DR. M.P.S. GROUP OF INSTITUTIONS

COURSE FILE

HTML, CSS & XML

VARUN MODI

What is an HTML File?

HTML stands for **H**yper **T**ext **M**arkup **L**anguage
An HTML file is a text file containing small **markup tags**
The markup tags tell the Web browser **how to display** the page An HTML file must have an **htm** or **html** file extension
An HTML file can be created using a **simple text editor**

If you are running Windows, start Notepad.

Type in the following text:

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

Save the file as "mypage.htm".

Start your Internet browser. Select "Open" (or "Open Page") in the File menu of your browser. A dialog box will appear. Select "Browse" (or "Choose File") and locate the HTML file you just created - "mypage.htm" - select it and click "Open". Now you should see an address in the dialog box, for example "C:\MyDocuments\mypage.htm". Click OK, and the browser will display the page.

Example Explained

The first tag in your HTML document is <html>. This tag tells your browser that this is the start of an HTML document. The last tag in your document is </html>. This tag tells your browser that this is the end of the HTML document.

The text between the <head> tag and the </head> tag is header information.
Header information is not displayed in the browser window.

The text between the <title> tags is the title of your document. The title is displayed in your browser's caption.

The text between the <body> tags is the text that will be displayed in your browser.

The text between the and tags will be displayed in a bold font.

HTM or HTML Extension?

When you save an HTML file, you can use either the .htm or the .html extension. We have used .htm in our examples. It might be a bad habit inherited from the past when some of the commonly used software only allowed three letter extensions.

With newer software we think it will be perfectly safe to use .html.

Note on HTML Editors:

You can easily edit HTML files using a WYSIWYG (what you see is what you get) editor like FrontPage, Claris Home Page, or Adobe PageMill instead of writing your markup tags in a plain text file.

But if you want to be a skillful Web developer, we strongly recommend that you use a plain text editor to learn your primer HTML.

HTML Elements

HTML documents are text files made up of HTML elements.

HTML elements are defined using HTML tags.

HTML Tags

- HTML tags are used to mark-up HTML **elements**
  - HTML tags are surrounded by the **two characters < and >** The surrounding characters are called **angle brackets**
- HTML tags normally **come in pairs** like and
  - The first tag in a pair is the **start tag,** the second tag is the **end tag** The text between the start and end tags is the **element content**
- HTML tags are **not case sensitive,** means the same as <B>

HTML Elements

Remember the HTML example from the previous page:

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

This is an HTML element:

```
<b>This text is bold</b>
```

The HTML element starts with a **start tag**:
The **content** of the HTML element is: This text is bold
The HTML element ends with an **end tag**:

The purpose of the tag is to define an HTML element that should be displayed as bold.

This is also an HTML element:

```
<body>
This is my first homepage. <b>This text is bold</b> </body>
```

This HTML element starts with the start tag <body>, and ends with the end tag </body>.

The purpose of the <body> tag is to define the HTML element that contains the body of the HTML document.

Why do We Use Lowercase Tags?

We have just said that HTML tags are not case sensitive: <B> means the same as . When you surf the Web, you will notice that most tutorials use uppercase HTML tags in their examples. We always use lowercase tags. Why?

If you want to prepare yourself for the next generations of HTML you should start using lowercase tags. The World Wide Web Consortium (W3C) recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags.

Tag Attributes

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page.

This tag defines the body element of your HTML page: <body>. With an added bgcolor attribute, you can tell the browser that the background color of your page should be red, like this: <body bgcolor="red">.

This tag defines an HTML table: <table>. With an added border attribute, you can
tell the browser that the table should have no borders: <table border="0">

Attributes always come in name/value pairs like this: name="value".

Attributes are always added to the start tag of an HTML element.

Quote Styles, "red" or 'red'?

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed.

In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:

name='John "ShotGun" Nelson'

Basic HTML Tags

The most important tags in HTML are tags that define headings, paragraphs and line breaks.

Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest heading. <h6> defines the smallest heading.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading.

Paragraphs

Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

HTML automatically adds an extra blank line before and after a paragraph.

Line Breaks

The <br> tag is used when you want to end a line, but don't want to start a new paragraph. The <br> tag forces a line break wherever you place it.

```
<p>This <br> is a para<br>graph with line breaks</p>
```

The <br> tag is an empty tag. It has no closing tag.

Comments in HTML

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Basic HTML Tags

Tag	Description
<html>	Defines an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines header 1 to header 6
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a horizontal rule
<!-->	Defines a comment

HTML Text Formatting

HTML defines a lot of elements for formatting output, like bold or italic text.

How to View HTML Source

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

To find out, simply click on the VIEW option in your browsers toolbar and select SOURCE or PAGE SOURCE. This will open a window that shows you the actual HTML of the page.

Text Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text

	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text

<ins>	Defines inserted text
<del>	Defines deleted text
<s>	Deprecated. Use <del> instead
<strike>	Deprecated. Use <del> instead
<u>	Deprecated. Use styles instead

HTML Character Entities

Some characters like the < character, have a special meaning in HTML, and therefore cannot be used in the text.

To display a less than sign (<) in HTML, we have to use a character entity.

Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

To display a less than sign in an HTML document we must write: **<** or **<**

The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Note that the entities are case sensitive.

Non-breaking Space

The most common character entity in HTML is the non-breaking space.

Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the   character entity.

The Most Common Character Entities:

Result	**Description**	**Entity Name**	**Entity Number**
	non-breaking space		
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&

"	quotation mark	&quot;	&#34;
'	apostrophe	&apos;	&#39;

Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

The Anchor Tag and the Href Attribute

HTML uses the <a> (anchor) tag to create a link to another document.

An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.

The syntax of creating an anchor:

```
<a href="url">Text to be displayed</a>
```

The <a> tag is used to create an anchor to link from, the href attribute is used to address the document to link to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.

This anchor defines a link to W3Schools:

```
<a href="http://www.w3schools.com/">Visit W3Schools!</a>
```

The line above will look like this in a browser:

Visit W3Schools!

The Target Attribute

With the target attribute, you can define **where** the linked document will be opened.

The line below will open the document in a new browser window:

```
<a href="http://www.w3schools.com/"
target="_blank">Visit W3Schools!</a>
```

The Anchor Tag and the Name Attribute

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly into a specific section on a page, instead of letting the user scroll around to find what he/she is looking for.

Below is the syntax of a named anchor:

```
<a name="label">Text to be displayed</a>
```

The name attribute is used to create a named anchor. The name of the anchor can be any text you care to use.

The line below defines a named anchor:

```
<a name="tips">Useful Tips Section</a>
```

You should notice that a named anchor is not displayed in a special way.

To link directly to the "tips" section, add a # sign and the name of the anchor to the end of a URL, like this:

```
<a href="http://www.w3schools.com/html_links.asp#tips"> Jump to the Useful
Tips Section</a>
```

A hyperlink to the Useful Tips Section from WITHIN the file "html_links.asp" will look like this:

```
<a href="#tips">Jump to the Useful Tips Section</a>
```

Link Tags

Tag	Description
<a>	Defines an anchor

HTML Frames

With frames, you can display more than one Web page in the same browser window.

Examples

Vertical frameset
This example demonstrates how to make a vertical frameset with three different documents.

Horizontal frameset
This example demonstrates how to make a horizontal frameset with three different documents.

How to use the <noframes> tag
This example demonstrates how to use the <noframes> tag.

(You can find more examples at the bottom of this page)

Frames

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

The web developer must keep track of more HTML documents It is difficult to print the entire page

The Frameset Tag

The <frameset> tag defines how to divide the window into frames Each frameset defines a set of rows **or** columns
The values of the rows/columns indicate the amount of screen area each row/column will occupy

The Frame Tag

The <frame> tag defines what HTML document to put into each frame

In the example below we have a frameset with two columns. The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%">
    <frame src="frame_a.htm">
    <frame src="frame_b.htm">
</frameset>
```

Basic Notes - Useful Tips

If a frame has visible borders, the user can resize it by dragging the border. To prevent a user from doing this, you can add noresize="noresize" to the <frame> tag.

Add the <noframes> tag for browsers that do not support frames.

Frame Tags

Tag	Description
<frameset>	Defines a set of frames
<frame>	Defines a sub window (a frame)
<noframes>	Defines a noframe section for browsers that do not handle frames
<iframe>	Defines an inline sub window (frame)

HTML Tables

With HTML you can create tables.

Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Tables and the Border Attribute

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.

To display a table with borders, you will have to use the border attribute:

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

```
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

Headings in a Table

Headings in a table are defined with the <th> tag.

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing.

To avoid this, add a non-breaking space ( ) to empty data cells, to make the borders visible:

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td> </td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

HTML Lists

HTML supports ordered, unordered and definition lists.

Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Here is how it looks in a browser:

- Coffee
- Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers.

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Definition Lists

A definition list is **not** a list of items. This is a list of terms and explanation of the terms.

A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag.

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser:

Coffee
    Black hot drink
Milk
    White cold drink

Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other lists, etc.

List Tags

Tag	Description
	Defines an ordered list
	Defines an unordered list
	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a definition term
<dd>	Defines a definition description
<dir>	Deprecated. Use instead
<menu>	Deprecated. Use instead

HTML Forms and Input

HTML Forms are used to select different kinds of user input.

Forms

A form is an area that can contain form elements.

Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

```
<form>
   <input>
   <input>
</form>
```

Input

The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

Text Fields

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
First name:
<input type="text" name="firstname"> <br>
Last name:
<input type="text" name="lastname">
</form>
```

How it looks in a browser:

First name:

Last name:

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default.

Radio Buttons

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex" value="male"> Male <br>
<input type="radio" name="sex" value="female"> Female </form>
```

How it looks in a browser:

- Male
- Female

Note that only one option can be chosen.

Checkboxes

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
<input type="checkbox" name="bike">
I have a bike
<br>
<input type="checkbox" name="car">
I have a car
</form>
```

How it looks in a browser:

 I have a bike

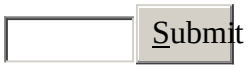
 I have a car

The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to another file. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_action.asp"
method="get">
Username:
<input type="text" name="user"> <input
type="submit" value="Submit"> </form>
```

How it looks in a browser:

Username: 

If you type some characters in the text field above, and click the "Submit" button, you will send your input to a page called "html_form_action.asp". That page will show you the received input.

Form Tags

Tag	Description
<form>	Defines a form for user input
<input>	Defines an input field

<textarea>	Defines a text-area (a multi-line text input control)
<label>	Defines a label to a control
<fieldset>	Defines a fieldset
<legend>	Defines a caption for a fieldset
<select>	Defines a selectable list (a drop-down box)
<optgroup>	Defines an option group
<option>	Defines an option in the drop-down box
<button>	Defines a push button
<isindex>	Deprecated. Use <input> instead

HTML Images

With HTML you can display images in a document.

The Image Tag and the Src Attribute

In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

The syntax of defining an image:

```
<img src="url">
```

The URL points to the location where the image is stored. An image named "boat.gif" located in the directory "images" on "www.w3schools.com" has the URL: http://www.w3schools.com/images/boat.gif.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

The Alt Attribute

The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text:

```
<img src="boat.gif" alt="Big Boat">
```

The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines an area inside an image map

HTML Backgrounds

A good background can make a Web site look really great.

Backgrounds

The <body> tag has two attributes where you can specify backgrounds. The background can be a color or an image.

Bgcolor

The bgcolor attribute sets the background to a color. The value of this attribute can be a hexadecimal number, an RGB value, or a color name.

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

The lines above all set the background color to black.

Background

The background attribute sets the background to an image. The value of this attribute is the URL of the image you want to use. If the image is smaller than the browser window, the image will repeat itself until it fills the entire browser window.

```
<body background="clouds.gif">
<body background="http://www.w3schools.com/clouds.gif">
```

The URL can be relative (as in the first line above) or absolute (as in the second line above).

HTML Fonts

The <font> tag in HTML is deprecated. It is supposed to be removed in a future version of HTML.

Even if a lot of people are using it, you should try to avoid it, and use styles instead.

The HTML <font> Tag

With HTML code like this, you can specify both the size and the type of the browser output :

```
<p>
<font size="2" face="Verdana">
This is a paragraph.
</font>
</p>
<p>
<font size="3" face="Times">
This is another paragraph.
</font>
</p>
```

Font Attributes

Attribute	Example	Purpose
size="number"	size="2"	Defines the font size
size="+number"	size="+1"	Increases the font size
size="-number"	size="-1"	Decreases the font size
face="face-name"	face="Times"	Defines the font-name
color="color-value"	color="#eeff00"	Defines the font color
color="color-name"	color="red"	Defines the font color

The <font> Tag Should NOT be Used

The <font> tag is deprecated in the latest versions of HTML (HTML 4 and XHTML).

CSS B a si cs

Chapter: 1 - Introduction to CSS

A CSS (cascading style sheet) file allows you to separate your web sites (X)HTML content from it's style. As always you use your (X)HTML file to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on...) are accomplished within a CSS.

At this point you have some choices of how to use the CSS, either internally or externally.

Internal Stylesheet

First we will explore the internal method. This way you are simply placing the CSS code within the <head></head> tags of each (X)HTML file you want to style with the CSS. The format for this is shown in the example below.

```
<head>
<title><title>
<style type="text/css">
CSS Content Goes  Here
</style>
</head>
<body>
```

With this method each (X)HTML file contains the CSS code needed to style the page. Meaning that any changes you want to make to one page, will have to be made to all. This method can be good if you need to style only one page, or if you want different pages to have varying styles.

External Stylesheet

Next we will explore the external method. An external CSS file can be created with any text or HTML editor such as "Notepad" or "Dreamweaver". A CSS file contains no (X)HTML, only CSS. You simply save it with the .css file extension. You can link to the file externally by placing one of the following links in the head section of every (X)HTML file you want to style with the CSS file.

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

Or you can also use the @import method as shown below

<style type="text/css">@import url(*Path To stylesheet.css*)</style>

Either of these methods are achieved by placing one or the other in the head section as shown in example below.

```
<head>
<title><title>
<link rel="stylesheet" type="text/css"href="style.css"    />
</head>
<body>
```

or

```
<head>
<title><title>
<style type="text/css"> @import url(Path To    stylesheet.css)
</style>
</head>
<body>
```

By using an external style sheet, all of your (X)HTML files link to one CSS file in order to style the pages. This means, that if you need to alter the design of all your pages, you only need to edit one .css file to make global changes to your entire website.

Here are a few reasons this is better.

- Easier Maintenance
- Reduced File Size
- Reduced Bandwidth
- Improved Flexibility

Cascading Order

In the previous paragraphs, I have explained how to link to a css file either internally or externally. If you understood, than I am doing a good job. If not don't fret, there is a long way to go before we are finished.
Assuming you have caught on already, you are probably asking, well can I do both? The answer is yes. You can have both internal, external, and now wait a minute a third way? Yes inline styles also.

Inline Styles

I have not mentioned them until now because in a way they defeat the purpose of using CSS in the first place. Inline styles are defined right in the (X)HTML file along side the element you want to style. See example below.

Some red text
Some red text

Inline styles will NOT allow the user to change styles of elements or text formatted this way

So, which is better?

So with all these various ways of inserting CSS into your (X)HTML files, you may now be asking well which is better, and if I use more than one method, in what order do these different ways load into my browser?

All the various methods will cascade into a new "pseudo" stylesheet in the following order:

1. Inline Style (inside (X)HTML element)
2. Internal Style Sheet (inside the <head> tag)
3. External Style Sheet

As far as which way is better, it depends on what you want to do. If you have only one file to style then placing it within the <head></head> tags (internal) will work fine. Though if you are planning on styling multiple files then the external file method is the way to go.

Choosing between the <link related=> & the @import methods are completely up to you. I will mention that the @import method may take a second longer to read the CSS file in Internet Explorer than the <link related=> option. To combat this see Flash of unstyled content

Users with Disabilities

The use of external style sheets also can benefit users that suffer from disabilities. For instance, a user can turn off your stylesheet or substitute one of there own to increase text size, change colors and so on. For more information on making your website accessible to all users please read Dive into accessibility

Power Users

Swapping stylesheets is beneficial not only for users with disabilities, but also power users who are particular about how they read Web documents.

Browser Issues

You will discover as you delve farther into the world of CSS that all browsers are not created equally, to say the least. CSS can and will render differently in various browsers causing numerous headaches.

Chapter 2 - CSS Syntax

The syntax for CSS is different than that of (X)HTML markup. Though it is not too confusing, once you take a look at it. It consists of only 3 parts.

```
selector { property: value  }
```

The selector is the (X)HTML element that you want to style. The property is the actual property title, and the value is the style you apply to that property.

Each selector can have multiple properties, and each property within that selector can have independent values. The property and value are seperated with a colon and contained within curly brackets. Multiple properties are seperated by a semi colon. Multiple values within a property are sperated by commas, and if an individual value contains more than one word you surround it with quotation marks. As shown below.

```
body {
   background: #eeeeee;
   font-family: "Trebuchet MS", Verdana, Arial,    serif;
}
```

As you can see in the above code I have seperated the color from the font-family with a semi-colon, seperated the various fonts with commas and contained the "Trebuchet MS" within quotations marks. The final result sets the body color to light grey, and sets the font to ones that most users will have installed on there computer.

I have changed the way I layout my code, but you can arrange it in one line if you choose. I find that it is more readable if I spread each property to a seperate line, with a 2 space indention.

Inheritance

When you nest one element inside another, the nested element will inherit the properties assigned to the containing element. Unless you modify the inner elements values independently.

For example, a font declared in the body will be inherited by all text in the file no matter the containing element, unless you declare another font for a specific nested element.

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

Now all text within the (X)HTML file will be set to Verdana.

If you wanted to style certain text with another font, like an h1 or a paragraph then you could do the following.

```
h1 {font-family: Georgia, sans-serif;} p {font-
family: Tahoma,  serif;}
```

Now all <h1> tags within the file will be set to Georgia and all <p> tags are set to Tahoma, leaving text within other elements unchanged from the body declaration of Verdana.

There are instances where nested elements do not inherit the containing elements properties.

For example, if the body margin is set to 20 pixels, the other elements within the file will not inherit the body margin by default.

```
body {margin: 20px;}
```

Combining Selectors

You can combine elements within one selector in the following fashion.

```
h1, h2, h3, h4, h5, h6 { color:
   #009900;
   font-family: Georgia, sans-serif;
}
```

As you can see in the above code, I have grouped all the header elements into one selector. Each one is seperated by a comma. The final result of the above code sets all headers to green and to the specified font. If the user does not have the first font I declared it will go to another sans-serif font the user has installed on there computer.

Comment tags

Comments can be used to explain why you added certain selectors within your css file. So as to help others who may see your file, or to help you remember what you we're thinking at a later date. You can add comments that will be ignored by browsers in the following manner.

```
/* This is a comment  */
```

You will note that it begins with a / (forward slash) and than an * (asterisks) then the comment, then the closing tag which is just backward from the opening tag * (asterisks) then the / (forward slash).

Chapter 3: CSS Classes

The class selector allows you to style items within the same (X)HTML element differently. Similiar to what I mentioned in the introduction about inline styles. Except with classes the style can be overwritten by changing out stylesheets. You can use the same class selector again and again within an (X)HTML file.

To put it more simply, this sentence you are reading is defined in my CSS file with the following.

```
p {

   font-size: small; color:
   #333333
}
```

Pretty simple, but lets say that I wanted to change the word "sentence" to green bold text, while leaving the rest of the sentence untouched. I would do the following to my (X)HTML file.

```
<p>
To put it more simply, this <span class="greenboldtext">sentence</span> you
are reading is styled in my CSS file by the following.
</p>
```

Then in my CSS file I would add this style selector:

```
.greenboldtext{ font-size:
   small; color: #008080;
   font-weight: bold;
}
```

The final result would look like the following:

To put it more simply, this **sentence** you are reading is styled in my CSS file by the following.

Please note that a class selector begins with a (.) period. The reason I named it "greenboldtext" is for example purposes, you can name it whatever you want. Though I do encourage you to use selector names that are descriptive. You can reuse the "greenboldtext" class as many times as you want.

Chapter 4: CSS IDs

IDs are similar to classes, except once a specific id has been declared it cannot be used again within the same (X)HTML file.

I generally use IDs to style the layout elements of a page that will only be needed once, whereas I use classes to style text and such that may be declared multiple times.

The main container for this page is defined by the following.

```
<div id="container">
Everything within my document is inside this division.
</div>
```

I have chosen the id selector for the "container" division over a class, because I only need to use it one time within this file.

Then in my CSS file I have the following:

```
#container{ width:
    80%; margin:
    auto;

    padding: 20px;
    border: 1px solid #666;
    background: #ffffff;
}
```

You will notice that the id selector begins with a (#) number sign instead of a (.) period, as the class selector does.

Chapter 5: CSS Divisions

Ok so you have finished the first 4 chapters in my series. You have learned the very basics of CSS, how the syntax works and a bit about classes and IDs. Now we are gonna take a quick break from CSS and focus on the (X)HTML side of using it.

Divsions

Divisions are a block level (X)HTML element used to define sections of an (X)HTML file. A division can contain all the parts that make up your website. Including additional divisions, spans, images, text and so on.

You define a division within an (X)HTML file by placing the following between the <body></body> tags:

```
<div>
```

```
Site contents go  here
</div>
```

Though most likely you will want to add some style to it. You can do that in the following fashion:

```
<div id="container"> Site
contents go  here
</div>
```

The CSS file contains this:

```
#container{ width:
    70%; margin:
    auto; padding:
    20px;
    border: 1px solid #666;
    background: #ffffff;
}
```

Now everything within that division will be styled by the "container" style rule, I defined within my CSS file. A division creates a linebreak by default. You can use both classes and IDs with a division tag to style sections of your website.

Chapter 6: CSS Spans

Spans are very similar to divisions except they are an inline element versus a block level element. No linebreak is created when a span is

declared.

You can use the span tag to style certain areas of text, as shown in the following:

```
<span class="italic">This text is   italic</span>
```

Then in my CSS file:

```
.italic{
    font-style: italic;
}
```

The final result is: *This text is italic.*

The purpose of the last 2 chapters was to provide you with a basis for using CSS in an (X)HTML file. For a more detailed explaination of XHTML please visit W3Schools

Chapter 7: CSS Margins

Inherited: No

As you may have guessed, the margin property declares the margin between an (X)HTML element and the elements around it. The margin property can be set for the top, left, right and bottom of an element. (see example below)

```
margin-top: length percentage or auto; margin-left:
length percentage or auto; margin-right: length
percentage or auto; margin-bottom: length
percentage or   auto;
```

As you can also see in the above example you have 3 choices of values for the margin property

- length
- percentage
- auto

You can also declare all the margins of an element in a single property as follows:

```
margin: 10px 10px 10px  10px;
```

If you declare all 4 values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left

If only one value is declared, it sets the margin on all sides. (see below)

```
margin: 10px;
```

If you only declare two or three values, the undeclared values are taken from the opposing side. (see below)

```
margin: 10px 10px; /* 2 values */ margin: 10px
10px 10px; /* 3 values */
```

You can set the margin property to negative values. If you do not declare the margin value of an element, the margin is 0 (zero).

```
margin: -10px;
```

Elements like paragraphs have default margins in some browsers, to combat this set the margin to 0 (zero).

```
p {margin: 0;}
```

Note: You do not have to add px (pixels) or whatever units you use, if the value is 0 (zero).

You can see in the example below, the elements for this site are set to be 20px (pixels) from the body

```
body{
    margin: 20px; background:
    #eeeeee; font-size: small;
    font-family: Tahoma, Arial, "Trebuchet MS", Helvetica, sans- serif;
    text-align: left;
}
```

Chapter 8: CSS Padding

Inherited: No

Padding is the distance between the border of an (X)HTML element and the content within it.

Most of the rules for margins also apply to padding, except there is no "auto" value, and negative values cannot be declared for padding.

```
padding-top: length percentage; padding-left:
length percentage; padding-right: length
percentage; padding-bottom: length percentage;
```

As you can also see in the above example you have 2 choices of values for the padding property

length

percentage

You can also declare all the padding of an element in a single property as follows:

```
padding: 10px 10px 10px  10px;
```

If you declare all 4 values as I have above, the order is as follows:

1. top
2. right
3. bottom
4. left

If only one value is declared, it sets the padding on all sides. (see below)

```
padding: 10px;
```

If you only declare two or three values, the undeclared values are taken from the opposing side. (see below)

```
padding: 10px 10px; /* 2 values */ padding: 10px
10px 10px; /* 3 values */
```

If you do not declare the padding value of an element, the padding is 0 (zero).

Note: You do not have to add px (pixels) or whatever units you use, if the value is 0 (zero).

You can see in the example below, the main container for this site has 30px (pixels) of padding between the border and the text.

```
#container{ width:
    70%; margin:
    auto; padding:
    30px;
    border: 1px solid #666;
    background: #ffffff;
}
```

Chapter 9: CSS Text Properties

Inherited: Yes

Color

You can set the color of text with the following:

```
color: value;
```

Possible values are

color name - example:(red, black...)

hexadecimal number - example:(#ff0000, #000000) RGB color

code - example:(rgb(255, 0, 0), rgb(0, 0, 0))

Letter Spacing

You can adjust the space between letters in the following manner. Setting the value to 0, prevents the text from justifying. You can use negative values.

letter-spacing: value;

Possible values are

normal

length

Example:

T h e s e l e t t e r s a r e s p a c e d a t 5 p x .

Text Align

You can align text with the following:

text-align: value;

Possible values are

left

right

center

justify

Examples:

This text is aligned left.

This text is aligned in the center.

This text is aligned right.

This text is justified.

Text Decoration

You can decorate text with the following:

text-decoration: value;

Possible values are

none underline
overline line
through blink

Examples:

This text is underlined. This

text is overlined.

~~This text has a line through it.~~

This text is blinking (not in internet explorer).

Text Indent

You can indent the first line of text in an (X)HTML element with the following:

text-indent: value;

Possible values are

length percentage

Examples:

This text is indented 10px pixels.

Text Transform

You can control the size of letters in an (X)HTML element with the following:

text-transform: value;

Possible values are

none capitalize

lowercase

Examples:

This First Letter In Each Word Is Capitalized, Though It Is Not In My File.

THIS TEXT IS ALL UPPERCASE, THOUGH IT IS ALL LOWERCASE IN MY FILE.

this text is all lowercase. though it is all uppercase in my file.

White Space

You can control the whitespace in an (X)HTML element with the following:

white-space: value;

Possible values are

normal
pre
nowrap

Word Spacing

You can adjust the space between words in the following manner. You can use negative values.

word-spacing: value;

Possible values are

normal
length

Example:

These words are spaced at 5px.

Inherited: Yes

Font

The font property can set the style, weight, variant, size, line height and font:

font: italic bold normal small/1.4em Verdana, sans-serif;

The above would set the text of an element to an italic style a bold weight a normal variant a relative size a line height of 1.4em and the font to Verdana or another sans-serif typeface.

Font-Family

You can set what font will be displayed in an element with the font- family property.

There are 2 choices for values:

family-name generic
family

If you set a family name it is best to also add the generic family at the end. As this is a priortized list. So if the user does not have the specified font name it will use the same generic family. (see below)

font-family: Verdana, sans-serif;

Font Size

You can set the size of the text used in an element by using the font- size property.

font-size: value;

There are alot of choices for values:

xx-large x-
large
larger
large
medium
small
smaller x-
small xx-

small
length
% (percent)

There is quite a bit to learn about font sizes with CSS so, I am not even going to try to explain it. Actually there are already some great resources on how to size your text. (see below)

What size text should I use in my css by Paul O'B Dive
into accessibility - Font Sizes

Font Style

You can set the style of text in a element with the font-style property

font-style: value;

Possible values are

normal
itailc
oblique

Font Variant

You can set the variant of text within an element with the font-variant property

font-variant: value;

Possible values are

normal small-
caps

Font Weight

You can control the weight of text in an element with the font-weight property:

font-weight: value;

Possible values are

lighter
normal
100
200
300
400
500
600
700
800
900
bold

Chapter 11: CSS Anchors, Links and Pseudo Classes

Below are the various ways you can use CSS to style links.

```
a:link {color: #009900;}
a:visited {color: #999999;}
a:hover {color:  #333333;}
a:focus {color:  #333333;}
a:active {color: #009900;}
```

Now lets take a look at what each one of the above link styles actually does.

a:link {color: #009900;}

The first on the list sets the color of a link when no event is occuring

a:visited {color: #999999;}

The second sets the color a link changes to, when the user has already visited that url

a:hover {color: #333333;}

The third sets the color a link changes to as the user places their mouse pointer over the link

a:focus {color: #333333;}

The fourth is primarilly for the same purpose as the last one, but this one is for users that are not using a mouse and are tabbing through the links via there

keyboards tab key, it sets the color a link changes to as the user tabs through the links

```
a:active {color: #009900;}
```

The fifth on the list sets the color a link changes to as it is pressed. Lets look at an

example: Google

If your last visit to Google is not stored in your cache than the above link to google is blue, if you have already been to google then the link should be grey. if you mouseover or tab through the links, the link will change to dark grey, and last but not least if you click and hold the link without releasing it you will see it return back to the original blue color.

You must declare the a:link and a:visited before you declare a:hover. Furthermore, you must declare a:hover before you can declare a:active.

Using the above code will style all links on your web page, unless you declare a seperate set of link styles for a certain area of your webpage.

Pseudo Classes

You can set links contained in different parts of your web page to be different colors by using the pseudo class. For example, lets say you want your links in the content area to have a different color then the links in the left or right column of your webpage.

You can do this in the following fashion:

```
#content a:link {color: #009900;} #content
a:visited {color: #999999;} #content a:hover
{color: #333333;} #content a:focus {color:
#333333;} #content a:active {color:  #009900;}
```

Now assuming that you have your main content in a division named "content" all links within that division will now be styled by this new style selector. Should your selector have a different name, just change the #content selector to match your division name.

Then for the links in a column you could use the following:

```
#column a:link {color: #009900;} #column
a:visited {color: #999999;} #column a:hover
{color: #333333;} #column a:focus {color:
#333333;} #column a:active {color:  #009900;}
```

Once again, this assumes the name of the column division, just change the name to match yours.

This same method can be accomplished by declaring a class instead of an id.

```
a.column:link {color: #009900;}
a.column:visited {color: #999999;}
a.column:hover {color:  #333333;}
a.column:focus {color:  #333333;}
a.column:active {color: #009900;}
```

Though in this case you will need to add a class to each link

```
<a class="column" href="" title="">some link    text</a>
```

But, there is still yet an easier way

```
.column a:link {color: #009900;}
.column a:visited {color:  #999999;}
.column a:hover {color:  #333333;}
.column a:focus {color:  #333333;}
.column a:active {color:  #009900;}
```

Then in the (X)HTML file

```
<div class="column">
 <a href="" title="">some link   text</a>
 </div>
```

There are other properties that can be added to links other than color, I was just trying to keep it simple. Almost any property that can be used to style text and fonts can be used to style links also

Chapter 12: CSS Backgro unds

Inherited: No

Background

You can style the background of an element in one declaration with the background property.

```
background: #ffffff url(path_to_image) top left no-repeat fixed;
```

Values:

attachment
color image
position
repeat

Or you can set each property individually

Background Attachment

If you are using an image as a background. You can set whether the background scrolls with the page or is fixed when the user scrolls down the page with the background-attachment property

background-attachment: value;

Values:

fixed
scroll

Background Color

You can specifically declare a color for the background of an element using the background-color property.

background-color: value;

Values:

color name hexadecimal
number RGB color code
transparent

Background Image

You can set an image for the background of an element using the background-image property.

background-image: url(path_to_image);

Values:

url
none

Background Position

You can position an image used for the background of an element using the background-position property.

background-position: value;

Values:

top left top
center top
right center
left
center center
center right
bottom left
bottom center
bottom right x-%
y-%
x-pos y-pos

Background Repeat

You can set if an image set as a background of an element is to repeat (across=x and/or down=y) the screen using the background-repeat property.

background-repeat: value;

Values:
repeat
repeat-x
repeat-y

Chapter 13: CSS Borders

Inherited: No

Border

You can set the color, style and width of the borders around an element in one declaration by using the border property.

border: 1px solid #333333;

Values:

color
style
width

Or you can set each property individually

Border Color

You can set the color of a border independently with the border-color property.

border-color: value;

Values:

color name hexadecimal
number RGB color code
transparent

Border Style

You can set the style of a border independently with the border-style property.

border-style: value;

Values:

dashed
dotted
double
groove
hidden
inset

none
outset
ridge
solid

Border Width

You can set the width of a border independently with the border-width property.

border-width: value;

Values:

Length
Thin
Medium
Thick

Or you can set the elements for each borders side individually

Border Bottom

You can set the color, style and width of the bottom border around an element in one declaration with the border-bottom property.

border -bottom: 1px solid #333333;

Values:

color
style
width

Or you can set each value individually

Border Bottom Color

You can set the color of the bottom border around an element with the border-bottom-color property.

border -bottom-color: value;

Border Bottom Style

You can set the style of the bottom border around an element with the border-bottom-style property.

```
border-bottom-style: value;
```

Border Bottom Width

You can set the width of the bottom border around an element with the border-bottom-width property.

```
border-bottom-width: value;
```

Border Left

You can set the color, style and width of the left border around an element with the border-left property.

```
border -left: 1px solid  #333333;
```

Values:

- color
- style
- width

Or you can set each value individually

Border Left Color

You can set the color of the left border around an element with the border-left-color property.

```
border-left-color: value;
```

Border Left Style

You can set the style of the left border around an element with the border-left-style property.

```
border-left-style: value;
```

Border Left Width

You can set the width of the left border around an element with the border-left-width property.

border-left-width: value;

Border Right

You can set the color, style and width of the right border around an element in one declaration with the border-right property.

border -right: 1px solid #333333;

Values:

color
style
width

Or you can set each value individually

Border Right Color

You can set the color of the right border around an element with the border-right-color property.

border-right-color: value;

Border Right Style

You can set the style of the right border around an element with the border-right-style property.

border-right-style: value;

Border Right Width

You can set the width of the right border around an element with the border-right-width property.

border-right-width: value;

Border Top

You can set the color, style and width of the top border around an element in one declaration with the border-top property.

```
border-top: 1px solid  #333333;
```

Values:

- color
- style
- width

Or you can set each value individually

Border Top Color

You can set the color of the top border around an element with the border-top-color property.

Border Top Style

You can set the style of the top border around an element with the border-top-style property.

```
border-top-style: value;
```

Border Top Width

You can set the width of the top border around an element with the border-top-width property.

```
border -top-width:  value;
```

Chapter 14 - CSS Ordered & Unordered Lists

Inherited: Yes

List Style

You can control the appearance of ordered and unordered lists in one declaration with the list-style property

```
list-style: value value;
```

Values:

image
position
type

Or you can control them individually

List Style Image

You can use an image for the bullet of unordered lists with the list-style property

list-style-image: url(path_to_image.gif, jpg or png);

If you use an image, it is a good idea to declare the list-style-type also in case the user has images turned off.

List Style Position

You can control the position of ordered and unordered lists with the list- style-position property

list-style-position: value;

Values

inside
outside

List Style Type

You can control the type of bullet ordered and unordered lists use with the list-style-type property

list-style-type: value;

Values

disc
circle
square

decimal
lower-roman
upper-roman
lower-alpha
upper-alpha
none

Chapter 15 - CSS Width and Height Properties

Inherited: No

Height

You can control the height of an element with the height

property height: value;

Values:

auto length
percentage

Line Height

You can control the height between lines with the line-height

property line-height: value;

Values:

normal number
length
percentage

Max Height

You can control the maximum height of an element with the max-height property

max-height: value;

Values:

none length
percentage

Min Height

You can control the minimum height of an element with the min-height property

min-height: value;

Values:

length percentage

Width

You can control the width of an element with the width

property width: value;

Values:

auto length
percentage

Chapter 16 - CSS Classification

Inherited: No

Clear

You can control if an element allows floated elements to its sides with the clear property

clear: value;

Values:

none
both
left
right

Now, what does all that mean?

None

This is the default setting, floated elements can appear on either side of the element set to clear: none;

Both

Setting the value to both, causes no floated elements to appear on either side of the element set to clear: both; **Left**

Setting the value to left, causes no floated elements to appear to the left side of the element set to clear: left;

Right

Setting the value to right, causes no floated elements to appear to the right side of the element set to clear: right;

Clip

You can control how much of an element is visible with the clip property

clip: value;

Values:

auto
shape

Currently the only shape recognized by the clip property is rect (rectangle)

clip: rect(10px, 10px, 10px, 10px);

Cursor

You can control the style of cursor to be used in an element with the cursor property

cursor: value;

Values:

auto
crosshair
default help
move
pointer text
url
wait
e-resize ne-
resize nw-
resize n-
resize se-
resize sw-
resize

s-resize

w-resize

If you choose to use a custom cursor, it is always a good idea to declare a generic one after the custom value.

```
cursor: url("image.cur"), default;
```

Display

You can control how an element is displayed with the display property

```
display: value;
```

Values:

block
inline list-
item none

Now, what does all that mean?

Block

Creates a line break before and after the element

Inline

No line break is created

List Item

Creates a line break before and after the element and adds a list item marker

None

Makes an element not display on the page

Float

The float property changes how text and or images within an element are displayed

```
float: value;
```

Values:

left
right
none

Now, what does all that mean?

Left

The image/text is displayed to the left of the parent element

Right

The image/text is displayed to the right of the parent element

None

There is no change in the way the image/text is displayed

Overflow

You can control what an elements contents will do if it overflows it boundaries with the overflow property

overflow: value;

Values:

auto
hidden
visible
scroll

Overflow Example

As you can see, with this property you can mimic an iframe. This box is set to an overflow value of "auto". Meaning that if the contents of the element break the boundaries it should add a scrollbar.

Here is what I have in my CSS file.

#overflow_box {width:200px; height:200px; border-top: 1px solid #eee; border-

left: 1px solid #eee; border -bottom: 1px solid #eee; padding: 10px; overflow: auto;}

Then in the (X)HTML file I have this:

Visibility

You can control if an element is visible or not with the visibility property

visibility: value;

Values:

hidden
visible

Z-Index

You can control the layer order of positioned elements with the z-index property

z-index: value;

Values:

auto
number

The higher the number the higher the level. Negative numbers are allowed

Chapter 17 - CSS Positioning

Inherited: No

Position

The position property (as you may have guessed) changes how elements are positioned on your webpage.

position: value;

Values:

static relative
absolute fixed

Now, what does all that mean?

Static

Static positioning is by default the way an element will appear in the normal flow of your (X)HTML file. It is not necessary to declare a position of static. Doing so, is no different than not declaring it at all.

```
position: static;
```

Relative

Positioning an element relatively places the element in the normal flow of your (X)HTML file and then offsets it by some amount using the properties left, right, top and bottom. This may cause the element to overlap other elements that are on the page, which of course may be the effect that is required.

```
position: relative;
```

Absolute

Positioning an element absolutely, removes the element from the normal flow of your (X)HTML file, and positions it to the top left of it's nearest parent element that has a position declared other than static. If no parent element with a position other than static exists then it will be positioned from the top left of the browser window.

```
position: absolute;
```

Fixed

Positioning an element with the fixed value, is the same as absolute except the parent element is always the browser window. It makes no difference if the fixed element is nested inside other positioned elements.

Furthermore, an element that is positioned with a fixed value, will not scroll with the document. It will remain in it's position regardless of the scroll position of the page.

At this time IE6 (Internet Explorer 6) does not support the fixed value for the positioning of an element. Thus it will not position fixed elements correctly and will still scroll with the page. To see this effect in action you will need to use a standards compliant browser, such as Firefox 1.0

position: fixed;

When positioning elements with relative, absolute or fixed values the following properties are used to offset the element:

top
left
right
bottom

position: absolute; top: 10px; right: 10px;

Chapter 18 - CSS Pseudo Elements

The Syntax

The syntax for pseudo elements is a bit different than that of regular CSS, but it's real close. If you have already read chapter 11 then you are slightly ahead of the game.

selector:pseudo-element {property: value}

As you can see the only difference is that you place the pseudo element after the selector, and divide the 2 with a (:) colon.

Or you can assign a class to a pseudo element as follows

selector.p:pseudo-element {property: value}

Using the above code would style all paragraphs within the declared selector with the pseudo element.

The elements:

first-line

first-letter

First Line

The first-line pseudo element styles the first line of text in a block level element.

```
p{font-size: small;}
p:first-line {font-size: medium; color:    #ff0000;}
```

As you can see in the above example paragraphs are set to be a small font size, but the p:first-line is set to be a medium size and a red color. The result is that the first line of all paragraphs will be red in color and a bit larger than the rest of the paragraph.

Though lets say you only want to style a certain paragraph of text with the first-line element. Thats where declaring a class to the pseudo element comes into play.

first -line with class

```
p.special:first-line {font-size: medium; color:     #ff0000;}
```

I have declared a class of special within my css file.

First-Line Example

This is a special sentence I wrote to demonstrate the use and look of the first-line pseudo element. As you can see the first line of this paragraph is styled differently than the rest of the text within it. All of
this was done by simply adding class="special" to the opening <p> tag for this paragraph.

```
<p class="special">the content</p>
```

Where the first-line ends depends on the width of the browser window or containing element, you can resize this page and see that it adjusts as you change the size of the browser window.

The following properties can be assigned to the first-line pseudo element:

background
clear
color
font
letter-spacing line-
height text-
decoration text-
transform vertical-
align word-spacing

XML Basics

1 Introducing XML

What is XML?

XML stands for e**X**tensible **M**arkup **L**anguage. It is a markup language much like HTML, but there are several differences between them:

- HTML includes a collection of predefined tags that you can use right away in editing your HTML files, e.g. <font> and <h1>, but XML tags are not predefined.
  To use XML, users have to define their own tags for their specific application before using them. For example, to describe a note, <note>, <to>, <from>, <heading>, and <body> are defined in advance and used in a nested fashion to present the following XML file as a note:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don’t forget the party!</body>
</note>
```

With XML, one can define whatever tags needed, which together compose a user-defined markup language similar to HTML, and then use the language to describe data. Specif- ically XML uses *Document Type Definition* (DTD) or an XML Schema to define tags. In this sense, XML is viewed as a meta-language since it can be used to define and de- scribe a markup language instead of concrete data directly. That is also why it is called extensible.

- XML was designed to describe data while HTML was designed for displaying data.

  If you remember, HTML tags control the way data is presented to browser users, color, font size, spacing, etc. Differently XML aims to deal with the logic meaning of data, or semantics. In the above example, the text wrapped in <from> and </from> is the name

of the sender of the note. This enables the fulfillment of the task of finding all the notes written by a specific person. So XML was designed to describe data and to focus on what data is while HTML was designed to display data and to focus on how data looks.

Actually XML and HTML can complement each other. For example, we use XML files to store data on a web server machine and when a request arrives, a servlet runs to retrieve data in XML, compose a HTML file, and finally output it to the client. This way you can concentrate on using HTML for data layout and display, and be sure that changes in the underlying data will not require any changes to your HTML.

Besides XML's role of storing data, with XML, data can be exchanged between incom- patible systems.

In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet.

Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications, since XML data is stored in plain text format, which is software- and hardware-independent.

The best description of XML may be this: XML is a cross-platform, software and hardware independent tool for transmitting information. Since the creation of XML, it has been amazing to see how quickly the XML standard has been developed and how quickly a large number of software vendors have adopted the standard. It is strongly believed by the IT community at large that XML will be as important to the future of the Web as HTML has been to the foundation of the Web and that XML will be the most common tool for all data manipulation and data transmission.

XML Syntax

The syntax rules of XML are very simple and very strict. The rules are very easy to learn, and very easy to use. Because of this, creating software that can read and manipulate XML is very easy to do.

An Example XML Document

XML documents use a self-describing and simple syntax. For example, the following is a complete
XML file presenting a note:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The first line in the document – the XML declaration – defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

The next line and the last line describe the root element of the document (like it was saying: "this document is a note"). So when you look at the document, you easily detect this is a note to Tove from Jani. So XML is pretty self-descriptive.

Rigid Syntax

Different from HTML, XML has a syntax that is much more rigid.

- With XML, it is illegal to omit the closing tag.

  In HTML some elements do not have to have a closing tag to be able to present content in the way the user wants them to. For example:

  ```
  <p>This is a paragraph
  <p>This is another paragraph
  ```

  In XML, however, all elements must have a closing tag, like this:

  ```
  <p>This is a paragraph</p>
  <p>This is another paragraph</p>
  ```

  Note that you might have noticed from the previous example that the XML declaration did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself. It is not an XML element, and it should not have a closing tag.

- Unlike HTML, XML tags are case sensitive.

  With XML, the tag <Letter> is different from the tag <letter>.
  Opening and closing tags must therefore be written with the same case:

```
<Message>This is incorrect</message>
<message>This is correct</message>
```

- Improper nesting of tags makes no sense to XML.

  In HTML some elements can be improperly nested within each other and still display content in the desired way like this:

  ```
  <b><i>This text is bold and italic</b></i>
  ```

  In XML all elements must be properly nested within each other like this:

  ```
  <b><i>This text is bold and italic</i></b>
  ```

- All XML documents must contain a single tag pair to define a root element.

  All other elements must be within this root element. All elements can have sub elements (child elements). Sub elements must be correctly nested within their parent element:

  ```
  <root>
    <child>
      <subchild>. </subchild>
    </child>
  </root>
  ```

- With XML, it is illegal to omit quotation marks around attribute values.

  XML elements can have attributes in name/value pairs just like in HTML. In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <note date=12/11/2002> <to>Tove</to>

  <from>Jani</from>
  </note>
  ```

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <note date="12/11/2002"> <to>Tove</to>

  <from>Jani</from>
  </note>
  ```

The error in the first document is that the date attribute in the note element is not quoted.

- With XML, the white space in your document is not truncated. This is unlike HTML. With HTML, a sentence like this:

  Hello my name is Tove,

  will be displayed like this:
  Hello my name is Tove,

  because HTML strips off the white space.

- With XML, CR/LF is converted to LF.

  Do you know what a typewriter is? Well, a typewriter is a mechanical device used in the previous century to produce printed documents. :-)

  After you have typed one line of text on a typewriter, you have to manually return the printing carriage to the left margin position and manually feed the paper up one line.

  In Windows applications, a new line is normally stored as a pair of characters: carriage return (CR) and line feed (LF). The character pair bears some resemblance to the type-writer actions of setting a new line. In Unix applications, a new line is normally stored as a LF character. Macintosh applications use only a CR character to store a new line.

- Comments in XML The syntax for writing comments in XML is similar to that of HTML.

  <!-- This is a comment -->

However as you see, there is nothing special about XML. It is just plain text with the addition of some XML tags enclosed in angle brackets.

Software that can handle plain text can also handle XML. In a simple text editor, the XML tags will be visible and will not be handled specially.

In an XML-aware application, however, the XML tags can be handled specially. The tags may or may not be visible, or have a functional meaning, depending on the nature of the application.

XML Elements

XML elements define the framework of an XML document and the structure of data items.

XML Elements are Extensible

XML was invented to be extensible and XML documents can be extended to carry more information.

Take the above note as an example again. Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:

MESSAGE
To: Tove
From: Jani

Don’t forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

<note>
<date>2002-08-01</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don’t forget me this weekend!</body>
</note>

Should the application break or crash? No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output. XML documents are Extensible.

XML Elements have Relationships

Elements are related as parents and children.

To understand XML terminology, you have to know how relationships between XML elements are named, and how element content is described.

Imagine that this is a description of a book:
My First XML

Introduction to XML

- What is HTML
- What is

XML XML

Syntax

- Elements must have a closing tag
- Elements must be properly nested Imagine

that this XML document describes the book:

```
<book>
<title>My First XML</title>
<prod id="33-657" media="paper"></prod>
<chapter>Introduction to XML
<para>What is HTML</para>
<para>What is XML</para>
</chapter>
<chapter>XML Syntax
<para>Elements must have a closing tag</para>
<para>Elements must be properly nested</para>
</chapter>
</book>
```

book is the root element. title, prod, and chapter are child elements of book. book is the parent element of title, prod, and chapter. title, prod, and chapter are siblings (or sister elements) because they have the same parent.

Elements have Content

Elements can have different content types.

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

An element can have *element content*, *mixed* content, *simple* content, or *empty* content. An element can also have *attributes*.

In the example above, book has element content, because it contains other elements. chapter has mixed content because it contains both text and other elements. para has simple content (or text content) because it contains only text. prod has empty content, because it carries no information.

In the example above only the prod element has attributes. The attribute named id has the value "33-657". The attribute named media has the value "paper".

Element Naming

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML or Xml ..)

- Names cannot contain spaces

Take care when you “invent” element names and follow these simple rules:

Any name can be used, no words are reserved, but the idea is to make names descriptive. Names with an underscore separator are nice, for example <first name> and <last name>.

Avoid “-” and “.” in names. For example, if you name something “first-name,” it could be a mess if your software tries to subtract name from first. Or if you name something “first.name,” your software may think that “name” is a property of the object ”first.”

Element names can be as long as you like, but don’t exaggerate. Names should be short and simple, like this: <book title> not like this: <the title of the book>.

XML documents often have a corresponding database, in which fields exist corresponding to elements in the XML document. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like o˙¸c are perfectly legal in XML element names, but watch out for problems if your software vendor doesn’t support them.

The “:” should not be used in element names because it is reserved to be used for something called namespaces.

XML Attributes

XML elements can have attributes in the start tag, just like HTML. Attributes are used to provide additional information about elements.

From HTML you will remember this: <IMG SRC="computer.gif">. The SRC attribute provides additional information about the IMG element.

In HTML (and in XML) attributes provide additional information about elements:

```
<img src="computer.gif">
<a href="demo.asp">
```

Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

```
<file type="gif">computer.gif</file>
```

As we said before, attribute values in XML must always be enclosed in quotes, but either single or double quotes can be used. Note that if the attribute value itself contains double quotes it is necessary to use single quotes and vice versa, like in this example:

```
<gangster name=’George "Shotgun" Ziegler’>
```

Sometimes information may be placed as attributes or child elements. Take a look at these examples:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

and

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

In the first example sex is an attribute. In the last, sex is a child element. Both examples provide the same information.

There are no rules about when to use attributes, and when to use child elements. Generally attributes are handy in HTML, but in XML you should try to avoid them. Use child elements if the information feels like data.

Here are some of the problems using attributes:

- attributes cannot contain multiple values (child elements can)
- attributes are not easily expandable (for future changes)
- attributes cannot describe structures (child elements can)
- attributes are more difficult to manipulate by program code
- attribute values are not easy to test against a Document Type Definition (DTD) - which is used to define the legal elements of an XML document

If you use attributes as containers for data, you end up with documents that are difficult to read and maintain. Try to use elements to describe data. Use attributes only to provide information that is not relevant to the data.

Don’t end up like this (if you think this looks like XML, you have not understood the point):

```
<note day="12" month="11" year="2002" to="Tove" from="Jani"
heading="Reminder" body="Don’t forget me this weekend!">
</note>
```

XML Validation

Similar to HTML, XML with correct syntax is *Well Formed XML*. That is a well formed XML document is a document that conforms to the XML syntax rules that were described in the previous sections.

More specifically, to be well formed, an XML document must be validated against a *Document Type Definition* (DTD). The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. a DTD can be specified internally or externally. The following is an example of internal DTD for the above note example:

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to  (#PCDATA)>
                        (#PCDATA)
  <!ELEMENT from    >
  <!ELEMENT heading
  (#PCDATA)>
                        (#PCDATA)
  <!ELEMENT body    >
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don’t forget me this weekend</body>
</note>
```

The DTD above is interpreted like this: !DOCTYPE note (in line 2) defines that this is a document of the type note. !ELEMENT note (in line 3) defines the note element as having four elements: “to,from,heading,body”. !ELEMENT to (in line 4) defines the to element to be of the type “#PCDATA”. !ELEMENT from (in line 5) defines the from element to be of the type “#PCDATA” and so on ...

If the DTD is external to your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<?xml   version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don’t forget me this weekend!</body>
</note>
```

And the following is a copy of the file "note.dtd" containing the DTD:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

W3C supports an alternative to DTD called *XML Schema*. If interested, you may read more about XML Schema in related books.

The W3C XML specification states that a program should not continue to process an XML document if it finds a validation error. The reason is that XML software should be easy to write, and that all XML documents should be compatible.

With HTML it was possible to create documents with lots of errors (like when you forget an end tag). One of the main reasons that HTML browsers are so big and incompatible, is that they have their own ways to figure out what a document should look like when they encounter an HTML error.

With XML this should not be possible.

Viewing XML Files

To view an XML document in IE 5.0 (and higher) you can click on a link, type the URL in the address bar, or double-click on the name of an XML file in a files folder. If you open an XML document in IE, it will display the document with color coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. If you want to view the raw XML source, you must select "View Source" from the browser menu.

To view an XML document in Netscape 6, you'll have to open the XML file and then right-click in XML file and select "View Page Source". If you open an XML document in Netscape 6, it will display the document with color coded root and child elements.

If an erroneous XML file is opened, the browser will report the error.

Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.

Without any information about how to display the data, most browsers will just display the XML document as it is. In the next section, we will take a look at different solutions to the display problem, using CSS and XSL.

Displaying XML with CSS

Before we have learned that CSS files may work together with HTML files in the way that the former is in charge of display and the latter provides concrete information. CSS can do the same thing with XML.

Below is a fraction of the XML file, with an added CSS style sheet reference. The second line, <?xml-stylesheet type="text/css" href="cd catalog.css"?>, links the XML file to the CSS file:

```
<?xml version="1.0" encoding="ISO-8859-1"?> <?xml-stylesheet
type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  .
  .
  .
  .
</CATALOG>
```

The CSS file cd_catalog.css goes as follows:

```
CATALOG {
    background-color:
    #ffffff; width: 100%;
  }

  CD {
    display: block;
    margin-bottom:
    30pt; margin-
    left: 0;
  }
```

```
TITLE {
  color:
  #FF0000;
  font-size:
  20pt;
}

ARTIST {

  color:
  #0000FF;
  font-size:
  20pt;
}

COUNTRY,PRICE,YEAR,COMPANY {
  display:
  block; color:
  #000000;
  margin-left:
  20pt;
}
```

where it is specified how to display each kind of elements.

Displaying XML with XSL

Besides CSS, XSL was invented just for displaying XML. The ***eXtensible Stylesheet Language*** (XSL) is far more sophisticated than CSS. XSL

consists of three parts:

- XSLT (XSL Transformations) is a language for transforming XML documents.

  XSLT is the most important part of the XSL Standards. It is the part of XSL that is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.

  XSLT can also add new elements into the output file, or remove elements. It can rearrange and sort elements, and test and make decisions about which elements to display, and a lot more.

  A common way to describe the transformation process is to say that XSLT transforms an XML source tree into an XML result tree.

XSLT uses XPath to define the matching patterns for transformations. In the transformation process, XSLT uses XPath to define parts of the source document that match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document. The parts of the source document that do not match a template will end up unmodified in the result document.

- XPath is a language for defining parts of an XML document.

  XPath uses path expressions to identify nodes in an XML document. These path expres- sions look very much like the expressions you see when you work with a computer file system:

  w3schools/xpath/default.asp

  Look at the following simple XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?> <catalog> <cd
  country="USA">
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <price>9.90</price>
  </cd>
  <cd country="USA">
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <price>9.90</price>
  </cd>
</catalog>
```

  The XPath expression below selects the ROOT element catalog:

  /catalog

  The XPath expression below selects all the cd elements of the catalog element:

  /catalog/cd

  The XPath expression below selects all the price elements of all the cd elements of the catalog element:

  /catalog/cd/price

  Note: If the path starts with a slash (‘/’) it represents an absolute path to an element!

XPath also defines a library of standard functions for working with strings, numbers and Boolean expressions. The XPath expression below selects all the cd elements that have a price element with a value larger than 10.80:

/catalog/cd[price>10.80]

- XSL-FO is a language for formatting XML documents.

Think of XSL as set of languages that can transform XML into XHTML, filter and sort XML data, define parts of an XML document, format XML data based on the data value, like displaying negative numbers in red, and output XML data to different media, like screens, paper, or voice.

One way to use XSL is to transform XML into HTML before it is displayed by the browser. Be- low is a fraction of the XML file, with an added XSL reference. The second line, <?xml-stylesheet type="text/xsl" href="simple.xsl"?>, links the XML file to the XSL file:

```
<?xml version="1.0" encoding="ISO-8859-1"?> <?xml-
stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast_menu>

  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
       two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

And the corresponding XSL file is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?> <!--
Edited with XML Spy v4.2 -->
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://www.w3.org/TR/xhtml1/strict">
  <body style="font-family:Arial,helvetica,sans-serif;font-
        size:12pt; background-color:#EEEEEE">
    <xsl:for-each select="breakfast_menu/food">
      <div style="background-color:teal;color:white;padding:4px"> <span
        style="font-weight:bold;color:white"> <xsl:value-of
        select="name"/></span>

        - <xsl:value-of select="price"/> </div>

      <div style="margin-left:20px;margin-bottom:1em;font-size:10pt">
        <xsl:value-of select="description"/>
```

```
        <span style="font-style:italic">
          (<xsl:value-of select="calories"/> calories per serving) </span>

      </div>
    </xsl:for-each>
  </body>
</html>
```

2 Programming with XML

Since Java has been commonly used for processing XML files, this section presents one popular Java-based method of parsing XML: the *Document Object Model* (DOM).

DOM represents an entire XML document in a tree-like data structure that can be easily manipulated by a Java program. The advantage of DOM is that it is relatively simple to use and you can modify the data structure in addition to extracting data from it. However, the disadvantage is that DOM always parses and stores the entire document, even if you only care about part of it. The *Simple API for XML* (SAX) is an alternative to avoid this problem.

To use DOM, you need have a DOM-compliant parser. A list of such parsers are given at http://www.xml.com/pub/rg/Java_Parsers. Besides, the *Java API for XML Processing* (JAXP) is needed and available from http://java.sun.com/. This API provides a small layer on top of DOM that lets you plug in different vendor's parsers without making any changes to your basic code.

The use of DOM goes as follows:

1. Tell the system which parser you want to use. There are many ways to do so, of which the system property is the easiest method. For example the following code permits users to specify the parser on the command line with -D option to java, and uses the Apache Xerces parser otherwise.

```
public static void main(String args[]) { String
  jaxPropertyName =
  "javax.xml.parsers.DocumentBuilderFactory"; if
  (System.getProperty(jaxPropertyName) == null) {

    String apacheXercesPropertyValue =
      "org.apache.xerces.jaxp.DocumentBuilderFactoryImpl";
            System.setProperty(jaxPropertyName, apacheXercesPropertyValue);
  }
  ...
}
```

2. Create a JAXP document builder. This is basically a wrapper around a specific XML parser.

```
    DocumentBuilderFactory builderFactory =
      DocumentBuilderFactory.newInstance();
DocumentBuilder builder =
  builderFactory.newDocumentBuilder();
```

3. Invoke the parser to create a Document representing an XML

```
document. Document document =

builder.parse(someInputStream);
```

4. Normalize the tree.

```
document.getDocumentElement().normalize();
```

5. Obtain the root node of the tree. This returns Element, which is a subclass of the more general Node class that represents an XML element.

```
Element rootElement = document.getDocumentElement();
```

6. Examine various properties of the node. Various methods are available for access: getNodeName(), getNodeValue(), getAttributes(), getChildNodes(), etc..

As you may image, you may simply replace the Property format configuration file for your own web server with an XML one, and use the above process to access the configuration information.

Dr. MPS College of Business Studies

Subject: Introduction to HTML	
Maximum Marks: 30	
Class: BCA	Semester: 1
Type: 1st Internal Examination – October 2019	Code: C104

Note: All Questions are Compulsory

Q1. What is world wide web? Differentiate between www and internet.

Q2. What is SMTP. Explain its working.

Q3. Make a webpage demonstrating the use of paragraphs, links, images and tables.

QUESTION BANK

What is HTML?

a)Hyper Text Marking Language b) Hyper Text Machine Language c)Hyper Text Middle Language d)Hyper Text Markup Language

1. In email address _____ character is essential

a) _ b) % c)@ d) *

LAN stand for _____

Local Area Network b) Linear Area Network c) Link Area Network d) None of these

MAN stand for _____

a)Main Area Network b) Middle Area Network c)Metropolitan Area Network d) Mix Area Network

WAN stand for _____

Wide Area Network b) Web Area Network c) World Area Network d) Width Area Network

FTP Stands for --------

Field Transfer Protocol b)File Transfer Protocol c)
Fix Transfer Protocol d) Flexible Transfer Protocol

Which tag is used to create a list that displays items with bullets?

<OL> b)<UL> c)<list> d)<DL>

Write a code for making text area?

<textarea> b) <input =”textarea”> c) <input type =”textbox”> d) <input type =”text area”>

Marquee tag has following attributes

a)behaviour b)direction c)position d)both a and b

For largest level of heading which tag is used?

<h1> b) <h6> c) <heading> d) <head>

To group sentences in paragraph which tag is used?

<para> b) <p> c) <b> d)

The language used to develop web page is ____

HTML b) HTTP c) Brower d)Protocol

Font tag has following attribute

type b) face c) value d) caption

Java Script is ____ side scripting language

Server b)Client c)Server & Client d) None of these

API stands for

a)Application Programming Interface b) Application Programming Index c) Applet

3. For handling user interaction ___side scripting is useful

Server b)Client c)Server & Client d) None of these

4. _____ Scripting is used for data entry validation

Client side b) Server side c) Client side & Server side d) None of these

5. _____ event occurs when the user clicks on a link or a form element

focus b) load c) change d) click

6. JAD stands for

Joint Application Development b) Joint Application Difference c) Joint Application Demo d) None of these

7. Primary aspects of Web design are content, Technology, economics & ___

visuals b) site c) person d) graphics

8. Form tag has 2 attributes

action & method b) form & action c) method & form d) all of these

9. _____ button send the current information held in each field of the form to the web server for further processing.'
a)submit b) reset c) normal d) all of these

10. Java script is scripting language introduce by

IE b) Microsoft c) Netscape d) SunJava

11. In table row is created by

<tr> b) <th> c) <ta> d) None of these

12. _____ tag is used for document header

<title> b) <head> c) <html> d) <body>

5. For inline images _____ tag is used in an HTML document a) <input> b) <map> c) <img> d) None of these
6. _____ are used to create imaginary sections of browser screen.

form b) frame c) input d) select

7. _____ value of method attribute send information to the server. a)POST b)GET c)form d)none of these
8. _____ provides the middleware between www server & external databases & information sources.
Perl b) CGI c) Java Script d) None of these

9. ____ tag is used with <option> tag to give list of items.

<input> b) <frame> c)<select> d) a and b

10. ____ element is used to check one element at a time.

Radio b) Menu c) List d)Checkbox

11. The word internet stands for ____

Internation Network b) Internal Network c) Intermediate Network d) none of these

12. _____ is used to set background image at web page.

a)<body background> b) <body bg ground> c)body back> d) None of these

36._____ is used to create a new paragraph

a)<form> b) <img> c) <P> d) <marque>

5. _____ is the text on clicking of which other web pages are opened

Markup b) Hypertext c) Link d) None of these

6. _____ tag is used to format the text matter

<format> b) <font >c)<form> d) <frame>

39 _____ tag is used to link a web page.

<a> b) <b> c) <td> d) <tr>

- _____ tag is used to display web pages information such as URL, authors name etc.

<address> b) <pre> c) d) <frame>

- _____ tag is used to create a caption on top or below the table.

<table> b) <caption> c) <align> d) <border>