Chapter 1

Developing Portals Using HTML

What is HTML?

HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most of markup (e.g. HTML) languages are human readable. Language uses tags to define what manipulation has to be done on the text.

The History of HTML

HTML was invented by Tim Berners-Lee, a physicist at the CERN research institute in Switzerland. He came up with the idea of an Internet-based hypertext system.

Hypertext means a text that contains references (links) to other texts that viewers can access immediately. He published the first version of HTML in 1991, consisting of 18 HTML tags. Since then, each new version of the HTML language came with new tags and attributes (tag modifiers) to the markup.

According to Mozilla Developer Network's HTML Element Reference, currently, there are 140 HTML tags, although some of them are already obsolete (not supported by modern browsers).

Due to a quick rise in popularity, HTML is now considered an official web standard. The HTML specifications are maintained and developed by the World Wide Web Consortium (W3C). You can check out the latest state of the language anytime on W3C's website.

The biggest upgrade of the language was the introduction of HTML5 in 2014. It added several new semantic tags to the markup, that reveal the meaning of their own content, such as
<article>, <header>, and <footer>.

HTML VERSION	YEAR
HTML 1.0	1991

HTML VERSION	YEAR
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML 5	2014

Introduction to HTML5

HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM).

Features:

- It has introduced new multimedia features which supports audio and video controls by using <audio> and <video> tags.
- There are new graphics elements including vector graphics and tags.
- Enrich semantic content by including <header> <footer>, <article>, <section> and <figure> are added.
- Drag and Drop- The user can grab an object and drag it further dropping it on a new location.
- Geo-location services- It helps to locate the geographical location of a client.
- Web storage facility which provides web application methods to store data on web browser.
- Uses SQL database to store data offline.
- Allows to draw various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e. <!doctype html>
- Easy character encoding i.e. <meta charset=”UTF-8">

Removed elements from HTML 5

There are many elements which are depreciated from HTML 5 are listed below:

REMOVED ELEMENTS	USE INSTEAD ELEMENTS
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
<center>	CSS
<dir>	<ul>
<font>	CSS
<frame>	
<frameset>	
<noframes>	
<isindex>	
<strike>	CSS, <s> or <del>
<tt>	CSS

New Added Elements in HTML 5

- **<article>**: The <article> tag is used to represent an article. More specifically, the content within the <article> tag is independent from the other content of the site (even though it can be related).
- **<aside>**: The <aside> tag is used to describe the main object of the web page in a shorter way like a highlighter. It basically identifies the content that is related to the primary

content of the web page but does not constitute the main intent of the primary page. The <aside> tag contains mainly author information, links, related content and so on.

- **<figcaption>**: The <figurecaption> tag in HTML is used to set a caption to the figure element in a document.
- **<figure>**: The <figure> tag in HTML is used to add self-contained content like illustrations, diagrams, photos or codes listing in a document. It is related to main flow but it can be used in any position of a document and the figure goes with the flow of the document and if remove it then it should not affect the flow of the document.
- **<header>**: It contains the section heading as well as other content, such as a navigation links, table of contents, etc.
- **<footer>**: The <footer> tag in HTML is used to define a footer of HTML document. This section contains the footer information (author information, copyright information, carriers etc). The footer tag are used within body tag. The <footer> tag is new in the HTML 5. The footer elements require a start tag as well as an end tag.
- **<main>**: Delineates the main content of the body of a document or web app.
- **<mark>**: The <mark> tag in HTML is used to define the marked text. It is used to highlight the part of the text in the paragraph.
- **<nav>**: The <nav> tag is used to declaring the navigational section in HTML documents. Websites typically have sections dedicated to navigational links, which enables user to navigate the site. These links can be placed inside a nav tag.
- **<section>**: It demarcates a thematic grouping of content.
- **<details>**: The <details> tag is used for the content/information which is initially hidden but could be displayed if the user wishes to see it. This tag is used to create interactive widget which user can open or close it. The content of details tag is visible when open the set attributes.
- **<summary>**: The <summary> tag in HTML is used to define a summary for the <details> element. The <summary> element is used along with the <details> element and provides a summary visible to the user. When the summary is clicked by the user, the content placed inside the <details> element becomes visible which was previously hidden. The
  <summary> tag was added in HTMl 5. The <summary> tag requires both starting and ending tag.

- <time>: The <time> tag is used to display the human-readable data/time. It can also be used to encode dates and times in a machine-readable form. The main advantage for users is that they can offer to add birthday reminders or scheduled events in their calender's and search engines can produce smarter search results.
- <bdi>: The <bdi> tag refers to the Bi-Directional Isolation. It differentiate a text from other text that may be formatted in different direction. This tag is used when a user generated text with an unknown directions.
- <wbr>: The <wbr> tag in HTML stands for word break opportunity and is used to define the position within the text which is treated as a line break by the browser. It is mostly used when the used word is too long and there are chances that the browser may break lines at the wrong place for fitting the text.
- <datalist>: The <datalist> tag is used to provide autocomplete feature in the HTML files. It can be used with input tag, so that users can easily fill the data in the forms using select the data.
- <keygen>: The <keygen> tag in HTML is used to specify a key-pair generator field in a form. The purpose of <keygen> element is to provide a secure way to authenticate users. When a from is submitted then two keys are generated, private key and public key. The private key stored locally, and the public key is sent to the server. The public key is used to generate client certificate to authenticate user for future.
- <output>: The <output> tag in HTML is used to represent the result of a calculation performed by the client-side script such as JavaScript.
- <progress>: It is used to represent the progress of a task. It is also define that how much work is done and how much is left to download a things. It is not used to represent the disk space or relevant query.
- <svg>: It is the Scalable Vector Graphics.
- <canvas>: The <canvas> tag in HTML is used to draw graphics on web page using JavaScript. It can be used to draw paths, boxes, texts, gradient and adding images. By default it does not contains border and text.
- <audio>: It defines the music or audio content.
- <embed>: Defines containers for external applications (usually a video player).
- <source>: It defines the sources for <video> and <audio>.

- <track>: It defines the tracks for <video> and <audio>.
- <video>: It defines the video content.

Advantages

- All browsers supported.
- More device friendly.
- Easy to use and implement.
- HTML 5 in integration with CSS, JavaScript, etc can help build beautiful websites.

Disadvantages:

- Long codes have to be written which is time consuming.
- Only modern browsers support it.

Difference between HTML and HTML5

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (at the note for computer) text so that a machine can understand it and manipulate text accordingly. Most of the markup (e.g. HTML) languages are human readable. The language uses tags to define what manipulation has to be done on the text. It is used for structuring and presenting the content on the web pages. HTML5 is the fifth version of HTML. Many elements are removed or modified from HTML5.

There are many differences between HTML and HTML5 which are discussed below:

HTML	HTML5
It didn't support audio and video without the use of flash player support.	It supports audio and video controls with the use of <audio> and <video> tags.
It uses cookies to store temporary data.	It uses SQL databases and application cache to store offline data.

Does not allow JavaScript to run in browser.	Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5.
Vector graphics is possible in HTML with the help of various technologies such as VML, Silver-light, Flash, etc.	Vector graphics is additionally an integral a part of HTML5 like SVG and canvas.
It does not allow drag and drop effects.	It allows drag and drop effects.
Not possible to draw shapes like circle, rectangle, triangle etc.	HTML5 allows to draw shapes like circle, rectangle, triangle etc.
It works with all old browsers.	It supported by all new browser like Firefox, Mozilla, Chrome, Safari, etc.
Older version of HTML are less mobile-friendly.	HTML5 language is more mobile-friendly.
Doctype declaration is too long and complicated.	Doctype declaration is quite simple and easy.
Elements like nav, header were not present.	New element for web structure like nav, header, footer etc.
Character encoding is long and complicated.	Character encoding is simple and easy.
It is almost impossible to get true GeoLocation of user with the help of browser.	One can track the GeoLocation of a user easily by using JS GeoLocation API.
It can not handle inaccurate syntax.	It is capable of handling inaccurate syntax.
Attributes like charset, async and ping are absent in HTML.	Attributes of charset, async and ping are a part of HTML 5.

What is CSS

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantages of CSS

- **CSS saves time** – You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** – Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

What is CSS3

CSS3 is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. It can also be used to display the web page differently which can change depending on your screen size. Changes to the design of a document can be applied quickly and easily.

Some of the most important CSS3 modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Basic structure of HTML

The <HTML> is a markup language which is used by the browser to manipulate text, images and other content to display it in required format.

Tags in HTML

Tags are one of the most important part in an HTML Document. HTML uses some predefined tags which tells the browser about content display property, that is how to display a particular given content. For Example, to create a paragraph, one must use the paragraph tags(<p> </p>) and to insert an image one must use the img tags(<img />).

There are generally two types of tags in HTML:

Paired Tags or Closed Tag: These tags come in pairs. That is they have both opening (< >) and closing(</ >) tags.

Singular Tags Open Tag: or Empty Tag: These tags do not required to be closed. Below is an example of (<b>) tag in HTML, which tells the browser to bold the text inside it.

An HTML Document is mainly divided into two parts:

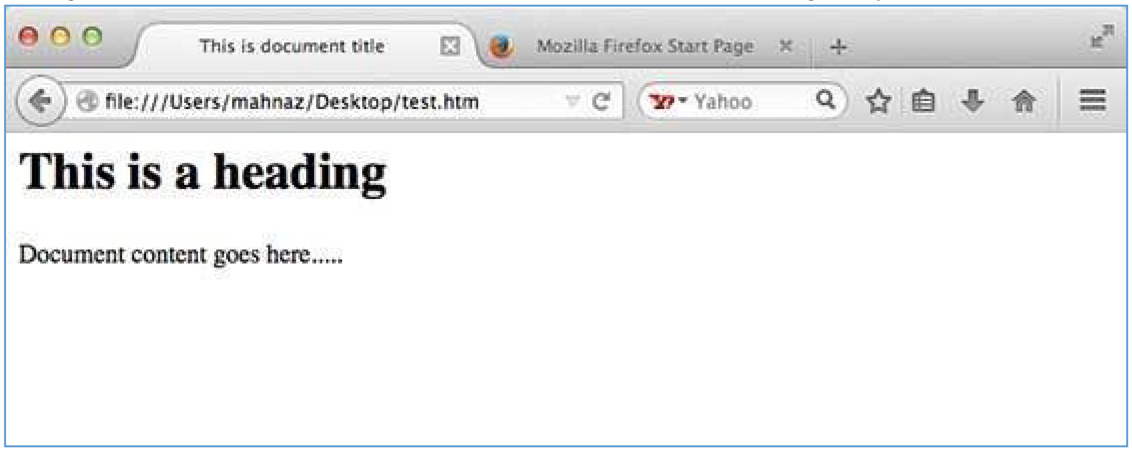
- HEAD: This contains the information about the HTML document. For Example, Head tag contains metadata, title, page CSS etc. All the HTML elements that can be used inside the
  <head> element are:
  - <style>
  - <title>
  - <base>
  - <noscript>
  - <script>
  - <meta>
- BODY: This contains everything you want to display on the Web Page. The most common elements to be placed in the HTML <body> tag are: <h1>, <p>, <div>, <table> tags etc.

Basic HTML Document

In its simplest form, following is an example of an HTML document:

```
<!DOCTYPE html>
<html>
<head>
<title>This is document title</title>
</head>
<body>
<h1>This is a heading</h1>
<p> Document content goes here .</p>
</body>
</html>
```

Either you can use **Try it** option available at the top right corner of the code box to check the result of this HTML code, or let's save it in an HTML file **test.htm** using your favorite text editor. Finally open it using a web browser like Internet Explorer or Google Chrome, or Firefox etc. It must show the following output:


This is document title
Mozilla Firefox Start Page
file:///Users/mahnaz/Desktop/test.htm
Yahoo

This is a heading

Document content goes here.....

The<!DOCTYPE>Declaration

The <!DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration:

```
<!DOCTYPE html>
```

There are many other declaration types which can be used in HTML document depending on what version of HTML is being used. We will see more details on this while discussing
<!DOCTYPE...> tag along with other HTML tags.

HeadingTags

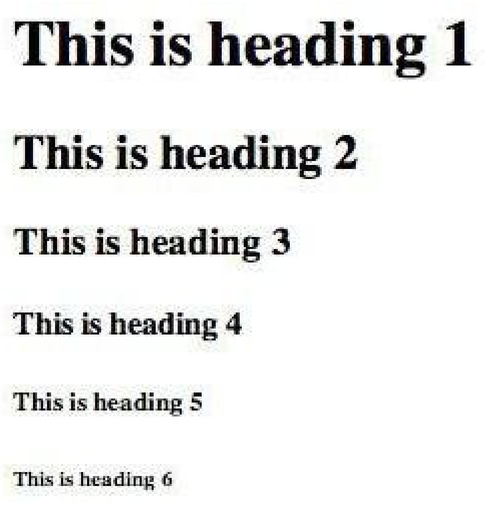
Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements **<h1>, <h2>, <h3>, <h4>, <h5>, and <h6>**. While displaying any heading, browser adds one line before and one line after that heading.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Heading Example</title>
</head>
<body>
<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>
</body>
</html>
```

This will produce the following result:



This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

ParagraphTag

The **<p>** tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening <p> and a closing </p> tag as shown below in the example:

```
<!DOCTYPE html>
<html>
<head>
<title>Paragraph Example</title>
</head>
<body>
<p>Here is a first paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
</body>
</html>
```

Here is a first paragraph of

text. Here is a second

paragraph of text.

This will produce the following result:

Line Break Tag

Whenever you use the **<br />** element, anything following it starts from the next line. This tag is an example of an **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <br /> tag has a space between the characters **br** and the forward slash. If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use <br> it is not valid in XHTML.

Horizontal Lines

Horizontal lines are used to visually break-up sections of a document. The <hr> tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

Again <hr /> tag is an example of the empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <hr /> element has a space between the characters hr and the forward slash. If you omit this space, older browsers will have trouble rendering the horizontal line, while if you miss the forward slash character and just use <hr> it is not valid in XHTML

HTML Attributes

We have seen few HTML tags and their usage like heading tags <h1>, <h2>, paragraph tag <p> and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.

An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts: a name and a value:

- The name is the property you want to set. For example, the paragraph <p> element in the example carries an attribute whose name is align, which you can use to indicate the alignment of paragraph on the page.
- The value is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: left, center and right.

- Attribute names and attribute values are case-insensitive. However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Align Attribute   Example</title>
</head>
<body>
<p align="left">This is left aligned</p>
<p align="center">This is center aligned</p>
<p align="right">This is right aligned</p>
</body>
</html>
```

This will display the following result:

```
This is left aligned
                                   This is center
                                       aligned
```

Core Attributes

The four core attributes that can be used on the majority of HTML elements (although not all) are:

- Id
- Title
- Class
- Style

The Id Attribute

The id attribute of an HTML tag can be used to uniquely identify any element within an HTML page. There are two primary reasons that you might want to use an id attribute on an element:

- If an element carries an id attribute as a unique identifier, it is possible to identify just that element and its content.
- If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.

```
<p id="html">This para explains what is HTML</p>
<p id="css">This para explains what is Cascading Style Sheet</p>
```

The title Attribute

The title attribute gives a suggested title for the element. They syntax for the title attribute is similar as explained for id attribute:

The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>The title Attribute Example</title>
</head>
<body>
<h3 title="Hello HTML!">Titled Heading Tag Example</h3>
</body>
</html>
```

This will produce the following result:

```
Titled Heading Tag Example
```

Now try to bring your cursor over "Titled Heading Tag Example" and you will see that whatever title you used in your code is coming out as a tooltip of the cursor.

The class Attribute

The **class** attribute is used to associate an element with a style sheet, and specifies the class of element. You will learn more about the use of the class attribute when you will learn

Cascading Style Sheet (CSS). So for now you can avoid it.

The value of the attribute may also be a space-separated list of class names. For example:

```
class="className1 className2 className3"
```

The style Attribute

The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element.

```
<!DOCTYPE html>
<html>
<head>
<title>The style Attribute</title>
</head>
<body>
<p style="font-family:arial; color:#FF0000;">Some text...</p>
</body>
</html>
```

This will produce the following result:

Some text...

HTML Formatting

If you use a word processor, you must be familiar with the ability to make text bold, italicized, or underlined; these are just three of the ten options available to indicate how text can appear in HTML and XHTML.

Bold Text

Anything that appears within <b>...</b> element, is displayed in bold as shown below:

```
<!DOCTYPE html>
<html>
<head>
<title>Bold Text Example</title>
</head>
<body>
<p>The following word uses a <b>bold</b> typeface.</p>
</body>
```

```
</html>
```

This will produce the following result:

The following word uses a **bold** typeface.

Italic Text

Anything that appears within <i>...</i> element is displayed in italicized.

Underline Text

Anything that appears within <u>...</u> element, is displayed with underline.

Superscript Text

The content of a <sup>...</sup> element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a character's height above the other characters.

Subscript Text

The content of a <sub>...</sub> element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a character's height beneath the other characters.

Fonts in HTML

Fonts play a very important role in making a website more user friendly and increasing content readability. Font face and color depends entirely on the computer and browser that is being used to view your page but you can use HTML <font> tag to add style, size, and color to the text on your website. You can use a <basefont> tag to set all of your text to the same size, face, and color.

The font tag is having three attributes called size, color, and face to customize your fonts. To change any of the font attributes at any time within your webpage, simply use the <font> tag. The text that follows will remain changed until you close with the </font> tag. You can change one or all of the font attributes within one <font> tag.

Set Font Size

You can set content font size using size attribute. The range of accepted values is from 1(smallest) to 7(largest). The default size of a font is 3.

```
<!DOCTYPE html>
<html>

   <head>
      <title>Setting Font Size</title>
   </head>

   <body>
      <font size = "1">Font size = "1"</font><br />
      <font size = "2">Font size = "2"</font><br />
      <font size = "3">Font size = "3"</font><br />
      <font size = "4">Font size = "4"</font><br />
      <font size = "5">Font size = "5"</font><br />
      <font size = "6">Font size = "6"</font><br />
      <font size = "7">Font size = "7"</font>
   </body>

</html>
```

This will produce the following result –

```
Font size = "1"
Font size = "2"
Font size =
"3" Font size
= "4"
Font size = "5"
Font size = "6"
Font size = "7"
```

Setting Font Color

You can set any font color you like using color attribute. You can specify the color that you want by either the color name or hexadecimal code for that color.

```
<!DOCTYPE html>
```

```
<html>

   <head>
      <title>Setting Font Color</title>
   </head>

   <body>
      <font color = "#FF00FF">This text is in pink</font><br />
      <font color = "red">This text is red</font>
   </body>

</html>
```

This will produce the following result –

This text is in

pink This text is

red

Comments in HTML

Comment is a piece of code which is ignored by any web browser. It is a good practice to add comments into your HTML code, especially in complex documents, to indicate sections of a document, and any other notes to anyone looking at the code. Comments help you and others understand your code and increases code readability.

HTML comments are placed in between <!-- ... --> tags. So, any content placed with-in <!-- ... --> tags will be treated as comment and will be completely ignored by the browser.

```
<!DOCTYPE html>
<html>

   <head>     <!-- Document Header Starts -->
      <title>This is document title</title>
   </head> <!-- Document Header Ends -->

   <body>
      <p>Document content goes here..........................</p>
   </body>

</html>
```

This will produce the following result without displaying the content given as a part of comments –

Document content goes here.....

HTML Images

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This tutorial will take you through simple steps to use images in your web pages.

Insert Image

You can insert any image in your web page by using <img> tag. Following is the simple syntax to use this tag.

```
<img src = "Image URL" ... attributes-list/>
```

The <img> tag is an empty tag, which means that, it can contain only list of attributes and it has no closing tag.

Example

To try following example, let's keep our HTML file test.htm and image file test.png in the same directory –

```
<!DOCTYPE html>
<html>

   <head>
      <title>Using Image in Webpage</title>
   </head>

   <body>
      <p>Simple Image Insert</p>
      <img src = "/html/images/test.png" alt = "Test Image" />
   </body>

</html>
```

You can use PNG, JPEG or GIF image file based on your comfort but make sure you specify correct image file name in src attribute. Image name is always case sensitive.

The alt attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.

Set Image Location

Usually we keep all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory images inside the home directory where we will keep our image test.png.

Example

Assuming our image location is "image/test.png", try the following example –

```
<!DOCTYPE html>
<html>

    <head>
        <title>Set Image Border</title>
    </head>

    <body>
        <p>Setting image Border</p>
        <img src = "/html/images/test.png" alt = "Test Image" border
= "3"/>
    </body>

</html>
```

Set Image Width/Height

You can set image width and height based on your requirement using width and height attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

Example

```
<!DOCTYPE html>
<html>

    <head>
        <title>Set Image Width and Height</title>
    </head>

    <body>
        <p>Setting image width and height</p>
        <img src = "/html/images/test.png" alt = "Test Image" width = "150" height =
"100"/>
    </body>

</html>
```

Set Image Border

By default, image will have a border around it, you can specify border thickness in terms of pixels using border attribute. A thickness of 0 means, no border around the picture.

Example

```
<!DOCTYPE html>
<html>

     <head>
          <title>Set Image Border</title>
     </head>

     <body>
          <p>Setting image Border</p>
          <img src = "/html/images/test.png" alt = "Test Image" border
= "3"/>
     </body>

</html>
```

Set Image Alignment

By default, image will align at the left side of the page, but you can use align attribute to set it in the center or right.

Example

```
<!DOCTYPE html>
<html>

     <head>
          <title>Set Image Alignment</title>
     </head>

     <body>
          <p>Setting image Alignment</p>
          <img src = "/html/images/test.png" alt = "Test Image" border
= "3" align = "right"/>
     </body>

</html>
```

Image Mapping

The HTML <map> tag is used for defining an image map along with <img> tag.

Example

```
<!DOCTYPE html>
<html>

     <head>
          <title>HTML map Tag</title>
     </head>

     <body>
          <img src = "/images/html.gif" alt = "HTML Map" border = "0" usemap = "#html"/>

          <!-- Create           Mappings -->
          <map name = "html">
               <area shape = "circle" coords = "154,150,59" href =
"about/about_team.htm"
                    alt = "Team" target = "_self" />
          </map>
     </body>

</html>
```

This will produce the following result, find the image map on bottom right.

HTML Table

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.

The HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells. The elements under <td> are regular and left aligned by default

Example

```
<!DOCTYPE html>
```

```
<html>

    <head>
        <title>HTML Tables</title>
    </head>

    <body>
        <table border = "1">
            <tr>
                <td>Row 1, Column 1</td>
                <td>Row 1, Column 2</td>
            </tr>

            <tr>
                <td>Row 2, Column 1</td>
                <td>Row 2, Column 2</td>
              </tr>
           </table>

    </body>
</html>
```

This will produce the following result –

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

Here, the border is an attribute of <table> tag and it is used to put a border across all the cells. If you do not need a border, then you can use border = "0".

Table Heading

Table heading can be defined using <th> tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Normally you will put your top row as table heading as shown below, otherwise you can use <th> element in any row. Headings, which are defined in
<th> tag are centered and bold by

```
<!DOCTYPE html>
<html>

    <head>
        <title>HTML Table Header</title>
    </head>

    <body>
        <table border = "1">
```

default. Example

```
            <tr>
                <th>Name</th>
                <th>Salary</th>
            </tr>
            <tr>
                <td>Ramesh Raman</td>
                <td>5000</td>
            </tr>

            <tr>
                <td>Shabbir Hussein</td>
                <td>7000</td>
              </tr>
           </table>
    </body>

</html>
```

This will produce the following result –

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Cellpadding and Cellspacing Attributes

There are two attributes called cellpadding and cellspacing which you will use to adjust the white space in your table cells. The cellspacing attribute defines space between table cells, while cellpadding represents the distance between cell borders and the content within a cell.

```
<!DOCTYPE html>
<html>

    <head>
        <title>HTML Table Cellpadding</title>
    </head>

    <body>
        <table border = "1" cellpadding = "5" cellspacing = "5">
            <tr>
                <th>Name</th>
                <th>Salary</th>
            </tr>
            <tr>
                <td>Ramesh Raman</td>
                <td>5000</td>
            </tr>
            <tr>
                <td>Shabbir Hussein</td>
```

Example

```
                <td>7000</td>
              </tr>
            </table>
      </body>

</html>
```

This will produce the following result –

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Colspan and Rowspan Attributes

You will use colspan attribute if you want to merge two or more columns into a single column. Similar way you will use rowspan if you want to merge two or more rows.

Example

```
<!DOCTYPE html>
<html>

      <head>
            <title>HTML Table Colspan/Rowspan</title>
      </head>

      <body>
            <table border = "1">
                  <tr>
                        <th>Column 1</th>
                        <th>Column 2</th>
                        <th>Column 3</th>
                  </tr>
                  <tr>
                        <td rowspan = "2">Row 1 Cell 1</td>
                        <td>Row 1 Cell 2</td>
                        <td>Row 1 Cell 3</td>
                  </tr>
                  <tr>
                        <td>Row 2 Cell 2</td>
                        <td>Row 2 Cell 3</td>
                  </tr>
                  <tr>
                        <td colspan = "3">Row 3 Cell 1</td>
                     </tr>
                </table>
      </body>

</html>
```

This will produce the following result –

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

Table Height and Width

You can set a table width and height using width and height attributes. You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

Example

```
<!DOCTYPE html>
<html>

    <head>
        <title>HTML Table Width/Height</title>
    </head>

    <body>
        <table border = "1" width = "400" height = "150">
            <tr>
                <td>Row 1, Column 1</td>
                <td>Row 1, Column 2</td>
            </tr>

            <tr>
                <td>Row 2, Column 1</td>
                <td>Row 2, Column 2</td>
              </tr>
           </table>
    </body>

</html>
```

This will produce the following result –

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

Table Caption

The caption tag will serve as a title or explanation for the table and it shows up at the top of the table. This tag is deprecated in newer version of HTML/XHTML.

Example

```
<!DOCTYPE html>
```

```
<html>

    <head>
        <title>HTML Table Caption</title>
    </head>

    <body>
        <table border = "1" width = "100%">
            <caption>This is the caption</caption>

            <tr>
                <td>row 1, column 1</td><td>row 1, columnn 2</td>
            </tr>

            <tr>
                <td>row 2, column 1</td><td>row 2, columnn 2</td>
              </tr>
           </table>
    </body>

</html>
```

This will produce the following result –

This is the caption	
row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

Hyperlink in HTML

A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks.

Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

Linking Documents

A link is specified using HTML tag <a>. This tag is called anchor tag and anything between the opening <a> tag and the closing </a> tag becomes part of the link and a user can click that part to reach to the linked document. Following is the simple syntax to use <a> tag.

<a href = "Document URL" ... attributes-list>Link
Text</a> Example

Let's try following example which links http://www.gpwfaridabad.edu.in at your page –

```
<!DOCTYPE html>
<html>

    <head>
        <title>Hyperlink Example</title>
    </head>

    <body>
        <p>Click following link</p>
        <a href = "https://www. gpwfaridabad.edu.in" target = "_self">Govt.
Polytechnic for Women, Faridabad</a>
    </body>

</html>
```

HTML Forms

HTML Forms are required, when you want to collect some data from the site visitor. For example, during user registration you would like to collect information such as name, email address, credit card, etc.

A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML <form> tag is used to create an HTML form and it has following syntax –

```
<form action = "Script URL" method = "GET|POST">
    form elements like input, textarea etc.
</form>
```

Form Attributes

Apart from common attributes, following is a list of the most frequently used form attributes –

Sr.No	Attribute & Description
1	action Backend script ready to process your passed data.

2	method Method to be used to upload data. The most frequently used are GET and POST methods.
3	target Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
4	enctype You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are – application/x-www-form-urlencoded – This is the standard method most forms use in simple scenarios. mutlipart/form-data – This is used when you want to upload binary data in the form of files like image, word file etc.

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form –

- Text Input Controls
- Checkboxes Controls
- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

Text Input Controls

There are three types of text input used on forms –

- **Single-line text input controls** – This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag. Here is a basic example of a single-line text input used to take first name and last name –

```
<!DOCTYPE html>
<html>

   <head>
      <title>Text Input Control</title>
   </head>

   <body>
      <form >
         First name: <input type = "text" name = "first_name" />
         <br>
         Last name: <input type = "text" name = "last_name" />
      </form>
   </body>

</html>
```

Firstname:

Last name:

Attributes

Following is the list of attributes for <input> tag for creating text field.

Sr.No	Attribute & Description
1	**type** Indicates the type of input control and for text input control it will be set to **text**.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** This can be used to provide an initial value inside the control.
4	**size** Allows to specify the width of the text-input control in terms of characters.

5	**maxlength** Allows to specify the maximum number of characters a user can enter into the text box.

- **Password input controls** – This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.

  Here is a basic example of a single-line password input used to take user password –

```
<!DOCTYPE html>
<html>

   <head>
      <title>Password Input Control</title>
   </head>

   <body>
      <form >
         User ID : <input type = "text" name = "user_id" />
         <br>
         Password: <input type = "password" name = "password" />
      </form>
   </body>

</html>
```

This will produce the following result –

UserID:

Password:

Attributes

Following is the list of attributes for <input> tag for creating password field.

Sr.No	Attribute & Description
1	**type** Indicates the type of input control and for password input control it will be set

	to **password**.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** This can be used to provide an initial value inside the control.
4	**size** Allows to specify the width of the text-input control in terms of characters.
5	**maxlength** Allows to specify the maximum number of characters a user can enter into the text box.

- **Multi-line text input controls** − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

  Here is a basic example of a multi-line text input used to take item description −

```
<!DOCTYPE html>
<html>

     <head>
          <title>Multiple-Line Input Control</title>
     </head>

     <body>
          <form>
               Description : <br />
               <textarea rows = "5" cols = "50" name = "description"> Enter description
                    here...
               </textarea>
          </form>
     </body>

</html>
```

Description:

Attributes

Following is the list of attributes for <textarea> tag.

Sr.No	Attribute & Description
1	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
2	**rows** Indicates the number of rows of text area box.
3	**cols** Indicates the number of columns of text area box

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox..

Example

Here is an example HTML code for a form with two checkboxes –

```
<!DOCTYPE html>
<html>

   <head>
      <title>Checkbox Control</title>
   </head>

   <body>
```

```
         <form>
              <input type = "checkbox" name = "maths" value = "on">
Maths
              <input type = "checkbox" name = "physics" value = "on">
Physics
         </form>
    </body>

</html>
```

This will produce the following result –

☐ Maths☐ Physics

Attributes

Following is the list of attributes for <checkbox> tag.

Sr.No	Attribute & Description
1	**type** Indicates the type of input control and for checkbox input control it will be set to **checkbox**.
2	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
3	**value** The value that will be used if the checkbox is selected.
4	**checked** Set to *checked* if you want to select it by default.

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

Example

Here is example HTML code for a form with two radio buttons –

```
<!DOCTYPE html>
<html>

     <head>
          <title>Radio Box Control</title>
     </head>

     <body>
          <form>
          <input type = "radio" name = "subject" value = "maths">Maths
          <input type= "radio" name= "subject" value= "physics">Physics
          </form>
     </body>

</html>
```

This will produce the following result –

◯ Maths ◯ Physics

Attributes

Following is the list of attributes for radio button.

Sr.No	Attribute & Description
1	type Indicates the type of input control and for checkbox input control it will be set to radio.
2	name Used to give a name to the control which is sent to the server to be recognized and get the value.
3	value The value that will be used if the radio box is selected.
4	checked

	Set to *checked* if you want to select it by default.

Select Box Control

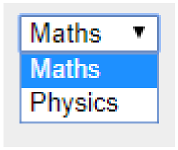
A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

Example

Here is example HTML code for a form with one drop down box

```
<!DOCTYPE html>
<html>

  <head>
    <title>Select Box Control</title>
  </head>

  <body>
    <form>
      <select name = "dropdown">
        <option value = "Maths" selected>Maths</option>
        <option value = "Physics">Physics</option>
      </select>
    </form>
  </body>

</html>
```

This will produce the following result –



Attributes

Following is the list of important attributes of <select> tag –

Sr.No	Attribute & Description
1	name

	Used to give a name to the control which is sent to the server to be recognized and get the value.
2	**size** This can be used to present a scrolling list box.
3	**multiple** If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option> tag –

Sr.No	Attribute & Description
1	**value** The value that will be used if an option in the select box box is selected.
2	**selected** Specifies that this option should be the initially selected value when the page loads.
3	**label** An alternative way of labeling options

File Upload Box

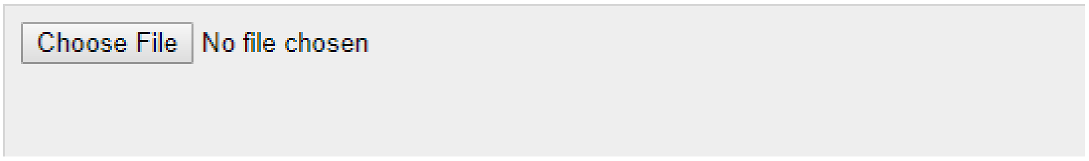
If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to **file**.

Example

Here is example HTML code for a form with one file upload box –

```
<!DOCTYPE html>
<html>

  <head>
    <title>File Upload Box</title>
  </head>
  <body>
    <form>
      <input type = "file" name = "fileupload" accept = "image/*" />
    </form>
  </body>

</html>
```

This will produce the following result –



Attributes

Following is the list of important attributes of file upload box –

Sr.No	Attribute & Description
1	**name** Used to give a name to the control which is sent to the server to be recognized and get the value.
2	**accept** Specifies the types of files that the server accepts.

Button Controls

There are various ways in HTML to create clickable buttons. You can also create a clickable button using <input>tag by setting its type attribute to **button**. The type attribute can take the following values –

Sr.No	Type & Description
1	**submit** This creates a button that automatically submits a form.

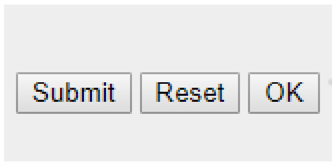
2	**reset** This creates a button that automatically resets form controls to their initial values.
3	**button** This creates a button that is used to trigger a client-side script when the user clicks that button.
4	**image** This creates a clickable button but we can use an image as background of the button.

Example

Here is example HTML code for a form with three types of buttons –

```
<!DOCTYPE html>
<html>

  <head>
    <title>File Upload Box</title>
  </head>

  <body>
    <form>
      <input type = "submit" name = "submit" value = "Submit" />
      <input type = "reset" name = "reset" value = "Reset" />
      <input type = "button" name = "ok" value = "OK" />
      <input type = "image" name = "imagebutton" src = "/html/images/logo.png" />
    </form>
  </body>

</html>
```

This will produce the following result –



Hidden Form Controls

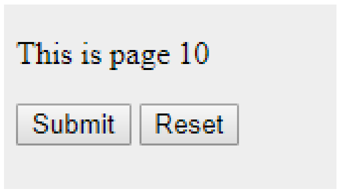
Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page.

Example

Here is example HTML code to show the usage of hidden control –

```
<!DOCTYPE html>
<html>

  <head>
    <title>File Upload Box</title>
  </head>

  <body>
    <form>
      <p>This is page 10</p>
      <input type = "hidden" name = "pagename" value = "10" />
      <input type = "submit" name = "submit" value = "Submit" />
      <input type = "reset" name = "reset" value = "Reset" />
    </form>
  </body>

</html>
```

This will produce the following result –


This is page 10

Style Sheets in HTML

Cascading Style Sheets (CSS) describe how documents are presented on screens, in print, or perhaps how they are pronounced. W3C has actively promoted the use of style sheets on the Web since the consortium was founded in 1994.

Cascading Style Sheets (CSS) provide easy and effective alternatives to specify various attributes for the HTML tags. Using CSS, you can specify a number of style properties for a given HTML element. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

Example

First let's consider an example of HTML document which makes use of <font> tag and associated attributes to specify text color and font size –

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML CSS</title>
   </head>

   <body>
      <p><font color = "green" size = "5">Hello, World!</font></p>
   </body>

</html>
```

We can re-write above example with the help of Style Sheet as follows –

```
<!DOCTYPE html>
<html>

   <head>
      <title>HTML CSS</title>
   </head>

   <body>
      <p style = "color:green; font-size:24px;" >Hello, World!</p>
   </body>

</html>
```

This will produce the following result –

Hello, World!

Type of CSS:

You can use CSS in three ways in your HTML document –

- **External Style Sheet** – Define style sheet rules in a separate .css file and then include that file in your HTML document using HTML <link> tag.
- **Internal Style Sheet** – Define style sheet rules in header section of the HTML document using <style> tag.
- **Inline Style Sheet** – Define style sheet rules directly along-with the HTML elements using **style** attribute.

External Style Sheet

If you need to use your style sheet to various pages, then its always recommended to define a common style sheet in a separate file. A cascading style sheet file will have extension as **.css** and it will be included in HTML files using <link> tag.

Example

Consider we define a style sheet file **style.css** which has following rules –

```
.red {
  color: red;
}
.thick {
  font-size:20px;
}
.green
  { color:gre
  en;
```

Here we defined three CSS rules which will be applicable to three different classes defined for the HTML tags. I suggest you should not bother about how these rules are being defined because you will learn them while studying CSS. Now let's make use of the above external CSS file in our following HTML document –

```
<!DOCTYPE html>
<html>

    <head>
        <title>HTML External CSS</title>
        <link rel = "stylesheet" type = "text/css" href = "/html/style.css">
    </head>

    <body>
        <p class = "red">This is red</p>
        <p class = "thick">This is thick</p>
        <p class = "green">This is green</p>
        <p class = "thick green">This is thick and green</p>
    </body>

</html>
```

This will produce the following result –

This is red

This is

thick This

is green

This is thick and green

Internal Style Sheet

If you want to apply Style Sheet rules to a single document only, then you can include those rules in header section of the HTML document using <style> tag.

Rules defined in internal style sheet overrides the rules defined in an external

CSS file. Example

Let's re-write above example once again, but here we will write style sheet rules in the same HTML document using <style> tag −

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Internal CSS</title>

    <style type = "text/css">
      .red {
        color: red;
      }
      .thick{
        font-size:20px;
      }
      .green
        { color:gre
        en;
      }
    </style>
  </head>

  <body>
    <p class = "red">This is red</p>
    <p class = "thick">This is thick</p>
    <p class = "green">This is green</p>
    <p class = "thick green">This is thick and green</p>
  </body>
```

This will produce the following result −

This is red

This is thick

This is green

This is thick and green

Inline Style Sheet

You can apply style sheet rules directly to any HTML element using **style** attribute of the relevant tag. This should be done only when you are interested to make a particular change in any HTML element only.

Rules defined inline with the element overrides the rules defined in an external CSS file as well as the rules defined in <style> element.

Example

Let's re-write above example once again, but here we will write style sheet rules along with the HTML elements using **style** attribute of those elements.

```
<!DOCTYPE html>
<html>

    <head>
        <title>HTML Inline CSS</title>
    </head>

    <body>
        <p style = "color:red;">This is red</p>
        <p style = "font-size:20px;">This is thick</p>
        <p style = "color:green;">This is green</p>
        <p style = "color:green;font-size:20px;">This is thick and green</p>
    </body>

</html>
```

This will produce the following result –

This is red

This is

thick This

is green

This is thick and green

Layouts in HTML

A webpage layout is very important to give better look to your website. It takes considerable time to design a website's layout with great look and feel.

Now-a-days, all modern websites are using CSS and JavaScript based framework to come up with responsive and dynamic websites but you can create a good layout using simple HTML tables or division tags in combination with other formatting tags. This chapter will give you few examples on how to create a simple but working layout for your webpage using pure HTML and its attributes.

HTML Layout - Using Tables

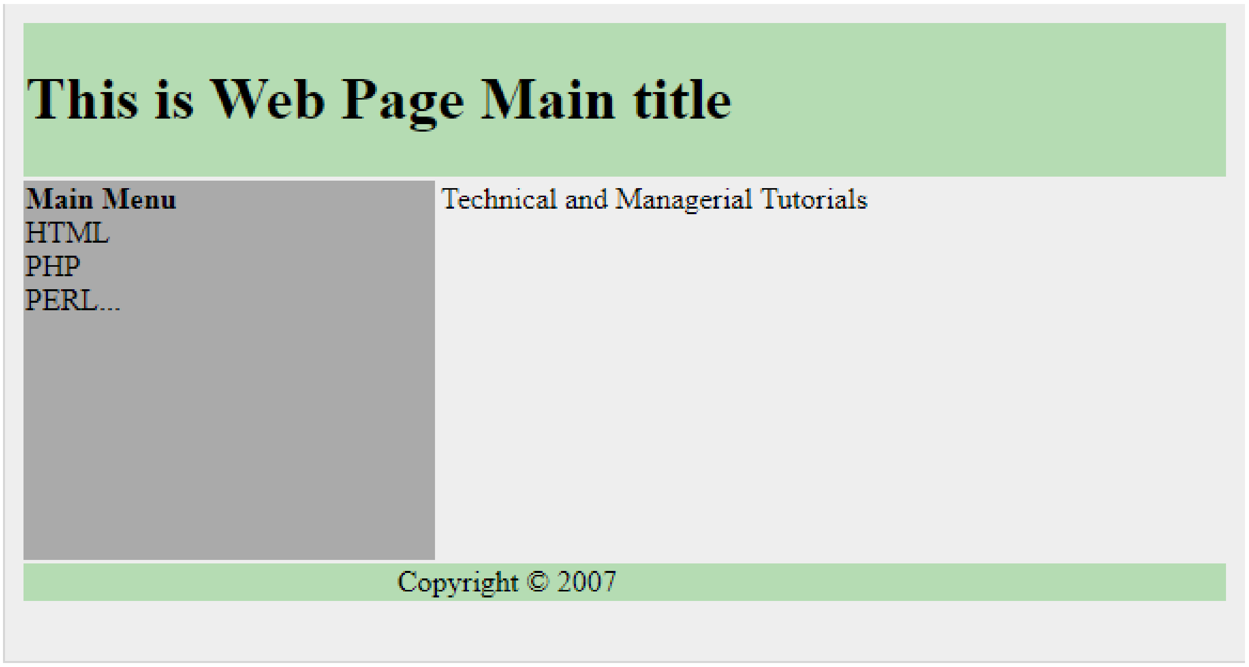
The simplest and most popular way of creating layouts is using HTML <table> tag. These tables are arranged in columns and rows, so you can utilize these rows and columns in whatever way you like.

Example

For example, the following HTML layout example is achieved using a table with 3 rows and 2 columns but the header and footer column spans both columns using the colspan attribute –

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Layout using Tables</title>
  </head>

  <body>
    <table width = "100%" border = "0">

      <tr>
     <td colspan = "2" bgcolor = "#b5dcb3">
       <h1>This is Web Page Main title</h1>
        </td>
      </tr>
      <tr valign = "top">
        <td bgcolor = "#aaa" width = "50">
          <b>Main Menu</b><br
          /> HTML<br />
          PHP<br
          /> PERL...
        </td>

        <td bgcolor = "#eee" width = "100" height =
          "200"> Technical and Managerial Tutorials
        </td>
      </tr>
      <tr>
        <td colspan = "2" bgcolor = "#b5dcb3">
          <center>
            Copyright © 2007
          </center>
        </td>
      </tr>

    </table>
  </body>

</html>
```

This will produce the following result –


This is Web Page Main title

Main Menu
HTML
PHP
PERL...

Technical and Managerial Tutorials

Copyright © 2007

Multiple Columns Layout - Using Tables

You can design your webpage to put your web content in multiple pages. You can keep your content in middle column and you can use left column to use menu and right column can be used to put advertisement or some other stuff. This layout will be very similar to what we have at our website tutorialspoint.com.

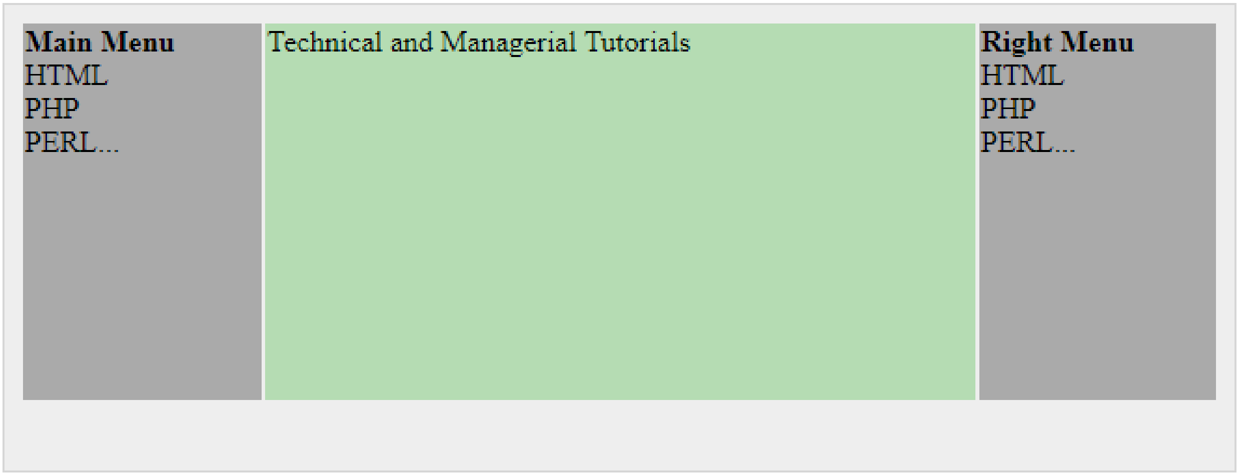
Example

Here is an example to create three column layout –

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>Three Column HTML Layout</title>
  </head>
  <body>
    <table width = "100%" border = "0">
            <tr valign = "top">
        <td bgcolor = "#aaa" width = "20%">
          <b>Main Menu</b><br
          /> HTML<br />
          PHP<br
          /> PERL...
        </td>

        <td bgcolor = "#b5dcb3" height = "200" width =
          "60%"> Technical and Managerial Tutorials
        </td>

        <td bgcolor = "#aaa" width = "20%">
          <b>Right Menu</b><br
          /> HTML<br />
          PHP<br
          /> PERL...
        </td>
      </tr>

    <table>
  </body>

</html>
```

This will produce the following result −


Main Menu
HTML
PHP
PERL...

Technical and Managerial Tutorials

Right Menu
HTML
PHP
PERL...

HTML Layouts - Using DIV, SPAN

The <div> element is a block level element used for grouping HTML elements. While the <div> tag is a block-level element, the HTML <span> element is used for grouping elements at an inline level.

Although we can achieve pretty nice layouts with HTML tables, but tables weren't really designed as a layout tool. Tables are more suited to presenting tabular data.

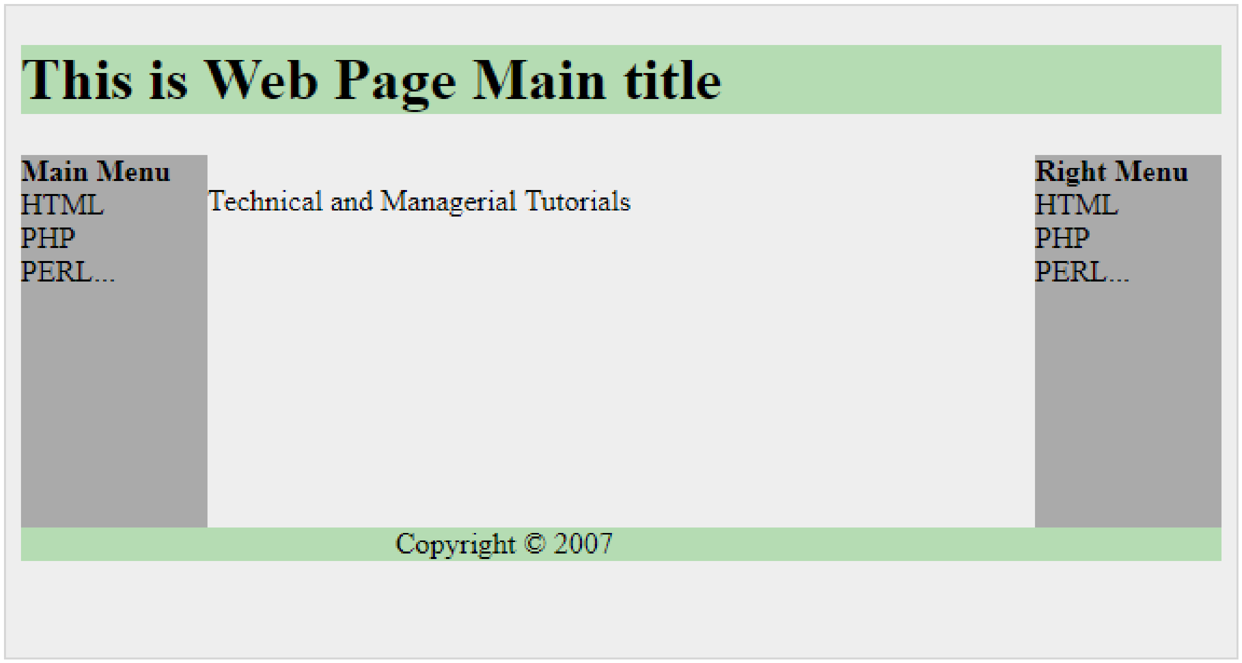
Note – This example makes use of Cascading Style Sheet (CSS), so before understanding this example you need to have a better understanding on how CSS works.

Example

Here we will try to achieve same result using <div> tag along with CSS, whatever you have achieved using <table> tag in previous example.

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Layouts using DIV, SPAN</title>
  </head>

  <body>
    <div style = "width:100%">

      <div style = "background-color:#b5dcb3; width:100%">
        <h1>This is Web Page Main title</h1>
      </div>

      <div style = "background-color:#aaa; height:200px; width:100px; float:left;">
        <div><b>Main
        Menu</b></div> HTML<br />
        PHP<br
        /> PERL...
      </div>

      <div style = "background-color:#eee; height:200px; width:350px; float:left;" >
        <p>Technical and Managerial Tutorials</p>
      </div>

      <div style = "background-color:#aaa; height:200px; width:100px; float:right;">
        <div><b>Right
        Menu</b></div> HTML<br />
        PHP<br
        /> PERL...
      </div>
      <div style = "background-color:#b5dcb3; clear:both">
        <center>
          Copyright © 2007
        </center>
      </div>

    </div>
  </body>

</html>
```

This will produce the following result –



This is Web Page Main title

Main Menu
HTML
PHP
PERL...

Technical and Managerial Tutorials

Right Menu
HTML
PHP
PERL...

Copyright © 2007

Chapter 2
PHP

Introduction

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.

- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

How PHP works & Basic PHP Syntax

PHP is server side scripting language. So, it is execute on the server, and the plain HTML result is sent back to the browser.

PHP File Structure

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

Syntax:

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

The default file extension for PHP files is .php.

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

Below, we have an example of a simple PHP file structure, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello PHP!" on a web page:

```
<html>
        <head>
        <title>Hello PHP</title>
    </head>

    <body>
        <?php echo "Hello, PHP!";?>
    </body>
```

```
</html>
```

It will produce following result –

```
Hello, PHP!
```

The php.ini File

The PHP configuration file, php.ini, is the final and most immediate way to affect PHP's functionality. The php.ini file is read each time PHP is initialized. In other words, whenever httpd is restarted for the module version or with each script execution for the CGI version. If your change isn't showing up, remember to stop and restart httpd. If it still isn't showing up, use phpinfo() to check the path to php.ini.

The configuration file is well commented and thorough. Keys are case sensitive, keyword values are not; whitespace, and lines beginning with semicolons are ignored. Booleans can be represented by 1/0, Yes/No, On/Off, or True/False. The default values in php.ini-dist will result in a reasonable PHP installation that can be tweaked later.

Here we are explaining the important settings in php.ini which you may need for your PHP Parser.

short_open_tag = Off

Short open tags look like this: <? ?>. This option must be set to Off if you want to use XML functions.

safe_mode = Off

If this is set to On, you probably compiled PHP with the --enable-safe-mode flag. Safe mode is most relevant to CGI use. See the explanation in the section "CGI compile-time options". safe_mode_exec_dir = [DIR]

This option is relevant only if safe mode is on; it can also be set with the --with-exec-dir flag during the Unix build process. PHP in safe mode only executes external binaries out of this directory. The default is /usr/local/bin. This has nothing to do with serving up a normal PHP/HTML Web page.

safe_mode_allowed_env_vars = [PHP_]

This option sets which environment variables users can change in safe mode. The default is only those variables prepended with "PHP_". If this directive is empty, most variables are alterable. safe_mode_protected_env_vars = [LD_LIBRARY_PATH]

This option sets which environment variables users can't change in safe mode, even if safe_mode_allowed_env_vars is set permissively

disable_functions = [function1, function2...]

A welcome addition to PHP4 configuration and one perpetuated in PHP5 is the ability to disable selected functions for security reasons. Previously, this necessitated hand-editing the C code from which PHP was made. Filesystem, system, and network functions should probably be the first to go because allowing the capability to write files and alter the system over HTTP is never such a safe idea.

max_execution_time = 30

The function set_time_limit() won.t work in safe mode, so this is the main way to make a script time out in safe mode. In Windows, you have to abort based on maximum memory consumed rather than time. You can also use the Apache timeout setting to timeout if you use Apache, but that will apply to non-PHP files on the site too.

PHP Variables

The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables –

- **Integers** – are whole numbers, without a decimal point, like 4195.
- **Doubles** – are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** – have only two possible values either true or false.
- **NULL** – is a special type that only has one value: NULL.
- **Strings** – are sequences of characters, like 'PHP supports string operations.'
- **Arrays** – are named and indexed collections of other values.
- **Objects** – are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** – are special variables that hold references to resources external to PHP (such as database connections).

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type. They correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so –

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is (2**31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2**31 . 1) (or .2,147,483,647).

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code –

```
<?php
  $many = 2.2888800;
  $many_2 = 2.2111200;
  $few = $many + $many_2;
  print("$many + $many_2 = $few <br>");
?>
```

It produces the following browser

output − 2.28888 + 2.21112 = 4.5

Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −

```
if (TRUE)
  print("This will always print<br>");

else
  print("This will never print<br>");
```

Interpreting other types as Booleans

Here are the rules for determine the "truth" of any value not already of the Boolean type −

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;
$true_str = "Tried and true"
$true_array[49] = "An array element";
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_str = "";
```

NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this –

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed –

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties –

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
  $variable = "name";
  $literally = 'My $variable will not print!';

  print($literally);
  print "<br>";

  $literally = "My $variable will
  print!"; print($literally);
?>
```

This will produce following result –

```
My $variable will not
print! My name will print
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP –

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are –

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types –

- Local variables

- Function parameters
- Global variables
- Static variables

Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function –

```
<?php
  $x = 4;

  function assignx () {
    $x = 0;
    print "\$x inside function is $x. <br />";
  }

  assignx();
  print "\$x outside of function is $x. <br />";
?>
```

This will produce the following result –

```
$x inside function is 0.
$x outside of function is 4.
```

Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be –

```
<?php
  // multiply a value by 10 and return it to the
  caller function multiply ($value) {
    $value = $value *
    10; return $value;
  }
    $retval = multiply (10);
  Print "Return value is $retval\n";
?>
```

This will produce the following result –

Return value is 100

Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword **GLOBAL** in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example –

```
<?php
  $somevar = 15;

  function addit()
    { GLOBAL
    $somevar;
    $somevar++;

    print "Somevar is $somevar";
  }

  addit();
```

This will produce the following
result – Somevar is 16

Static Variables

The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

```
<?php
  function keep_track()
    { STATIC $count =
    $count++;
    print $count;
    print "<br />";
  }

  keep_track(
  );
  keep_track(
  );
```

This will produce the following result −

```
1
2
3
```

PHP Conditional Statements

Like most programming languages, PHP also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time. This means, you can create test conditions in the form of expressions that evaluates to either true or false and based on these results you can perform certain actions.

There are several statements in PHP that you can use to make decisions:

- The **if** statement
- The **if...else** statement
- The **if...elseif....else** statement
- The **switch...case** statement

We will explore each of these statements in the coming sections.

The if Statement

The *if* statement is used to execute a block of code only if the specified condition evaluates to true. This is the simplest PHP's conditional statements and can be written like:

```
if(condition){
   // Code to be executed
}
```

The following example will output "Have a nice weekend!" if the current day is Friday:

Example

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
}
?>
```

The if...else Statement

You can enhance the decision making process by providing an alternative choice through adding an *else* statement to the *if* statement. The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false. It can be written, like this:

```
if(condition){
    // Code to be executed if condition is true
} else{
    // Code to be executed if condition is false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!"

Example

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} else{
    echo "Have a nice day!";
}
?>
```

The if...elseif...else Statement

The *if...elseif...else* a special statement that is used to combine multiple *if...else* statements.

```
if(condition1){
    // Code to be executed if condition1 is true
} elseif(condition2){
    // Code to be executed if the condition1 is false and condition2 is true
} else{
    // Code to be executed if both condition1 and condition2 are false
}
```

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday, otherwise it will output "Have a nice day!"

Example

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} elseif($d == "Sun"){
    echo "Have a nice Sunday!";
} else{
    echo "Have a nice day!";
}
?>
```

The Ternary Operator

The ternary operator provides a shorthand way of writing the *if...else* statements. The ternary operator is represented by the question mark (?) symbol and it takes three operands: a condition to check, a result for true, and a result for false.

To understand how this operator works, consider the following examples:

Example

```
<?php
if($age < 18){
    echo 'Child'; // Display Child if age is less than 18
} else{
    echo 'Adult'; // Display Adult if age is greater than or equal to 18
}
?>
```

Using the ternary operator the same code could be written in a more compact way:

Example

```
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

The ternary operator in the example above selects the value on the left of the colon (i.e. 'Child') if the condition evaluates to true (i.e. if $age is less than 18), and selects the value on the right of the colon (i.e. 'Adult') if the condition evaluates to false.

Operators in PHP

Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language – Assume variable A holds 10 and variable B holds 20 then –

Show Examples

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators

There are following comparison operators supported by PHP

language Assume variable A holds 10 and variable B holds 20 then

–

Show Examples

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

There are following logical operators supported by PHP

language Assume variable A holds 10 and variable B holds

20 then – Show Examples

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.

or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

There are following assignment operators supported by PHP language – Show Examples

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A

/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax –

Show Examples

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Operators Categories

All the operators we have discussed above can be categorised into following categories –

- Unary prefix operators, which precede a single operand.
- Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
- The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Assignment operators, which assign a value to a variable.

Precedence of PHP Operators

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator –

For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.

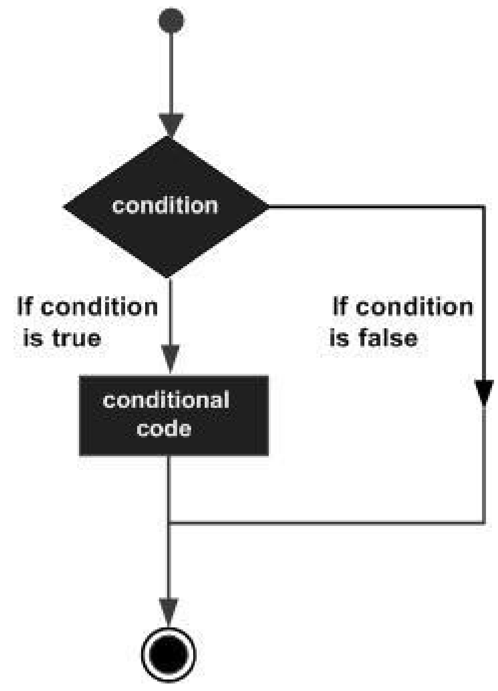
Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

Decision Making

The if, elseif ...else and switch statements are used to take decision based on the different condition.

You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements –



- **if...else statement** – use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** – is used with the if...else statement to execute a set of code if **one** of the several condition is true
- **switch statement** – is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if. else statement.

Syntax

if (*condition*)
  code to be executed if condition is true;
else
  code to be executed if condition is false;

Example

The following example will output "Have a nice weekend!" if the current day is Friday, Otherwise, it will output "Have a nice day!":

```
<html>
  <body>

    <?php
      $d = date("D");

      if ($d == "Fri")
        echo "Have a nice weekend!";

      else
        echo "Have a nice day!";
    ?>

  </body>
</html>
```

It will produce the following

result − Have a nice weekend!

The ElseIf Statement

If you want to execute some code if one of the several conditions are true use the elseif statement

Syntax

if (*condition*)
  code to be executed if condition is true;
elseif (*condition*)
  code to be executed if condition is true;
else
  code to be executed if condition is false;

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise, it will output "Have a nice day!"
−

```
<html>
  <body>

    <?php
      $d = date("D");

      if ($d == "Fri")
        echo "Have a nice weekend!";

      elseif ($d == "Sun")
        echo "Have a nice Sunday!";

      else
        echo "Have a nice day!";
    ?>

  </body>
</html>
```

It will produce the following result –

```
Have a nice Weekend!
```

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch
  (expression){ case
  label1:
    code to be executed if expression = label1;
    break;

  case label2:
    code to be executed if expression = label2;
    break;
    default:

  code to be executed
  if expression is different
  from both label1 and
  label2;
}
```

Example

The *switch* statement works in an unusual way. First it evaluates given expression then seeks a lable to match the resulting value. If a matching value is found then the code associated with the matching label will be executed or if none of the lable matches then statement will execute any specified default code.

```
<html>
  <body>

    <?php
      $d = date("D");

      switch
        ($d){ case
        "Mon":
          echo "Today is
          Monday"; break;

        case "Tue":
          echo "Today is
          Tuesday"; break;

        case "Wed":
          echo "Today is
          Wednesday"; break;

        case "Thu":
          echo "Today is
          Thursday"; break;

        case "Fri":
          echo "Today is
          Friday"; break;

        case "Sat":
          echo "Today is
          Saturday"; break;

        case "Sun":
          echo "Today is
          Sunday"; break;

        default:
          echo "Wonder which day is this ?";
      }
    ?>
  </body>
</html>
```

It will produce the following result –

```
Today is Monday
```

Loops in PHP

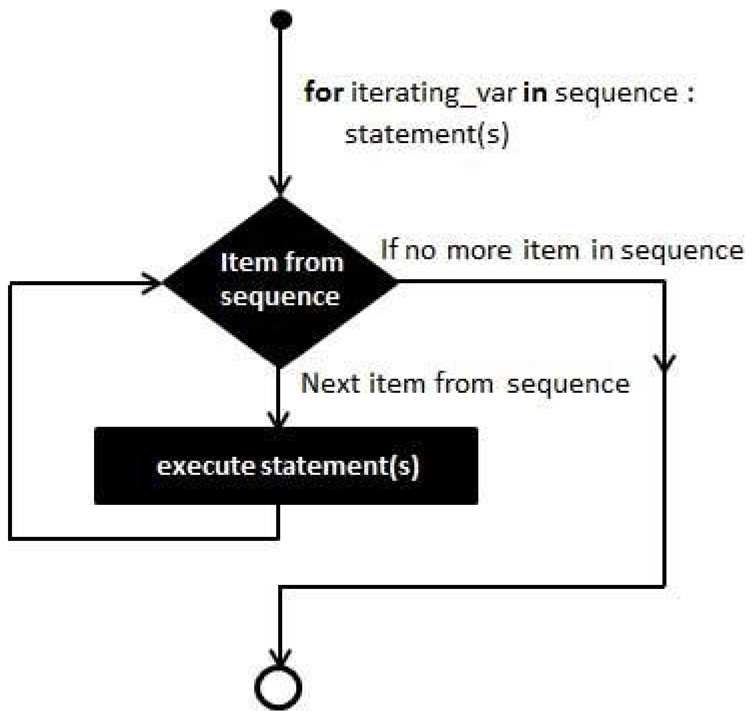
Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

- **for** – loops through a block of code a specified number of times.
- **while** – loops through a block of code if and as long as a specified condition is true.
- **do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** – loops through a block of code for each element in an array.

We will discuss about **continue** and **break** keywords used to control the loops execution.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.



Syntax
for (*initialization*; *condition*;
  increment){ *code to be executed;*
}

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop –

```
<html>
  <body>

    <?php
      $a = 0;
      $b = 0;

      for( $i = 0; $i<5; $i++ ) {
        $a += 10;
        $b += 5;
      }

      echo ("At the end of the loop a = $a and b = $b" );
    ?>

  </body>
</html>
```

This will produce the following

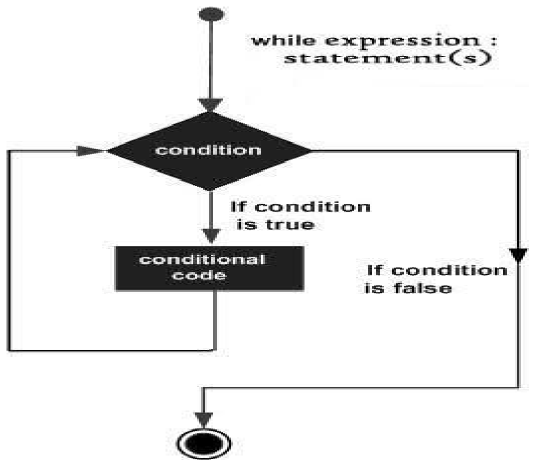
result – At the end of the loop a = 50

and b = 25

The while loop statement

The while statement will execute a block of code if and as long as a test expression is true.

If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.



Syntax

while (*condition*) {
  code to be executed;
}

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

```
<html>
  <body>

    <?php
      $i = 0;
      $num = 50;

      while( $i < 10) {
        $num--;
        $i++;
      }

      echo ("Loop stopped at i = $i and num = $num" );
    ?>

  </body>
</html>
```

This will produce the following

result – Loop stopped at i = 10 and

num = 40

The do...while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Synta
x do {
  code to be executed;
}
while

(*condition*);

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 –

```
<html>
  <body>

    <?php
      $i = 0;
      $num = 0;

      do {
        $i++;
      }

      while( $i < 10 );
      echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

This will produce the following

result – Loop stopped at i = 10

The foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

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

Example

Try out following example to list out the values of an array.

```
<html>
  <body>

    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value )
        { echo "Value is $value <br
        />";
      }
    ?>
  </body>
</html>
```

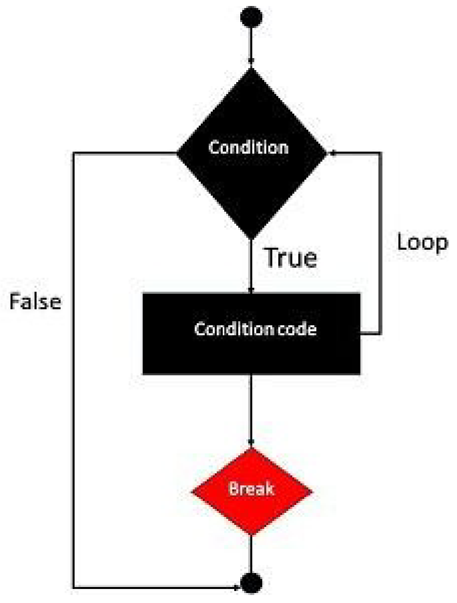
This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

The break statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely.

The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.



Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<html>
  <body>
  
    <?php
      $i = 0;
      
      while( $i < 10) {
        $i++;
```

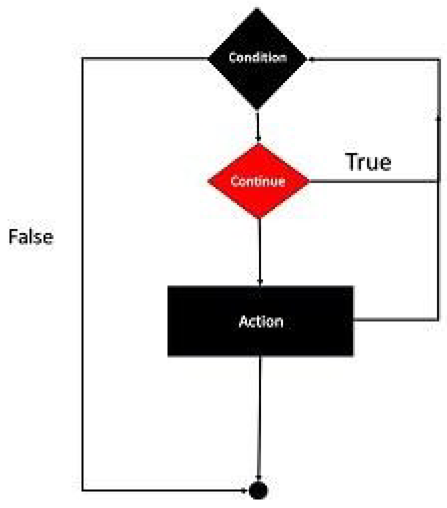
```
         if( $i == 3 )break;
       }
       echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

This will produce the following result − Loop stopped at i = 3

The continue statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.

Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.



Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
  <body>

    <?php
       $array = array( 1, 2, 3, 4, 5);

       foreach( $array as $value
          ) { if( $value == 3
```

```
         echo "Value is $value <br />";
       }
    ?>

  </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 4
Value is 5
```

Array

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

- **Numeric array** – An array with a numeric index. Values are stored and accessed in linear fashion.
- **Associative array** – An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array** – An array containing one or more arrays and values are accessed using multiple indices

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
  <body>

    <?php
      /* First method to create array. */
      $numbers = array( 1, 2, 3, 4, 5);

      foreach( $numbers as $value )
        { echo "Value is $value <br
        />";
      }

      /* Second method to create array. */
      $numbers[0] = "one";
      $numbers[1] = "two";
      $numbers[2] = "three";
      $numbers[3] = "four";
      $numbers[4] = "five";
```

```
         echo "Value is $value <br />";
       }
     ?>

  </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is
three Value
is four Value
is five
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE − Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
  <body>

    <?php
      /* First method to associate create array. */
      $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);

      echo "Salary of mohammad is ". $salaries['mohammad'] .
      "<br />"; echo "Salary of qadir is ". $salaries['qadir']. "<br />";
      echo "Salary of zara is ". $salaries['zara']. "<br />";

      /* Second method to create array. */
      $salaries['mohammad'] = "high";
      $salaries['qadir'] = "medium";
```

```
        $salaries['zara'] = "low";

        echo "Salary of mohammad is ". $salaries['mohammad'] .
        "<br />"; echo "Salary of qadir is ". $salaries['qadir']. "<br />";
        echo "Salary of zara is ". $salaries['zara']. "<br />";
     ?>

  </body>
</html>
```

This will produce the following result −

```
Salary of mohammad is
2000 Salary of qadir is
1000 Salary of zara is 500
Salary of mohammad is
high Salary of qadir is
medium Salary of zara is
low
```

Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects −

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
  <body>

     <?php
        $marks =
           array( "mohammad"
           => array (
              "physics" => 35,
              "maths" => 30,
              "chemistry" => 39
           ),

           "qadir" => array
              ( "physics" =>
              30,
              "maths" => 32,
              "chemistry" => 29
```

```
    "zara" => array
      ( "physics" =>
      31,
      "maths" => 22,
      "chemistry" => 39
    )
  );

  /* Accessing multi-dimensional array
  values */ echo "Marks for mohammad in
  physics : " ; echo
  $marks['mohammad']['physics'] . "<br />";

  echo "Marks for qadir in maths : ";
  echo $marks['qadir']['maths'] . "<br
  />";

  echo "Marks for zara in chemistry : " ;
  echo $marks['zara']['chemistry'] . "<br
  />";
```

This will produce the following result −

```
Marks for mohammad in physics :
35 Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

Strings in PHP

They are sequences of characters, like "PHP supports string

operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = "This is a somewhat longer, singly quoted string";
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
  $variable = "name";
  $literally = 'My $variable will not print!\\n';

  print($literally);
  print "<br />";

  $literally = "My $variable will

  print!\\n"; print($literally);
?>
```

This will produce the following result –

```
My $variable will not
print!\n My name will
print!\n
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP –

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of

their values. The escape-sequence replacements are –

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator –

```
<?php
  $string1="Hello World";
  $string2="1234";

  echo $string1 . " " . $string2;
?>
```

This will produce the following

result – Hello World 1234

If we look at the code above you see that we used the concatenation operator two times. This is
because we had to insert a third string.

Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

Using the strlen() function

The strlen() function is used to find the length of a

string. Let's find the length of our string "Hello world!" –

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

This will produce the following

result – 12

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

Using the strpos() function

The strpos() function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string –

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

This will produce the following

result – 6

As you see the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

GET & POST Methods

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

name1=value1&name2=value2&name3=value3

Spaces are removed and replaced with the *+* character and any other nonalphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

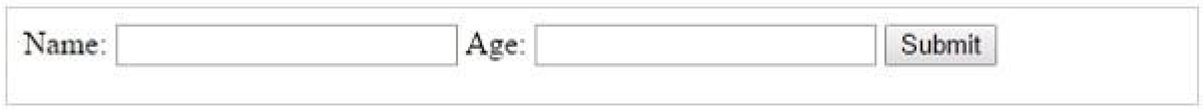
http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_GET["name"] || $_GET["age"] ) {
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";

    exit();
  }
?>
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method =
      "GET"> Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>

  </body>
</html>
```

It will produce the following result −



The POST Method

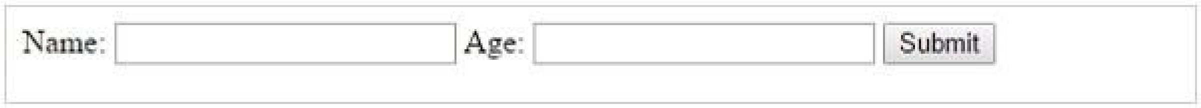
The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

Try out following example by putting the source code in test.php script.

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    if (preg_match("/[^A-Za-z'-]/",$_POST['name'] ))
      { die ("invalid name and name should be
      alpha");
    }
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";

    exit();
  }
?>
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method =
      "POST"> Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>

  </body>
```

It will produce the following result –



The $_REQUEST variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE. We will discuss $_COOKIE variable when we will explain about cookies.

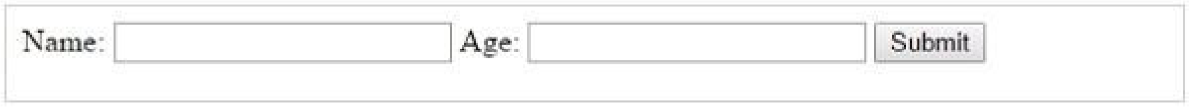
The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

Try out following example by putting the source code in test.php script.

```
<?php
  if(  $_REQUEST["name"]  ||  $_REQUEST["age"]  )
    {  echo  "Welcome  ".  $_REQUEST['name'].  "<br
    />"; echo "You are ". $_REQUEST['age']. " years
    old."; exit();
  }
?>
<html>
  <body>
```

```
     <form action = "<?php $_PHP_SELF ?>" method =
       "POST"> Name: <input type = "text" name = "name" />
       Age: <input type = "text" name = "age" />
       <input type = "submit" />
     </form>

   </body>
</html>
```

Here $_PHP_SELF variable contains the name of self script in which it is being called. It will produce the following result –



Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts which should be clear to you –

- Creating a PHP Function
- Calling a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Please refer to PHP Function Reference for a complete set of useful functions.

Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below –

```
<html>

  <head>
    <title>Writing PHP Function</title>
  </head>

  <body>

    <?php
      /* Defining a PHP Function */
      function writeMessage() {
        echo "You are really a nice person, Have a nice time!";
      }

      /* Calling a PHP Function
      */ writeMessage();
    ?>

  </body>
```

```
</html>
```

This will display following result –

You are really a nice person, Have a nice time!

PHP Functions with Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

```
<html>

  <head>
    <title>Writing PHP Function with Parameters</title>
  </head>

  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
      }

      addFunction(10, 20);
    ?>

  </body>
</html>
```

This will display following result

– Sum of the two numbers is :

30

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

```
<html>

  <head>
```

```
    <title>Passing Argument by Reference</title>
  </head>

  <body>

    <?php
      function addFive($num) {
        $num += 5;
      }

      function addSix(&$num) {
        $num += 6;
      }

      $orignum = 10;
      addFive( $orignum );

      echo "Original Value is $orignum<br

      />"; addSix( $orignum );
      echo "Original Value is $orignum<br />";
    ?>

  </body>
</html>
```

This will display following result −

```
Original Value is
10 Original Value
is 16
```

PHP Functions returning value

A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

You can return more than one value from a function using **return array(1,2,3,4)**.

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that **return** keyword is used to return a

```
<html>

  <head>
    <title>Writing PHP Function which returns value</title>
  </head>

  <body>

```

value from a function.

```
    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        return $sum;
      }
      $return_value = addFunction(10, 20);

      echo "Returned value from the function : $return_value";
    ?>

  </body>
</html>
```

This will display following result –

```
Returned value from the function :
30
```

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it. Following function prints NULL in case use does not pass any value to this function.

```
<html>

  <head>
    <title>Writing PHP Function which returns value</title>
  </head>

  <body>

    <?php
      function printMe($param = NULL)
        { print $param;
      }

      printMe("This is
      test"); printMe();
    ?>

  </body>
</html>
```

This will produce following result – This is test

Dynamic Function Calls

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behaviour.

```
<html>

  <head>
    <title>Dynamic Function Calls</title>
  </head>

  <body>

    <?php
      function sayHello()
        { echo "Hello<br
        />";
      }

      $function_holder = "sayHello";
      $function_holder();
    ?>

  </body>
```

This will display following result

– Hello

Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users –

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

Setting Cookies with PHP

PHP provided **setcookie()** function to set a cookie. This function requires upto six arguments and should be called before <html> tag. For each cookie this function has to be called separately.

setcookie(name, value, expire, path, domain,

security); Here is the detail of all the arguments

–

- **Name** – This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value** – This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** – This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** – This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

- **Domain** – This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** – This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.

```
<?php
  setcookie("name", "John Watkin", time()+3600, "/","", 0);
  setcookie("age", "36", time()+3600, "/", "", 0);
?>
<html>

  <head>
    <title>Setting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Set Cookies"?>
  </body>

</html>
```

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.

```
<html>

  <head>
    <title>Accessing Cookies with PHP</title>
  </head>

  <body>

    <?php
```

```
      echo $_COOKIE["name"]. "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["name"]. "<br />";

      echo $_COOKIE["age"] . "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["age"] . "<br />";
    ?>

  </body>
</html>
```

You can use isset() function to check if a cookie is set or not.

```
<html>

  <head>
    <title>Accessing Cookies with PHP</title>
  </head>

  <body>

    <?php
      if( isset($_COOKIE["name"]))
        echo "Welcome " . $_COOKIE["name"] . "<br />";

      else
        echo "Sorry... Not recognized" . "<br />";
    ?>

  </body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired –

```
<?php
  setcookie( "name", "", time()- 60, "/","", 0);
  setcookie( "age", "", time()- 60, "/","", 0);
?>
```

```
<html>

  <head>
    <title>Deleting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Deleted Cookies" ?>
  </body>

</html>
```

Sessions

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Before using any session variable make sure you have setup this path.

When a session is started following things happen −

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session

A PHP session is easily started by making a call to the **session_start()** function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.

Session variables are stored in associative array called **$_SESSION[]**. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session.

Make use of **isset()** function to check if session variable is already set or not. Put this code in a test.php file and load this file many times to see the result –

```
<?php
  session_start
  ();

  if( isset( $_SESSION['counter'] ) ) {
     $_SESSION['counter'] += 1;
  }else {
     $_SESSION['counter'] = 1;
  }

  $msg = "You have visited this page ". $_SESSION['counter'];
  $msg .= "in this session.";
?>

<html>

  <head>
     <title>Setting up a PHP session</title>
  </head>

  <body>
     <?php echo ( $msg ); ?>
  </body>
```

It will produce the following result –

You have visited this page 1in this session.

Destroying a PHP Session

A PHP session can be destroyed by **session_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use **unset()** function to unset a session variable.

Here is the example to unset a single variable –

```
<?php
  unset($_SESSION['counter']
  );
```

Here is the call which will destroy all the session variables –

```
<?php
  session_destroy
  ();
```

Turning on Auto Session

You don't need to call start_session() function to start a session when a user visits your site if you can set **session.auto_start** variable to 1 in **php.ini** file.

Sessions without cookies

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant SID which is defined if the session started. If the client did not send an appropriate session cookie, it has the form session_name=session_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

```
<?php
  session_start
  ();

  if (isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  }else {
    $_SESSION['counter']++;
  }

  $msg = "You have visited this page ". $_SESSION['counter'];
  $msg .= "in this session.";

  echo ( $msg );
?>

<p>
  To continue click following link <br />

  <a href = "nextpage.php?<?php echo htmlspecialchars(SID); ?>">
```

It will produce the following result −

```
You have visited this page 1in this
session. To continue click following
link
```

Event management in PHP

Event Management System is a web application developed with PHP and MYSQL to make the events management effective. Event Management is necessary because the more popular a brand is, the lesser reluctant individuals will be for experimenting with new items propelled by that brand. Occasion the executive's abilities are, in this way, fundamental for the organization to get the required introduction and assemble a positive picture of the general organization just as any brand specifically. They do not just fill in as an opportunity for an entrenched organization to recover its significance by pulling in an expanding number of imminent clients yet in addition empower a sprouting organization to develop a feeling of enthusiasm for the everyday citizens about the items and administrations they offer.

Using the software, we can perform a function related to what events management company performs that includes creating events, choose resources and location, login and manage/edit/update/delete events and location list, make new announcements of their company and so on. Users can send any inquiries using the contact form and the message is stored on a database and can be seen there only.

Features of Event Management Software

- This system is based on the event management records and to add and edit.
- The GUI is user-friendly and responsive.
- Less Time Consumed.
- The system makes easy to use to manage the event.
- Create new Events along with event name, event start date, event end date, event location, event total cost.
- See all list of events created with their proper details.
- Add new locations for events that includes Name, address, location manager name, number, numbers of people capacity and number of facilities included in the location.
- Edit/update and delete locations.
- Announce events on the website pages
- Login protection system
- Send Contact message

Introduction to content management systems based on PHP

Content development has become a sensational means of sharing information over the internet. Even the non-technical users got the ability to publish content easily and quickly on the World Wide Web. It is all possible because of the easy-use of content management tools available and is widely used by firms, news organizations, educational institutions, and other businesses. In this chapter, you will learn about the concept of CMS and why it is widely accepted in the market.

What is Content Management System?

Content Management System (CMS) can be defined as a tool or software program containing a set of interrelated programs used for creating and managing different digital or online content. Some famous examples of CMS software are Joomla, Drupal, WordPress, TYPO3, etc. The typical use of CMSes are in two areas:

1. Enterprise Content Management (ECM) and
2. Web Content Management (WCM)

In the majority of the cases, it can support many users, letting them work in the association. For example, WordPress makes it possible to create many administrative users, where each one has different privileges hence making the work progress in parallel. Content management systems also comprise of text as well as the layout and design features like the facility to upload multimedia content like photos, videos, audio, maps, or even any source code.

Components of Content Management System

A content management system is composed of two major components. These are:

- A content management application (CMA) is a graphical user interface that allows its users to create, delete, modify, and publish content even without the knowledge of HTML or other programming languages that are necessary to create web pages.
- A content delivery application (CDA) is responsible for the back-end services. It manages as well as delivers content after framed in the CMA.

Features of Content Management System

- User Management: This permits the management of user information like the roles of different users allotted to work simultaneously, such as creating or deleting the user, change the username, password, and other related information.
- Theme System: This allows us to modify the site view as well as functionality using stylesheets, images, and templates.
- Extending Plugins: Different plugins are offered, which gives custom functionalities and features to create the CMS site.
- Search Engine Optimization: It is embedded with a lot of search engine optimization (SEO) tools making content SEO more straightforward.
- Media Management: is used for managing the media files and folder, with uploading media contents easy and effortless.
- Multilingual: Translation of the language, as preferred by the user, is possible through CMS.

Advantages of Content Management System

- Most of the CMS is open source and is available for free.
- Easy and quick uploading of media files can be done.
- Several SEO tools make on-site SEO simpler.
- Easy customization is possible as per the need of the user.
- It can modify CSS files as per the design needed by the user.
- Many templates and plugins are available for free. Customization of plugins is also possible.
- Content editing is also more comfortable as it uses the WYSIWYG editor.

Disadvantages of Content Management System

- CMS software needs a time-to-time update, and hence the user needs to look out for an updated version.

- The use of different plugins can make your website heavy and challenging to run.
- CMS hosting is quite expansive.
- PHP knowledge is required to modify or change the WordPress website.

Chapter 3

PHP and MySQL

Database

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A Relational DataBase Management System (RDBMS) is a software that –

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

- **Database** – A database is a collection of tables, with related data.
- **Table** – A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column** – One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row** – A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy** – Storing data twice, redundantly to make the system faster.

- **Primary Key** – A primary key is unique. A key value cannot occur twice in one table. With a key, you can only find one row.
- **Foreign Key** – A foreign key is the linking pin between two tables.
- **Compound Key** – A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index** – An index in a database resembles an index at the back of a book.
- **Referential Integrity** – Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons –

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Connecting to MySQL

In order to store or access the data inside a MySQL database, you first need to connect to the MySQL database server. PHP offers two different ways to connect to MySQL server: MySQLi (Improved MySQL) and PDO (PHP Data Objects) extensions.

While the PDO extension is more portable and supports more than twelve different databases, MySQLi extension as the name suggests supports MySQL database only. MySQLi extension however provides an easier way to connect to, and execute queries on, a MySQL database server. Both PDO and MySQLi offer an object-oriented API, but MySQLi also offers a procedural API which is relatively easy for beginners to understand.

In PHP you can easily do this using the mysqli_connect() function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:

Syntax: MySQLi, Procedural way

```
$link = mysqli_connect("hostname", "username", "password", "database");
```

Syntax: MySQLi, Object Oriented way

```
$mysqli = new mysqli("hostname", "username", "password", "database");
```

The hostname parameter in the above syntax specify the host name (e.g. localhost), or IP address of the MySQL server, whereas the username and password parameters specifies the credentials to access MySQL server, and the database parameter, if provided will specify the default MySQL database to be used when performing queries.

```
<?php
$link = mysqli_connect("localhost", "root", "");
// Check
connection
if($link === false)
```

```
{
```

```
die("ERROR: Could not connect. " . mysqli_connect_error());
}
// Print host information
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);
?>
```

Closing the MySQL Database Server Connection

The connection to the MySQL database server will be closed automatically as soon as the execution of the script ends. However, if you want to close it earlier you can do this by simply calling the PHP mysqli_close() function.

```
<?php
$link = mysqli_connect("localhost", "root", "");
// Check
connection
if($link === false)
{
die("ERROR: Could not connect. " . mysqli_connect_error());
}
// Print host information
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);
//
Close connection mysqli_close($link);
?>
```

Create a MySQL Database

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
$conn = mysqli_connect($servername, $username, $password);
// Check
connection if
(!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Create a MySQL Table

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

```
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 DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP

)
```

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

```
<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check

connection if

(!$conn) {

  die("Connection failed: " . mysqli_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 DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP

)";

if (mysqli_query($conn, $sql)) {

  echo "Table MyGuests created successfully";

}

else

{

  echo "Error creating table: " . mysqli_error($conn);

}

mysqli_close($conn);

?>
```

Insert Data Into MySQL

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,...)

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

```
<?php

$servername = "localhost";

$username = "username";
```

```
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check
connection if
(!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

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

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Delete Data From a MySQL

The DELETE statement is used to delete records from a

table: DELETE FROM table_name

WHERE some_column =

some_value Let's look at the

"MyGuests" table:

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check
connection if
(!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

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

if (mysqli_query($conn, $sql))
  { echo "Record deleted
  successfully";
} else {
  echo "Error deleting record: " . mysqli_error($conn);
}
```

```
mysqli_close($conn);
```

?>

Select Data From a MySQL

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

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

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check

connection if

(!$conn) {

  die("Connection failed: " . mysqli_connect_error());

}


$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = mysqli_query($conn, $sql);
```

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

mysqli_close($conn);
?>
```