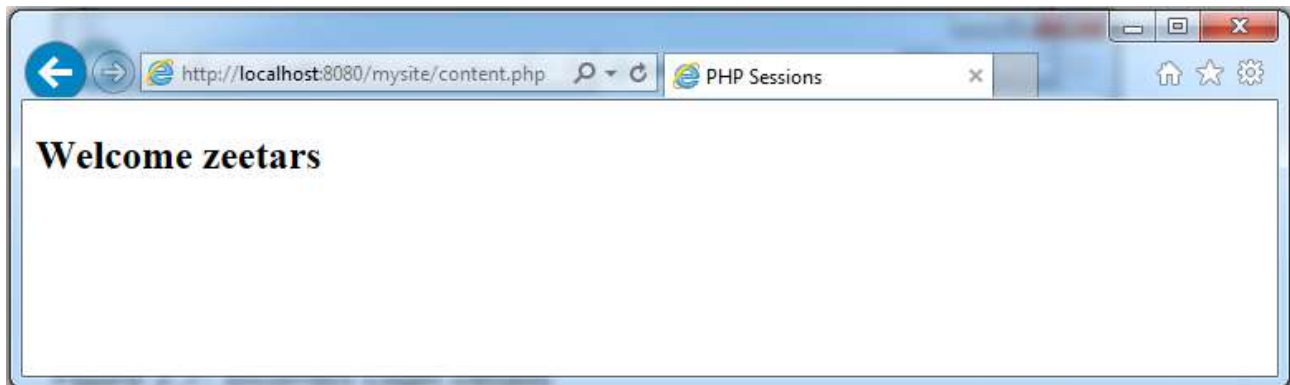


Figure 2.9: Incorrect Login Details

Source: Taruvinga 2016

However, if we try to visit the content.php page without logging in (by typing its URL in the address bar), an error message will be generated as follows:



Figure 2.10: Accessing the content.php page directly

Source: Taruvinga 2016

We can edit our **content.php** page to the following so that the user will be redirected to the login page (index.php) if the page is accessed directly:

```
1  <?php
2      session_start();
3      if(!isset($_SESSION['username']))
4      {
5          header('Location:index.php');
6      }
7  ?>
8  <!DOCTYPE html>
9  <html lang="en">
10 <head>
11     <meta charset="utf-8">
12     <title>PHP Sessions</title>
13 </head>
14 <body>
15     <h2>Welcome <?php echo $_SESSION['username']; ?></h2>
16     <?php session_destroy(); ?>
17 </body>
18 </html>
```

Figure 2.11: Using the **!isset()** (not `isset()`)

Source: Taruvinga 2016

Line 3 will determine whether the username variable for our session is set or not. If it's set we can view the restricted content otherwise the user will be redirected back to the login page.