



RC:1337346

Gr8tson Technologies Limited

WEB DESIGN

think BIG, start SMALL



Knowledge is Power

OVERVIEW

About the Tutorial

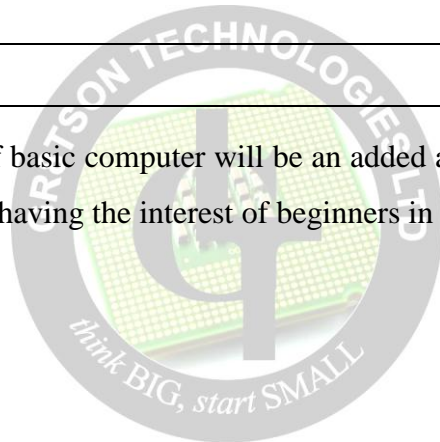
This Web Design Tutorial allows web developers to learn the elements of web development using Hypertext Markup Language (HTML), Cascading Style Sheet (CSS) and elements of JavaScript in building websites. This tutorial will help you understand the basics of web development and how to put it into practice.

Audience

This tutorial has been designed to meet the requirements of all those readers who are keen to learn the basics of Web Development.

Prerequisites

Although a prior knowledge of basic computer will be an added advantage in understanding web design, this material is written having the interest of beginners in web development in mind.



Contents

| | |
|---------------------------------------|----|
| OVERVIEW..... | 2 |
| WELCOME TO THE INTERNET | 9 |
| WORKING WITH WEB BROWSERS | 9 |
| UNDERSTANDING WEB PAGES | 11 |
| CREATING WEB PAGES..... | 13 |
| SOME IMPORTANT WEB TOOLS | 13 |
| CHAPTER ONE | 15 |
| Uses of HTML | 15 |
| CHAPTER TWO | 21 |
| HTML – BASIC TAGS | 21 |
| Heading Tags..... | 21 |
| Paragraph Tag | 22 |
| Line Break Tag..... | 22 |
| Centering Content..... | 23 |
| Horizontal Lines | 23 |
| Preserve Formatting | 24 |
| Nonbreaking Spaces..... | 24 |
| CHAPTER THREE | 25 |
| HTML – ELEMENTS..... | 25 |
| HTML Tag vs. Element..... | 25 |
| Nested HTML Elements..... | 25 |
| CHAPTER FOUR | 27 |
| HTML – ATTRIBUTES | 27 |
| Core Attributes..... | 27 |
| Internationalization Attributes | 28 |
| Generic Attributes..... | 30 |
| CHAPTER FIVE | 31 |
| HTML – FORMATTING..... | 31 |
| Bold Text | 31 |
| Italic Text..... | 31 |
| Underlined Text | 31 |
| Strike Text | 32 |

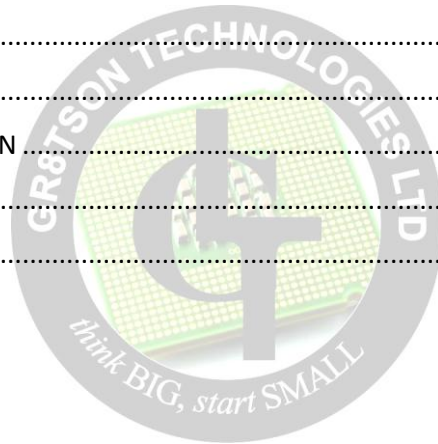
| | |
|--|----|
| Monospaced Font | 32 |
| Superscript Text | 33 |
| Subscript Text..... | 33 |
| Inserted / Deleted Text | 34 |
| Larger Text | 35 |
| Smaller Text | 36 |
| Grouping Content | 36 |
| CHAPTER SIX..... | 39 |
| HTML – PHRASE TAGS | 39 |
| Emphasized Text | 39 |
| Marked Text | 39 |
| Strong Text..... | 39 |
| Text Abbreviation..... | 40 |
| Acronym Element..... | 40 |
| Text Direction..... | 41 |
| Special Terms | 41 |
| Quoting Text | 42 |
| Short Quotations..... | 42 |
| Text Citations | 43 |
| Computer Code..... | 43 |
| Keyboard Text | 44 |
| Program Output | 44 |
| Address Text..... | 44 |
| CHAPTER SEVEN | 46 |
| HTML – META TAGS | 46 |
| Adding Meta Tags to Your Documents | 46 |
| Specifying Keywords | 47 |
| Document Description | 47 |
| Document Revision Date | 48 |
| Document Refreshing | 48 |
| Page Redirection | 48 |
| Setting Cookies..... | 49 |
| Setting Author Name | 50 |

| | |
|--------------------------------------|----|
| Specify Character Set | 50 |
| CHAPTER EIGHT | 51 |
| HTML – COMMENTS | 51 |
| Valid vs Invalid Comments | 51 |
| Multiline Comments | 52 |
| Conditional Comments | 52 |
| Using Comment Tag | 53 |
| Commenting Script Code | 53 |
| Commenting Style Sheets | 54 |
| CHAPTER NINE | 56 |
| HTML – IMAGES | 56 |
| Insert Image | 56 |
| Set Image Location | 56 |
| Set Image Width/Height | 57 |
| Set Image Border | 57 |
| Set Image Alignment | 58 |
| Free Web Graphics | 58 |
| CHAPTER TEN | 59 |
| Table Caption | 64 |
| Table Header, Body, and Footer | 65 |
| Nested Tables | 66 |
| CHAPTER ELEVEN | 68 |
| HTML – LISTS | 68 |
| HTML Unordered Lists | 68 |
| The type Attribute | 68 |
| HTML Ordered Lists | 69 |
| The type Attribute | 69 |
| The start Attribute | 70 |
| HTML Definition Lists | 71 |
| CHAPTER TWELVE | 73 |
| HTML LINKS | 73 |
| Html – Text Links | 73 |
| Linking Documents | 73 |

| | |
|------------------------------------|-----|
| The target Attribute | 74 |
| Linking to a Page Section | 74 |
| Setting Link Colors..... | 75 |
| Download Links | 76 |
| Html – Image Links | 76 |
| Html – Email Links | 77 |
| HTML Email Tag..... | 77 |
| Default Settings..... | 78 |
| CHAPTER THIRTEEN..... | 79 |
| HTML – FRAMES..... | 79 |
| Disadvantages of Frames | 79 |
| Creating Frames | 79 |
| CHAPTER FOURTEEN | 89 |
| HTML – IFRAMES..... | 89 |
| CHAPTER FIFTEEN | 92 |
| HTML – BLOCKS..... | 92 |
| Block Elements | 92 |
| Inline Elements..... | 92 |
| Grouping HTML Elements | 92 |
| HTML – Backgrounds | 94 |
| Html Background with Colors | 95 |
| CHAPTER SIXTEEN | 99 |
| HTML – COLORS | 99 |
| HTML Color Coding Methods..... | 99 |
| HTML Colors - Color Names | 99 |
| HTML Colors - Hex Codes..... | 100 |
| HTML Colors - RGB Values | 101 |
| Browser Safe Colors | 102 |
| CHAPTER SEVENTEEN..... | 103 |
| HTML – FONTS | 103 |
| Setting Font Face | 105 |
| Specify alternate font faces | 106 |
| Setting Font Color | 106 |

| | |
|--|-----|
| CHAPTER EIGHTEEN | 108 |
| HTML – FORMS | 108 |
| Form Attributes..... | 108 |
| HTML Form Controls | 109 |
| Text Input Controls | 109 |
| Single-line text input controls | 109 |
| Password Input controls | 110 |
| Multiple-Line Text Input Controls | 111 |
| Checkbox Control..... | 112 |
| Radio Button Control | 113 |
| Select Box Control..... | 114 |
| File Upload Box | 115 |
| Button Controls..... | 116 |
| Hidden Form Controls | 117 |
| CHAPTER NINETEEN | 119 |
| HTML – EMBED MULTIMEDIA..... | 119 |
| CHAPTER TWENTY..... | 123 |
| HTML – MARQUEES | 123 |
| HTML – HEADER..... | 125 |
| CHAPTER TWENTY TWO..... | 129 |
| HTML – STYLE SHEET..... | 129 |
| UNDERSTANDING CASCADING STYLE SHEETS | 129 |
| Ease of modification | 129 |
| Advanced design options..... | 129 |
| Media targeting..... | 130 |
| Key CSS Concepts | 131 |
| The Cascading Principle | 131 |
| The Inheritance Principle | 131 |
| The Specificity Principle | 132 |
| Working with CSS Placement..... | 133 |
| External style sheets | 133 |
| Embedded styles..... | 135 |
| Inline styles | 135 |

| | |
|--------------------------------------|-----|
| Tags | 136 |
| IDs | 136 |
| Classes..... | 136 |
| compound selectors..... | 137 |
| External Style Sheet | 139 |
| Internal Style Sheet..... | 140 |
| Inline Style Sheet..... | 141 |
| CHAPTER TWENTY THREE | 142 |
| HTML JAVASCRIPT..... | 142 |
| External JavaScript | 142 |
| Internal Script..... | 143 |
| Event Handlers | 143 |
| CHAPTER TWENTY FOUR..... | 145 |
| HTML – LAYOUTS | 145 |
| HTML Layout - Using Tables..... | 146 |
| HTML Layouts - Using DIV, SPAN | 148 |
| CHAPTER TWENTY FIVE..... | 157 |
| HTML EVENTS REFERENCE | 157 |



WELCOME TO THE INTERNET

The internet is a world-wide, publicly accessible series of interconnected networks that transmit data packet switching, using the standard internet protocol. The internet is the underlying telecommunication infrastructure that makes the worldwide web possible. The function of the internet is to help users communicate (move data) from one host machine to another.

There are hundreds of millions of computers on the internet but they do not all do the same thing. Some are simply used to store information and pass it on when required, these are known as **SERVERS** while other computers that get information from a server computer are called **CLIENTS**.

The web environment is the collection of technologies and software that enable us to use the world wide web, these include; web servers, database servers, and other programming languages as well as browsers which gives a graphic interpretation of the web.

WORKING WITH WEB BROWSERS

A web browser is a software application used for retrieving, presenting and traversing information resources on the W3, the most popular of which include Google chrome, Mozilla Firefox, safari and internet explorer. The information resource is identified by a URL (Uniform resource Locator) that may be a webpage, a multimedia item (video/ image).

The first web browser was invented in 1990 by Sir Tim Berners-Lee. Berners-Lee is the director of the World Wide Web Consortium (W3C), which oversees the Web's continued development, and is also the founder of the World Wide Web Foundation. His browser was called World Wide Web and later renamed Nexus. World Wide Web for NeXT, released in 1991, was the first web browser.

The first commonly available web browser with a graphical user interface was Erwise. The development of Erwise was initiated by Robert Cailliau.

In 1993, browser software was further innovated by Marc Andreessen with the release of Mosaic, "the world's first popular browser", which made the World Wide Web system easy to use and more accessible to the average person. Andreessen's browser sparked the internet boom of the 1990s. The introduction of Mosaic in 1993 – one of the first graphical web browsers – led to an explosion in web use. Andreessen, the leader of the Mosaic team at National Center for

Supercomputing Applications (NCSA), soon started his own company, named Netscape, and released the Mosaic-influenced Netscape Navigator in 1994, which quickly became the world's most popular browser, accounting for 90% of all web use at its peak.

Microsoft responded with its Internet Explorer in 1995, also heavily influenced by Mosaic, initiating the industry's first browser war. Bundled with Windows, Internet Explorer gained dominance in the web browser market; Internet Explorer usage share peaked at over 95% by 2002.

Opera debuted in 1996; it has never achieved widespread use, having less than 2% browser usage share as of February 2012 according to Net Applications. Its Opera-mini version has an additive share, in April 2011 amounting to 1.1% of overall browser use, but focused on the fast-growing mobile phone web browser market, being preinstalled on over 40 million phones. It is also available on several other embedded systems, including Nintendo's Wii video game console.

In 1998, Netscape launched what was to become the Mozilla Foundation in an attempt to produce a competitive browser using the open source software model. That browser would eventually evolve into Firefox, which developed a respectable following while still in the beta stage of development; shortly after the release of Firefox 1.0 in late 2004, Firefox (all versions) accounted for 7% of browser use. As of August 2011, Firefox has a 28% usage share.

Apple's Safari had its first beta release in January 2003; as of April 2011, it had a dominant share of Apple-based web browsing, accounting for just over 7% of the entire browser market.

The most recent major entrant to the browser market is Chrome, first released in September 2008. Chrome's take-up has increased significantly year by year, by doubling its usage share from 8% to 16% by August 2011. This increase seems largely to be at the expense of Internet Explorer, whose share has tended to decrease from month to month. In December 2011, Chrome overtook Internet Explorer 8 as the most widely used web browser but still had lower usage than all versions of Internet Explorer combined. Chrome's user-base continued to grow and in May 2012, Chrome's usage passed the usage of all versions of Internet Explorer combined. By April 2014, Chrome's usage had hit 45%.

Internet Explorer was deprecated in Windows 10, with Microsoft Edge replacing it as the default web browser.

Most major web browsers have these user interface elements in common:

- Back and forward buttons to go back to the previous resource and forward respectively.
- A refresh or reload button to reload the current resource.
- A stop button to cancel loading the resource. In some browsers, the stop button is merged with the reload button.
- A home button to return to the user's home page.
- An address bar to input the Uniform Resource Identifier (URI) of the desired resource and display it.
- A search bar to input terms into a web search engine. In some browsers, the search bar is merged with the address bar.
- A status bar to display progress in loading the resource and also the URI of links when the cursor hovers over them, and page zooming capability.
- The viewport, the visible area of the webpage within the browser window.
- The ability to view the HTML source for a page.
- Major browsers also possess incremental find features to search within a web page.

UNDERSTANDING WEB PAGES

A web page, or webpage, is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device. The web page is what displays, but the term also refers to a computer file, usually written in HTML or comparable markup language. Web browsers coordinate the various web resource elements for the written web page, such as style sheets, scripts, and images, to present the web page.

Typical web pages provide hypertext that includes a navigation bar or a sidebar menu to other web pages via hyperlinks, often referred to as links.

On a network, a web browser can retrieve a web page from a remote web server. On a higher level, the web server may restrict access to only a private network such as a corporate intranet or it provides access to the World Wide Web. On a lower level, the web browser uses the Hypertext Transfer Protocol (HTTP) to make such requests.

A static web page is delivered exactly as stored, as web content in the web server's file system, while a dynamic web page is generated by a web application that is driven by server-side software or client-side scripting. Dynamic website pages help the browser (the client) to enhance the web page through user input to the server.

A web page, as an information set, can contain numerous types of information, which is able to be seen, heard or interacted with by the end user:

- Textual information: with diverse render variations.
- Non-textual information:
- Static images may be raster graphics, typically GIF, JPEG or PNG; or vector formats such as SVG or Flash.
- Animated images typically Animated GIF and SVG, but also may be Flash, Shockwave, or Java applet.
- Audio, typically MP3, Ogg or various proprietary formats.
- Video, WMV (Windows), RM (RealMedia), FLV (Flash Video), MPG, MOV (QuickTime)
- Interactive information: see interactive media.
- For "on page" interaction:
- Interactive text: see DHTML.
- Interactive illustrations: ranging from "click to play" images to games, typically using script orchestration, Flash, Java applets, SVG, or Shockwave.
- Buttons: forms providing an alternative interface, typically for use with script orchestration and DHTML.
- For "between pages" interaction:
- Hyperlinks: standard "change page" reactivity.
- Forms: providing more interaction with the server and server-side databases.
- Internal (hidden) information:
- Comments
- Linked Files through Hyperlink (Like DOC, XLS, PDF, etc.)
- Metadata with semantic meta-information, Charset information, Document Type Definition (DTD), etc.
- Diagrammatic and style information: information about rendered items (like image size attributes) and visual specifications, as Cascading Style Sheets (CSS).
- Scripts, usually JavaScript, complement interactivity, and functionality.

Note: on server-side the web page may also have "Processing Instruction Information Items".

The web page can also contain dynamically adapted information elements, dependent upon the rendering browser or end-user location (through the use of IP address tracking and/or "cookie"

information). From a more general/wide point of view, some information (grouped) elements, like a navigation bar, are uniform for all website pages, like a standard. This kind of "website standard information" is supplied by technologies like web template systems.

CREATING WEB PAGES

To create a web page, a text editor or a specialized HTML editor is needed. In order to upload the created web page to a web server, traditionally an FTP client is needed.

The design of a web page is highly personal. A design can be made according to one's own preference, or a premade web template can be used. Web templates let web page designers edit the content of a web page without having to worry about the overall aesthetics. Many people turn to all-in-one sites for web domain purchases, web hosting service and templates to build customized websites. Web publishing tools such as Tripod and WordPress offer free page creation and hosting up to a certain size limit. Other ways of making a web page is to download specialized software, like a Wiki, CMS, or forum. These options allow for quick and easy creation of a web page which is typically dynamic.

In order to graphically display a web page, a web browser is needed. This is a type of software that can retrieve web pages from the Internet. Most current web browsers include the ability to view the source code. Viewing a web page in a text editor will also display the source code.

SOME IMPORTANT WEB TOOLS

Several software applications are vital to creating and managing websites. Some important ones are described below.

1. **Text Editor:** A text editor is a software used to create web pages, an example of which is notepad or wordpad. Sometimes, one may use specialized HTML editors such as Netbeans, Dreamweaver, Notepad++, etc.
2. **SQL Server:** This is a software that enables a local machine to be morphed (albeit temporarily) into a web server to enable users view source codes. XAMPP and WAMP are very popular examples.

FTP Client: FTP is the acronym for file transfer protocol and it is used to upload web pages to an actual web server. A very good example is FileZilla.



CHAPTER ONE

HTML OVERVIEW

What is HTML

HTML stands for **H**ypertext **M**arkup **L**anguage, which simply means it is a language for describing web-pages using ordinary text. It is the most widely used language to write Web Pages.

HTML is *not* a complex programming language.

Hypertext - Hypertext is text which contains links to other texts.

Markup/Markup languages

Markup languages such as HTML are designed for the **processing, definition** and **presentation** of text. A Markup language specifies the code for formatting both the layout and style, within a text file. HTML is an example of a widely known and used **markup** language.

Markup refers to the sequence of characters or other symbols that you insert at certain places in a text to indicate how the file should look like when it is printed or displayed.

Language - the method of human communication, either spoken or written, consisting of the use of words in a structured and conventional way.

Uses of HTML

HTML is the language for describing the structure of Web pages. HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips.

Web pages/Website

Every **web page** is actually an HTML file. Each HTML file is just a plain-text file, but with a **.html** file extension instead of .txt, and is made up of many HTML tags as well as the content for a web page.

A **web** site will often contain many html files that link (connect) to each other. You can edit HTML files with your favorite editor.


HTML Editors

These are text editors used to create web pages. Examples include Notepad, Netbeans, Dreamweaver, Notepad++, and Sublime 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>
```



Saving An HTML File

To save an HTML file and name it as test, we need to save it as **test.htm** or **test.html**. An HTML file can be saved on the desktop, document or inside the website folder on the server such as htdocs when using XAMPP server.

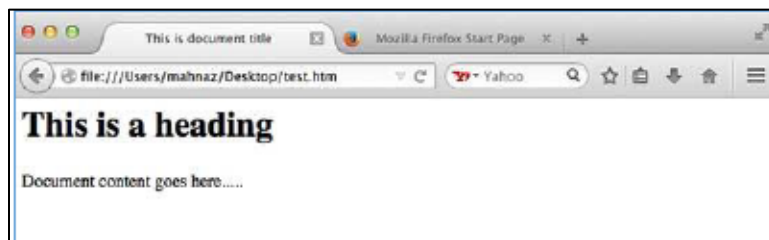
Note:

When saving an html page that you intend to use other programming language code in them, you have to save the web page with the file extension of that programming language instead of the html extension. For example, if the test web page is also to contain php programming language codes inside, it will rather be saved as test.php instead of test.htm otherwise the php codes in the file will not be recognized as php codes. Also, a php file is recommended to be saved inside the website folder inside the server that is used to process php web pages.

Viewing HTML or PHP Web Pages

Web pages can be viewed using web browsers such as Internet Explorer, Mozilla Firefox or Google Chrome etc.

For a web page that contains only html text and is saved outside the server like on the desktop or documents location, when you double click on it, it opens in the browser. For example, the test.html page above displays as shown below when viewed in the web browser.



Note:

For files that need to be executed on the server such as XAMPP server, the server has to be setup and running and then the address of the web page on the server can be entered on the address bar of the browser. For example, assuming the **test page** were a php file, then the following address will be entered in the browser's address bar:

Localhost/websitefolder/test.php

HTML Tags

HTML is a Markup Language and makes use of various symbols called tags to format the content of a web page. Tags names are usually enclosed within angle braces in this form: **<TagName>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing **</html>** tag and **<body>** tag has its closing **</body>** tag etc.

Tags specify the formatting within the markup language. Tags simply tell the browser how to structure the HTML file to display. They define how your web browser must format and display the content of a web page.

Most tags must have two parts, an opening and a closing part. For example, **<html>** is the opening tag and **</html>** is the closing tag. Note that the closing tag has the same text as the opening tag, but has an additional forward-slash (/) character before the tag name. There are

some tags that are an exception to this rule, where a corresponding closing tag is not required. Such tags are called empty tags. Examples include ``, `
`, `<hr>`, etc.

Essential HTML Tags

There are four sets of HTML tags that are needed to form the basic structure for every HTML file:

- `<html></html>`
- `<head></head>`
- `<title></title>`
- `<body></body>`

Definition - `<html> </html>`

This basically defines the document as a web page. It also identifies the beginning and end of the HTML document. All other tags must fall between the html tags.

Header - `<head> </head>`

The header contains information about the document that will not appear on the actual page, such as the title of the document, the author.

Title - `<title> </title>`

The title tag defines the title that will appear in the title bar of your web browser. The title must appear between the head tags.

Body - `<body> </body>`

The body tags contain all the information and other visible content on the page. All your images, links and plain text must go between the `<body>` and `</body>` tags.

These four tags are special. There must only be one set of each and they must be in the correct order as shown in the basic HTML structure.

| | |
|--|--|
| Above example of HTML document uses the following tags: Tag | Description |
| <code><!DOCTYPE...></code> | This tag defines the document type and HTML version. |

| | |
|---------|---|
| <html> | This tag encloses the complete HTML document and mainly comprises the document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags. |
| <head> | This tag represents the document's header which can keep other HTML tags like <title>, <link> etc. |
| <title> | The <title> tag is used inside the <head> tag to mention the document title. |
| <body> | This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc. |
| <h1> | This tag represents the heading. |
| <p> | This tag represents a paragraph. |

To learn HTML, you will need to study various tags and understand how they behave, while formatting a textual document. Learning HTML is simple as users have to learn the usage of different tags in order to format the text or images to make a beautiful web page.

The World Wide Web Consortium (W3C) recommends use of lowercase tags starting from HTML 4

HTML Document Structure

A typical HTML document will have the following structure:

Document declaration tag

```
<html>
```

```
    <head>
```

Document header related tags

```
    </head>
```

```
    <body>
```

Document body related tags

`</body>`

`</html>`

The `<!DOCTYPE>` Declaration

This is used to tell the web browser what version of HTML been used in the document. The current version is HTML is 5. Its declaration is as follows:

`<!DOCTYPE html>`



CHAPTER TWO

HTML – BASIC TAGS

Heading Tags

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>`.

`<h1>` produces the largest heading while `<h6>` produces the smallest heading.

While displaying any heading, the browser adds one line before and one line after that heading.

```
<!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

Assignment to students: Read basic tags, phrase tags, comments, images, and tables.

Paragraph Tag

The **<p>** tag is used to specify a paragraph. Each paragraph of text should go in between an opening **<p>** and a closing **</p>** tag.

Example

```
<p>Here is a paragraph of text.</p>
```

Line Break Tag

Whenever you use the **
** element, anything following it starts from the next line.

This tag is an example of an **empty** element, where you do not need corresponding closing tags, as there is nothing to go in between them as the content.

The **
** 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 **
** it is not valid in XHTML.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Line Break Example</title>
  </head>
  <body>
    <p>Hello<br />
      You delivered your assignment on time.<br />
      Thanks<br />Michael
    </p>
  </body>
</html>
```

Result:

Hello

You delivered your assignment on time.

Thanks

Michael

Centering Content

To align content to the center of the page of any table cell, use **<center>** tag. In the code below, the **<center>** tag is used to align the paragraph to the center.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Line Break Example</title>
  </head>
  <body>
    <center>
      <p>This text is in the center.</p>
    </center>
  </body>
</html>
```

Result:

This text is in the center

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. In the code below, a horizontal line will be drawn in between the two paragraphs.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Horizontal Line Example</title>
  </head>
  <body>
    <p>This is paragraph one and should be on top of the line</p>
    <hr />
    <p>This is paragraph two and should be below the line</p>
  </body>
</html>
```

Result:

This is paragraph one and should be on top

This is paragraph two and should be at bottom

Preserve Formatting

To make a text remain the way it is formatted in the HTML document when displaying on the browser, we use the preformatted tags.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title> Preserve Formatting Example</title>
  </head>
  <body>
    <pre>This is paragraph one and should be on top</pre>
  </body>
</html>
```

Nonbreaking Spaces

If you don't want a browser to break a group of words into separate lines, you can use ** ** between the words. To always display the words GR8TSON TECHNOLOGIES LIMITED without breaking it into separate lines, we do the following:

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title> Nonbreaking Spaces Example </title>
  </head>
  <body>
    <p>An The best ICT Company is
    "GR8TSON&nbsp;TECHNOLOGIES&nbsp;LIMITED."</p>
  </body>
</html>
```


CHAPTER THREE

HTML – ELEMENTS

An HTML element is made up of an opening tag, a closing tag and a content between the opening and the closing tag. Some elements such as `<hr />`, `
`, `` don't have content and as such do not need a closing tag.

| Start Tag | Content | End Tag |
|---------------------------|----------------------------|---------------------------|
| <code><p></code> | This is paragraph content. | <code></p></code> |
| <code><h1></code> | This is heading content. | <code></h1></code> |
| <code><div></code> | This is division content. | <code></div></code> |
| <code> </code> | | |
| <code><hr /></code> | | |

HTML Tag vs. Element

A tag is made up of two angular braces with the name of the tag in between them while an element is made up two tags and a content. Only few elements do not have content and as such do not need a closing tag. Elements without a content are called empty elements.

Nested HTML Elements

When an element is included in another element, this process is known as nesting. The element inside the other is known as the nested element. In the example below, the underline element, `<u>...</u>` is included inside the paragraph tag. The underline element is therefore nested.

Note: the closing tag for the nested element, `<u>...</u>` must end before the closing tag of the outside element, `<p></p>`

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title> Nested Elements Example</title>
  </head>
  <body>
    <p>This is <u>underlined</u> paragraph</p>
```

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



CHAPTER FOUR

HTML – ATTRIBUTES

Extra bits of information which can be added to the opening tag of an HTML element to add meaning to it is referred to as an attribute. All attributes are made up of two parts: a **name** and a **value** with an equal sign (=) in between them. The value is usually enclosed in double quotes. In the example below, we use the align attribute to align text to the center of the paragraph.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title> Align Attribute Example</title>
  </head>
  <body>
    <p align="center">This paragraph will be aligned to the center of the page.</p>
  </body>
</html>
```

Core Attributes

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

- i. Id – this is used to uniquely identify an element within an HTML page.

Two primary reasons for using an Id attribute:

- a. It makes it possible to identify an element and its content
- b. Id attribute can be used to distinguish between two elements with the same name within the same web page (or style sheet).

Example

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

- ii. Title – the title attribute gives a suggested title to the element. The way to write the title attribute is the same as that of the id attribute.

The behavior of this attribute will depend upon the element that carries it. 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>
```

- iii. Class – A class attribute is used to associate an element to a style sheet, and specifies the class of element. The value of the attribute may also be a space-separated list of class names. For example:

class="className1 className2 className3"

- iv. Style – the style attribute allows you to specify cascading style sheet rules within the element.

Example

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

Result:

This will produce the following result:

Some text...

Internationalization Attributes

These are attributes which work very well irrespective of the culture, or region of users involved.

They include:

i. **dir** attribute

The **dir** attribute allows you to indicate to the browser about the direction in which the text should flow. The dir attribute can take one of two values:

- a. ltr – left to right
- b. rtl – right to left (for languages such as Hebrew and Arabic that are read from right to left)

Note: when the dir attribute is included in the HTML tag, it determines how text will be presented within the entire document. When used within another tag, it controls the text's direction for just the content of that tag.

ii. **lang** attribute

The **lang** attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML.

This attribute has been replaced by the **xml:lang** attribute in new XHTML documents.

The values of the *lang* attribute are ISO-639 standard two-character language codes

Example

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>English Language Page</title>
  </head>
  <body>
    This page is using English Language
  </body>
</html>
```

iii. **xml:lang** attribute

The *xml:lang* attribute is the XHTML replacement for the *lang* attribute. The value of the *xml:lang* attribute should be an ISO-639 country code

Generic Attributes

Here's a table of some other attributes that are readily usable with many of the HTML tags.

| Attribute | Options | Function |
|------------|----------------------------------|--|
| align | right, left, center | Horizontally aligns tags |
| valign | top, middle, bottom | Vertically aligns tags within an HTML element. |
| bgcolor | numeric, hexadecimal, RGB values | Places a background color behind an element |
| background | URL | Places a background image behind an element |
| id | User Defined | Names an element for use with Cascading Style Sheets. |
| class | User Defined | Classifies an element for use with Cascading Style Sheets. |
| width | Numeric Value | Specifies the width of tables, images, or table cells. |
| height | Numeric Value | Specifies the height of tables, images, or table cells. |
| title | User Defined | "Pop-up" title of the elements. |

CHAPTER FIVE

HTML – FORMATTING

Just as text in Word processing application packages can be made to be bold, italic, underlined etc. HTML also has elements that can be used to format text on the web page.

Bold Text

Anything that appears in between the `` element appears bold on the web page.

Example

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

Result:

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

Italic Text

```
<!DOCTYPE html>
<html>
  <head>
    <title>Italic Text Example</title>
  </head>
  <body>
    <p>The following word uses a <i>italicized</i> typeface.</p>
  </body>
</html>
```

Result:

The following word uses a *italicized* typeface.

Underlined Text

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

```
<!DOCTYPE html>
<html>
  <head>
    <title>Underlined Text Example</title>
  </head>
  <body>
    <p>The following word uses a <u>underlined</u> typeface.</p>
  </body>
</html>
```

Result:

The following word uses an underlined typeface.

Strike Text

Anything that appears within `<strike>...</strike>` element is displayed with strikethrough, which is a thin line through a text.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Strike Text Example</title>
  </head>
  <body>
    <p>The following word uses a <strike>strikethrough</strike> typeface.</p>
  </body>
</html>
```

Result:

The following word uses a ~~strikethrough~~ typeface.

Monospaced Font

The content of a `<tt>...</tt>` element is written in monospaced font. Most of the fonts are known as variable-width fonts because different letters are of different widths (for example, the letter 'm' is wider than the letter 'i'). In a monospaced font, however, each letter has the same width.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Monospaced Font Example</title>
  </head>
  <body>
    <p>The following word uses a <tt>monospaced</tt> typeface.</p>
  </body>
</html>
```

Result:

The following word uses a `monospaced` typeface.

Superscript Text

The content of a `^{...}` 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.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Superscript Text Example</title>
  </head>
  <body>
    <p>The following word uses a <sup>superscript</sup> typeface.</p>
  </body>
</html>
```

Result:

The following word uses a ^{superscript} typeface.

Subscript Text

The content of a `_{...}` 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.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Subscript Text Example</title>
  </head>
  <body>
    <p>The following word uses a <sub>subscript</sub> typeface.</p>
  </body>
</html>
```

Result:

The following word uses a _{subscript} typeface.

Inserted / Deleted Text

Sometimes, you might want to show that a text has been deleted and another is inserted in its place. In this case, the `...` and `<ins>...</ins>` are useful. The deleted text still shows but with a strike through it while the inserted text is underlined.

Anything that appears within `<ins>...</ins>` element is displayed as inserted text.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inserted Text Example</title>
  </head>
  <body>
    <p>I want to drink <del>cola</del> <ins>wine</ins></p>
  </body>
</html>
```

Result:

I want to drink ~~cola~~ wine

Larger Text

The content of the `<big>...</big>` element is displayed one font size larger than the rest of the text surrounding it.



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

Result:

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

Smaller Text

The content of the `<small>...</small>` element is displayed one font size smaller than the rest of the text surrounding it.

Example

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

Result:

The following word uses a small typeface.

Grouping Content

The `<div>` and `` elements allow you to group together several elements to create sections or subsections of a page.

For example, you might want to put all of the footnotes on a page within a `<div>` element to indicate that all of the elements within that `<div>` element relate to the footnotes. You might then attach a style to this `<div>` element so that they appear using a special set of style rules.

Example:

```

<!DOCTYPE html>
<html>
  <head>
    <title>Div Tag Example</title>
  </head>
  <body>
    <div id="menu" align="middle" >
      <a href="/index.htm">HOME</a>
      <a href="/about/contact_us.htm">CONTACT</a>
      <a href="/about/index.htm">ABOUT</a>
    </div>
    <div id="content" align="left" bgcolor="white">
      <h5>Content Articles</h5>
      <p>Actual content goes here.....</p>
    </div>
  </body>
</html>

```

Result:



The `` element, on the other hand, can be used to group inline elements only. So, if you have a part of a sentence or paragraph which you want to group together, you could use the `` element as follows

Example

```

<!DOCTYPE html>
<html>
  <head>
    <title>Span Tag Example</title>
  </head>
  <body>
    <p>This is the example of <span style="color:green">span tag</span> and the <span style="color:red">div tag</span> along with CSS</p>
  </body>
</html>

```

Result:

This is the example of span tag and the div tag along with CSS

Span and div tags are used with CSS to allow you to attach a style to a section of a page.



CHAPTER SIX

HTML – PHRASE TAGS

The phrase tags are used the same ways the <i>,<pre>, and the <tt> tags are written.

Emphasized Text

```
<!DOCTYPE html>
<html>
  <head>
    <title>Emphasized Text Example</title>
  </head>
  <body>
    <p>The following word uses a <em>emphasized</em> typeface.</p>
  </body>
</html>
```

Result:

The following word uses an *emphasized* typeface.

Marked Text

Anything that appears with-in <mark>...</mark> element, is displayed as marked with yellow ink.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Marked Text Example</title>
  </head>
  <body>
    <p>The following word has been <mark>marked</mark> with yellow</p> .
  </body>
</html>
```

Result:

The following word has been **marked** with yellow

Strong Text

Anything that appears within `...` element is displayed as important text.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Strong Text Example</title>
  </head>

  <body>
    <p>The following word uses a <strong>strong</strong> typeface.</p>
  </body>
</html>
```

Result:

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

Text Abbreviation

You can abbreviate a text by putting it inside opening `<abbr>` and closing `</abbr>` tags. If present, the title attribute must contain this full description and nothing else.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Text Abbreviation</title>
  </head>
  <body>
    <p>My best friend's name is <abbr title="Abhishek">Abhy</abbr>.</p>
  </body>
</html>
```

Result:

My best friend's name is Abhy.

Acronym Element

The **<acronym>** element allows you to indicate that the text between **<acronym>** and **</acronym>** tags is an acronym.

At present, the major browsers do not change the appearance of the content of the **<acronym>** element.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Acronym Example</title>
  </head>
  <body>
    <p>This chapter covers marking up text in <acronym>XHTML</acronym>.</p>
  </body>
</html>
```

Result:

This chapter covers marking up text in XHTML.

Text Direction

The **<bdo>...</bdo>** element stands for Bi-Directional Override and it is used to override the current text direction.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Text Direction Example</title>
  </head>
  <body>
    <p>This text will go left to right.</p>
    <p><bdo dir="rtl">This text will go right to left.</bdo></p>
  </body>
</html>
```

Result:

This will produce the following result:

This text will go left to right.

.tfel ot thgir og lliw txet sihT

Special Terms

The **<dfn>...</dfn>** element (or HTML Definition Element) allows you to specify that you are introducing a special term. It's usage is similar to italic words in the midst of a paragraph.

Typically, you would use the **<dfn>** element the first time you introduce a key term. Most recent browsers render the content of a **<dfn>** element in an italic font.

Quoting Text

When you want to quote a passage from another source, you should put it in between **<blockquote>...</blockquote>** tags.

Text inside a **<blockquote>** element is usually indented from the left and right edges of the surrounding text, and sometimes uses an italicized font.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example on how to quote text</title>
  </head>
  <body>
    <p>Quoted text will be indented left and right.</p>
    <blockquote>Text gotten from another source</blockquote>
  </body>
</html>
```

Result:

Quoted text will be indented left and right.

This is a quoted Text from another source

Short Quotations

The **<q>...</q>** element is used when you want to add a double quote within a sentence.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using short quote</title>
  </head>
  <body>
    <p>With commitment, <q>success is possible</q>.</p>
  </body>
</html>
```

Result:

With commitment, “success is possible”.

Text Citations

If you are quoting a text, you can indicate the source placing it between an opening `<cite>` tag and closing `</cite>` tag

As you would expect in a print publication, the content of the `<cite>` element is rendered in italicized text by default.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Citations Example</title>
  </head>
  <body>
    <p>This HTML tutorial is derived from <cite>W3 Standard for HTML</cite>.</p>
  </body>
</html>
```

Result:

This HTML tutorial is derived from *W3 Standard for HTML*.

Computer Code

Any programming code to appear on a Web page should be placed inside `<code>...</code>` tags. Usually the content of the `<code>` element is presented in a monospaced font, just like the code in most programming books.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Computer Code Example</title>
  </head>
  <body>
    <p>Regular text. <code>This is a coded text.</code> Regular text.</p>
  </body>
</html>
```

Result:

Regular text. This is a coded text. Regular text.

Keyboard Text

When you are talking about computers, if you want to tell a reader to enter some text, you can use the `<kbd>...</kbd>` element to indicate what should be typed in, as in this example.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Keyboard Text Example</title>
  </head>
  <body>
    <p>Regular text. <kbd>This is inside kbd element</kbd> Regular text.</p>
  </body>
</html>
```

Result:

Regular text. This is inside kbd element Regular text.

Program Output

The `<samp>...</samp>` element indicates sample output from a program, and script etc. Again, it is mainly used when documenting programming or coding concepts.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Program Output Example</title>
  </head>
  <body>
    <p>Result produced by the program is <samp>Hello World!</samp></p>
  </body>
</html>
```

Result:

Result produced by the program is Hello World!

Address Text

The `<address>...</address>` element is used to contain any address.

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

```
<head>
  <title>Address Example</title>
</head>
<body>
  <address>No. 1. Kuje Town Hall Admin block, FCT-Abuja, Nigeria</address>
</body>
</html>
```

Result:

No. 1. Kuje Town Hall Admin block, FCT-Abuja, Nigeria



CHAPTER SEVEN

HTML – META TAGS

HTML lets you specify metadata - additional important information about a document in a variety of ways. The META elements can be used to include name/value pairs describing properties of the HTML document, such as author, expiry date, a list of keywords, document author etc.

The **<meta>** tag is used to provide such additional information. This tag is an empty element and so does not have a closing tag but it carries information within its attributes.

You can include one or more meta tags in your document based on what information you want to keep in your document but in general, meta tags do not impact physical appearance of the document so from appearance point of view, it does not matter if you include them or not.

Adding Meta Tags to Your Documents

You can add metadata to your web pages by placing **<meta>** tags inside the header of the document which is represented by **<head>** and **</head>** tags. A meta tag can have following attributes in addition to core attributes:

| Attribute | Description |
|------------|---|
| Name | Name for the property. Can be anything. Examples include, keywords, description, author, revised, generator etc. |
| content | Specifies the property's value. |
| scheme | Specifies a scheme to interpret the property's value (as declared in the content attribute). |
| http-equiv | Used for http response message headers. For example, http-equiv can be used to refresh the page or to set a cookie. Values include content-type, expires, refresh and set-cookie. |

Specifying Keywords

You can use <meta> tag to specify important keywords related to the document and later these keywords are used by the search engines while indexing your webpage for searching purpose.

Example

Following is an example, where we are adding HTML, Meta Tags, Metadata as important keywords about the document.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tags Example</title>
    <meta name="keywords" content="HTML, Meta Tags, Metadata" />
  </head>
  <body>
    <p>The meta tag in the head section is used to indicate that HTML, Meta Tags and
Metadata are key words about the document!</p>
  </body>
</html>
```

Result:

The meta tag in the head section is used to indicate that HTML, Meta Tags and Metadata are key words about the document!

Document Description

You can use <meta> tag to give a short description about the document. This again can be used by various search engines while indexing your webpage for searching purpose.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tags Example</title>
    <meta name="description" content="Learning about Meta Tags." />
  </head>
  <body>
    <p>Describing a document with Meta tag!</p>
  </body>
</html>
```

Result:

Describing a document with Meta tag!

Document Revision Date

You can use <meta> tag to give information about when last time the document was updated. This information can be used by various web browsers while refreshing your webpage.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tags Example</title>
    <meta name="revised" content="gr8tson web design material, 11/1/2020" />
  </head>
  <body>
    <p>Meta tag used to show document revision date</p>
  </body>
</html>
```

Result:

Meta tag used to show document revision date

Document Refreshing

A <meta> tag can be used to specify a duration after which your web page will keep refreshing automatically. If you want your page keep refreshing after every 5 seconds then use the following syntax.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tags Example</title>
    <meta http-equiv="refresh" content="5" />
  </head>
  <body>
    <p>Meta tag used to refresh a web page after 5 seconds</p>
  </body>
</html>
```

Result:

Meta tag used to refresh a web page after 5 seconds

Page Redirection

You can use <meta> tag to redirect your page to any other webpage. You can also specify a duration if you want to redirect the page after a certain number of seconds. Following is an example of redirecting current page to another page after 5 seconds. If you want to redirect page immediately then do not specify *content* attribute.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tag used to redirect to new web page</title>
    <meta http-equiv="refresh" content="5; url=http://www.gr8tson.com.ng" />
  </head>
  <body>
    <p>This page is redirected to www.gr8tson.com.ng after 5 seconds</p>
  </body>
</html>
```

Result:

This page is redirected to www.gr8tson.com.ng after 5 seconds

Setting Cookies

Cookies are data, stored in small text files on your computer and it is exchanged between web browser and web server to keep track of various information based on your web application need.

You can use <meta> tag to store cookies on client side and later this information can be used by the Web Server to track a site visitor.

If you do not include the expiration date and time, the cookie is considered a session cookie and will be deleted when the user exits the browser.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tag used to set cookie on a web page</title>
    <meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15
23:59:59 GMT;" />
  </head>
  <body>
    <p>Cookie is set on this page</p>
  </body>
</html>
```

Result:

Cookie is set on this page

Setting Author Name

The Author's name can be set using the meta tag as shown below:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tag used to set Name of Author on a web page</title>
    <meta name="author" content="Adu Leanat" />
  </head>
  <body>
    <p>Author's name is set using the Meta tag on this page</p>
  </body>
</html>
```

Result:

Author's name is set using the Meta tag on this page

Specify Character Set

You can use <meta> tag to specify character set used within the webpage.

Example

By default, Web servers and Web browsers use ISO-8859-1 (Latin1) encoding to process Web pages. Following is an example to set UTF-8 encoding:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meta Tag used to set Character set encoding on this web page</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body>
    <p>Character set encoding using Meta tag on this page</p>
  </body>
</html>
```

Result:

Character set encoding using Meta tag on this page

CHAPTER EIGHT

HTML – COMMENTS

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>
```

Valid vs Invalid Comments

Comments do not nest which means a comment cannot be put inside another comment. Second the double-dash sequence `--` may not appear inside a comment except as part of the closing `-->` tag. You must also make sure that there are no spaces in the start-of-comment string.

Note that if there is a space between the left angle bracket and the exclamation mark, it makes the comment invalid.

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
        <title>Invalid Comment Example</title>
```

```
    </head>
```

```
<body>
    <!-- This is not a valid comment -->
    <p>Document content goes here.....</p>
</body>
</html>
```

Multiline Comments

You can comment multiple lines by the special beginning tag `<!--` and ending tag `-->` placed before the first line and end of the last line as shown in the given example below.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Multiline Comments</title>
    </head>
    <body>
        <!--
        This is a multiline comment and it can
        span through as many as lines you like.
        -->
        <p>Document content goes here.....</p>
    </body>
</html>
```

Result:

Document content goes here.....

Conditional Comments

Conditional comments only work in Internet Explorer (IE) on Windows but they are ignored by other browsers. They are supported from Explorer 5 onwards, and you can use them to give conditional instructions to different versions of IE.

```
<!DOCTYPE html><html>
  <head>
    <title>Conditional Comments</title>
    <!--[if IE 6]>
      Special instructions for IE 6 here
    <![endif]-->
  </head>
  <body>
    <p>Document content goes here.....</p>
  </body>
</html>
```

Using Comment Tag

There are few browsers that support `<comment>` tag to comment a part of HTML code. Internet explorer support it.

```
<!DOCTYPE html><html>
  <head>
    <title>Using Comment Tag</title>
  </head>
  <body>
    <p>This is <comment>not</comment> Internet Explorer.</p>
  </body>
</html>
```

Commenting Script Code

If you are using Java Script or VB Script in your HTML code then it is recommended to put that script code inside proper HTML comments so that old browsers can work properly.

```
<!DOCTYPE html><html>
  <head>
    <title>Commenting Script Code</title>
    <script>
      <!--
      document.write("Hello World!")
      //-->
    </script>
  </head>
  <body>
    <p>Hello , World!</p>
  </body>
</html>
```

Result:

Hello World!

Hello, World!

Commenting Style Sheets

Note that if you are using Cascading Style Sheet (CSS) in your HTML code, then it is recommended to put that style sheet code inside proper HTML comments so that old browsers can work properly.

```
<!DOCTYPE html><html>
  <head>
    <title>Commenting Style Sheets</title>
    <style>
      <!--
      .example {
        border:1px solid #4a7d49;
      }
      //-->
    </style>
  </head>
  <body>
    <div class="example">Hello , World!</div>
  </body>
</html>
```

Result:

Hello, World!



CHAPTER NINE

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 **** tag. Following is the simple syntax to use this tag.

```

```

If we have our LOGO.png image in the same directory with our index.html file, we shall add the image to our code using this format below:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using Image in Webpage</title>
  </head>
  <body>
    <p>Simple Image Insert</p>
    
  </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 index.html in our home directory and create a subdirectory **images** inside the home directory where we will keep our image LOGO.png.

Example

Assuming our image location is "image/LOGO.png", try the following example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using Image in Webpage</title>
  </head>
  <body>
    <p>Simple Image Insert</p>
    
  </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.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Set Image Width and Height</title>
  </head>
  <body>
    <p>Setting image width and height</p>
    
  </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.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Set Image Border</title>
  </head>
  <body>
    <p>Setting image Border</p>
    
  </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.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Set Image Alignment</title>
  </head>
  <body>
    <p>Setting image Alignment</p>
    
  </body>
</html>
```

Free Web Graphics

You can find more web graphics by visiting the website address below.

http://www.tutorialspoint.com/free_web_graphics.htm

CHAPTER TEN

HTML – TABLES

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.

```
<!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>
```

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.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table Header</title>
  </head>
  <body>
    <table border="1">
      <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>
```

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 the width of the border, 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>
    <td>7000</td>
</tr>
</table>
</body>
</html>

```

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.

```

<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Colspan/Rowspan</title>
    </head>
    <body>
        <table border="1">
            <tr>
                <th>Column heading 1</th>
                <th>Column heading 2</th>
                <th>Column heading 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>
        <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>

```

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 Backgrounds

You can set table background using one of the following two ways:

- **bgcolor** attribute - You can set background color for whole table or just for one cell.
- **background** attribute - You can set background image for whole table or just for one cell.

You can also set border color also using **bordercolor** attribute.

```

<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Background</title>
    </head>
    <body>
        <table border="1" bordercolor="green" bgcolor="yellow">
            <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>

```

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 | | |

Here is an example of using **background** attribute. Here we will use an image available in /images directory.

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table Background</title>
  </head>
  <body>
    <table border="1" bordercolor="green" background="/images/LOGO.png">
      <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>
```

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.

```
<!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>
```

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.


```

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

```

Result:

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

Table Header, Body, and Footer

Tables can be divided into three portions: a header, a body, and a foot. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content holder of the table.

The three elements for separating the head, body, and foot of a table are:

- **<thead>** - to create a separate table header.
- **<tbody>** - to indicate the main body of the table.
- **<tfoot>** - to create a separate table footer.

A table may contain several **<tbody>** elements to indicate different *pages* or groups of data. But it is notable that **<thead>** and **<tfoot>** tags should appear before **<tbody>**

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table</title>
  </head>


```

```

<body>
  <table border="1" width="40%">
    <thead>
      <tr>
        <td colspan="4">This is the head of the table</td>
      </tr>
    </thead>
    <tfoot>
      <tr>
        <td colspan="4">This is the foot of the table</td>
      </tr>
    </tfoot>
    <tbody>
      <tr>
        <td>Cell 1</td>
        <td>Cell 2</td>
        <td>Cell 3</td>
        <td>Cell 4</td>
      </tr>
    </tbody>
  </table>
</body>
</html>

```

Result:



| | | | |
|-------------------------------|--------|--------|--------|
| This is the head of the table | | | |
| Cell 1 | Cell 2 | Cell 3 | Cell 4 |
| This is the foot of the table | | | |

Nested Tables

You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag <td>.

Example

Following is the example of using another table and other tags inside a table cell.

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table</title>
  </head>
  <body>
    <table border="1" width="100%">
      <tr>

```

```
<td>
  <table border="1" width="100%">
    <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>
</td>
</tr>
</table>
</body>
</html>
```

Result:

| Name | Salary |
|-----------------|--------|
| Ramesh Raman | 5000 |
| Shabbir Hussein | 7000 |

CHAPTER ELEVEN

HTML – LISTS

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

- **** - An unordered list. This will list items using plain bullets.
- **** - An ordered list. This will use different schemes of numbers to list your items.
- **<dl>** - A definition list. This arranges your items in the same way as they are arranged in a dictionary.

HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML **** tag. Each item in the list is marked with a bullet.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Unordered List</title>
  </head>
  <body>
    <ul>
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ul>
  </body>
</html>
```

Result:

- Beetroot
- Ginger
- Potato
- Radish

The type Attribute

You can use **type** attribute for tag to specify the type of bullet you like. By default, it is a disc. Following are the possible options:

```
<ul type="square">
```

```
<ul type="disc">
```

```
<ul type="circle">
```

HTML Ordered Lists

If you are required to put your items in a numbered list instead of bulleted, then HTML ordered list will be used. This list is created by using tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with .

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>HTML Ordered List</title>
```

```
  </head>
```

```
  <body>
```

```
    <ol>
```

```
      <li>Beetroot</li>
```

```
      <li>Ginger</li>
```

```
      <li>Potato</li>
```

```
      <li>Radish</li>
```

```
    </ol>
```

```
  </body>
```

```
</html>
```

Result:

1. Beetroot
2. Ginger
3. Potato
4. Radish

The type Attribute

You can use **type** attribute for tag to specify the type of numbering you like. By default, it is a number. Following are the possible options:

<ol type="1"> - Default-Case Numerals.

<ol type="I"> - Upper-Case Numerals.

<ol type="i"> - Lower-Case Numerals.

<ol type="a"> - Lower-Case Letters.

<ol type="A"> - Upper-Case Letters.

Following is an example where we used <ol type="1">

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Ordered List</title>
  </head>
  <body>
    <ol type="1">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ol>
  </body>
</html>
```

Result:

1. Beetroot
2. Ginger
3. Potato
4. Radish



The start Attribute

You can use **start** attribute for tag to specify the starting point of numbering you need.

Following are the possible options:

<ol type="1" start="4"> - Numerals starts with 4.

<ol type="I" start="4"> - Numerals starts with IV.

<ol type="i" start="4"> - Numerals starts with iv.

<ol type="a" start="4"> - Letters starts with d.

<ol type="A" start="4"> - Letters starts with D.

Following is an example where we used <ol type="i" start="4" >

```
<!DOCTYPE html>
```

```

<html>
  <head>
    <title>HTML Ordered List</title>
  </head>
  <body>
    <ol type="i" start="4">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ol>
  </body>
</html>

```

Result:

- iv. Beetroot
- v. Ginger
- vi. Potato
- vii. Radish

HTML Definition Lists

HTML and XHTML supports a list style which is called **definition lists** where entries are listed like in a dictionary or encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags:

- <dl> - Defines the start of the list
- <dt> - A term
- <dd> - Term definition
- </dl> - Defines the end of the list

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML Definition List</title>
  </head>
  <body>
    <dl>
      <dt><b>HTML</b></dt>
      <dd>This stands for Hyper Text Markup Language</dd>

      <dt><b>HTTP</b></dt>

```

```
                <dd>This stands for Hyper Text Transfer Protocol</dd>
            </dl>
    </body>
</html>
```

Result:

HTML

This stands for Hyper Text Markup Language

HTTP

This stands for Hyper Text Transfer Protocol



CHAPTER TWELVE

HTML LINKS

Html – Text Links

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.

For more understanding on links, visit:

www.tutorialspoint.com/html/understanding_url_tutorials.htm

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 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.

Link Text

Let's try following example which links <http://www.gr8tson.com.ng> at your page:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hyperlink Example</title>
  </head>
  <body>
    <p>Click following link</p>
    <a href="http://www.gr8tson.com.ng" target="_self">GR8TSON WEB
    PAGE</a>
  </body>
</html>
```

Result:

[GR8TSON WEB PAGE](http://www.gr8tson.com.ng)

The target Attribute

We have used **target** attribute in our previous example. This attribute is used to specify the location where linked document is opened. Following are the possible options:

| Option | Description |
|-------------|---|
| _blank | Opens the linked document in a new window or tab. |
| _self | Opens the linked document in the same frame. |
| _parent | Opens the linked document in the parent frame. |
| _top | Opens the linked document in the full body of the window. |
| targetframe | Opens the linked document in a named <i>targetframe</i> |

Lets try the following example to understand basic difference in few options given for target attribute.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hyperlink Example</title>
    <base href="http://www.gr8tson.com.ng/">
  </head>
  <body>
    <p>Click any of the following links</p>
    <a href="/html/index.htm" target="_blank">Opens in New</a> |
    <a href="/html/index.htm" target="_self">Opens in Self</a> |
    <a href="/html/index.htm" target="_parent">Opens in Parent</a> |
    <a href="/html/index.htm" target="_top">Opens in Body</a>
  </body>
</html>
```

Result:

[Opens in New](#) | [Opens in Self](#) | [Opens in Parent](#) | [Opens in Body](#)

Linking to a Page Section

You can create a link to a particular section of a given webpage by using **name** attribute. This is a two-step process.

First create a link to the place where you want to reach with-in a webpage and name it using `<a...>` tag as follows:

```
<h1>HTML Text Links <a name="top"></a></h1>
```

Second step is to create a hyperlink to link the document and place where you want to reach:

```
<a href="html_text_links.htm#top">Go to the Top</a>
```

Result:

[Go to the Top](#)

Setting Link Colors

You can set colors of your links, active links and visited links using **link**, **alink** and **vlink** attributes of `<body>` tag.

These attributes control the colors of the different link states:

- LINK - initial appearance – default = Blue
- VLINK - visited link – default = Purple
- ALINK - active link being clicked – default = Red

Many web developers will set the link colors of their documents to flow with the color scheme of the site. The format for setting these attributes is:

```
<BODY BGCOLOR="#FFFFFF" TEXT="#FF0000" LINK="#0000FF"
VLINK="#FF00FF" ALINK="FFFF00">
```

The results of the above BODY element would be a white background with links being blue, visited links as magenta and active links colored in yellow.

Let us see how **link**, **alink** and **vlink** attribute work.

```
<!DOCTYPE html>
<html>
  <head>
```

```

        <title>Hyperlink Example</title>
        <base href="http://www.gr8tson.com.ng/">
        </head>
        <body BGCOLOR="#FFFFFF" TEXT="#FF0000" link="#040404" vlink="#F40633"
        alink="#54A250">
            <p>Click following link</p>
            <a href="/html/index.htm" target="_blank" >HTML Tutorial</a>
        </body>
    </html>

```

Download Links

You can create text link to make your PDF, or DOC or ZIP files downloadable. This is very simple; you just need to give complete URL of the downloadable file as follows:

```

<!DOCTYPE html>
<html>
    <head>
        #!/usr/bin/perl
        # Additional HTTP Header
        print "Content-Type:application/octet-stream; name=\"FileName\"\\r\\n";
        print "Content-Disposition: attachment; filename=\"FileName\"\\r\\n\\n";
        # Open the target file and list down its content as follows
        open( FILE, "<FileName" );
        while(read(FILE, $buffer, 100)){
            print("$buffer");
        }
        <title>Hyperlink Example</title>
    </head>
    <body>
        <a href="http://www.gr8tson.com.ng/page.pdf">Download PDF File</a>
    </body>
</html>

```

Result:

[Download PDF File](#)

Html – Image Links

We have seen how to create hypertext link using text and we also learnt how to use images in our web pages. Now, we will learn how to use images to create hyperlinks.

It's simple to use an image as hyperlink. We just need to use an image inside hyperlink at the place of text as shown below:

```

<!DOCTYPE html>
<html>
  <head>
    <title>Image Hyperlink Example</title>
  </head>
  <body>
    <p>Click following link</p>
    <a href="http://www.gr8tson.com.ng" target="_self">
      
    </a>
  </body>
</html>

```

Result:



Html – Email Links

It is not difficult to put an HTML email link on your webpage but it can cause unnecessary spamming problem for your email account. There are people, who can run programs to harvest these types of emails and later use them for spamming in various ways.

You can have another option to facilitate people to send you emails. One option could be to use HTML forms to collect user data and then use PHP or CGI script to send an email.

A simple example, check our **Contact Us** Form. We take user feedback using this form and then we are using one CGI program which is collecting this information and sending us email to the one given email ID.

Note: You will learn about HTML Forms in **HTML Forms** and you will learn about CGI in our another tutorial **Perl CGI Programming**.

HTML Email Tag

HTML **<a>** tag provides you option to specify an email address to send an email. While using **<a>** tag as an email tag, you will use **mailto: email address** along with *href* attribute. Following is the syntax of using **mailto** instead of using http.

```
<a href= "mailto: abc@example.com">Send Email</a>
```

This code will generate the following link which you can use to send email.

Send Email

Now, if a user clicks this link, it launches one Email Client (like Lotus Notes, Outlook Express etc.) installed on your user's computer. There is another risk to use this option to send email because if user do not have email client installed on their computer then it would not be possible to send email.

Default Settings

You can specify a default *email subject* and *email body* along with your email address. Following is the example to use default subject and body.

```
<a href="mailto:abc@example.com?subject=Feedback&body=Message">
```

Result:

Send Feedback



CHAPTER THIRTEEN

HTML – FRAMES

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages:

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back button* might not work as the user hopes.
- There are still few browsers that do not support frame technology.

Creating Frames

To use frames on a page, we use `<frameset>` tag instead of `<body>` tag. The `<frameset>` tag defines, how to divide the window into frames. The **rows** attribute of `<frameset>` tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by `<frame>` tag and it defines which HTML document shall open into the frame.

Shown below is how to create a three rows horizontal frames.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
</head>
<frameset rows="10%,80%,10%">
    <frame name ="top" src="topframepage.php" />
    <frame name ="main" src="mainframepage.php" />
    <frame name ="bottom" src="bottomframepage.php" />
```

```

        </noframes>
        <body>
        Your browser does not support frames.
        </body>
        </noframes>
    </frameset>
</html>

```

The three html documents (**topframepage.php**, **mainframepage.php**, **bottomframepage.php**) that will load into the frames are as included below:

topframepage.php code:

```

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Untitled Document</title>
    </head>
    <body>
        <table border="1" bordercolor="#7BBF91" bgcolor="#92E389" height="100"
        width="80%">
            <tr>
            <td></td>
            </tr>
        </table>
    </body>
</html>

```



mainframepage.php code:

```

<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Untitled Document</title>
    </head>

    <body>
        <table border="1" bordercolor="#C1CF75" bgcolor="#BEE95C" height="800"
        width="80%">
            <tr>
            <td></td>
            </tr>
        </table>
    </body>
</html>

```


bottomframepage.php code:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Untitled Document</title>
  </head>

  <body>
    <table border="1" bordercolor="#CBE8DE" bgcolor="rgba(75,231,88,1.00)"
    height="100" width="80%">
      <tr>
        <td></td>
      </tr>
    </table>
  </body>
</html>
```

Result:



The code below also shows a frame of two columns with the associated code for the HTML documents attached.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Untitled Document</title>
  </head>
  <frameset cols="25%,75%">
    <frame name ="left" src="leftframepage.php" scrolling="no" />
    <frame name ="center" src="rightframepage.php" />
  </frameset>
</html>
```

```

</noframes>
<body>
Your browser does not support frames.
</body>
</noframes>
</frameset>
</html>

```

For the sake of navigation, a two column frameset is included so that a left column will hold the frame displaying the links while the right column will display the frames for the three rows discussed above.

The two column tables are as shown below:

leftframepage.php code

```

<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Untitled Document</title>
  </head>
  <body>
    <table border="1" bordercolor="#CBE8DE" bgcolor="rgba(75,231,88,1.00)"
    height="650" width="100%">
      <tr>
        <td></td>
      </tr>
    </table>
  </body>
</html>

```

rightframepage.php code:

```

!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Untitled Document</title>
  </head>

  <body>
    <table border="1" bordercolor="#7BBF91" bgcolor="#92E389" height="650"
    width="100%">
      <tr>
        <td></td>
      </tr>
    </table>
  </body>
</html>

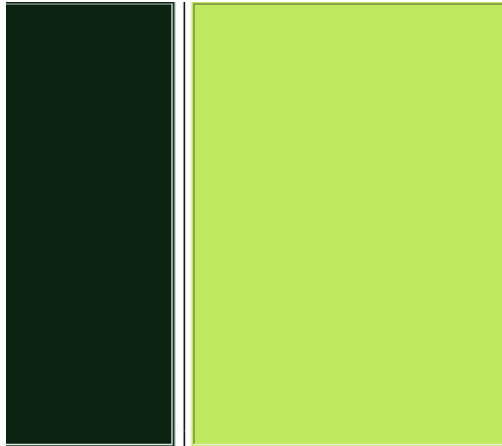
```

```

        </table>
    </body>
</html>

```

Result:



The <frameset> Tag Attributes

| Attribute | Description |
|-----------|---|
| cols | <p>Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways:</p> <ul style="list-style-type: none"> • Absolute values in pixels. For example, to create three vertical frames, use <code>cols="100, 500, 100"</code>. • A percentage of the browser window. For example, to create three vertical frames, use <code>cols="10%, 80%, 10%"</code>. • Using a wildcard symbol. For example, to create three vertical frames, use <code>cols="10%, *, 10%"</code>. In this case wildcard takes remainder of the window. • As relative widths of the browser window. For example, to create three vertical frames, use <code>cols="3*, 2*, 1*"</code>. This is an alternative to percentages. You can use relative widths of the |

| | |
|--------------|--|
| | browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth. |
| rows | This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use <code>rows="10%, 90%"</code> . You can specify the height of each row in the same way as explained above for columns. |
| border | This attribute specifies the width of the border of each frame in pixels. For example, <code>border="5"</code> . A value of zero means no border. |
| frameborder | This attribute specifies whether a three-dimensional border should be displayed between frames. |
| framespacing | This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example <code>framespacing="10"</code> means there should be 10 pixels spacing between each frames. |

The <frame> Tag Attributes

Following are the important attributes of <frame> tag:

| Attribute | Description |
|-----------|--|
| src | This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, <code>src="top_frame.htm"</code> will load an HTML file available in html directory. |
| name | This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link. |

| | |
|--------------|--|
| frameborder | This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no). |
| marginwidth | This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10". |
| marginheight | This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10". |
| noresize | By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize". |
| scrolling | This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars. |
| longdesc | This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm" |

Browser Support for Frames

If a user is using any old browser or any browser, which does not support frames then <noframes> element should be displayed to the user.

So you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.

You can put some nice message for your user having old browsers. For example, *Sorry!! your browser does not support frames.* as shown in the above example.

Frame's name and target attributes

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.

The example below shows how the target and the name attributes help to load a frame into another:

This is a frameset containing two frames. The `menu_page.php` will load inside the first frame while the `main_page.php` will load inside the second frame. The `menu_page.php` and `main_page.php` contain pages which are targeted at the `menu_page` frame and the `main_page` frame.

The page **framesetwithframes.php** contains frames have name attributes which are given the values `menu_page` and `main_page` respectively.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" http-equiv="refresh" content="30">
    <title>Untitled Document</title>
  </head>
  <frameset cols="25%,75%" frameborder="0" framespacing="10">
    <frame name ="menu_page" src="menu_page.php" scrolling="no" />
    <frame name ="main_page" src="main_page.php" />

    <noframes>
    <body>
      Your browser does not support frames.
    </body>
    </noframes>
  </frameset>
</html>
```

menu_page.php code:

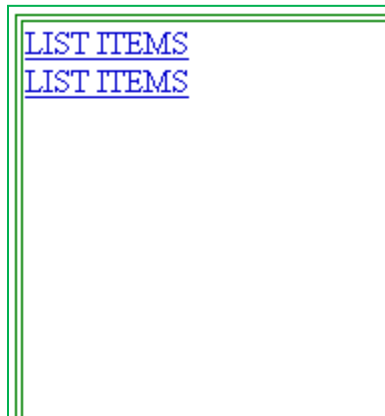
```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Untitled Document</title>
  </head>
  <body>
    <table border="1" bordercolor="green" bgcolor="white" height="650"
      width="100%">
      <tr valign="top">
```

```

        <td>
        <a href="listitems.php" target="main_page">LIST ITEMS</a><br />
        <a href="index.php" target="main_page">LIST ITEMS</a>
        </td>
    </tr>
</table>
</body>
</html>

```

Result:



main_page.php code:

```

<!doctype html>
<html>
    <head>
        <style type="text/css">
            #round{ border-radius: 100px, 100px, 100px, 100px;
            border-color:rgba(244,206,206,1.00);

            }
        </style>
        <meta charset="utf-8">
        <title>VISA REGISTRATION PROCESS</title>
    </head>

    <body>
        <table border="1" bordercolor="#C1CF75" bgcolor="#BEE95C" height="650"
        width="100%">
            <tr valign="top">
                <td><p align="center">
                    <strong><marquee direction="up">MAIN
                    PAGE</marquee></strong></p>

```

```

        <table width="100%" height="600" bgcolor="#C6EDAB" border="1"
id="round">
        <tr>
        <td bgcolor="#5BB860"></td>
        </tr>
        </table>

</td>
</tr>
</table>

</body>
</html>

```

Combined result:



The links on the left frame are targeted to the main page. Once clicked upon, they load inside the main page.

The *target* attribute can also take one of the following values:

| Option | Description |
|-------------|---|
| _self | Loads the page into the current frame. |
| _blank | Loads a page into a new browser window. |
| _parent | Loads the page into the parent window, which in the case of a single frameset is the main browser window. |
| _top | Loads the page into the browser window, replacing any current frames. |
| targetframe | Loads the page into a named targetframe. |

CHAPTER FOURTEEN

HTML – IFRAMES

You can define an inline frame with HTML tag **<iframe>**. The **<iframe>** tag is not somehow related to **<frameset>** tag, instead, it can appear anywhere in your document. The **<iframe>** tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders.

The **src** attribute is used to specify the URL of the document that occupies the inline frame.

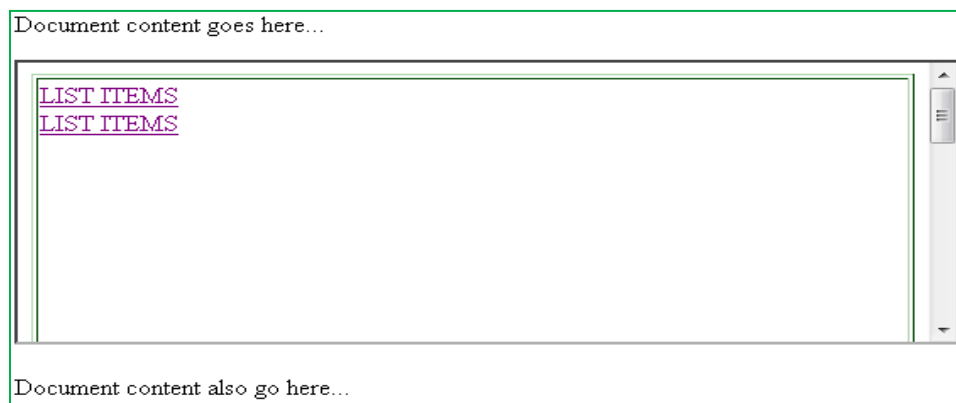
Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Iframes</title>
  </head>
  <body>
    <p>Document content goes here...</p>

    <iframe src="menu_page.php" width="555" height="200">
      Sorry your browser does not support inline frames.
    </iframe>

    <p>Document content also go here...</p>
  </body>
</html>
```

Result:



The **<iframe>** Tag Attributes

Most of the attributes of the <iframe> tag, including *name*, *class*, *frameborder*, *id*, *longdesc*, *marginheight*, *marginwidth*, *name*, *scrolling*, *style*, and *title* behave exactly like the corresponding attributes for the <frame> tag.

| Attribute | Description |
|--------------|--|
| src | This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src="top_frame.htm" will load an HTML file available in html directory. |
| name | This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link. |
| frameborder | This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no). |
| marginwidth | This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10". |
| marginheight | This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10". |
| noresize | By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize". |
| scrolling | This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have |

| | |
|----------|---|
| | scroll bars. |
| longdesc | This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm" |



CHAPTER FIFTEEN

HTML – BLOCKS

All the HTML elements can be categorized into two categories:

- (a) Block Level Elements
- (b) Inline Elements.

Block Elements

Block elements appear on the screen as if they have a line break before and after them. For example, the `<p>`, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, ``, ``, `<dl>`, `<pre>`, `<hr />`, `<blockquote>`, and `<address>` elements are all block level elements. They all start on their own new line, and anything that follows them appears on its own new line.

Inline Elements

Inline elements, on the other hand, can appear within sentences and do not have to appear on a new line of their own. The ``, `<i>`, `<u>`, ``, ``, `<sup>`, `<sub>`, `<big>`, `<small>`, ``, `<ins>`, ``, `<code>`, `<cite>`, `<dfn>`, `<kbd>`, and `<var>` elements are all inline elements.

Grouping HTML Elements

There are two important tags which we use very frequently to group various other HTML tags (i) `<div>` tag and (ii) `` tag

The `<div>` tag

This is the very important block level tag which plays a big role in grouping various other HTML tags and applying CSS on group of elements. Even now `<div>` tag can be used to create webpage layout where we define different parts (Left, Right, Top etc.) of the page using `<div>` tag. This tag does not provide any visual change on the block but it has more meaning when it is used with CSS.

The code below shows how to use div tag.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML div Tag</title>
```

```

</head>
<body>
  <!-- First group of tags -->

  <div style="color:red">
    <h4>This is first group of tags</h4>
    <p>Following is a list of vegetables</p>
    <ul>
      <li>Carot</li>
      <li>Ginger</li>
      <li>Galic</li>
      <li>Onions</li>
    </ul>
  </div>

  <!-- Second group of tags -->
  <div><h4>This is second group</h4>
  <p>Following is a list of fruits</p>
  <ul>
    <li>Apple</li>
    <li>Banana</li>
    <li style="color:green">Mango</li>
    <li>Strawberry</li>
  </ul>
  </div>
</body>
</html>

```



Result:

THIS IS FIRST GROUP

Following is a list of vegetables

- Beetroot
- Ginger
- Potato
- Radish

THIS IS SECOND GROUP

Following is a list of fruits

Apple

- Banana

- Mango
- Strawberry

The tag

The HTML is an inline element and it can be used to group inline elements in an HTML document. This tag also does not provide any visual change on the block but has more meaning when it is used with CSS.

The difference between the tag and the <div> tag is that the tag is used with inline elements whereas the <div> tag is used with block-level elements.

Following is a simple example of tag. We will learn Cascading Style Sheet (CSS) in a separate chapter but we used it here to show the usage of tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML span Tag</title>
  </head>
  <body>
    <p>This is <span style="color:red">red</span> and this is <span
      style="color:green">green</span></p>
  </body>
</html>
```

Result:

This is red, and this is green

HTML – Backgrounds

By default, your webpage background is white in color. HTML provides the following two ways to decorate your webpage background.

- Html Background with Colors
- Html Background with Images

Now let's see both the approaches one by one using appropriate examples.

Html Background with Colors

The **bgcolor** attribute is used to control the background of an HTML element, specifically page body and table backgrounds. Following is the syntax to use bgcolor attribute with any HTML tag.

It follows this syntax:

```
<tagname bgcolor="color_value"...>
```

This color_value can be given in any of the following formats:

<!-- Format 1 - Use color name -->

```
<table bgcolor="lime" >
```

<!-- Format 2 - Use hex value -->

```
<table bgcolor="#f1f1f1" >
```

<!-- Format 3 - Use color value in RGB terms -->

```
<table bgcolor="rgb(0,0,120)" >
```

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>HTML Background Colors</title>
```

```
  </head>
```

```
  <body>
```

```
    <!-- Format 1 - Use color name -->
```

```
    <table bgcolor="yellow" width="100%">
```

```
      <tr><td>
```

```
        This background is yellow
```

```
      </td></tr>
```

```
    </table>
```

```
    <!-- Format 2 - Use hex value -->
```

```
    <table bgcolor="#6666FF" width="100%">
```

```
      <tr><td>
```

```
        This background is sky blue
```

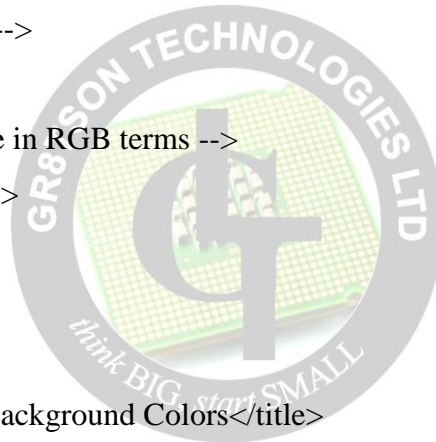
```
      </td></tr>
```

```
    </table>
```

```
    <!-- Format 3 - Use color value in RGB terms -->
```

```
    <table bgcolor="rgb(255,0,255)" width="100%">
```

```
      <tr><td>
```



```

        This background is green
    </td></tr>
</table>
</body>
</html>

```

Result:



Html Background with Images

The **background** attribute can also be used to control the background of an HTML element, specifically page body and table backgrounds. You can specify an image to set background of your HTML page or table. Following is the syntax to use background attribute with any HTML tag.

Note: The *background* attribute is deprecated and it is recommended to use Style Sheet for background setting.

```
<tagname background="Image URL"...>
```

The most frequently used image formats are JPEG, GIF and PNG images

Here are the examples to set background images of a table.

```

<!DOCTYPE html>
<html>
    <head>
        <title>HTML Background Images</title>
    </head>
    <body>
        <!-- Set table background -->
        <table background="images/html.gif" width="100%" height="100">
            <tr><td>
                This background is filled up with HTML image.
            </td></tr>
        </table>
    </body>
</html>

```


Result:



Patterned & Transparent Backgrounds

You might have seen many pattern or transparent backgrounds on various websites. This simply can be achieved by using patterned image or transparent image in the background.

It is suggested that while creating patterns or transparent GIF or PNG images, use the smallest dimensions possible even as small as 1x1 to avoid slow loading.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Background Images</title>
  </head>
  <body>
    <!-- Set a table background using pattern -->
    <table background="/images/pattern1.gif" width="100%" height="100">
      <tr><td>
        This background is filled up with a pattern image.
      </td></tr>
    </table>

    <!-- Another example on table background using pattern -->
    <table background="images/pattern2.gif" width="100%" height="100">
      <tr><td>
        This background is filled up with a pattern image.
      </td></tr>
    </table>
  </body>
</html>
```

Result:

This background is filled up with a pattern image.

This background is filled up with a pattern image.



CHAPTER SIXTEEN

HTML – COLORS

Colors are very important to give a good look and feel to your website. You can specify colors on page level using <body> tag or you can set colors for individual tags using **bgcolor** attribute.

The <body> tag has following attributes which can be used to set different colors:

- **bgcolor** - sets a color for the background of the page.
- **text** - sets a color for the body text.
- **alink** - sets a color for active links or selected links.
- **link** - sets a color for linked text.
- **vlink** - sets a color for *visited links* - that is, for linked text that you have already clicked on.

HTML Color Coding Methods

There are following three different methods to set colors in your web page:

- **Color names** - You can specify color names directly like green, blue or red.
- **Hex codes** - A six-digit code representing the amount of red, green, and blue that makes up the color.
- **Color decimal or percentage values** - This value is specified using the rgb() property.

Now we will see these coloring schemes one by one.

HTML Colors - Color Names

You can directly specify a color name to set text or background color. W3C has listed 16 basic color names that will validate with an HTML validator but there are over 200 different color names supported by major browsers.

You can visit www.tutorialspoint.com for more color names.

W3C Standard 16 Colors

Here is the list of W3C Standard 16 Colors names and it is recommended to use them.

| | | | |
|--------|-------|--------|---------|
| Black | Gray | Silver | White |
| Yellow | Lime | Aqua | Fuchsia |
| Red | Green | Blue | Purple |
| Maroon | Olive | Navy | Teal |

Here are the examples to set background of an HTML tag by color name:









```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Colors by Name</title>
  </head>
  <body text="blue" bgcolor="green">
    <p>Use different color names for for body and table and see the result.</p>
    <table bgcolor="black">
      <tr>
        <td>
          <font color="white">This text will appear white on black background.</font>
        </td>
      </tr>
    </table>
  </body>
</html>
```

HTML Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Paintshop Pro or MS Paint.

Each hexadecimal code will be preceded by a pound or hash sign #. Following is a list of few colors using hexadecimal notation.

| Color | Color HEX |
|---|-----------|
|  | #000000 |
|  | #FF0000 |
|  | #00FF00 |
|  | #0000FF |
|  | #FFFF00 |
|  | #00FFFF |
|  | #FF00FF |
|  | #C0C0C0 |
| | #FFFFFF |










Here are the examples to set background of an HTML tag by color code in hexadecimal:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Colors by Hex</title>
  </head>
  <body text="#0000FF" bgcolor="#00FF00">
    <p>Use different color hexa for for body and table and see the result.</p>
    <table bgcolor="#000000">
      <tr>
        <td>
          <font color="#FFFFFF">This text will appear white on black background.</font>
        </td>
      </tr>
    </table>
  </body>
</html>
```

HTML Colors - RGB Values

This color value is specified using the **rgb()** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

Note: It is not all browsers that support rgb() property of color so it is recommended not to use it. Following is a list to show few colors using RGB values.

| Color | Color RGB |
|---|------------------|
|  | rgb(0,0,0) |
|  | rgb(255,0,0) |
|  | rgb(0,255,0) |
|  | rgb(0,0,255) |
|  | rgb(255,255,0) |
|  | rgb(0,255,255) |
|  | rgb(255,0,255) |
|  | rgb(192,192,192) |
|  | rgb(255,255,255) |

Here are the examples to set background of an HTML tag by color code using rgb() values:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Colors by RGB code</title>
  </head>
  <body text="rgb(0,0,255)" bgcolor="rgb(0,255,0)">
    <p>Use different color code for for body and table and see the result.</p>
    <table bgcolor="rgb(0,0,0)">
      <tr>
        <td>
          <font color="rgb(255,255,255)">This text will appear white on black
          background.</font>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Browser Safe Colors

Here is the list of 216 colors which are supposed to be safest and computer independent colors. These colors vary from hexa code 000000 to FFFFFFFF and they will be supported by all the computers having 256 color palette.

| | | | | | |
|--------|--------|--------|--------|--------|--------|
| 000000 | 000033 | 000066 | 000099 | 0000CC | 0000FF |
| 003300 | 003333 | 003366 | 003399 | 0033CC | 0033FF |
| 006600 | 006633 | 006666 | 006699 | 0066CC | 0066FF |
| 009900 | 009933 | 009966 | 009999 | 0099CC | 0099FF |
| 00CC00 | 00CC33 | 00CC66 | 00CC99 | 00CCCC | 00CCFF |
| 00FF00 | 00FF33 | 00FF66 | 00FF99 | 00FFCC | 00FFFF |
| 330000 | 330033 | 330066 | 330099 | 3300CC | 3300FF |
| 333300 | 333333 | 333366 | 333399 | 3333CC | 3333FF |
| 336600 | 336633 | 336666 | 336699 | 3366CC | 3366FF |
| 339900 | 339933 | 339966 | 339999 | 3399CC | 3399FF |
| 33CC00 | 33CC33 | 33CC66 | 33CC99 | 33CCCC | 33CCFF |
| 33FF00 | 33FF33 | 33FF66 | 33FF99 | 33FFCC | 33FFFF |
| 660000 | 660033 | 660066 | 660099 | 6600CC | 6600FF |
| 663300 | 663333 | 663366 | 663399 | 6633CC | 6633FF |
| 666600 | 666633 | 666666 | 666699 | 6666CC | 6666FF |
| 669900 | 669933 | 669966 | 669999 | 6699CC | 6699FF |
| 66CC00 | 66CC33 | 66CC66 | 66CC99 | 66CCCC | 66CCFF |
| 66FF00 | 66FF33 | 66FF66 | 66FF99 | 66FFCC | 66FFFF |
| 990000 | 990033 | 990066 | 990099 | 9900CC | 9900FF |
| 993300 | 993333 | 993366 | 993399 | 9933CC | 9933FF |
| 996600 | 996633 | 996666 | 996699 | 9966CC | 9966FF |
| 999900 | 999933 | 999966 | 999999 | 9999CC | 9999FF |
| 99CC00 | 99CC33 | 99CC66 | 99CC99 | 99CCCC | 99CCFF |
| 99FF00 | 99FF33 | 99FF66 | 99FF99 | 99FFCC | 99FFFF |
| CC0000 | CC0033 | CC0066 | CC0099 | CC00CC | CC00FF |
| CC3300 | CC3333 | CC3366 | CC3399 | CC33CC | CC33FF |

CHAPTER SEVENTEEN

HTML – FONTS

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 **** 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** which are used to customize your fonts. To change any of the font attributes at any time within your webpage, simply use the **** tag. The text that follows will remain changed until you close with the **** tag. You can change one or all of the font attributes within one **** tag.

Note: The font and basefont tags are deprecated and it is supposed to be removed in a future version of HTML. So they should not be used rather, it's suggested to use CSS styles to manipulate your fonts. But still for learning purpose, this chapter will explain font and basefont tags in detail.

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" |

Relative Font Size

You can specify how many sizes larger or how many sizes smaller than the preset font size should be. You can specify it like `` or ``

```
<!DOCTYPE html>
<html>
  <head>
    <title>Relative Font Size</title>
  </head>
  <body>
    <font size="-1">Font size="-1"</font><br />
    <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>
  </body>
</html>
```

This will produce the following result:

| | |
|------|-----------|
| Font | size="-1" |
| Font | size="+1" |
| Font | size="+2" |
| Font | size="+3" |
| Font | size="+4" |

Setting Font Face

You can set font face using *face* attribute but be aware that if the user viewing the page doesn't have the font installed, they will not be able to see it. Instead user will see the default font face applicable to the user's computer.

```

<!DOCTYPE html>
<html>
  <head>
    <title>Font Face</title>
  </head>
  <body>
    <font face="Times New Roman" size="5">Times New Roman</font><br />
    <font face="Verdana" size="5">Verdana</font><br />
    <font face="Comic sans MS" size="5">Comic Sans MS</font><br />
    <font face="WildWest" size="5">WildWest</font><br />
    <font face="Bedrock" size="5">Bedrock</font><br />
  </body>
</html>

```

This will produce the following result:

Times New Roman
 Verdana
 Comic Sans MS
 WildWest
 Bedrock



Specify alternate font faces

A visitor will only be able to see your font if they have that font installed on their computer. So, it is possible to specify two or more font face alternatives by listing the font face names, separated by a comma.

```
<font face="arial,helvetica">
```

```
<font face="Lucida Calligraphy,Comic Sans MS,Lucida Console">
```

When your page is loaded, their browser will display the first font face available. If none of the given fonts are installed, then it will display the default font face *Times New Roman*.

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

The <basefont> Element:

The <basefont> element is supposed to set a default font size, color, and typeface for any parts of the document that are not otherwise contained within a tag. You can use the elements to override the <basefont> settings.

The <basefont> tag also takes color, size and face attributes and it will support relative font setting by giving size a value of +1 for a size larger or -2 for two sizes smaller.

```

<!DOCTYPE html>
<html>
<head>
    <title>Setting Basefont Color</title>
</head>
<body>
    <basefont face="arial, verdana, sans-serif" size="2" color="#ff0000">
    <p>This is the page's default font.</p>
    <h2>Example of the &lt;basefont&gt; Element</h2>
    <p><font size="+2" color="darkgray">
    This is darkgray text with two sizes larger
    </font></p>
    <p><font face="courier" size="-1" color="#000000">
    It is a courier font, a size smaller and black in color.
    </font></p>
</body>
</html>

```

CHAPTER EIGHTEEN

HTML – FORMS

HTML Forms are required when data is to be collected 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 server-side processing script such as ASP Script or PHP script etc. This script will then 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 the 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:

| Attribute | Description |
|-----------|--|
| action | Backend script ready to process your passed data. |
| method | Method to be used to upload data. The most frequently used are GET and POST methods. |
| target | Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc. |
| 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.
- **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.
- **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.

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>

```

Result:

First name

Last name:

Attributes

| Attribute | Description |
|-----------|--|
| | |
| type | Indicates the type of input control and for password input control it will be set to password. |
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| value | This can be used to provide an initial value inside the control. |
| size | Allows to specify the width of the text-input control in terms of characters. |
| 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 but type attribute is set to **password**.

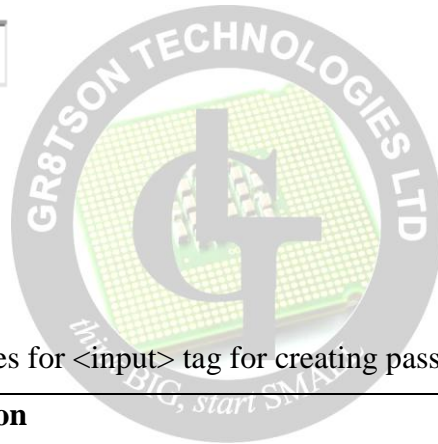
An example of a single-line password input used to take user password is as shown below:

```
<!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>
```

Result:

User ID:

Password:



Attributes

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

| Attribute | Description |
|-----------|--|
| type | Indicates the type of input control and for password input control it will be set to password. |
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| value | This can be used to provide an initial value inside the control. |
| size | Allows to specify the width of the text-input control in terms of characters. |
| maxlength | Allows to specify the maximum number of characters a user can enter into the text box. |

Multiple-Line Text Input Controls

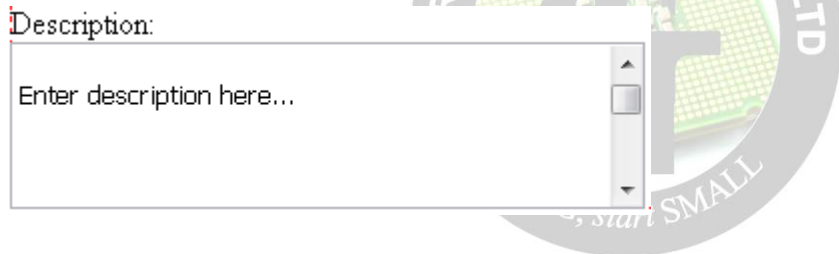
This is used when the user is required to give details that may be longer than a single sentence.

Multi-line input controls are created using HTML <textarea> tag.

An example of a multi-line text input used to take item description is as shown below:

```
<!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>
```

Result:



Description:
Enter description here...

Attributes

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

| Attribute | Description |
|-----------|--|
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| rows | Indicates the number of rows of text area box. |
| 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**.

An example HTML code for a form with two checkboxes is as shown below:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Checkbox Control</title>
  </head>
  <body>
    <form>
      <input type="checkbox" name="maths" value="Maths"> Maths
      <input type="checkbox" name="physics" value="Physics"> Physics
    </form>
  </body>
</html>
```

Result:



Attributes

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

| Attribute | Description |
|-----------|--|
| type | Indicates the type of input control and for checkbox which is checkbox |
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| value | The value that will be used if the checkbox is selected. |
| checked | Set to <i>checked</i> 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**

An example HTML code for a form with two radio buttons is as shown below:

```
<!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>
```

Result:



Attributes

Following is the list of attributes for radio button

| Attribute | Description |
|-----------|--|
| type | Indicates the type of input control and for radio button input control it will be set to radio. |
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| value | The value that will be used if the radio box is selected. |
| checked | Set to <i>checked</i> 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.

An example HTML code for a form with one drop down box is as shown below:

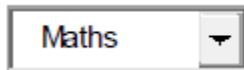
```
<!DOCTYPE html>
<html>
  <head>
```

```

        <title>Select Box Control</title>
    </head>
    <body>
        <form>
            <select name="courses">
                <option value="Maths" selected>Maths</option>
                <option value="Physics">Physics</option>
            </select>
        </form>
    </body>
</html>

```

Result:



Attributes

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

| Attribute | Description |
|-----------|--|
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| size | This can be used to present a scrolling list box. |
| 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:

| Attribute | Description |
|-----------|--|
| value | The value that will be used if an option in the select box is selected. |
| selected | Specifies that this option should be the initially selected value when the page loads. |
| 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**.

An example HTML code for a form with one file upload box is as shown below:

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

Result:



Attributes

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

| Attribute | Description |
|-----------|--|
| name | Used to give a name to the control which is sent to the server to be recognized and get the value. |
| 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:

| Type | Description |
|--------|--|
| submit | This creates a button that automatically submits a form. |
| reset | This creates a button that automatically resets form controls to |

| | |
|--------|--|
| | their initial values. |
| button | This creates a button that is used to trigger a client-side script when the user clicks that button. |
| image | This creates a clickable button but we can use an image as background of the button. |

An example HTML code for a form with three types of buttons is as shown below:

```
<!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="images/logo.png" />
    </form>
  </body>
</html>
```

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.

An example HTML code to show the usage of hidden control is as shown below:

```
<!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>
```

Result:

This is page 10

| | |
|---------------------------------------|-------------------------------------|
| <input type="submit" value="Submit"/> | <input type="reset" value="Reset"/> |
|---------------------------------------|-------------------------------------|



CHAPTER NINETEEN

HTML – EMBED MULTIMEDIA

Sometimes you need to add music or video into your web page. The easiest way to add video or sound to your web site is to include the special HTML tag called **<embed>**. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports **<embed>** tag and given media type.

You can also include a **<noembed>** tag for the browsers which don't recognize the **<embed>** tag. You could, for example, use **<embed>** to display a movie of your choice, and **<noembed>** to display a single JPG image if browser does not support **<embed>** tag.

Example:

```
DOCTYPE html>
<html>
<head>
    <title>HTML embed Tag</title>
</head>
<body>
    <embed src="/html/yourfile.mid" width="100%" height="60" >
    <noembed></noembed>
</embed>
</body>
</html>
```

Result:



You can put any media file of your choice in the source attribute and try it.

The **<embed>** Tag Attributes

| Following is the list of important attributes which can be used with <embed> tag. Attribute | Description |
|--|---|
| align | Determines how to align the object. It can be set to either <i>center</i> , <i>left</i> or <i>right</i> . |
| autostart | This boolean attribute indicates if the media should start automatically. You can set it either true or false. |
| loop | Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false) |

| | |
|-----------|---|
| playcount | Specifies the number of times to play the sound. This is alternate option for <i>loop</i> if you are using IE. |
| hidden | Specifies if the multimedia object should be shown on the page. A false value means no and true values means yes. |
| width | Width of the object in pixels |
| height | Height of the object in pixels |
| name | A name used to reference the object. |
| src | URL of the object to be embedded. |
| volume | Controls volume of the sound. Can be from 0 (off) to 100 (full volume). |

Supported Video Types

You can use various media types like Flash movies (.swf), AVI's (.avi), and MOV's (.mov) file types inside embed tag.

- i. .swf files - are the file types created by Macromedia's Flash program.
- ii. .wmv files - are Microsoft's Window's Media Video file types.
- iii. .mov files - are Apple's Quick Time Movie format.
- iv. .mpeg files - are movie files created by the Moving Pictures Expert Group.

Background Audio

You can use HTML **<bgsound>** tag to play a soundtrack in the background of your webpage. This tag is supported by Internet Explorer only and most of the other browsers ignore this tag. It downloads and plays an audio file when the host document is first downloaded by the user and displayed. The background sound file also will replay whenever the user refreshes the browser. This tag is having only two attributes *loop* and *src*. Both these attributes have same meaning as explained above.

Here is a simple example to play a small midi file:


```

<!DOCTYPE html>
<html>
<head>
    <title>HTML embed Tag</title>
</head>
<body>
    <bgsound src="/html/yourfile.mid">
    <noembed></noembed>
    </bgsound>
</body>
</html>

```

This file plays an audio with the option to download

```

<!DOCTYPE html>
<html>
<head>
    <title>HTML audio Tag</title>
</head>
<body>
    <audio id="Test_Audio" controls>
        <source src="media/Sleep Away.mp3" type="audio/ogg">
    </audio>
</body>
</html>

```

HTML Object tag

HTML 4 introduces the **<object>** element, which offers an all-purpose solution to generic object inclusion. The **<object>** element allows HTML authors to specify everything required by an object for its presentation by a user agent.

Example

We can use this option to include a PDF material. In the example below, the PDF page displays in the html file when it loads.

```

<!DOCTYPE html>
<html>

```

```
<head>
<title>HTML PDF inclusion Tag</title>
</head>
<body>
    <object data="data/Gr8tson.pdf" type="application/pdf" width="500" height="400">
        alt : <a href="data/Gr8tson.pdf">Gr8tson PROGRAMMING PDF MATERIAL</a>
    </object>
</body>
</html>
```



CHAPTER TWENTY

HTML – MARQUEES

An HTML marquee is a scrolling piece of text displayed either horizontally across or vertically down your webpage depending on the settings. This is created by using HTML <marquees> tag.

Note: The HTML <marquee> tag may not be supported by various browsers so it is not recommended to rely on this tag, instead you can use JavaScript and CSS to create such effects.

Syntax

A simple syntax to use HTML <marquee> tag is as follows:

```
<marquee attribute_name="attribute_value"....more attributes>
```

One or more lines or text message or image

```
</marquee>
```

The <marquee> Tag Attributes

| Following is the list of important attributes which can be used with <marquee> tag. Attribute | Description |
|---|---|
| width | This specifies the width of the marquee. This can be a value like 10 or 20% etc. |
| height | This specifies the height of the marquee. This can be a value like 10 or 20% etc. |
| direction | This specifies the direction in which marquee should scroll. This can be a value like <i>up</i> , <i>down</i> , <i>left</i> or <i>right</i> . |
| behavior | This specifies the type of scrolling of the marquee. This can have a value like <i>scroll</i> , <i>slide</i> and <i>alternate</i> . |
| scrolldelay | This specifies how long to delay between each jump. This will have a value like 10 etc. |
| scrollamount | This specifies the speed of marquee text. This can have a value like 10 etc. |
| loop | This specifies how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly. |
| bgcolor | This specifies background color in terms of color name or color hex value. |
| hspace | This specifies horizontal space around the marquee. This can be a value like 10 or 20% etc. |
| vspace | This specifies vertical space around the marquee. This can be a value like 10 or 20% etc. |

Examples of Marquee

The Marquee

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>HTML marquee Tag</title>
```

```
</head>
```

```
<body>
```

```
    <marquee width="30%" direction="up">This example will take only 50%  
    width</marquee>
```

```
</body>
```

```
</html>
```



CHAPTER TWENTY ONE

HTML – HEADER

We have learnt that a typical HTML document will have following structure:

Document declaration tag

```
<html>
```

```
<head>
```

Document header related tags

```
</head>
```

```
<body>
```

Document body related tags

```
</body>
```

```
</html>
```

This chapter will give a little more detail about header part which is represented by HTML

`<head>` tag. The `<head>` tag is a container of various important tags like `<title>`, `<meta>`, `<link>`, `<base>`, `<style>`, `<script>`, and `<noscript>` tags.

The HTML `<title>` Tag

The HTML `<title>` tag is used for specifying the title of the HTML document. Following is an example to give a title to an HTML document:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>HTML Title Tag Example</title>
```

```
</head>
```

```
<body>
```

```
<p>Hello, World!</p>
```

```
</body>
```

```
</html>
```

The HTML `<meta>` Tag

The HTML `<meta>` tag is used to provide metadata about the HTML document which includes information about page expiry, page author, list of keywords, page description etc.

Following are few of the important usages of `<meta>` tag inside an HTML document:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```

<title>HTML Meta Tag Example</title>

<!-- Provide list of keywords -->
<meta name="keywords" content="C, C++, Java, PHP, Perl, Python">

<!-- Provide description of the page -->
<meta name="description" content="Simply Easy Learning by Tutorials Point">

<!-- Author information -->
<meta name="author" content="Tutorials Point">

<!-- Page content type -->
<meta http-equiv="content-type" content="text/html; charset=UTF-8">

<!-- Page refreshing delay -->
<meta http-equiv="refresh" content="30">

<!-- Page expiry -->
<meta http-equiv="expires" content="Wed, 21 June 2006 14:25:27 GMT">

<!-- Tag to tell robots not to index the content of a page -->
<meta name="robots" content="noindex, nofollow">
</head>
<body>
<p>Hello, World!</p>
</body>
</html>

```

The HTML <base> Tag

The HTML <base> tag is used for specifying the base URL for all relative URLs in a page, which means all the other URLs will be concatenated into base URL while locating for the given item.

For example, all the given pages and images will be searched after prefixing the given URLs with base URL <http://www.tutorialspoint.com/> directory:

```

<!DOCTYPE html>
<html>
<head>
    <title>HTML Base Tag Example</title>
    <base href="http://www.tutorialspoint.com/" />
</head>
<body>
    
    <a href="/html/index.htm" title="HTML Tutorial">HTML
    Tutorial</a>
</body>

```

```
</html>
```

The HTML <link> Tag

The HTML <link> tag is used to specify relationships between the current document and external resource. Following is an example to link an external style sheet file available in **css** sub-directory within web root:

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML link Tag Example</title>
    <base href="http://www.tutorialspoint.com/" />
    <link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
    <p>Hello, World!</p>
</body>
</html>
```

The HTML <style> Tag

The HTML <style> tag is used to specify style sheet for the current HTML document. Following is an example to define few style sheet rules inside <style> tag:

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML style Tag Example</title>
    <base href="http://www.tutorialspoint.com/" />
    <style type="text/css">
        .myclass{
            background-color: #aaa;
            padding: 10px;
        }
    </style>
</head>
<body>
    <p class="myclass">Hello, World!</p>
</body>
</html>
```

Note: To learn about how Cascading Style Sheet works, kindly check a separate tutorial available at <http://www.tutorialspoint.com/css>

The HTML <script> Tag

The HTML <script> tag is used to include either external script file or to define internal script for the HTML document. Following is an example where we are using JavaScript to define a simple JavaScript function:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML script Tag Example</title>
  <base href="http://www.tutorialspoint.com/" />
  <script type="text/JavaScript">
    function Hello(){
      alert("Hello, World");
    }
  </script>
</head>
<body>
  <input type="button" onclick="Hello();" name="ok" value="OK" />
</body>
</html>
```

This will produce a button on the page such that once the button is click, it displays Hello, World

To learn about how JavaScript works, kindly check a separate tutorial available at: <http://www.tutorialspoint.com/javascript>

CHAPTER TWENTY TWO

HTML – STYLE SHEET

CSS, short for cascading style sheets, is the look-and-feel for HTML content. With CSS, you can change how text, images, and links appear quickly and easily, on a single web page or across an entire site — and what's more, the content's appearance can change based on the medium presenting it. CSS is a powerful technology, tightly intertwined with HTML in the building of modern websites. In this lesson, you learn the basics of CSS, including key concepts, where to store your CSS rules, and how to work with primary selectors.

UNDERSTANDING CASCADING STYLE SHEETS

Before CSS gained popularity, HTML pages were styled with tag attributes. For example, if you wanted to make a particular heading red, your tag would look like this:

```
<h1 color="red">Listen Up!</h1>
```

The problem with this approach is that the styling of the content is very tightly tied to the content itself. Though changing a single tag is easy enough, what if your design called for all `<h1>` tags to be red? If your color scheme changed so that every heading needed to be blue, you'd have to update every tag, one at a time. Using CSS instead of tags for styling allows you to easily make general (global) formatting changes. This approach brings numerous benefits, including:

Ease of modification

With CSS, you can style all the `<h1>` tags — or any other tags or custom selected content — in an entire site by changing values in one place.

Advanced design options

Current CSS implementations enable rich background elements, pixel-perfect positioning, and robust padding and margin possibilities. The next generation of CSS, much of which is available today in modern browsers, extends the designer's palette with rounded corners, drop shadows, and gradients, among many other features.

Media targeting

Today's digital content isn't just for the computer screen: you can easily print a web page, view it on your smart phone, or even see it on your TV. CSS makes it possible to change the look-and-feel of your content to suit the output device with different layouts, removal or inclusion of page sections, and a completely different color scheme.

With CSS, web page styles are made up of one or more *rules*. A CSS rule is comprised of three main parts: the selectors, the properties, and the values.

Considering the css rule used above,

h1 is the selector, color is the property, and red is the value.

After the selector, properties and values (collectively referred to as a *declaration*) are enclosed in a set of curly braces. Properties and values are separated by a colon and each declaration must end with a semicolon. You can include multiple declarations for any selector. For example:

```
h1 {  
color: red;  
margin: 0;  
padding: 5px;  
}
```

As with HTML, white space is ignored in CSS, so you can apply line-returns and indentation as needed to make your CSS rules more readable.

Moreover, you can specify multiple selectors for any set of declarations in a comma-separated list:

```
h1, h2, h3, h4 {  
color: red;  
margin: 0;  
padding: 5px;  
}
```

CSS is truly an integral part of modern web page creation and a further understanding of its basic tenets and how it can be used as discussed in the following sections will further your work with HTML.

Key CSS Concepts

To represent CSS rules consistently, browsers follow a set of implementation guidelines that adhere to three main principles in CSS

- Cascading
- Inheritance
- Specificity

The Cascading Principle

The “cascading” in cascading style sheets refers to the idea that, given two identical CSS rules, the one closest to the targeted element wins. For example, say you have the following two rules, one after the other:

```
h1 { color: red; }
```

```
h1 { color: blue; }
```

In this situation, the second rule — with the declaration that the color should be blue — would take effect and the heading would be blue. As you learn later in this lesson, CSS rules can be located in one of three places: an external style sheet, embedded in the <head> of a document, or inline with the affected tag. The cascade concept dictates that in any CSS rule conflict, an embedded rule would beat the one in an external style sheet and an inline rule would beat them both.

The Inheritance Principle

You’ve seen how much of HTML is based on the principle of nested tags, where, for example, all content is within the <body> tag. CSS rules defined to target outer or *parent* tags also affect inner or *child* tags, thanks to the principle of inheritance. For instance, this rule, which uses the

<body>tag as the selector, also sets the font-family property for every other text element on the page unless otherwise specified:

```
body {  
font-family: Verdana, Arial, Helvetica, sans-serif;  
}
```

Many style sheets start with a series of so-called reset statements that rely on the inheritance principle to establish a baseline of values for a wide spectrum of tags.

As you learn later in this lesson, CSS declarations can be applied to more than just tags. You can also create custom selectors, called classes and IDs. You've seen what happens when two rules with identical selectors conflict — thanks to the cascading principle, the one closest to the actual tag overcomes the other — but what happens when two rules with different selectors affect the same tag? For example, say you have one rule that sets the color of <h1> tags to red, like this:

```
h1 { color: red; }
```

Furthermore, say there is a second rule that uses a custom CSS selector called a class with the name

```
.alert:
```

```
.alert { color: purple; }
```

How do you think the browser is supposed to render the following tag?

```
<h1 class="alert">Attention site visitors!</h1>
```

In this situation, the CSS principle of specificity comes into play.

The Specificity Principle

A class selector is regarded as being more specific than that of a tag selector, so, in this example, the text would be purple. The hierarchy of selectors from least to most specific looks like this:

1. Tags
2. Classes
3. IDs

4. Inline styles

I want to take a look at an example to demonstrate how specificity works. Say that you have a page like this:

```
<body>

<div id="content">

<h1 class="mainTopic">When in Doubt, Be Specific!</h1>

</div>

</body>
```

Furthermore, assume you declared a CSS rule that made all h1 tags green, like this:

```
h1 { color: green; }
```

Now, if the client decides he or she wants h1 tags to be green in general, but those that are within a mainTopic class to be red, you could keep your original CSS rule and write another, like this:

```
.mainTopic h1 { color:red; }
```

Because this CSS rule has a higher specificity, the heading in this section would be red. If, for whatever reason, the client decides that this one particular heading has to be purple, you could inject an inline style into the HTML source code:

```
<h1 class="mainTopic" style="color:purple;">When in Doubt, Be Specific!</h1>
```

As noted previously, inline styles are generally frowned upon by web designers because they are difficult to quickly modify. Specificity is a fundamental principle to keep in mind when you're debugging your CSS styles.

Working with CSS Placement

As mentioned earlier, CSS rules can be integrated into an HTML page in a number of ways: as an external style sheet, embedded within the HTML page itself, and inline as an attribute within the tag. This section takes a look at each approach in turn.

External style sheets

External style sheets are used to provide a consistent look-and-feel to any number of related pages, up to and including an entire website. An external style sheet is connected to an HTML page in one of two ways: either with a <link> tag, or with an @import directive within a <style> tag. For example, say you wanted to include the CSS rules written in a file called main.css. The <link> syntax would look like this:

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

The href attribute provides the path to the external style sheet, and type specifies the kind of document the browser can expect. The relationship of the HTML page to the linked file is defined by the rel attribute; the two possible values are stylesheet and alternate stylesheet.

If you wanted to use the @import syntax, you would write code like this:

```
<style>
```

```
@import { url("styles/main.css"); }
```

```
</style>
```

Notice that @import is actually a CSS rule with the single url property, written somewhat differently from standard CSS declarations. When used with an HTML page, the @import rule must be within a <style> tag.

Complex site designs often use the @import rule within an external style sheet to include additional style sheets. When used in an external style sheet, the @import rule does not require a <style>tag.

So, when do you use <link> and when do you use @import? It really is a matter of choice at this point in time. All modern browsers recognize both options. I prefer to use the <link> syntax because it involves a single tag instead of a tag and a rule when associating an external style sheet with an HTML page, and save @import for incorporating additional style sheets into CSS files.

Whichever technique you use, external style sheets have the tremendous advantage of being able to affect multiple HTML pages simultaneously. Change any CSS rule, save the external style sheet, publish it, and immediately the modification can be seen by any site visitor to any of the

associated pages. You can see why external style sheets are widely used by web designers across the industry.

Embedded styles

CSS rules can also be included in an HTML page, typically in the <head> section of the document. This technique is known as *embedding*. CSS rules are embedded through use of the <style> tag, like this:

```
<style type="text/css">

body {
margin: 0;
padding: 0;
background-color: white;
}

h1, h2, h3, h4 {
color: red;
margin: 0;
padding: 5px;
}

</style>
```



As mentioned earlier, if the same CSS rule is both included in an external style sheet and embedded, the embedded rule has precedence. The obvious disadvantage to embedding rather than linking to an external style sheet is that CSS modifications apply only to a single page. Updates to multiple pages with embedded styles require multiple steps.

Inline styles

The final method for styling HTML tags is called *inline* styles. An inline style is applied by use of the style attribute within an HTML tag. For example, if you want to color an <h1> tag red with in an inline style, your code would look this:

```
<h1 style="color:red;">Important Message Ahead</h1>
```

You'll notice the resemblance between the inline style and the pre-CSS technique for changing the look-and-feel of a tag. Not surprisingly, the inline style has the same drawback as the pre-CSS attribute-based method of being difficult to update. For this reason, inline styles are rarely used by designers when creating web pages.

Tags

The use of HTML tags as a CSS selector is very straightforward. When an HTML tag, such as `<p>`, is defined as a selector with CSS, all `<p>` tags are immediately affected unless another CSS style overrules it. With tag selectors, it is easy to implement broad modifications to existing web pages. This ability is both a blessing and a curse. You'll need to make sure that any tag styles created work well in all page variations.

IDs

A CSS ID is a custom selector intended for use once per HTML page. To define an ID selector, use a leading number sign symbol, like this:

```
#id_name {  
width: 960px;  
}
```

e.g

```
#header {  
width: 960px;  
}
```

An ID is applied to an HTML tag with the ID attribute:

```
<div id="header">
```

Note that the # symbol is only used when defining the CSS rule, not when applying it.

Classes



The class selector is similar to the ID, except it may be used multiple times on a single page. Additionally, instead of a leading number sign, a period is used to define a class selector, like this:

```
.courses {  
font-size: small;  
}
```

To apply the class selector to an HTML tag, use the class attribute:

```
<div class="courses">
```

The names of classes and IDs must begin with a letter and not contain any white spaces or other special characters. Similarly, classes and IDs are case-sensitive. In other words, `.courses` is not the same as `. courses`.

Compound selectors

It is often beneficial to limit CSS rules to a tightly defined section of the page. Rather than create a specific ID or class for such sections, designers often opt for a compound selector that targets the area contextually.

Say, for example, that you need to make all `<h1>` tags in the sidebar green. Instead of creating and applying a series of classes, you can define a compound selector, like this:

```
#sidebar h1 {  
  
color: green;  
  
}
```

This selector will apply to any `<h1>` tag within any HTML element with an ID of sidebar. Compound selectors can utilize any combination of tags, IDs, and classes. You'll be using CSS - with a full range of selectors, properties, and values — throughout the book to help you better understand how HTML5 tags are used to create coherent web pages.

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 (;).

Let's consider an example of HTML document which makes use of 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>
```

Result:

Hello, World!

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:green;
}
```

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="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>
```

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.

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:green;
      }
      .thick green{
        color:green;
        font-size:20px;
      }
    </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>
</html>
```

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

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>
```

Result:

This is red

This is thick

This is green

This is thick and green

CHAPTER TWENTY THREE

HTML JAVASCRIPT

A **script** is a small piece of program that can add interactivity to your website. For example, a script could generate a pop-up alert box message, or provide a dropdown menu. This script could be written using JavaScript or VBScript.

You can write various small functions, called event handlers using any of the scripting language and then you can trigger those functions using HTML attributes.

Now-a-days, only **JavaScript** and associated frameworks are being used by most of the web developers, VBScript is not even supported by various major browsers.

You can keep JavaScript code in a separate file and then include it wherever it's needed, or you can define functionality inside HTML document itself. Let's see both the cases one by one with suitable examples.

External JavaScript

If you are going to define a functionality which will be used in various HTML documents then it's better to keep that functionality in a separate JavaScript file and then include that file in your HTML documents. A JavaScript file will have extension as **.js** and it will be included in HTML files using `<script>` tag.

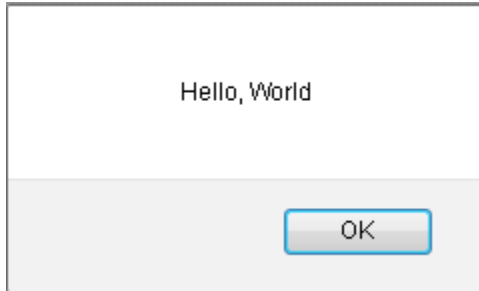
Consider we define a small function using JavaScript in **script.js** which has following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript External Script</title>
    <script src="script.js" type="text/JavaScript"/></script>
  </head>
  <body>
    <input type="button" onclick="Hello();" name="ok" value="Click Me" />
  </body>
</html>
```

Result:



After clicking the button, the following interface will appear requesting you to click OK.



Internal Script

You can write your script code directly into your HTML document. Usually we keep script code in header of the document using `<script>` tag, otherwise there is no restriction and you can put your source code anywhere in the document but inside `<script>` tag.

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript Internal Script</title>

    <script type="text/JavaScript">

      function Hello(){
        alert("Hello, World");
      }
    </script>
  </head>
  <body>
    <input type="button" onclick="Hello();" name="ok" value="Click Me" />
  </body>
</html>
```

Event Handlers

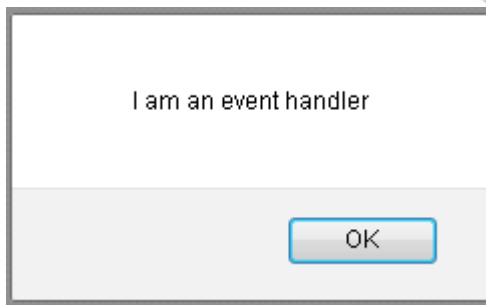
Event handlers are nothing but simply defined functions which can be called against any mouse or keyboard event. You can define your business logic inside your event handler which can vary from a single to 1000s of line code.

Following example explains how to write an event handler. Let's write one simple function `EventHandler()` in the header of the document. We will call this function when any user brings mouse over a paragraph.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Event Handlers Example</title>

    <script type="text/JavaScript">
      function EventHandler(){
        alert("I'm event handler!!");
      }
    </script>
  </head>
  <body>
    <p onmouseover="EventHandler();">Bring your mouse here to see how an even
    handler alert works</p>
  </body>
</html>
```

Result:



Hide Scripts from Older Browsers

Although most (if not all) browsers these days support JavaScript, but still some older browsers don't. If a browser doesn't support JavaScript, instead of running your script, it would display the code to the user. To prevent this, you can simply place HTML comments around the script as shown below.

JavaScript Example:


```
<script type="text/JavaScript">
    <!--
    document.write("Hello JavaScript!");
    //-->
</script>
```

The <noscript> Element

You can also provide alternative info to the users whose browsers don't support scripts and for those users who have disabled script option their browsers. You can do this using the <noscript> tag.

JavaScript Example:

```
<script type="text/JavaScript">
<!--
document.write("Hello JavaScript!");
//-->
</script>
<noscript>Your browser does not support JavaScript!</noscript>
```

Default Scripting Language

There may be a situation when you will include multiple script files and ultimately using multiple <script> tags. You can specify a default scripting language for all your *script* tags.

This saves you from specifying the language every time you use a script tag within the page.

Below is the example:

```
<meta http-equiv="Content-Script-Type" content="text/JavaScript" />
```

Note that you can still override the default by specifying a language within the script tag.

CHAPTER TWENTY FOUR

HTML – LAYOUTS

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.

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 Tutorialspoint.com
        </center>
      </td>
    </tr>
  </table>
```

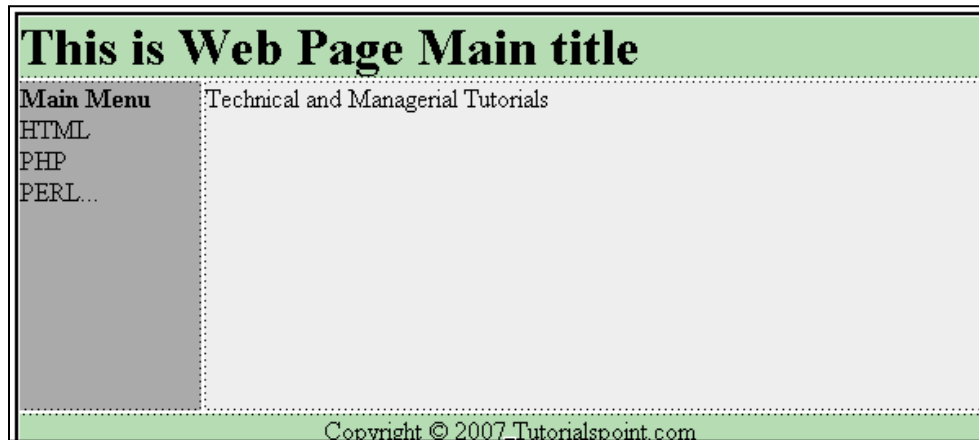
```

        </td>
      </tr>

    </table>
  </body>
</html>

```

Result:



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.

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>

```

```

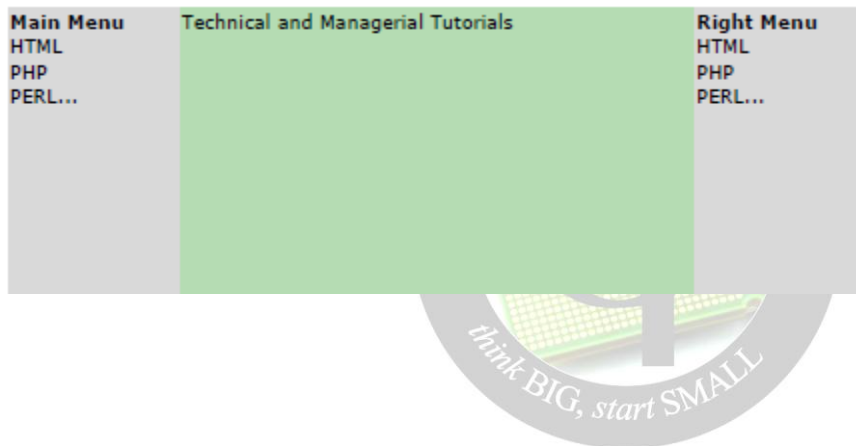
        <td bgcolor="#b5dcb3" height="200" width="60%">
        Technical and Managerial Tutorials
        </td>

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

    </tr>
</table>
</body>
</html>

```

Result:



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 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.

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>

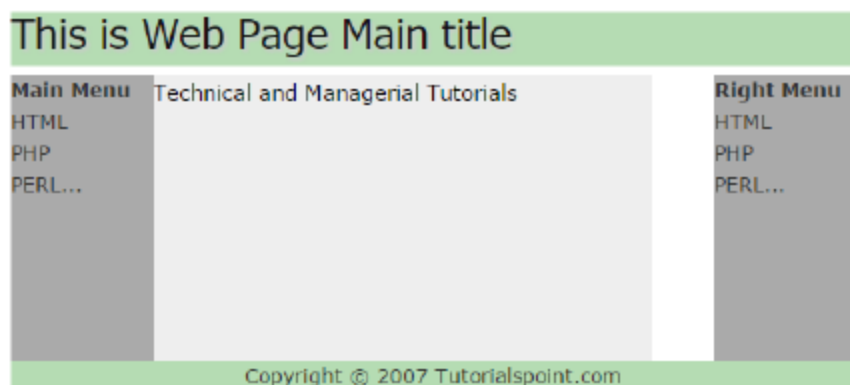
      <div style="background-color:#aaa; height:200px;width:100px;float:left;">
        <div><b>Main Menu</b></div>
        HTML<br />
        PHP<br />
        PERL...
      </div>

      <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 Tutorialspoint.com
        </center>
      </div>
    </div>
  </body>
</html>

```



You can create better layout using DIV, SPAN along with CSS. For more information on CSS, please refer to CSS Tutorial.



CHAPTER TWENTY FIVE

HTML EVENTS REFERENCE

When users visit your website, they do things like click various links, bring mouse over text and images etc. These are examples of what we call events in JavaScript and VBScript terminologies.

We can write our event handlers using JavaScript or VBScript and can specify some actions to be taken against these events. Though these are the events but they will be specified as attributes for the HTML tags.

Window Events Attributes

HTML EVENTS REFERENCE

| Events | Description |
|----------------|---|
| onafterprint | Triggers after a document is printed |
| onbeforeprint | Triggers before a document is printed |
| onbeforeunload | Triggers before a document loads |
| onerror | Triggers when an error occurs |
| onhaschange | Triggers when a document has changed |
| onload | Triggers when a document loads |
| onmessage | Triggers when a message is triggered |
| onoffline | Triggers when a document goes offline |
| ononline | Triggers when a document comes online |
| onpagehide | Triggers when a window is hidden |
| onpageshow | Triggers when a window becomes visible |
| onpopstate | Triggers when a window's history changes |
| onredo | Triggers when a document performs a redo |
| onresize | Triggers when a window is resized |
| onstorage | Triggers when a document loads |
| onundo | Triggers when a document performs an undo |
| onunload | Triggers when a user leaves the document |

Form Events

| Events | Description |
|---------------|---|
| onblur | Triggers when a window loses focus |
| onchange | Triggers when an element changes |
| oncontextmenu | Triggers when a context menu is triggered |
| onfocus | Triggers when a window gets focus |
| onformchange | Triggers when a form changes |
| onforminput | Triggers when a form gets user input |
| oninput | Triggers when an element gets user input |

| | |
|-----------|--------------------------------------|
| oninvalid | Triggers when an element is invalid |
| onreset | Triggers when a form is reset |
| onselect | Triggers when an element is selected |
| onsubmit | Triggers when a form is submitted |

Keyboard Events

| Events | Description |
|-------------------|---|
| onkeydown | Triggers when a key is pressed |
| onkeypress | Triggers when a key is pressed and released |
| onkeyup | Triggers when a key is released |

Mouse Events

| Events | Description |
|--------------|--|
| onclick | Triggers on a mouse click |
| ondblclick | Triggers on a mouse double-click |
| ondrag | Triggers when an element is dragged |
| ondragend | Triggers at the end of a drag operation |
| ondragenter | Triggers when an element has been dragged to a valid drop target |
| ondragleave | Triggers when an element leaves a valid drop target |
| ondragover | Triggers when an element is being dragged over a valid drop target |
| ondragstart | Triggers at the start of a drag operation |
| ondrop | Triggers when a dragged element is being dropped |
| onmousedown | Triggers when a mouse button is pressed |
| onmousemove | Triggers when the mouse pointer moves |
| onmouseout | Triggers when the mouse pointer moves out of an element |
| onmouseover | Triggers when the mouse pointer moves over an element |
| onmouseup | Triggers when a mouse button is released |
| onmousewheel | Triggers when the mouse wheel is being rotated |
| onscroll | Triggers when an element's scrollbar is being scrolled |

Media Events

| | |
|------------------|--|
| onabort | Triggers on an abort event |
| oncanplay | Triggers when a media can start play, but might has to stop for buffering |
| oncanplaythrough | Triggers when a media can be played to the end, without stopping for buffering |
| ondurationchange | Triggers when the length of a media is changed |
| onemptied | Triggers when a media resource element suddenly becomes empty. |
| onended | Triggers when a media has reached the end |
| onerror | Triggers when an error occurs |
| onloadeddata | Triggers when media data is loaded |
| onloadedmetadata | Triggers when the duration and other media data of a media element is loaded |

| | |
|--------------------|--|
| onloadstart | Triggers when the browser starts loading the media data |
| onpause | Triggers when media data is paused |
| onplay | Triggers when media data is going to start playing |
| onplaying | Triggers when media data has started playing |
| onprogress | Triggers when the browser is fetching the media data |
| onratechange | Triggers when the playing rate of media data has changed |
| onreadystatechange | Triggers when the ready-state changes |
| onseeked | Triggers when the seeking attribute of a media element is no longer true, and the seeking has ended |
| onseeking | Triggers when the seeking attribute of a media element is true, and the seeking has begun |
| onstalled | Triggers when there is an error in fetching media data |
| onsuspend | Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched |
| ontimeupdate | Triggers when media changes its playing position |
| onvolumechange | Triggers when a media changes the volume, also when volume is set to "mute" |
| onwaiting | Triggers when media has stopped playing, but is expected to resume |



CHAPTER TWENTY FIVE

HTML FONTS REFERENCE

Fonts are specific to platform. You will have different look and feel of a web page on different machines running different operating systems like Windows, Linux or Mac iOS. Here we are giving a list of fonts which are available in various operating systems.

HTML tag is deprecated in version 4.0 onwards and now all fonts are set by using CSS.

Here is the simple syntax of setting font of a body of web page.

```
body { font-family: "new century schoolbook"; }
```

or

```
<body style="font-family:new century schoolbook;">
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Font Setting Using CSS</title>
```

```
</head>
```

```
<body>
```

```
    <p>Change any of the style and try it.</p>
```

```
    <div style="font-family:verdana;">This is demo for font family</div>
```

```
    <br />
```

```
    <div style="font-size:120%;">This is demo for font size</div>
```

```
    <br />
```

```
    <div style="font-size:14pt;">This is demo for font size</div>
```

```
</body>
```

```
</html>
```

Fonts for Microsoft Systems

Following is the list of fonts supported by most Microsoft System variants

| Font | Font | Font |
|---------------------|--------------------|-------------------------|
| Andale Mono | Arial | Arial Bold |
| Arial Italic | Arial Bold Italic | Arial Black |
| Comic Sans MS | Comic Sans MS Bold | Courier New |
| Courier New Bold | Courier New Italic | Courier New Bold Italic |
| Georgia | Georgia Bold | Georgia Italic |
| Georgia Bold Italic | Impact | Lucida Console |

| | | |
|--------------------------|-----------------------------|----------------------|
| Lucida Sans Unicode | Marlett | Minion Web |
| Symbol | Times New Roman | Times New Roman Bold |
| Times New Roman Italic | Times New Roman Bold Italic | Tahoma |
| Trebuchet MS | Trebuchet MS Bold | Trebuchet MS Italic |
| Trebuchet MS Bold Italic | Verdana | Verdana Bold |
| Verdana Italic | Verdana Bold Italic | Webdings |

Fonts for Macintosh Systems

Following is the list of fonts supported by Macintosh System 7 and higher versions

| Font | Font | Font |
|------------------------|---------------|-----------------|
| American Typewriter | Andale Mono | Apple Chancery |
| Arial | Arial Black | Brush Script |
| Baskerville | Big Caslon | Comic Sans MS |
| Copperplate | Courier New | Gill Sans |
| Futura | Herculanum | Impact |
| Lucida Grande | Marker Felt | Optima |
| Trebuchet MS | Verdana | Webdings |
| Palatino | Symbol | Times |
| Osaka | Papyrus | Times New Roman |
| Textile | Zapf Dingbats | Zapfino |
| Techno | Hoefler Text | Skia |
| Hoefler Text Ornaments | Capitals | Charcoal |
| Gadget | Sand | |

Fonts for Unix Systems

Following is the list of fonts supported by most Unix System variants

| Font | Font | Font |
|---------------|-------------------|------------------------|
| Charter | Clean | Courier |
| Fixed | Helvetica | Lucida |
| Lucida bright | Lucida Typewriter | New Century Schoolbook |
| Symbol | Terminal | Times |
| Utopia | | |



You Can Also Acquire More Skills In:

***QUICKBOOKS**

***DATABASE DESIGN & MANAGMENT**

***NETWORKING**

***PROGRAMMING**

***GRAPHICS DESIGN**

***COMPUTER ENGINEERING**

*** COMPUTER APPRECIATION**

***SEMINARS**

*** WEB DESIGN & HOSTING**

***BULK SMS**

***RECHARGE CARD PRINTING**

***FOREIGN LANGUAGES**

***MOBILE APPS DEVELOPMENT**

***POS SOFTWARE (for supermarket/Pharmacy)**

***PAYROLL SOFTWARE**

***SOLAR ENERGY TRAINING & INSTALLATION**

***SCHOOL MANAGEMENT SOFTWARE**

***ICT CLUB ACTIVITIES**

***COOPERATIVE SOCIETY SOFTWARE**

07089059571

Email: gr8tsontechnologies@gmail.com, jamesmygreatson@gmail.com

Web: www.gr8tson.com.ng