IT614 Responsive Web Development with HTML5, CSS and JQuery

Chapter 1: Getting Started with HTML 5

1. Hypertext Markup Language

When you read a page of text, normally you read from top down line by line. A hypertext document is a text which you don't necessarily read top down line by line. In the middle of the page you can jump to some other page and then come back to the original page and continue reading. The web pages are called hypertext documents.

Hypertext Markup Language, HTML for short, is a Language that web browsers understand and interpret. HTML is not a programming language like Java or C. You embed HTML codes within your text and a Web browser reads the file interprets these codes. A page, which contains HTML code is called an HTML document. An HTML document file name must have the extension html or htm. You just need a regular text editor to create HTML pages.

HTML5

The current HTML version is HTML5. The previous version of HTML, HTML 4.01, came in 1999. HTML5 has lot more features than the previous versions. In addition to being a much improved version of HTML 4.01, HTML5, by design, works well with CSS3 and JavaScript API. A study of HTML5, therefore, is not complete without a study of CSS3 and JavaScript API. HTML5 provides structure, CSS3 presentation and JavaScript interactivity.

- **Note:**

HTML5 is still evolving. Some of the features may not work in some of the browsers. To see the latest status of HTML 5 go to the source: http://www.w3.org/TR/html-markup. Also see, http://caniuse.com/#cats=HTML5

- **Note:**

If there are syntax errors in a Java or C program, the compiler will identify them. Not so in HTML. When there are syntax errors in an HTML file and open it in a browser, the browser will not identify the syntax errors. The browser renders the page ignoring the mistakes. If you want your HTML file to follow the rules correctly, you need to pass it through a "validator." This online validator provided by W3 is good (and free): http://validator.w3.org. Try it.

Another validator: http://html5.validator.nu/

2. HTML Tags and Elements:

HTML markup codes consist of tags. There are two kinds of tags: start tag and end tag. A start tag consists of a tag name, sometimes followed by an optional list of tag attributes, placed between opening and closing brackets (< and >). The end tag corresponding to a start tag will have the same tag name, enclosed between </ and >. The end tag will have a slash (/) before the name. The entire structure from the start tag to the end tag is called an HTML element.

A typical format of an HTML element is,
<tagName attributes> element text </tagName>

Example:
<p>New York City</p>

Note:

- Some elements will not have any content between tags :<*tagName* … ></*tagName*>. Such an element is called an empty element. In such cases, the element can be represented using only one tag: <*tagName* …/>. Such a tag is called a self-closing tag.
- Tag names are case insensitive. They could be written in all uppercase, or all lowercase or mixed case.
- Within each tag, whitespace is permitted after the tag name, but it is not before the tag name.
- If there are syntax errors in tags, they are ignored by browsers. The text, however, is just displayed.
- Not all browsers support all tags. If a browser does not understand a tag, it just ignores the tag (but displays the text, if any, in default font).
- When you nest tags, they must not overlap - one must be completely inside the other.

Void elements

Some empty elements are specified by a start tag only – no end tag.

Examples of void elements:

Tag	Meaning
**
**	Line break.
<hr>	Horizontal rule
<link>	The link is for linking to other resources. Details later.
<meta>	Used to provide various types of metadata, such as the application-name or specifying the documents character encoding.

Note: The self-closing tag syntax may be used in writing void elements.

3. Attributes

Tags may have attributes that are used to specify additional information about them. Attributes are added to a tag to provide the browser with more information about how the content should appear or behave. Attributes consist of a name and a value separated by an equals (=) sign, with the value generally surrounded by double quotes. Attributes are placed within a start tag and are separated from the tag name and from each other by whitespace. They must not be specified within an end tag.

- **Example:**

```
<body attribute="example">
    …..
</body>
```

- **Note:**

Don't include numeric values in quotes.

Global Attributes

The following attributes are standard across all HTML 5 tags.

- accesskey
- class
- contenteditable
- contextmenu
- dir
- draggable
- dropzone
- hidden
- id
- inert
- itemid
- itemprop
- itemref
- itemscope
- itemtype
- lang
- spellcheck
- style
- tabindex
- title
- translate

We will discuss global attributes later.

Some commonly used important attributes:

class – This attribute provides a mechanism to group similar elements.

Id – This attributes assigns a unique identifier. JavaScripts and CSS use the id value of to refer to the element.

name – This is a unique name given to an element. This name is used by server side programs to access the value of the element.

style – This element is used to assign CSS styles to the element.

We will discuss these attributes in details when we use them.

4. The general structure of an HTML page

```
<!DOCTYPE html>
<html>
   <head>
        <meta charaset="utf-8">
        <title>A title for the page</title>
   </head>
   <body>
        Other HTML tags and content
   </body>
</html>
```

The Document Type Declaration, **<!DOCTYPE html>** needs to be present at the beginning of a document that uses the HTML5 syntax. This line must be the first line in every HTML document – even a space is not permitted before this.

Two commonly used tags within **<head>…</head>** are **<meta>** and **<title>…</title>.**

Note: In all my examples, I use the above template. The code in the examples will not show <!DOCTYPE html>, etc. Before trying hands-on, use the above template and include the code in the examples between **<body>** and **</body>.**

- **The <meta> tag**

The **<meta>** tag is used for several purposes:

<meta name="keywords" content="*several keywords separated by commas*" >

Search engines used the keywords to create page rankings. Modern browsers have stopped doing this.

<meta charset="utf-8">

This defines the character coding used in the page. UTF-8 is a variable width Unicode format that is compatible with ASCII or plain text for the basic alphanumeric characters.

- The **<title>…</title> tag**

This used to specify a title for the page. Only one title can be in the document. Many search engines to help classify the document and will be the name which bookmarks and search engine listings display use the title. So be certain the title describes your page well.

---

5. Selected basic tags

- **The <p> tag**

Browsers ignore spaces, tabs and line feeds in the source HTML document. Even if the source HTML document contains line or paragraph breaks, the browsers will combine different lines into one line. The <P> tag produces a line break. The end tag </p> is optimal.

- **The <br> tag**

The HTML <br> tag is used for specifying a line break.

The <br> tag is an empty tag. In other words, it has no end tag.

- **The <hr> tag**

The <hr> tag produces a horizontal rule, across the screen. This is useful if you want some kind of a separation of text in the output. The <hr> tag is an empty tag. In other words, it has no end tag.

- **Heading tags**

Heading tags are used to divide a page into sections. Headings are displayed in a larger and/or bolder font than normal body text.
Tag IDs H1, H2, H3, H4, H5, and H6 define headings of different levels. They are in the decreasing order of size. H1 being the largest (first level) and H6 being the smallest (last level). The heading elements must have end tags.

- **Example**

```
<TITLE> Heading tags example </TITLE>
<H1> School of Computer Science and Information Systems </H1>
<H2> Department of Computer Science </H2>
<H3> Pleasantville </H3>
<H4> (914) 773-3702 </H4>
<H5> Goldstein center </H5>
<H6> Room 308 </H6>
```

- **NOTE:**

A heading element gives a paragraph break before and after.

➢ **Logical style tags**

Tag	**Description**
<em> … </em>	Emphasis - typically italics
<strong>… </strong>	Strong emphasis - typically bold
<cite>…</cite>	Used for indicating a citation. Typically italics. Text enclosed in <cite> tags is intended to represent the title of a work (e.g. a book, a paper, an essay, a poem, a score, a song, a script, a film, a TV show, a game, a sculpture, a painting, a theatre production, a play, an opera, a musical, an exhibition, etc).
<code>…</code>	Used for indicating a piece of code. The code tag surrounds the code being marked up. Typically a monospaced font such as Courier.
<kbd> …</kbd>	Used for indicating the text to be entered by the user.
<samp>…</samp>	Used for indicating sample output from a computer program, script etc. Usually bold-fixed font.
<dfn>…<dfn>	Used for indicating a definition. Usually italics.
<address>…</address>	Author’s identification - typically italics
<blockquote>…</blockquote>	Long quotation - indented in italics
<var>…</var>	Used for indicating an instance of a variable. Typically italics.

- **Note:**

EM and CITE both result in italics.
CODE, KBD, and SAMP all give the same font.
Try them with the browser available to you.

- **Physical style tags**

The rendering of the logical style tags, as I mentioned before, depends on the browser. The following tags, called physical tags, do not depend on the browser. These are expected to work as described.

Tag	**Description**
<b>…</b>	Bold
<i>…</i>	Italics
<u>…</u>	Underlined
<s>…</s>	Strikeout
<sub>…</sub>	Subscript
<sup>…</sup>	Superscript
<pre>…</pre>	Preformatted - retains spaces, tabs, carriage return etc.
<small> … <small>	Used for specifying small font size.

- **The <div> tag**

The <div>…</div> tag is used for defining a section of the document. This helps in applying formatting to a section of the text easily. Details later.

- **The <span> tag**

The HTML <span> tag is used for grouping and applying styles to inline elements.
The difference between the <span> tag and the <div> tag is that the <span> tag is used with inline elements whilst the<div> tag is used with block-level content.

6. Embedding images

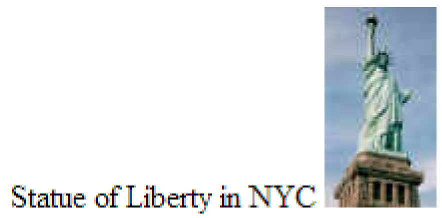
The tag to include an image in an HTML document is **<img>.** This is an empty tag – meaning there is no **</img>** tag.

Syntax:

<img src=”image location”>

- **Example:**

Statue of Liberty in NYC
<img src="NYC1.jpg">

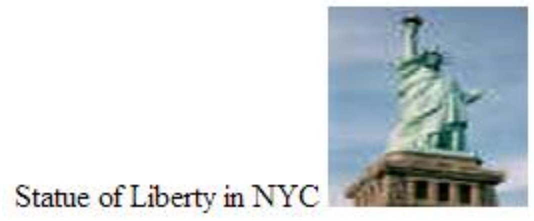

Statue of Liberty in NYC

Other <img> attributes

alt = "*text***"**
Alternate text. The specified text will be displayed if the browser cannot render the image.
width=*w*
Use specified width (in pixels) to render the image.
height=*h*
Use specified height (in pixels) to render the image.

- **Example:**

Statue of Liberty in NYC
<img src="NYC1.jpg" alt="Statue of Liberty" width=100 height=100>


Statue of Liberty in NYC

7. Creating links

From the middle of the current home page, you can make the reader jump to another HTML page, by providing a "link" to that page. The link tag is <a>,

A typical syntax is:
<a href="*target url*"> *linkText* </a>

On the browser screen, *linkText* will displayed underlined, in blue color (default). When the user clicks this text, the indicated page will be loaded and displayed. By default, a link after visiting becomes purple.

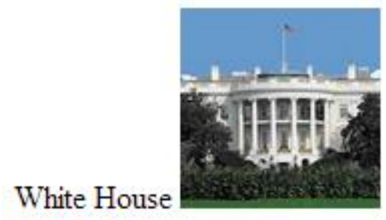
- **Example:**

<a href="http://www.whitehouse.gov">The White House </a>

Image as a link
Instead of a link text, you can include an image, using the **<img>** tag.

- **Example:**

White House
<a href="http://www.whitehouse.gov"><img src="whitehouse.jpg" alt="The White House" width=100 height=100> </a>



Here the image itself is a link.

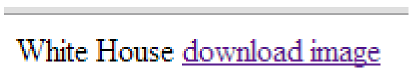
Besides, **href**, the <a> tag supports several attributes:

- **download=***"filename"*

When the user clicks the link, the file will be downloaded.

Example:
White House
<a href="http://www.whitehouse.gov" download="whitehuse.jpg">download image </a>

Output:


White House download image

- **target**=”_blank”

By default when you click a link, the new Web page is loaded into the current window. The attribute **target=”_blank”** opens the new page in a new tab.

- **Example:**

White House
<a href="http://www.whitehouse.gov" target="_blank">White House </a>

8. Lists

HTML includes sets of codes that create several types of preformatted lists. The three most commonly used are:

- ordered (numbered)
- unordered (bulleted)
- definition (formatted so that the item to be defined is left justified and a definition on the following line is indented)

Ordered and unordered lists

The ordered and unordered lists are most commonly used. They work identically except ordered gives you numbered list and unordered gives bulleted list.

<ol> and </ol> mark the start and end of an ordered (numbered) list.
<ul> and </ul> mark the start and end of an unordered (bulleted) list.
<li>: precedes each bulleted or numbered item in either type of list. li stands for List Item. There is no official ending tag for <li>, however, increasingly editors, especially WYSIWYG editors like Dreamweaver and others are including </li> since they need to define the end of a list item. The browsers pay no attention to the </li>.

- **Note:**

Since <ul> and <ol> result in an indentation from the left the included text item, they can also be used for indenting. If there is no <li> to supply a bullet or number, only the indenting will show. Beware, however, that all lists add a blank line above and below the list.

The <ol> tag

The OL tag is used to create ordered list.

Syntax:

```
<ol>
<li> a menu item
<li> a menu item
<li> a menu item
....
....
</ol>
```

The output will be a menu, items labeled with numbers 1, 2, 3, ...

Using type attribute in <ol> tag

The default numbering (1, 2, 3 ...) can be changed using the **type** attribute in **<ol>** tag. Various values you can use with **type** attribute and their results are given below:

<ol> tag with type attribute	**Result**
<ol type=”A”>… </ol>	List labels will be A, B, C, …
<ol type=”a”>… </ol>	List labels will be a, b, c, …
<ol type=”I”>… </ol>	List labels will be I, II, III, …
<ol type=”i”>… </ol>	List labels will be I, ii, iii, …
<ol type=”1”>… </ol>	List labels will be 1, 2, 3 … (default)

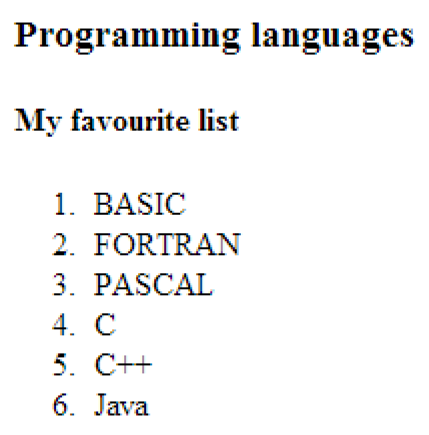
The **value** attribute can be used with **li**, to change the value of the label for an item: value=k, where k is the value for that label. Subsequent labels are incremented by 1.

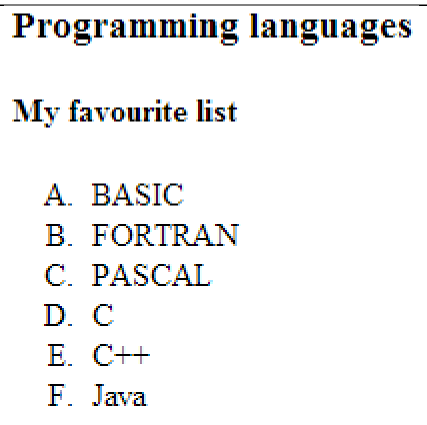
The attribute **start** in **ol** tag lets you start the list with a number other than 1. If you assign, for example, start=3, then the first label will be 3, or ’C’, ’c’, ’III’, or ’iii’, depending on the **type** value.

Another interesting attribute in **<ol>** is **reversed**. If this attribute is present (no value, just the attribute name **reversed**), the list will be displayed with reverse order of numbers.

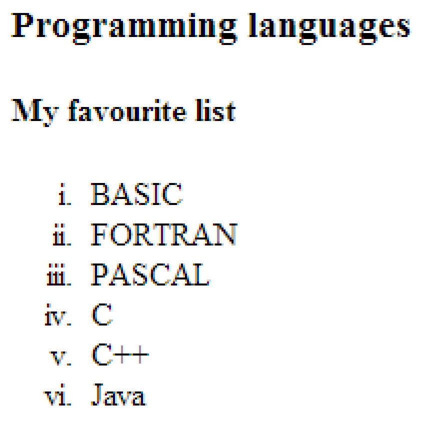
- **Examples**

Code	Browser Output

`<h3>Programming languages</h3>`
`<h4>My favourite list</h4>`
`<ol>`
`<li> BASIC`
`<li> FORTRAN`
`<li> PASCAL`
`<li> C`
`<li> C++`
`<li> Java`
`</ol>`
	 **Programming languages**
	My favourite list
	1. BASIC
	2. FORTRAN
	3. PASCAL
	4. C
	5. C++
	6. Java

`<h3>Programming languages</h3>`
`<h4>My favourite list</h4>`
`<ol type="A">`
`<li> BASIC`
`<li> FORTRAN`
`<li> PASCAL`
`<li> C`
`<li> C++`
`<li> Java`
`</ol>`
	 **Programming languages**
	My favourite list
	A. BASIC
	B. FORTRAN
	C. PASCAL
	D. C
	E. C++
	F. Java

```
<h3>Programming languages</h3>
<h4>My favourite list</h4>
<ol type="i">
<li> BASIC
<li> FORTRAN
<li> PASCAL
<li> C
<li> C++
<li> Java
</ol>
```

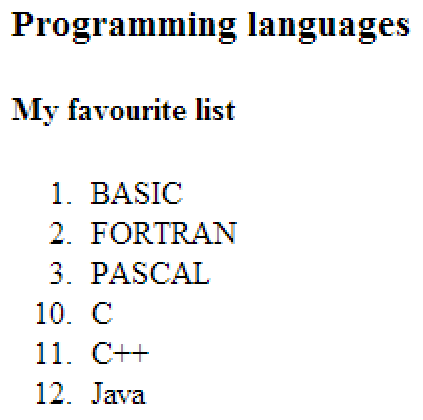


Programming languages

My favourite list

```
  i. BASIC
 ii. FORTRAN
iii. PASCAL
 iv. C
  v. C++
 vi. Java
```

```
<h3>Programming languages</h3>
<h4>My favourite list</h4>
<ol >
<li> BASIC
<li> FORTRAN
<li> PASCAL
<li value=10> C
<li> C++
<li> Java
</ol>
```

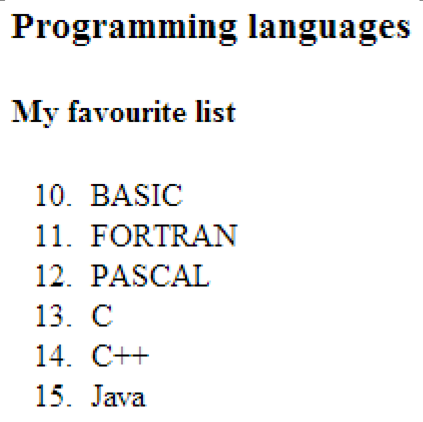


Programming languages

My favourite list

```
 1. BASIC
 2. FORTRAN
 3. PASCAL
10. C
11. C++
12. Java
```

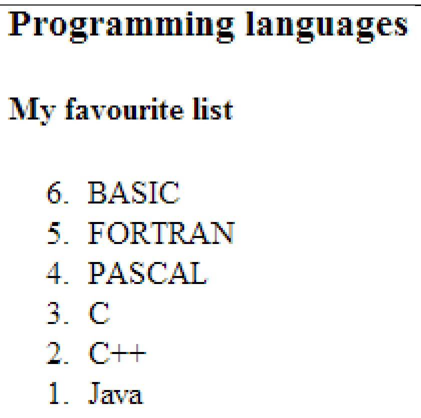
```
<h3>Programming languages</h3>
<h4>My favourite list</h4>
<ol start=10>
<li> BASIC
<li> FORTRAN
<li> PASCAL
<li> C
<li> C++
<li> Java
</ol>
```



Programming languages

My favourite list

```
10. BASIC
11. FORTRAN
12. PASCAL
13. C
14. C++
15. Java
```

<h3>Programming languages</h3> <h4>My favourite list</h4> <ol reversed> <li> BASIC <li> FORTRAN <li> PASCAL <li> C <li> C++ <li> Java </ol>	 **Programming languages** **My favourite list** 6. BASIC 5. FORTRAN 4. PASCAL 3. C 2. C++ 1. Java

The <UL> tag

The UL tag is used to create unordered lists.

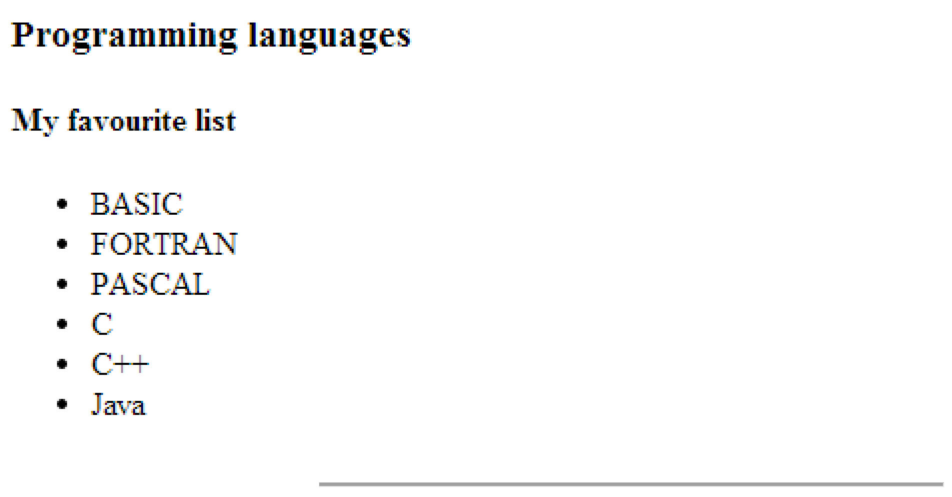
Syntax:

```
<ul>
<li> a menu item
<li> a menu item
<li> a menu item
....
....
</ul>
```

In this case, the labels for menu items will be "bullets".

- **Example:**

```
<h3>Programming languages</h3>
<h4>My favourite list</h4>
<ul>
<li> BASIC
<li> FORTRAN
<li> PASCAL
<li> C
<li> C++
<li> Java
</ul>
```



Programming languages

My favourite list

- BASIC
- FORTRAN
- PASCAL
- C
- C++
- Java

- **The definition list tag IDs DL, DT and DD**

They are used together to develop pages including definitions.

Syntax:

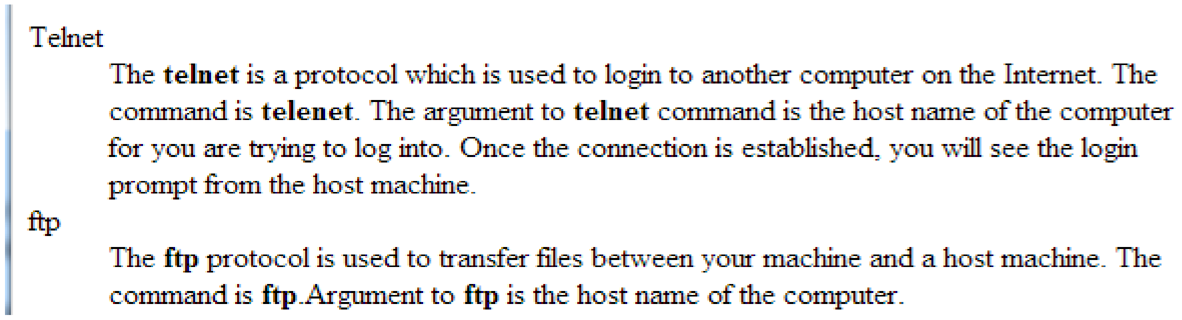
```
<DL>
<DT> a term
<DD> its definition
<DT> a term
<DD> its definition
<DT> a term
<DD> its definition
....
....
</DL>
```

The end tag </dt>may be omitted if the <dt> element is immediately followed by another <dt> tag or a <dd> tag. The end tag </dd> may be omitted if the <dd> tag is immediately followed by another <dd> tag or a <dt> tag, or if there is no more content in the parent element.

- **Practice Example 5:**

```
<TITLE> Definition Lists </TITLE>
<DL>
<DT> Telnet
<DD> The <STRONG>telnet </STRONG> is a protocol which is used to login to another
computer on the Internet. The command is <STRONG>telnet</STRONG>. The argument to
```

<B>telnet</B> command is the host name of the computer for you are trying to log into. Once the connection is established, you will see the login prompt from the host machine.
<DT>ftp
<DD> The <B>ftp</B> protocol is used to transfer files between your machine and a host machine. The command is <B>ftp</B>.Argument to <B>ftp</B> is the host name of the computer.
</DL>



Telnet
The **telnet** is a protocol which is used to login to another computer on the Internet. The command is **telenet**. The argument to **telnet** command is the host name of the computer for you are trying to log into. Once the connection is established, you will see the login prompt from the host machine.
ftp
The **ftp** protocol is used to transfer files between your machine and a host machine. The command is **ftp**.Argument to **ftp** is the host name of the computer.

9. Tables

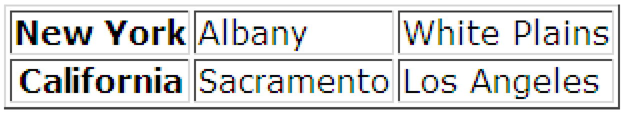
A table consists of several rows and several columns. The entire matrix is enclosed in <table> and </table>. This is a complex feature consisting of many options which give you considerable control. In addition to being useful for presenting grids of data, tables solve many of the routine alignment problems that occur in HTML. A table cell can contain text, images, hyperlinks, form elements or another table.

Each row is started with the tag <tr>.
Each heading cell is enclosed between tags <th> and </th>.
Each data cell is enclosed between tags <td> and </td>.
The only difference between <th> and <td> is that <th> prints the enclosed text bold.

- **Example:**

```
<table border>
<tr> <th> New York </th> <td> Albany </td> <td> White Plains </td> <tr>
<tr> California </tr> <td>Sacramento </td><td>Los Angeles </td>
</teble>
```

A browser renders this as follows:



New York	Albany	White Plains
California	Sacramento	Los Angeles

➢ **Adding comments**

Sometimes you want to include information in your HTML source document that should not be displayed on the screen by the browser. One way to do this is to use a comment tag: <!-- … -->. The text of the comment is included between the dashes (e.g. <!-- Created by Me -->). There is no end tag for this.

Comments can be used as notes to yourself when you are creating a document or notes about complex pieces of code. They are particularly useful if more than one person is working on the same document. Each author can enter a comment at the beginning of the document briefly describing the latest changes made. The browser ignores the comment tag.

- **Viewing HTML source code**

The document containing the HTML codes is referred to as the source document of the Web page. To view the HTML code of the Web page you are viewing in a browser screen, right click the mouse button and press “View page source”.

- **Including special characters**

As you know some characters, such as <, >, have special meaning in an HTML document. So you cannot use these special characters as characters in your HTML document. To get special character output, you need to use a special sequence of characters, called entities. All these sequences, start with a & and end with a ;. In between include a predefined symbol (case sensitive) as shown in the following table. Instead of a predefined symbol, you can also use the ASCII value preceded by #:

Character	**Entity**	**ASCII format**
<	&lt;	&#60;
>	&gt;	&#62;
“	&quot;	&#34;
‘	&apos;	&#39;
&	&amp;	&#38;
Space	&nbsp;	&#160;

There is a large number of such codes for other special codes and international symbols. See: http://dev.w3.org/html5/html-author/charref for details.

10. Selected list of tags and attributes new in HTML5

- **The <header>…</header> tags**

The **<header>…</header>** tag specifies a header for a document or section. Don't confuse the <header> tag for heading tags (<h1>, <h2>, ..) or <head> tag. <header> You normally include heading tags (<h1>…) within <header> tags. The <header> element can also be used to wrap a section's table of contents, introductory contents, and so on. You can include several <header> elements in a document.

The<header>tag cannot be placed within a<footer>, <address> or another <header> element.

Note:

- The contents of the <header>…</header> are not automatically placed at the beginning of the document browser.

- **The <footer>…</footer> tags**

The **<footer>…</footer>** tag is used for defining the footer of an HTML document or section. A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc. A document/section can have more than one footer.

- **Note:**
- The contents of the <footer>…</footer> are not automatically placed at the end of the document browser.

- **The <nav>…</nav> element**

The **<nav>… </nav>** element is used for declaring a navigational section of the HTML document. Websites typically have sections dedicated to navigational links – links that enable the user to navigate the site. These links should be placed inside a <nav> …</nav> element. Bulleted lists are generally placed between <nav>…</nav> element.

- **The <section>…</section> element**

The **<section> … </section>** represents a generic section of a document, i.e., a thematic grouping of content, typically with a heading.

- **The <aside>…</aside> element**

The **<aside>…</aside>** element represents a section of a page that consists of content that is tangentially related to the content around it, which could be considered separate from that content. Such sections are often represented as sidebars or as inserts. They often contain side explanations, like a glossary definition; more loosely related stuff, like advertisements; the biography of the author; or in web-applications, profile information or related blog links.

- **The <article> … <article> element**

The **<article> … </article>** element represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable, e.g., in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

- **The <hgroup>…<hgroup> element**

The <hgroup> … </hgroup> element has now been removed from the HTML5 specification. So don't use it.

- **The <figure> … </figure> and <figcaption> … </figcaption> elements**

The <figure> … </figure> element is a block display element used for annotating illustrations, diagrams, photos, code listings, etc. The <figure> … </figure> is frequently contains <figcaption>…</figcaption> element, which adds a caption to the content of <figure> … </figure>.

- **Example:**

```
<figure>
    <img src="ESB.jpg" width=150 height=150>
    <figcaption> Empire State Building</figcaption>
  </figure>
```

- **The <details>…</details and <summary>…</summary> elements**

The <details>…</details> element helps us hide certain portion of the page. The <details> … </details> element normally contains the <summary> … </summary> element.

Syntax:
<details><summary>*text***</summary>***detailed text …….***</details>**
The text within **<summary>…</summary>** will be displayed along with a small arrow, and the text between **<details>…</details>** will be hidden. When you click the small arrow, the detailed text will be displayed.

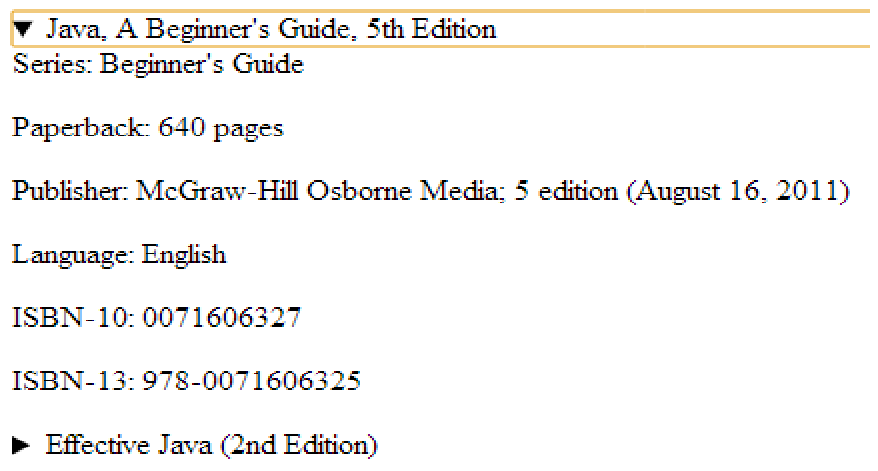
- **Example:**

<details><summary>Java, A Beginner's Guide, 5th Edition</summary>
Series: Beginner's Guide<P>
Paperback: 640 pages<P>
Publisher: McGraw-Hill Osborne Media; 5 edition (August 16, 2011)<P>
Language: English<P>
ISBN-10: 0071606327<P>
ISBN-13: 978-0071606325<P>
</details>

<details><summary>Effective Java (2nd Edition)</summary>
Paperback: 346 pages<p>
Publisher: Addison-Wesley; 2 edition (May 28, 2008)<p>
Language: English<p>
ISBN-10: 0321356683<p>

ISBN-13: 978-0321356680<p>
</details>

Chrome output:


▼ Java, A Beginner's Guide, 5th Edition
Series: Beginner's Guide

Paperback: 640 pages

Publisher: McGraw-Hill Osborne Media; 5 edition (August 16, 2011)

Language: English

ISBN-10: 0071606327

ISBN-13: 978-0071606325

▶ Effective Java (2nd Edition)

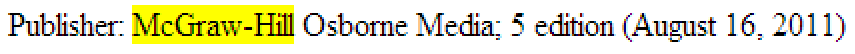
➢ **The <mark> … </mark> element**

The **<mark>***text***</mark>** is used to highlight a text.

Example:

Publisher: <mark>McGraw-Hill</mark> Osborne Media; 5 edition (August 16, 2011)

Chrome Output:


Publisher: McGraw-Hill Osborne Media; 5 edition (August 16, 2011)

➢ **The <wbr> element**

The **<wbr>** tag is used for specifying a line break when there is a long string (without a space) and you want to provide a break in such cases. Without the **<wbr>** tag, these long strings of text could either wrap in strange place (making it difficult to read), or not wrap at all – inadvertently pushing the page layout to the side (again, making it difficult to read and view the document as intended).

The difference between the **<wbr>** tag and the **<br>** tag is that the **<br>** tag forces a line break. The **<wbr>** tag, on the other hand, simply represents a line break opportunity – the browser should only render a line-break if necessary.

The **<wbr>** tag is an empty tag. In other words, it has no end tag.

- **The <time> … </time>element**

The HTML <time>…</time> element is used for declaring the date and/or time within an HTML document.
Example:

```
<h2> My appointment is at <time>9:00</time></h2>
```

The <time>…</time> element does not render special formatting.

The <time> attribute datetime:
The value of the **datetime** attribute can be in several formats. Some of the permitted formats are: YYYY, YYYY-MM, YYYY-MM-DD, MM-DD.

The general format is YYYY-MM-DDThh:mm:ssTZD.
Where,
T – a required separator if time is also specified
hh – hour (e.g. 22 for 10.00pm)
mm – minutes (e.g. 55)
ss – seconds (e.g. 03)
TZD – Time Zone Designator (Z denotes Zulu, also known as Greenwich Mean Time)

This can be helpful for user agents to offer any event scheduling for user's calendar. The machine readable version is the value of the datetime attribute. The human readable version is the content of the element.

Example:

<time datetime="2005-11-01">November 1st 2005</time>

- **The <data>…</data> element**

The <data>…</data> element is used for providing a machine-readable version of its own contents.

The <data> attribute value:
<data value="…">text</data>
We will see later how JavaScript uses this to process the data.

- **The data-* attribute**

The W3C specification for data-attributes states that:
Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.

The syntax for the data attribute is : **data-***anything*. This attribute can be used with practically any tag.

Example:
<span data-serial="1243536">product 1</span>.

This value can be processed elsewhere (using JavaScript, for example). We will see more on this later.

- **The ping attribute**

The ping attribute is used with <a> tag (also <area> tag). The value of ping attribute is a space-separated list of URLs which have to be pinged when a hyperlink is followed.

Example:
<a href="http://www.pace.edu" ping="http://www.pace.edu/track">Goto Pace</a>
When the user clicks the link, the information automatically goes to http://www.pace.edu/track.

- **The translate attribute**

This attribute does not work in any of the major browsers.

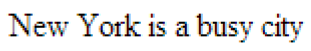
➢ **The contenteditable attribute**

This attribute makes the content of the tag editable. This attribute can be used with any HTML tag. The value of the attribute is either "true" or "false".

Example:

```
<p contenteditable="true">New York is a busy city</p>
```

Output:



New York is a busy city

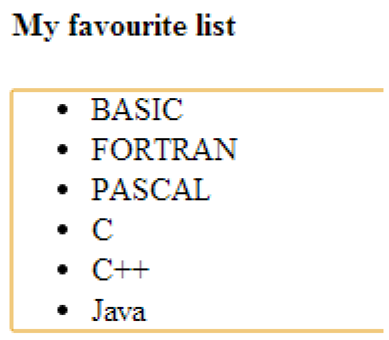
You can make changes to the text.

Example:

```
<h4>My favourite list</h4>
<ul contenteditable="true">
<li > BASIC
<li> FORTRAN
<li> PASCAL
<li> C
<li> C++
<li> Java
</ul>
```

The entire list is editable.

Output:


My favourite list

- BASIC
- FORTRAN
- PASCAL
- C
- C++
- Java

➢ **The spellcheck attribute**

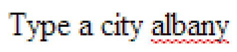
This attribute checks for spelling when the user types content. This attribute can be used with any tag which requires input from the user. The value of this attribute is either "true" or "false".

Example:

Type a city <span contenteditable="true" spellcheck="true"> Here</span>

Output:

In the output I type Albany in place of Here:

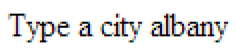

Type a city albany

Notice the spelling error (a is not capitalized).

Example:

Type a city <span contenteditable="true" spellcheck="false"> Here</span>

Output:


Type a city albany

---

11. The <iframe> tag

The HTML **<iframe>** tag provides a rectangular area, called an inline frame, on your HTML page, which can be used to embed another HTML page. This concept is called nested browsing context.

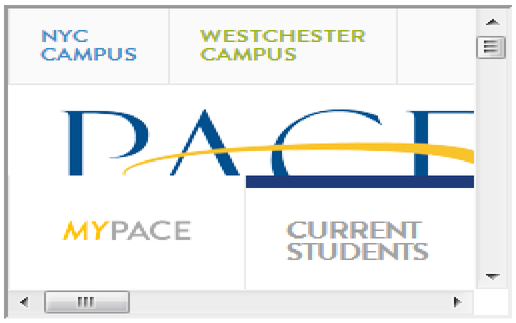
Syntax:

<**iframe src**="*URL to another Web site*" **width**=w **height**=h ></**iframe**>

This creates a rectangular area on your current HTML page and loads the specified Web page within the rectangle. The values w and h (integers specifying pixels) of attributes **width** and **height** specify width and the height of the frame dimensions.

Example:

<iframe src=http://www.pace.edu width=25 height=200></iframe>



Note: for some technical reason src=https://google.com does not work.

Other attributes:

Attribute	**Possible Values**	**Description**
name	Any name	Used in referencing the iframe elsewhere.
seamless	None	Allows the inline frame to appear as though it is being rendered as part of the containing document

sandbox	**allow-same-origin**	Allows the iframe content to be treated as being from the same origin as the containing document
	allow-top-navigation	Allows the iframe content to navigate (load) content from the containing document
	allow-forms	Allows form submission
	allow-scripts	Allows script execution
	""	imposes all above restrictions

12. The <embed> tag

The <embed> tag is newly introduced in HTML5. This is an empty tag (no end tag).
The HTML <embed> Element represents an integration point for an external application or interactive content (in other words, a plug-in). In practice, you can use <embed > tag to add images, audio or video etc.

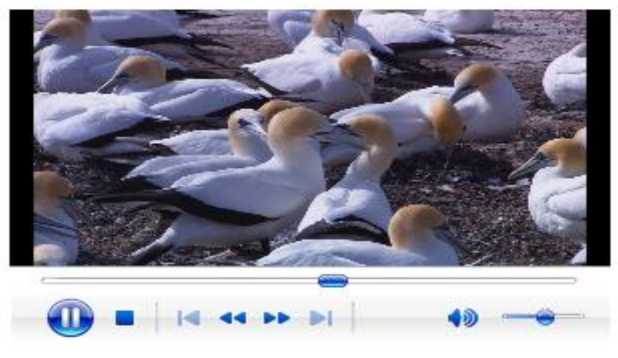
Syntax:

<embed src=”filename”>

Example:

<embed src="Wildlife.wmv" >

IE plays the wmv file. But my Chrome does not.
Output in IE:



13. Responsive Web design

Responsive Web development is an approach to web design that adjusts to the user, from varying browser sizes to changes in device. Responsive design has become a hot topic now because of exponential growth of tablets and smartphones.

Responsive websites work by using media queries to detect the device or the resolution of the device being used to access them. Once this determination has been made, a combination of flexible images, type, and grids adjust to fit the screen on which the site is being viewed. The number of users accessing the Internet via a device other than a desktop computer is rapidly increasing, making it necessary for developers to think about all the different ways their information is being viewed.

(http://www.adobe.com/devnet/html5/articles/ten-things-you-need-to-know-about-responsive-design.html)

14. References

http://dev.w3.org/html5/html-author/

http://dev.w3.org/html5/markup/elements.html

http://www.tutorialspoint.com/html5/html5_syntax.htm

http://www.w3schools.com/html/html5_intro.asp

http://www.quackit.com/html/

http://hscripts.com/tutorials/html/form-radiobutton.php

http://diveintohtml5.info/

http://www.htmlgoodies.com/html5/client/how-to-embed-video-using-html5.html#fbid=5zlWmyrpMAS

http://www.adobe.com/devnet/html5.html

http://www.adobe.com/devnet/html5/articles/ten-things-you-need-to-know-about-responsive-design.html

http://html5test.com/

http://www.chromeexperiments.com/

There are plenty of ebooks available. Search for "html5 ebooks free download."