

LAB MANUAL

Web Engineering ***SEWE-346***



DEPARTMENT OF SOFTWARE ENGINEERING
FACULTY OF ENGINEERING & CS
NATIONAL UNIVERSITY OF MODERN LANGUAGES
ISLAMABAD

Preface

This lab manual has been prepared to facilitate the students of Web engineering in studying and implementing Web Technologies. The students will Learn technologies and web development languages like HTML , CSS, JavaScript and PHP. The students can learn and implement ways to create web pages and websites. After the completion of this course students will be able to practically develop websites including front end and backend of websites.-

Tools/ Technologies

- HTML
- CSS
- Javascript
- PHP

TABLE OF CONTENTS

Preface	2
Tools/ Technologies	2
LAB 1: Introduction to HTML Basics	4
LAB 2: Exploring HTML Semantic elements, File Paths, Text formatting.....	9
LAB 3: Exploring HTML for website development.	13
Lab 04: Exploring HTML forms elements	19
LAB 5: Exploring CSS for website development	23
LAB 6 : Exploring CSS for website development	28
LAB 7 Positioning elements using CSS	29
LAB 8 : Mid Lab Exam.....	45
LAB 9: Introduction to JavaScript for website development	46
LAB 10: Implementing JavaScript for Front-End development	2
LAB 11: Lab Exam.....	5
LAB 12: Introduction to PHP for website development	6
LAB 13: Basic PHP Commands for website backend	12
LAB 14: Functions in PHP	20
LAB 15: MYSQL operations in PHP.....	31
LAB 16: Project Evaluation	38

LAB 1: Introduction to HTML Basics

Objectives

1. To learn how to inspect code.
2. Understand the working of HTML and its basic tags
3. Browser inspection

Tool:

1. We are going to use notepad++ Step 1: Open it
2. Write Some HTML Save the HTML Page
3. You can use either .htm or .html as file extension. There is no difference, it is up to you.
4. Step 4: View the HTML Page in Your Browser.

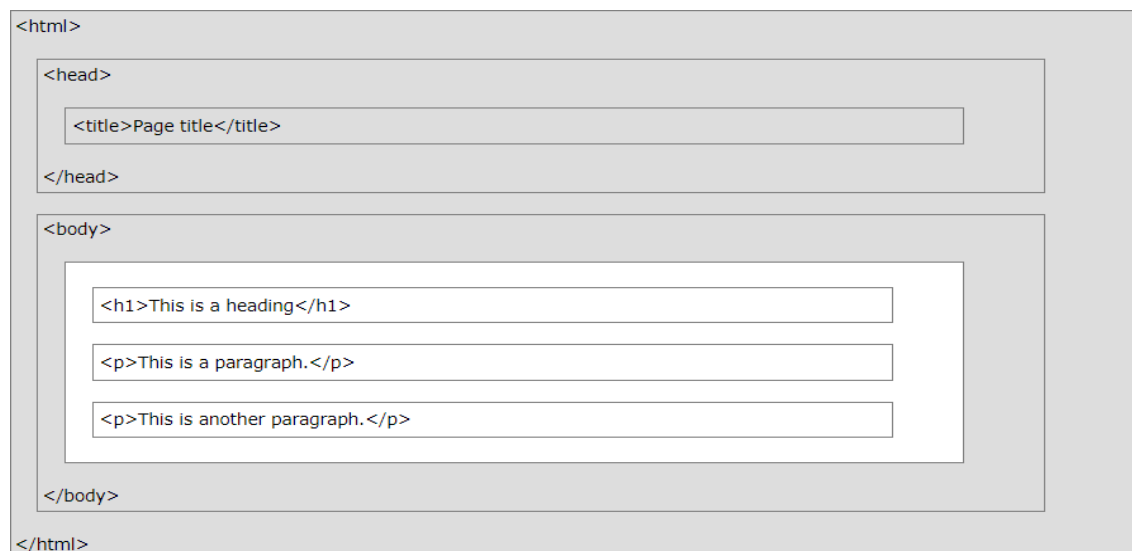
Theoretical Description

1 What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language.
- HTML describes the structure of Web pages using markup.
- HTML elements are the building blocks of HTML pages.
- HTML elements are represented by tags.
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on.
- Browsers do not display the HTML tags but use them to render the content of the page.
- Father of HTML is **Tim Burner Lee**.

HTML Basic structure:



2 HTML Tags:

HTML tags are element names surrounded by angle brackets:

`<tagname> content goes here... </tagname>`

- `<tagname>` is called as ‘**Starting Tag**’.
- `</tagname>` is called as ‘**Closing/Ending Tag**’.

3 Types of HTML Tags:

Different types of HTML tags are explained in the following:

1. HTML Document:

- All HTML documents must start with a document type declaration:
`<!DOCTYPE html>`.
- The HTML document itself begins with `<html>` and ends with `</html>`.
- The visible part of the HTML document is between `<body>` and `</body>`.

Example

```
<!DOCTYPE >
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

The `<!DOCTYPE>` declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The `<!DOCTYPE>` declaration is not case sensitive

2 HTML Headings:

- HTML headings are defined with the `<h1>` to `<h6>` tags.
- `<h1>` defines the most important heading.
- `<h6>` defines the least important heading:
`<h1>`This is heading 1`</h1>`
`<h2>`This is heading 2`</h2>`
`<h3>`This is heading 3`</h3>`
`<h4>`This is heading 4`</h4>`
`<h5>`This is heading 5`</h5>`
`<h6>`This is heading 6`</h6>`

3 HTML Paragraphs:

- HTML paragraphs are defined with the `<p>` tag:

4 HTML Links/Anchor Tag:

- i. HTML links are defined with the **<a>** tag:
- ii. The link's destination is specified in the **href attribute**.

`This is a link`

5. Html link syntax

`link text`

The most important attribute of the `<a>` element is the `href` attribute, which indicates the link's destination.

The link text is the part that will be visible to the reader.

Clicking on the link text, will send the reader to the specified URL address.

Example :to create a link to W3Schools.com

`Visit W3Schools.com!`

2. HTML Attributes:

Attributes provide additional information about HTML elements.

- All HTML elements can have **attributes**.
- Attributes provide **additional information** about an element.
- Attributes are always specified in **the start tag**.
- Attributes usually come in name/value pairs like: **name="value"**.

Attribute	Description
Alt	Specifies an alternative text for an image when the image cannot be displayed.
Disabled	Specifies that an input element should be disabled.
Href	Specifies the URL (web address) for a link.
Id	Specifies a unique id for an element.
Src	Specifies the URL (web address) for an image.
Style	Specifies an inline CSS style for an element.
Title	Specifies extra information about an element (displayed as a tool tip).

2 HTML <head> Element:

- The HTML **<head>** element has nothing to do with HTML headings.
- The **<head>** element is a container for metadata. HTML metadata is data about the HTML document. Metadata is not displayed.
- The **<head>** element is placed between the **<html>** tag and the **<body>** tag:

3 HTML Line Breaks:

- The HTML **
** element defines a **line break**.
- Use **
** if you want a line break (a new line) without starting a new paragraph:

`<p>This is
a paragraph
with line breaks.</p>`

4 How to View HTML Source?

Have you ever seen a Web page and wondered "Hey! How did they do that?"

View HTML Source Code:

Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in Edge), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element:

Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

LAB ACTIVITY/ CODE

```
<!DOCTYPE html>
<html>

<head>

  <title>My First HTML</title>
  <meta charset="UTF-8">

</head>
<body>

<p> The HTML head element contains meta data. </p>
<p> Meta data is data about the HTML document. </p>

</body>
</html>
```

LAB TASK

1. Build a simple Web page using Notepad or Notepad++ With a specific title. Your web page must have:
 - a. An image file
 - b. A description about the image
 - c. Some Description about the title.
2. Open Code Inspection via Developer tools. Experiment around Web pages and look to it via changing styles.

3. Explain the output of following code with reasoning and also list down the errors in it

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Links</h2>
<h3>HTML Links</h3>
<p>HTML links are defined with the a tag:</p>
<p>HTML links are defined with the no tag:</p>
</body>
<a href="https://www.IBM.com">This is IBM Website</a>
```


LAB 2: Exploring HTML Semantic elements, File Paths, Text formatting

Objectives

1. Learning most common HTML tags for website development
2. Learn HTML Semantic elements
3. Learn about Absolute and Relative File Paths

Theoretical Description

1 HTML File Paths

A file path describes the location of a file in a web site's folder structure.

File paths are used when linking to external files, like: Web pages, Images, Style sheets

JavaScripts

2 Absolute File Paths

An absolute file path is the full URL to a file:

File Path Examples

Path	Description
<code></code>	The "picture.jpg" file is located in the same folder as the current page
<code></code>	The "picture.jpg" file is located in the images folder in the current folder
<code></code>	The "picture.jpg" file is located in the images folder at the root of the current web
<code></code>	The "picture.jpg" file is located in the folder one level up from the current folder

3 HTML File Paths

.File path describes the location of a file in a web site's folder structure.

le paths are used when linking to external files, like:

- Web pages
- Images
- Style sheets
- JavaScripts

4 Absolute File Paths

In absolute file path is the full URL to a file:

```

```

5 Relative File Paths

.The relative file path points to a file relative to the current page.

.In the following example, the file path points to a file in the images folder located at the root of the current web:

Example

``

In the following example, the file path points to a file in the images folder located in the current folder:

``

In the following example, the file path points to a file in the images folder located in the folder one level up from the current folder:

``

5 HTML Text Formatting

HTML contains several elements for defining text with a special meaning.

Example

his text is bold

his text is italic

his is _{subscript} and ^{superscript}

6 HTML Formatting Elements

Formatting elements were designed to display special types of text:

- `` - Bold text
- `` - Important text
- `<i>` - Italic text
- `` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

7 HTML `` and `` Elements

he HTML `` element defines bold text, without any extra importance.

Example

``This text is bold``

ie HTML `` element defines text with strong importance. The content inside is typically displayed in bold.

Example

``This text is important!``

8 HTML `<i>` and `` Elements

ie HTML `<i>` element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.

p: The `<i>` tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

Example

`<i>`This text is italic`</i>`

ie HTML `` element defines emphasized text. The content inside is typically displayed in italic.

p: A screen reader will pronounce the words in `` with an emphasis, using verbal stress.

Example

``This text is emphasized``

9 HTML `<small>` Element

he HTML `<small>` element defines smaller text:

Example

`<small>`This is some smaller text.`</small>`

10 HTML `<mark>` Element

ie HTML `<mark>` element defines text that should be marked or highlighted:

Example

`<p>`Do not forget to buy `<mark>`milk`</mark>` today.`</p>`

11 HTML `` Element

ie HTML `` element defines text that has been deleted from a document. Browsers will usually strike a line through leted text:

Example

`<p>`My favorite color is ``blue`` red.`</p>`

12 HTML `<ins>` Element

The HTML `<ins>` element defines a text that has been inserted into a document. rowers will usually underline inserted text:

Example

`<p>`My favorite color is ``blue`` `<ins>`red`</ins>`.`</p>`

13 HTML `<sub>` Element

ie HTML `<sub>` element defines subscript text. Subscript text appears half a character below the normal line, and is metimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H₂O:

Example

`<p>`This is `_{`subscripted`}` text.`</p>`

14 HTML `<sup>` Element

he HTML `<sup>` element defines superscript text. Superscript text appears half character above the normal line, and is sometimes rendered in a smaller font. uperscript text can be used for footnotes, like WWW^[1]:

Example

`<p>`This is `^{`superscripted`}` text.`</p>`

15 `<abbr>` for Abbreviations

The HTML `<abbr>` tag defines an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".

Tip: Use the global title attribute to show the description for the Abbreviation/acronym when you mouse over the element.

Example

`<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>`

Lab Activity/Code

Lab Task

1. Create a Wiki page related to a topic of your choice. Create at least two HTML pages for references and link the main Wiki page to those references. Use necessary File paths to images and Add formatted textual information on your Wiki. Include a table like the following in your Wiki page
2. Create a Simple web page about any latest topic of your choice e.g “ emerging software technologies now days”. Following things should be considered 1. Use all the appropriate Html Commands 2. Write Description of topic 3. Use animated images as well as use those images as a link 4. Make usage of mailto
3. Explore HTML Emojis from the following link:
https://www.w3schools.com/html/html_emojis.asp

LAB 3: Exploring HTML for website development.

Objectives

- 1 To learn how to add tables in a html webpage.
2. Learning most common HTML tags for website development

Theoretical Description

1 HTML Tables

HTML tables allow web developers to arrange data into rows and columns.

Define an HTML Table

The `<table>` tag defines an HTML table. Each table row is defined with a `<tr>` tag. Each table header is defined with a `<th>` tag. Each table data/cell is defined with a `<td>` tag. By default, the text in `<th>` elements are bold and centered. By default, the text in `<td>` elements are regular and left-aligned

Example:







```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

2 HTML Table border

```
<style>
table, th, td {
  border: 1px solid black;
}
</style>
```

With the `border-style` or `border` property, you can set the appearance of the border

```
th, td {
  border-style: dotted;
}
```

- `dotted` 
- `dashed` 
- `solid` 
- `double` 
- `groove` 
- `ridge` 

- 10

```
th, td {
    background-color: #96D4D4;
}
```

[illegible]

```
<table style="width:100%">
  <tr>
    <th style="width:70%">Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
```

to make a cell span more than one row, use the **rowspan** attribute:

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
  </tr>
</table>
```

```
</tr>
</table>
```

Cell that spans two rows

To make a cell span more than one row, use the rowspan attribute.

Name:	Bill Gates
Telephone:	55577854
	55577855

```
<table>
<tr>
  <th>Name</th>
  <th colspan="2">Telephone</th>
</tr>
<tr>
  <td>Bill Gates</td>
  <td>55577854</td>
  <td>55577855</td>
</tr>
</table>
```

HTML Table - Add a Caption

To add a caption to a table, use the <caption> tag:

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
```


HTML Table Colgroup

If you want to style the two first columns of a table, use the `<colgroup>` and `<col>` elements.

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

The `<colgroup>` element should be used as a container for the column specifications.

```
<table style="width: 100%;">
<colgroup>
  <col span="2" style="background-color: #d6eeee">
</colgroup>
<tr>
<th>mon</th>
<th>tue</th>
<th>wed</th>
<th>thu</th>
<th>fri</th>
<th>sat</th>
<th>sun</th>
</tr>
```

Example

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada

```

<style>
table {
  font-family: arial, sans-serif;
  border-collapse: collapse;
  width: 100%;
}


td, th {
  border: 1px solid #dddddd;
  text-align: left;
  padding: 8px;
}
tr:nth-child(even) {
  background-color: #dddddd;
}
</style>
</head>
<body>
<table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>

```

Lab Task

- 1 Using table tag, Create a web page CV with your skills. Give it a name About me.html

About Me



Name: Muhammad Umer

Date of Birth: 04/06/1996

Gender: Male

Education Background

Sr. No.	Degree	Institute	City	CGPA/Grade
1.	BS Software Engineering	National University of Modern Languages	Islamabad	3.99
2.	Higher Secondary School Certificate	Punjab College of Information Technology	Rawalpindi	A
3.	Secondary School Certificate	Divisional Public School & College	D.G.Khan	A+

[Return to Home Page](#)

- 2 Create a web page containing description list , table styling, rowspan ,column span and column group

Lab 04: Exploring HTML forms elements

Objectives

Exploring html form elements and form tags

Theoretical Description

Form tags, Script tags, Radio buttons etc. are part of the control tags

1. Input Tag

- The **<input>** tag specifies an input field where the user can enter data.
- **<input>** elements are used within a **<form>** element to declare input controls that allow users to input data.
- An input field can vary in many ways, depending on the type attribute.

```
<form>
```

First name:

```
<input type="text" name="fname"><br>
```

Last name:

```
<input type="text" name="lname"><br>
```

```
<input type="submit" value="Submit">
```

```
</form>
```

First name:

Last name:

Note:

The **<input>** element is empty, it contains attributes only.

Use the **<label>** element to define labels for **<input>** elements.

2. Select Tag

- The **<select>** element is used to create a drop-down list.
- The **<option>** tags inside the **<select>** element define the available options in the list.

```
<select>
```

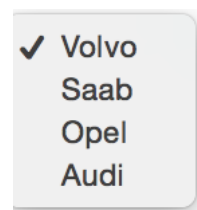
```
<option value="volvo">Volvo</option>
```

```
<option value="saab">Saab</option>
```

```
<option value="mercedes">Mercedes</option>
```

```
<option value="audi">Audi</option>
```

```
</select>
```



Note: The **<select>** element is a form control and can be used in a form to collect user input.

3. Button Tag

- The **<button>** tag defines a clickable button.
- Inside a **<button>** element you can put content, like text or images. This is the difference between this element and buttons created with the **<input>** element.

`<button type="button">Click
Me!</button>`

a. Types of buttons:

There are three types of buttons in html:

1. Simple Button:

The button is a clickable button.

`<button type="button">Simple Button</button>`

2. Submit Button:

The button is a submit button (submits form-data).

`<button type="submit">Submit Button</button>`

3. Reset Button:

The button is a reset button (resets the form-data to its initial values).

`<button type="reset">Reset Button</button>`

2 Access Identifiers/Selectors:

Selectors are patterns used to select the element(s) you want to style.

Attribute	Belongs to	Description
Id	Global Attributes	Specifies a unique id for an element
Name	<button>, <form>, <select>, <textarea>, <input>	Specifies the name of an element
Class	Global Attributes	Specifies one or more classnames for an element

Lab Task

1. Design a signup form for creating an account of student in the database system.

Sign Up Form

First Name	<input type="text"/>	Last Name	<input type="text"/>
Create Password	<input type="password"/>	Confirm Password	<input type="password"/>
<input type="submit" value="Submit"/>			

2 design following



NUML Student Management Information System

Username/Email

Password

Need an account? [Click here for Signup](#)

2. Lab assignment



Sign Up Form

First Name

Last Name

Choose your username

Create a Password

Confirm a Password

Birthday

Month 	Day <input type="text"/>	Year <input type="text"/>
---	--------------------------	---------------------------

Gender

- ☐ Male
- ☐ Female
- ☐ Other

Mobile phone

Country

☐ Yes, I agree to the terms

LAB 5: Exploring CSS for website development

Objectives

1. To learn how to use CSS in a HTML page
2. Learning most common CSS properties for website development

Theoretical Description

1 CSS (Cascading Style Sheets)

- CSS stands for Cascading Style Sheets.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- CSS saves a lot of work. It can control the layout of multiple web pages all at once.
- External stylesheets are stored in **CSS files**.

How to add CSS

- ✓ **Inline CSS:**
- ✓ **Internal CSS:**
- ✓ **External CSS:**

2 CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

1 The element Selector:

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Example:

```
p {  
    text-align:  
center; color:  
red;  
}
```

Source Code:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
> p {  
    text-align:  
center; color:  
red;  
}  
</head>  
<body>  
<p>Every paragraph will be affected  
by the style.</p>  
<p>Me too!</p>  
<p>And me!</p>  
</body>  
</html>
```

Output:

Every paragraph will be affected by the style.

Me too!

And me!

2 The id Selector:

The id selector uses the id attribute of an HTML element to select a specific element. The id of an element should be unique within a page, so the id selector is used to select one unique element!. To select an element with a specific id, write a hash (#) character, followed by the id of the element. The style rule below will be applied to the HTML element with id="para1":

Example:

```
#para1 {  
  text-align:  
  center; color:  
  red;  
}
```

Source Code:

```
<!DOCTYPE html>    </head>  
  
<html>              <body>  
  
<head>              <p id="#para1">Hello  
                    World!</p>  
  
<style  
>                  <p>This paragraph is not  
#para1             affected by the style.</p>  
{                 </body>  
  text-align:      </html>  
  center; color:   </html>  
  red;}  
</style>
```


Output:

Hello World!

This paragraph is not affected by the style.

3 The class Selector:

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class. In the example below, all HTML elements with class="center" will be red and center-aligned:

Example:

```
.center {  
  text-align:  
  center; color:  
  red;  
}
```

Source Code:

```
.center {  
text-align: center; color: red;  
}  
</style>  
</head>  
<body>  
<h1 class="center">Red and center-aligned heading</h1>  
<p class="center">Red and center-aligned paragraph.</p>  
</body>  
</html>
```

Output:

Red and center-aligned heading

Red and center-aligned paragraph.

4 CSS Borders

The CSS border properties allow you to specify the style, width, and color of an element's border.

I have borders on all sides.

I have a red bottom border.

I have rounded borders.

The **border-style** property specifies what kind Of border to display. The following values are allowed:

- **dotted** - Defines a dotted border
- **dashed** - Defines a dashed border
- **solid** - Defines a solid border
- **double** - Defines a double border
- **groove** - Defines a 3D grooved border. The effect depends on the border-color value
- **ridge** - Defines a 3D ridged border. The effect depends on the border-color value
- **inset** - Defines a 3D inset border. The effect depends on the border-color value
- **outset** - Defines a 3D outset border. The effect depends on the border-color value
- **none** - Defines no border
- **hidden** - Defines a hidden border
- The **border-style** property can have from one to four values (for the top border, right border, bottom border, and the left border)

5 CSS Margins

The CSS **margin** properties are used to create space around elements, outside of any defined borders. With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- margin-left

Example:

```
p {  
margin-top: 100px; margin-  
bottom: 100px; margin-right:
```

150px; margin-left: 80px;

}

CSS Padding

The CSS **padding** properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

Lab Task

Q1. Convert the following image into HTML 5 & CSS. Use semantic elements. HTML Code elements and Quotation blocks to improve the SEO of webpage.

 **Zozor**
Travel diaries

HOME BLOG RESUME CONTACT


Reflections on my holiday in the United States... [See article ▶](#)

 **I'M A GREAT TRAVELLER**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam nec sagittis massa. Nulla facilisi. Cras id arcu lorem, et semper purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis vel enim mi, in lobortis sem. Vestibulum luctus elit eu libero ultrices id fermentum sem sagittis. Nulla imperdiet mauris sed sapien dignissim id aliquam est aliquam. Maecenas non odio ipsum, a elementum nisi. Mauris non erat eu erat placerat convallis. Mauris in pretium urna. Cras laoreet molestie odio, consequat consequat velit commodo eu. Integer vitae lectus ac nunc posuere pellentesque non at eros. Suspendisse non lectus lorem.

Vivamus sed libero nec mauris pulvinar facilisis ut non sem. Quisque mollis ullamcorper diam vel faucibus. Vestibulum sollicitudin facilisis feugiat. Nulla euismod sodales hendrerit. Donec quis orci arcu. Vivamus fermentum magna a erat ullamcorper dignissim pretium nunc aliquam. Aenean pulvinar condimentum enim a dignissim. Vivamus sit amet lectus at ante adipiscing adipiscing eget vitae felis. In at fringilla est. Cras id velit ut magna rutrum commodo. Etiam ut scelerisque purus. Duis risus elit, venenatis vel rutrum in, imperdiet in quam. Sed vestibulum, libero ut bibendum consectetur, eros ipsum ultrices nisl, in rutrum diam augue non tortor. Fusce nec massa et risus dapibus aliquam vitae nec diam.

Phasellus ligula massa, congue ac vulputate non, dignissim at augue. Sed auctor fringilla quam quis porttitor. Praesent vitae dignissim magna. Pellentesque quis sem purus, vel elementum mi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Maecenas consectetur euismod urna. In hac habitasse platea dictumst. Quisque tincidunt porttitor vestibulum. Ut iaculis, lacus at molestie lacinia, ipsum mi adipiscing ligula, vel mollis sem risus eu lectus. Nunc elit quam, rutrum ut dignissim sit amet, egestas at sem.


ABOUT THE AUTHOR

Let me introduce myself: My name's Zozor. I was born on 23 November 2005.

A bit meager, is it not? This is why I've now decided to write my biography to let my readers know who I really am.



MY LAST TWEET

Hee-haw!
12/05 23:12

MY PICTURES



MY FRIENDS

» Pupi the rabbit
» Mr Baobab
» Kaiwaii
» Perceval.eu

» Ji
» Super cucumber
» Prince
» Mr Fan

LAB 6 : Exploring CSS for website development

Objectives

- 1 To learn how to use CSS in a HTML page
2. Learning advanced CSS properties for website development

Theoretical Description

Go to the link:

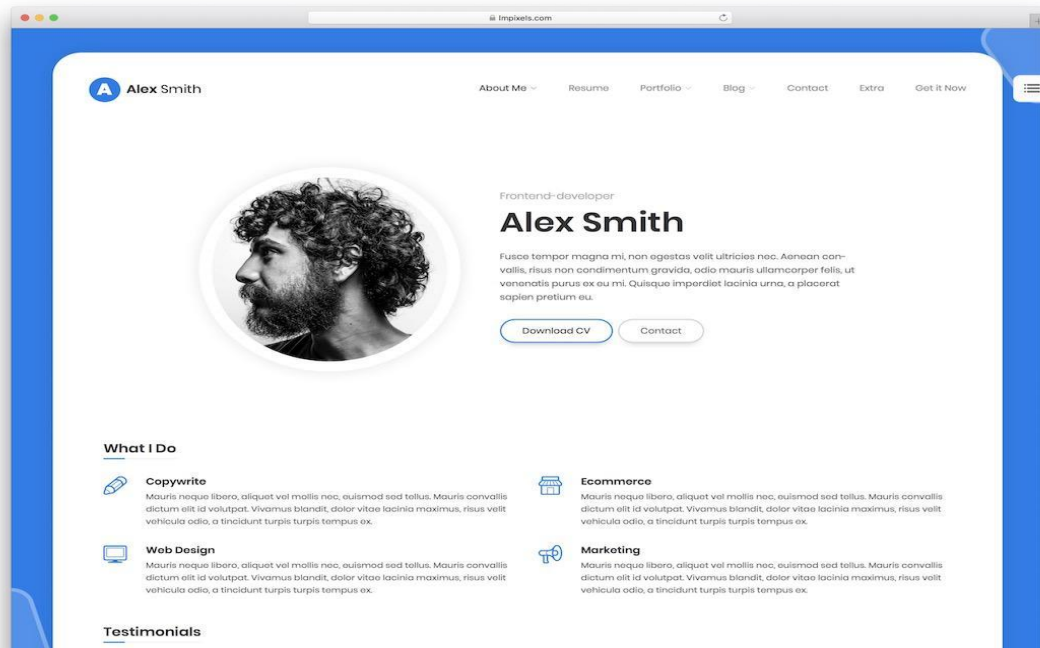
<https://www.w3schools.com/css/>

Perform the exercises on the following CSS topics:

- Font properties (e.g Web Fonts)
- Text properties (e.g Text Effects, Shadows)
- Backgrounds
- Rounded Corners:
- Border Images
- border properties (border block property, Border bottom etc)
- Lists and table styling
- Colors
- Keywords
- Gradients
- 2D Transformation
- 3D transformation
- Transitions
- Animation

Lab Task

1. Create a Personal portfolio page in HTML and CSS like following



LAB 7 Positioning elements using CSS

Objectives

Theoretical Description

Positioning Elements Using CSS

1. Position Static

By default all elements on the web page are using **position:static** without even needed to specify it. This is the *normal positioning method* which is what is used in the normal document flow. Chances are, you'll never need to specify position:static.

2 Position Relative

If you assign **position:relative** to an element, what that does is allow you to offset the box precisely to the top, right, bottom, or left by a given amount *relative to its original position in the document flow*. The space that element would have occupied is preserved and continues to have an effect on the layout of surrounding content. To demonstrate the position Relative technique, we will use the following markup. We remove the floats from our <section> elements so they now stack on top of each other. We also assign a width to the wrapping div, and set the margin to auto so that our <section> elements line up right in the center of the page.

html

```
<html>
```

```
<head>
```

```
<title>CSS Positioning Tutorial</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

  <div class="container">

    <section class="one">

      Lorem ipsum dolor sit...

    </section>

    <section class="two">

      Lorem ipsum dolor sit...

    </section>

    <section class="three">

      Lorem ipsum dolor sit...

    </section>

  </div>

</body>

</html>
```

Markup

Copy

css

```
section {
```

```
border: 2px solid lightblue;

padding: 10px;

margin: 3px;
}

img {

float: left;

padding-right: 10px;
}

.container {

width: 500px;

margin: auto;
}
```

CSS
Copy



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Aliquam error animi distinctio
veritatis, voluptas quaerat
laborum dolore atque saepe
earum exercitationem soluta?
Praesentium odio ipsum at
culpa dolorum repellat.

Accusamus amet similique cum aut dolore vero possimus
rerum, omnis est officiis sapiente, sunt quia molestiae!



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Quas at cupiditate labore
dignissimos voluptatem placeat
explicabo vel officia magni?
Fugit quos, veritatis iusto
laudantium non, vitae dolorum
sunt et dolorem hic accusamus

perferendis, doloremque praesentium fuga dolore corporis
assumenda nesciunt quia provident architecto harum!



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Quas at cupiditate labore
dignissimos voluptatem placeat
explicabo vel officia magni?
Fugit quos, veritatis iusto
laudantium non, vitae dolorum
sunt et dolorem hic accusamus

perferendis, doloremque praesentium fuga dolore corporis
assumenda nesciunt quia provident architecto harum!

What we want to do now is apply a `position: relative` to the second `<section>` so that we can move it relative to where it would normally appear in the document flow. This following CSS says to offset that `<section>` 100 pixels from the left of where it would have normally appeared. This gives the effect of shifting the `<section>` 100 pixels to the right.

```
section.two {  
  position: relative;
```



```
left: 100px;
```

```
}
```

CSS



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Aliquam error animi distinctio
veritatis, voluptas quaerat
laborum dolore atque saepe
earum exercitationem soluta?
Praesentium odio ipsum at
culpa dolorum repellat.

Accusamus amet similique cum aut dolore vero possimus
rerum, omnis est officiis sapiente, sunt quia molestiae!



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Quas at cupiditate labore
dignissimos voluptatem placeat
explicabo vel officia magni?
Fugit quos, veritatis iusto
laudantium non, vitae dolorum
sunt et dolorem hic accusamus

perferendis, doloremque praesentium fuga dolore corporis
assumenda nesciunt quia provident architecto harum!



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Quas at cupiditate labore
dignissimos voluptatem placeat
explicabo vel officia magni?
Fugit quos, veritatis iusto
laudantium non, vitae dolorum
sunt et dolorem hic accusamus

perferendis, doloremque praesentium fuga dolore corporis
assumenda nesciunt quia provident architecto harum!

You can also move it to the left by offsetting from the right.

```
section.two {
```

```
position: relative;
```

```
right: 100px;
```

```
}
```

CSS



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Aliquam error animi distinctio
veritatis, voluptas quaerat
laborum dolore atque saepe
earum exercitationem soluta?
Praesentium odio ipsum at
culpa dolorum repellat.

Accusamus amet similique cum aut dolore vero possimus
rerum, omnis est officiis sapiente, sunt quia molestiae!



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Quas at cupiditate labore
dignissimos voluptatem placeat
explicabo vel officia magni?
Fugit quos, veritatis iusto
laudantium non, vitae dolorum
sunt et dolorem hic accusamus

perferendis, doloremque praesentium fuga dolore corporis
assumenda nesciunt quia provident architecto harum!



Lorem ipsum dolor sit, amet
consectetur adipisicing elit.
Quas at cupiditate labore
dignissimos voluptatem placeat
explicabo vel officia magni?
Fugit quos, veritatis iusto
laudantium non, vitae dolorum
sunt et dolorem hic accusamus

perferendis, doloremque praesentium fuga dolore corporis
assumenda nesciunt quia provident architecto harum!

If we were to set either the top or bottom offset, the content would begin to overlap but you get the idea of how this works. It is important to note that when you use `position: relative` on an element, it is *not removed from normal document flow*. The element still occupies the original position in the document, it is just offset visually to the end-user. Position relative is very good if you want to simply tweak an element just a bit in the overall layout.

Consider the three sections are now back to a float layout where we have three columns. We could use this CSS to create a neat offset effect vertically now.

```
section.two {  
  position: relative;  
  top: 40px;  
}
```

```
section.three {  
  position: relative;  
  top: 80px;  
}
```

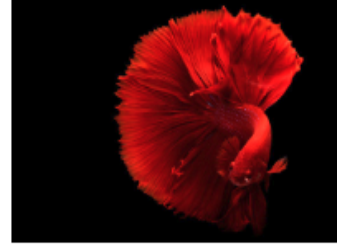
CSS



Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aliquam error animi distinctio veritatis, voluptas quaerat laborum dolore atque saepe earum exercitationem soluta? Praesentium odio ipsum at culpa dolorum repellat. Accusamus amet similique cum aut dolore vero possimus rerum, omnis est officiis sapiente, sunt quia molestiae!



Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quas at cupiditate labore dignissimos voluptatem placeat explicabo vel officia magni? Fugit quos, veritatis iusto laudantium non, vitae dolorum sunt et dolorem hic accusamus perferendis, doloremque praesentium fuga dolore corporis assumenda nesciunt quia provident architecto harum!



Lorem ipsum dolor sit, amet consectetur adipisicing elit. Quas at cupiditate labore dignissimos voluptatem placeat explicabo vel officia magni? Fugit quos, veritatis iusto laudantium non, vitae dolorum sunt et dolorem hic accusamus perferendis, doloremque praesentium fuga dolore corporis assumenda nesciunt quia provident architecto harum!

The key points to remember about Relative Positioning are:

- The element remains in the normal document flow
- The original space in the document is preserved
- Be careful for overlapping content as you move the element

2. Position Absolute

The **position:absolute** value takes elements out of the normal document flow. This means the element can be placed absolutely anywhere on the web page that the designer chooses. Absolute positioning is likely the most commonly used and versatile of the various positioning methods. Absolutely positioned elements will continue to scroll with the page. The **position:absolute** value is most often used on an element which has a parent that has its positioning set to *position:relative*. It's a little confusing, so let's see an example.

We will start with just a <div> element which has an and then an <h1> element like so.

html

```
<html>

<head>

  <title>CSS Positioning Tutorial</title>

  <link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

  <div class="container">

    <div id="banner">

      <h1>Hanging Out With Friends</h1>

    </div>

  </div>

</body>

</html>
```



Hanging Out With Friends

With that in place, our goal is to take that `<h1>` headline and position it absolutely overlapping the image to give it a cool banner effect. It would almost be like the text itself is part of the banner. This is a two-step process. First, we assign *position: relative* to the parent of the element we want to absolutely position. So the parent of the `<h1>` is the div with an id of `#banner`. Once that is in place, we target the `<h1>` itself and apply **position: absolute**.

```
#banner {  
  position: relative;  
}
```

```
h1 {
```



```
position: absolute;
```

```
bottom: 10px;
```

```
left: 140px;
```

```
}
```

CSS

Copy



Looks pretty good! Absolutely positioned elements work in conjunction with **containing blocks**. Since overlap occurs with absolute positioning, you can make use of the z-index to adjust the stacking order of elements on the page to get the result you are looking for. So with Position Absolute, we can remember that:

- The element is taken out of the normal document flow
 - The original space in the document is no longer preserved
 - Will often use with a parent element whose position is relative
-

3. Position Fixed

Position fixed is like taking an element and tacking it directly to the screen. In other words, the element stays fixed in one position in the window even when the user scrolls the page. Of course this means that fixed elements are taken out of the normal document flow and are positioned relative only to the browser window itself. Fixed positioning can get you in to trouble, so be careful with it. One use you might see with fixed positioning however is a fixed navigation bar. Let's see an example of that.

Here is out HTML which now has a nav area near the top of the page.

```
<html>

<head>

  <title>CSS Positioning Tutorial</title>

  <link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

  <nav>

    <ul>

      <li>Link 1</li>

      <li>Link 2</li>

      <li>Link 3</li>

      <li>Link 4</li>

      <li>Link 5</li>

    </ul>

  
```



```
</nav>
```

```
<div class="container">
```

```
  <section>
```

```
    
```

Lorem ipsum dolor sit...

```
  </section>
```

```
  <section>
```

```
    
```

Lorem ipsum dolor sit...

```
  </section>
```

```
  <section>
```

```
    
```

Lorem ipsum dolor sit...

```
  </section>
```

```
  <section>
```

```
    
```

Lorem ipsum dolor sit...

```
  </section>
```

```
  <section>
```

```
    
```

Lorem ipsum dolor sit...

</section>

<section>

Lorem ipsum dolor sit...

</section>

<section>

Lorem ipsum dolor sit...

</section>

<section>

Lorem ipsum dolor sit...

</section>

<section>

Lorem ipsum dolor sit...

</section>

<section>

Lorem ipsum dolor sit...

</section>

```
</div>

</body>

</html>
```

Markup

Our goal is to make this nav bar sticky to it sticks to the top of the page and is visible no matter what, even when scrolling. To do this, we can assign the nav a position of fixed, and then set the top to zero and the left to zero. This effectively sticks this navigation menu to the top of the page, and it will continue to be visible during scrolling. This is a common use case for a fixed positioning.

CSS

```
nav {

    width: 100%;

    background-color: #333;

    padding: 20px;

    position: fixed;

    top: 0;

    left: 0;

}

nav li {

    list-style-type: none;

    margin: 0 10px;

    color: white;

    float: left;
```

```
}  
  
ul::after {  
  
    display: block;  
  
    content: "";  
  
    clear: both;  
  
}
```

In summary, fixed positioning has these traits:

- The element is removed from normal document flow
- Offset values for fixed elements are always relative to the viewport
- The element retains position even during web page scrolling

Lab Task

Create web pages on any emerging topic using positioning elements

LAB 8 : Mid Lab Exam

Objectives

To Evaluate Practical concepts for front end development.

Theoretical Description

None

Lab Task

Lab Exam Task

LAB 9: Introduction to JavaScript for website development

Objectives

To Understand basic programming in Javascript and to use Javascript for front end development.

Theoretical Description

Introduction to JavaScript

- JavaScript (JS) is a scripting language, primarily used on the Web.
- It is used to enhance HTML pages and is commonly found embedded in HTML code.
- JavaScript is an interpreted language.
- Thus, it doesn't need to be compiled.
- JavaScript renders web pages in an interactive and dynamic fashion.
- This allowing the pages to react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user's browser, etc.

1. Advantages and Disadvantages of JavaScript (JS):

JavaScript is one of the most simple, versatile and effective languages used to extend functionality in websites. Uses range from on screen visual effects to processing and calculating data on web pages with ease as well as extended functionality to websites using third party scripts among several other handy features, however it also possesses some negative effects that might make you want to think twice before implementing JavaScript on your website. Let's look at some of its pros and cons.

Advantages:

JavaScript is executed on the client side

This means that the code is executed on the user's processor instead of the web server thus saving bandwidth and strain on the web server.

JavaScript is a relatively easy language

The JavaScript language is relatively easy to learn and comprises of syntax that is close to English. It uses the DOM model that provides plenty of prewritten functionality to the various objects on pages making it a breeze to develop a script to solve a custom purpose.

JavaScript is relatively fast to the end user

As the code is executed on the user's computer, results and processing is completed almost instantly depending on the task (tasks in JavaScript on web pages are usually simple so as to prevent being a memory hog) as it does not need to be processed in the site's web server and sent back to the user consuming local as well as server bandwidth.

Extended functionality to web pages

Third party add-ons like Greasemonkey enable JavaScript developers to write snippets of JavaScript which can execute on desired web pages to extend its functionality. If you use a website and require a certain feature to be included, you can write it yourself and use an add-on like Greasemonkey to implement it on the web page.

Disadvantages:

Security Issues

JavaScript snippets, once appended onto web pages execute on client servers immediately and therefore can also be used to exploit the user's system. While a certain restriction is set by modern web standards on browsers, malicious code can still be executed complying with the restrictions set.

JavaScript rendering varies

Different layout engines may render JavaScript differently resulting in inconsistency in terms of functionality and interface. While the latest versions of JavaScript and rendering have been geared towards a universal standard, certain variations still exist. Website Usability Consultants all over the world make a living on these differences, but it enrages thousands of developers on a daily basis.

1 Usage and Benefits of JavaScript (JS):

With the new “JavaScript” approach to web apps development, the server is largely removed from the process of front-end rendering. Rather, a sophisticated JavaScript layer running in the browser gets data from the server and then handles all front-end rendering and logic.

Let's look at some of its benefits & usage of JavaScript:

- Fast and Responsive
- Universal Front-end Platform
- JavaScript is a relatively easy language
- JavaScript is a relatively easy language
- Framework for the next generation
- Extended functionality to web pages
- Executed on the client side
- Offline Support and App Stores
- Industry Force - app development

2 Conditional Statements:

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

The if Statement:

Use the **if** statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition) {  
    block of code to be executed if the condition  
    is true  
}
```

Example

Make a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {  
    greeting = "Good day";  
}
```

The result of greeting will be:

Good day

Source Code

<pre><html> <body> <p>Display "Good day!" if the hour is less than 18:00:</p> <p id="demo">Good Evening!</p> <script> if (new Date().getHours() < 18) {</pre>	<pre>document.getElementById("demo").innerHTML = "Good day!"; } </script> </body> </html></pre>
--	---

Output

Display "Good day!" if the hour is less than 18:00:

Good day!

The else Statement:

Use the **else** statement to specify a block of code to be executed if the condition is false.

Syntax

```
if (condition) {  
    block of code to be executed if the condition  
    is true  
} else {  
    block of code to be executed if the condition  
    is false  
}
```

Example

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```

The result of greeting will be:

Good day

Source Code

<pre><html> <body> <p>Click the button to display a time- based greeting:</p> <button onclick="myFunction()">Try it</button> <p id="demo"></p> <script> function myFunction() { var hour = new Date().getHours(); var greeting; if (hour < 18) {</pre>	<pre> greeting = "Good day"; } else { greeting = "Good evening"; } document.getElementById("demo").innerHTML = greeting; } </script> </body> </html></pre>
--	--

Output

Click the button to display a time-based greeting:

Try it

Good day

The else if Statement:

Use the **else if** statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {  
    block of code to be executed if condition1 is  
    true  
} else if (condition2) {  
    block of code to be executed if the condition1  
    is false and condition2 is true  
} else {  
    block of code to be executed if the condition1  
    is false and condition2 is false  
}
```

3

Example

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {  
    greeting = "Good morning";  
} else if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```

The result of greeting will be:

Good day

Source Code

```
<!DOCTYPE html>  
<html>  
<body>  
  
<p>Click the button to get a time-based  
greeting:</p>  
  
<button onclick="myFunction()">Try  
it</button>  
  
<p id="demo"></p>  
  
<script>  
function myFunction() {  
    var greeting;
```

```
    var time = new Date().getHours();  
    if (time < 10) {  
        greeting = "Good morning";  
    } else if (time < 20) {  
        greeting = "Good day";  
    } else {  
        greeting = "Good evening";  
    }  
    document.getElementById("demo").innerHT  
ML = greeting;  
}  
</script>  
  
</body>  
</html>
```

Output

Click the button to get a time-based greeting:

Try it

Good day

4.1 The switch Statement:

- The **switch** statement is used to perform different actions based on different conditions.
- Use the **switch** statement to select one of many blocks of code to be executed.

Syntax

```
switch(expression) {  
    case n:  
        code block  
        break;  
    case n:  
        code block  
        break;  
    default:  
        code block  
}
```

Note:

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

Example

The `getDay()` method returns the weekday as a number between 0 and 6.

(Sunday=0, Monday=1, Tuesday=2 ..)

This example uses the weekday number to calculate the weekday name:

```

switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}

```

The result of day will be:

Wednesday

Note:

- i) The break Keyword
 - When JavaScript reaches a break keyword, it breaks out of the switch block.
 - This will stop the execution of more code and case testing inside the block.
 - It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.
- ii) The default Keyword

The default keyword specifies the code to run if there is no case match.

4.1.1 Source Code

```

<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>

```

```

var day;
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";

```

```
break;
case 2:
  day = "Tuesday";
  break;
case 3:
  day = "Wednesday";
  break;
case 4:
  day = "Thursday";
  break;
case 5:
```

```
day = "Friday";
break;
case 6:
  day = "Saturday";
}
document.getElementById("demo").innerHTML = "Today is " + day;
</script>

</body>
</html>
```

4.1.2 Output

Today is Wednesday

2. Iteration Statements/Loops:

Loops can execute a block of code a number of times.

- Loops are handy, if you want to run the same code over and over again, each time with a different value.
- Often this is the case when working with arrays.

JavaScript supports different kinds of loops:

- **for** - loops through a block of code a number of times.
- **while** - loops through a block of code while a specified condition is true.
- **do/while** - also loops through a block of code while a specified condition is true.

5.1 The for Loop:

The **for** loop is often the tool you will use when you want to create a loop.

Syntax

```
for (statement 1; statement 2; statement 3) {
  code block to be executed
}
```

- **Statement 1** is executed before the loop (the code block) starts.
- **Statement 2** defines the condition for running the loop (the code block).
- **Statement 3** is executed each time after the loop (the code block) has been executed.

Example

```
for (i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
}
```

Source Code

```
<!DOCTYPE html>  
<html>  
<body>  
  
<h2>JavaScript Loops</h2>  
  
<p id="demo"></p>  
  
<script>  
var text = "";  
var i;  
for (i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
}  
document.getElementById("demo").innerHTML = text;  
</script>  
  
</body>  
</html>
```

5.1.1 Output

JavaScript Loops

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4

5.2 The while Loop:

The **while** loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {  
    code block to be executed  
}
```

Example

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```

Source Code

<pre><!DOCTYPE html> <html> <body> <h2>JavaScript while</h2> <p id="demo"></p> <script> var text = "";</pre>	<pre>var i = 0; while (i < 10) { text += "
The number is " + i; i++; } document.getElementById("demo").innerHTML = text; </script> </body> </html></pre>
---	---

5.2.1 Output

JavaScript while

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

5.3 The do/while Loop:

The **do/while** loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop

as long as the condition is true.

Syntax

```
do {  
    code block to be executed  
}  
while (condition);
```

Example

The example below uses a **do/while** loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
do {  
    text += "The number is " + i;  
    i++;  
}  
while (i < 10);
```

Source Code


```

<!DOCTYPE html>
<html>
<body>

<h2>JavaScript do ... while</h2>

<p id="demo"></p>

<script>
var text = ""
var i = 0;

do {
    text += "<br>The number is " + i;
    i++;
}
while (i < 10);

document.getElementById("demo").innerHTML
= text;
</script>

</body>
</html>

```

JavaScript do ... while

```

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

```

3. JavaScript Output/Display Statements:

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Lab Task

1. Implement the above manual and all the code snippets for developing javascript understanding
2. Identify the differences between document

LAB 10: Implementing JavaScript for Front-End development

Objectives

To implement Front end using Javascript by working with HTML DOM

Theoretical Description

To access an HTML element, JavaScript can use the **document.getElementById(id)** method. The id attribute defines the HTML element. The innerHTML property defines the HTML content:

Example

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Output

My First Web Page

My First Paragraph.

11

6.1 Using document.write():

For testing purposes, it is convenient to use **document.write()**:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Using console.log():

For debugging purposes, you can use the **console.log()** method to display data.



Example

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

Output

Activate debugging with F12

Select "Console" in the debugger menu. Then click Run again.

Lab Task

Design a **Marks Calculator** using JavaScript which prompts user to enter marks of three subjects and give result in the textbox.

Marks Calculator

Subject 1	<input type="text" value="80"/>
Subject 2	<input type="text" value="85"/>
Subject 3	<input type="text" value="92"/>
Result	<input type="text" value="257"/>

- Create a Simple calculator using Javascript

LAB 11: Lab Exam

Objectives

To Evaluate Practical concepts for front end development.

Theoretical Description

None

Lab Task

Lab Exam Task

LAB 12: Introduction to PHP for website development

Objectives

To learn how to install XAMPP server and Perform basic PHP commands

Theoretical Description

PHP Introduction

- 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 is free to download and use.

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.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Servers in PHP

To start using PHP, you can:

- Find a web host with **PHP** and **MySQL** support
- Install a web server on your own PC, and then install PHP and MySQL

There are generally three types of servers used to run PHP:

- i. wamp
 - ii. xampp
 - iii. apache
- All the above servers include PHP and MySQL
 - The extension of PHP files are **.php**

- These files are located at : **xampp/htdocs/index.php**
- **index.php** page (first page) is known as **Landing Page**
- While configuring, go to **php.ini** and **php.config**, then enter the port number where required

How to run PHP pages

- IP of localhost is 127.0.0.1
- The default url to run php file is <http://localhost:8080>
- where **8080** is by default port, that runs on web application
- PHP files are located at : **xamp\htdocs\index.php**
- To run PHP files on a browser, type: <http://localhost:1218/index.php> (press ENTER)

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**

```
<?php
// PHP code goes here
?>
```

Note:

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Example

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:

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

4 PHP Comments

A comment in PHP code is a line that is not read/executed as 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 what you are doing
- 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:

3.1 Single-Line Comment

The single line comment can be used as:

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

3.2 Multiple-Line Comment

The multiple line comment can be used as:

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

Example

```
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// 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  
*/  
  
// You can also use comments to leave out parts of a code line  
$x = 5 /* + 15 */ + 5;  
echo $x;  
?>  
  
</body>  
</html>
```

5 PHP Variables

Variables are "containers" for storing information.

4.1 Declaring (Creating) PHP Variables

In PHP, a variable starts with the \$ sign, followed by the name of the variable:

Example

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

After the execution of the statements above, the variable **\$txt** will hold the value **Hello world!**, the variable **\$x** will hold the value **5**, and the variable **\$y** will hold the value **10.5**.

Output

```
Hello world!
5
10.5
```

Note:

- When you assign a text value to a variable, put quotes around the value.
- Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

6 PHP Strings

A string is a sequence of characters, like **"Hello world!"**.

5.1 String Functions

In this section, we will look at some commonly used functions to manipulate strings.

5.1.1 strlen()

The PHP **strlen()** function returns the length of a string.

5.1.1.1 Example

The example below returns the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

5.1.1.2 Output

The output of the code above will be: **12**

5.1.2 str_word_count()

The PHP **str_word_count()** function counts the number of words in a string:

5.1.2.1 Example

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

5.1.2.2 Output

The output of the code above will be: **2**

5.1.3 strrev()

The PHP **strrev()** function reverses a string:

5.1.3.1 Example

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

5.1.3.2 Output

The output of the code above will be: **!dlrow olleH**

5.1.4 strpos()

The PHP **strpos()** function searches for a specific text within a string.

- If a match is found, the function returns the character position of the first match.
- If no match is found, it will return FALSE.

5.1.4.1 Example

The example below searches for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

5.1.5 str_replace()

The PHP **str_replace()** function replaces some characters with some other characters in a string.

5.1.5.1 Example

The example below replaces the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello
Dolly!
?>
```

5.1.5.2 Output

The output of the code above will be: **Hello Dolly!**

Lab Task

1. Download and Install Apache. Setup working environment for PHP development.
2. Declare your name in a variable and apply string functions in PHP

Lab Task

LAB 13: Basic PHP Commands for website backend

Objectives

To Perform basic PHP commands and practice Logic building with PHP using PHP operators.

Theoretical Description

1. 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

Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

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 (Introduced in PHP 5.6)

Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

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

<code>x *= y</code>	<code>x = x * y</code>	Multiplication
<code>x /= y</code>	<code>x = x / y</code>	Division
<code>x %= y</code>	<code>x = x % y</code>	Modulus

Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
<code>=</code>	Equal	<code>\$x = \$y</code>	Returns true if \$x is equal to \$y
<code>===</code>	Identical	<code>\$x === \$y</code>	Returns true if \$x is equal to \$y, and they are of the same type
<code>!=</code>	Not Equal	<code>\$x != \$y</code>	Returns true if \$x is not equal to \$y
<code><></code>	Not Equal	<code>\$x <> \$y</code>	Returns true if \$x is not equal to \$y
<code>!==</code>	Not Identical	<code>\$x !== \$y</code>	Returns true if \$x is not equal to \$y, or they are not of the same type
<code>></code>	Greater than	<code>\$x > \$y</code>	Returns true if \$x is greater than \$y
<code><</code>	Less than	<code>\$x < \$y</code>	Returns true if \$x is less than \$y
<code>>=</code>	Greater than or Equal to	<code>\$x >= \$y</code>	Returns true if \$x is greater than or equal to \$y
<code><=</code>	Less than or Equal to	<code>\$x <= \$y</code>	Returns true if \$x is less than or equal to \$y

Logical Operators

The PHP logical operators are used to combine conditional statements.

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

7 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

2.1 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;  
}
```

Example

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
}  
?>
```

2.2 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;  
}
```

Example

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php  
$t = date("H");  
  
if ($t < "20") {  
    echo "Have a good day!";  
}
```

```

} else {
    echo "Have a good night!";
}
?>

```

2.3 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 this condition is true;
} else {
    code to be executed if all conditions are false;
}

```

Example

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```

<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>

```

2.4 The switch Statement

The switch statement is used to perform different actions based on different conditions.

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.

Example

```
<?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!";
}
?>
```

PHP Iteration Statements

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **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 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;  
}
```

Example

The example below first sets a variable \$x to 1 (\$x = 1). Then, the while loop will continue to run as long as \$x is less than, or equal to 5 (\$x <= 5). \$x will increase by 1 each time the loop runs (\$x++):

```
<?php  
$x = 1;  
  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

3.2 The 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);
```

Example

The example below first sets a variable \$x to 1 (\$x = 1). Then, the do while loop will write some output, and then increment the variable \$x with 1. Then the condition is checked (is \$x less than, or equal to 5?), and the loop will continue to run as long as \$x is less than, or equal to 5:

```
<?php  
$x = 1;  
  
do {  
    echo "The number is: $x <br>";  
    $x++;  
}
```

```
} while ($x <= 5);  
?>
```

Note:

Notice that in a do while loop the condition is tested **AFTER** executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

3.3 The 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;  
}
```

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

Example

The example below displays the numbers from 0 to 10:

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

3.4 The foreach Loop

The foreach loop works only on arrays and is used to loop through each key/value pair in an array.

Syntax

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

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

Example

The following example demonstrates a loop that will output the values of the given array (\$colors):

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

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

Lab Task:

1. Write a PHP program to find all numbers which occurs odd number of times and other numbers occur even number of times

Input : 4, 5, 4, 4, 2, 2, 3, 3, 2, 4

Output :

Odd number of times: 2, 5

Even number of times: 3, 4

2. In mathematics, an arithmetic progression or arithmetic sequence is a sequence of numbers such that the difference between the consecutive terms is constant. For example, the sequence 5, 8, 11, 14... is an arithmetic progression with common difference of 3.

Write a PHP program to check whether a sequence of numbers is an arithmetic progression or not.

Input : array(5, 8, 11, 14)

Output:

Arithmetic Progression

difference => 3

LAB 14: Functions in PHP

Objectives

To understand the use of functions and Global Variables in PHP

Theoretical Description

PHP Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

A function will be executed by a call to the function.

Syntax

A user-defined function declaration starts with the word **function**:

```
function functionName() {  
    code to be executed;  
}
```

Example

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

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

Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (\$fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

```
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

Default Argument Value

To let a function, return a value, use the **return** statement:

```
<?php
function sum($x, $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);
?>
```

PHP Arrays

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

2.1 Create an Array

In PHP, the `array()` function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

Indexed arrays - Arrays with a numeric index

Associative arrays - Arrays with named keys

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";
```

Example

The following example creates an indexed array named \$cars, assigns three elements to it, and then prints a text containing the array values:

```
<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";  
?>
```

Length of an Array – The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

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

Loop through an Indexed Array

To loop through and print all the values of an indexed array, you could use a **for** loop, like this:

Example

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

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

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";
```

The named keys can then be used in a script:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop through Associative Arrays

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

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

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

PHP Global Variables

Several 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.

The PHP **superglobal** variables are:

`$GLOBALS`

`$_SERVER`

`$_REQUEST`

`$_POST`

`$_GET`

`$_FILES`

`$_ENV`

`$_COOKIE`

`$_SESSION`

PHP \$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.

Example

The example below shows how to use the super global variable \$GLOBALS

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

Note: In the example above, since z is a variable present within the \$GLOBALS array, it is also accessible from outside the function!

PHP \$_SERVER

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

Example

The example below shows how to use some of the elements in \$_SERVER:

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

The following table lists the most important elements that can go inside \$_SERVER:

PHP \$_REQUEST

PHP \$_REQUEST is used to collect data after submitting an HTML form.

Example

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable \$_REQUEST to collect the value of the input field:

```
<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 \$_POST

PHP \$_POST is widely used to collect form data after submitting an HTML form with method="post". \$_POST is also widely used to pass variables.

Example

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable \$_POST to collect the value of the input field:

```
<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 \$_GET

PHP \$_GET can also be used to collect form data after submitting an HTML form with method="get".

\$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a
href="test_get.php?subject=PHP&web=W3schools.com">Test
$GET</a>

</body>
</html>
```

When a user clicks on the link "Test \$GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with \$_GET

Example

The example below shows the code in "test_get.php":

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```

Lab Task

Implement Functions in PHP as per given manual

Implement A Sign in and Sign Up page for your project . Also include A log out functionality without using a database

LAB 15: MYSQL operations in PHP

Objectives

To implement CRUD operations in PHP

Theoretical Description

PHP MySQL

MySQL is a database system used on the web

MySQL is a database system that runs on a server

MySQL is ideal for both small and large applications

MySQL is very fast, reliable, and easy to use

MySQL uses standard SQL

MySQL compiles on a number of platforms

MySQL is free to download and use

MySQL is developed, distributed, and supported by Oracle Corporation

MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

Employees

Products

Customers

Orders

Database Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

The query above selects all the data in the "LastName" column from the "Employees" table.

MySQL Connect

Before we can access data in the MySQL database, we need to be able to connect to the server:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```
// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

MySQL Create DB

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

MySQL Create Table

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```
CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
```



```
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg_date TIMESTAMP  
)
```

Example

The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our Data Types reference.

After the data type, you can specify other optional attributes for each column:

NOT NULL - Each row must contain a value for that column, null values are not allowed

DEFAULT value - Set a default value that is added when no other value is passed

UNSIGNED - Used for number types, limits the stored data to positive numbers and zero

AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added

PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}  
  
// sql to create table  
$sql = "CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg_date TIMESTAMP  
)";  
  
if ($conn->query($sql) === TRUE) {  
    echo "Table MyGuests created successfully";  
} else {
```

```
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

MySQL Insert Data

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

The SQL query must be quoted in PHP

String values inside the SQL query must be quoted

Numeric values must not be quoted

The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Example

The following examples add a new record to the "MyGuests" table:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

MySQL Select Data

The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

Example

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
        $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

MySQL Delete Data

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

Example

The following examples delete the record with id=3 in the "MyGuests" table:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

MySQL Update Data

The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

Example

The following examples update the record with id=2 in the "MyGuests" table:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

Lab Task:

Implement CRUD operations in PHP for a project

Create a simple To Do App in PHP using a Database

LAB 16: Project Evaluation

Objectives

To evaluate student PHP projects with respect to front end, backend, CRUD operations

Theoretical Description

None

Lab Task

Viva Voce