

Web Development and Database Administration Level I

Based on March 2022, Curriculum Version 1

Company Logo	Company Name			
Home	Services	Products	About Us	Search Box
Page specific information				Social Media Icons as required
Privacy	Copyright	Terms and Conditions	Site Map	

Module Title: **Creating a Simple Markup Language Document**

Module Code: **EIS WDDBA1 M10 0322**

Nominal Duration: **70 Hours**

Table of Contents

Acknowledgment	3
Acronym	4
Introduction to the Module	5
Unit One: Review Requirements	6
1.1. Reviewing Document Requirements	7
1.2. Selecting Markup Language Based on Organizational Standards	10
1.3. Reviewing Document Structure	14
Self Check - 1	21
Operation Sheet - 1	22
Lap Test - 1	23
Unit Two: Create Document Structure	25
2.1. Introduction to HTML	26
2.2. Creating Basic Elements of Document	29
2.3. Depicting Document Structure of Markup Sections	36
2.4. Writing Simple Markup Language	54
Self Check - 2	72
Operation Sheet - 2.1	73
Operation Sheet - 3.2	74
Operation Sheet - 4.3	75
Lap Test - 2	76
Unit Three: Validate Documents	77
3.1. Validating Markup Language Document	78
3.2. Validating Markup Language Document in Different Browsers	81
3.3. Validating Simple Markup Language Document	86
Self Check - 3	87
Operation Sheet - 5	88
Lap Test - 3	89
Reference	90
Developers Profile	91

Acknowledgment

Ministry of Labor and Skills wish to extend thanks and appreciation to the many representatives of TVET instructors and respective industry experts who donated their time and expertise to the development of this Teaching, Training and Learning Materials (TTLM).

Page 3 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Acronym

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
XHTML	Extensible HyperText Markup Language
XML	eXtensible Mark-up Language
WML	Wireless Mark-up Language
DHTML	Dynamic HTML
SGML	Standard Generalized Mark-up Language
SVG	Scalable Vector Graphics
XBRL	eXtensible Business Reporting Language

Introduction to the Module

This module describes the performance outcomes, skills and knowledge required to design, create and save a basic markup language document using text editor.

This module covers the units :

- Review requirements
- Create document structure
- Validate documents

Learning Objective of the Module

- Review requirements
- Create document structure
- Validate documents

Module Instruction

For effective use this modules trainees are expected to follow the following module instruction:

1. Read the information written in each unit
2. Accomplish the Self-checks at the end of each unit
3. Perform Operation Sheets which were provided at the end of units
4. Do the “LAP test” giver at the end of each unit and
5. Read the identified reference book for Examples and exercise

Page 5 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Unit One: Review Requirements

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Introduction to Mark-up language
- Reviewing document requirements
- Selecting Mark-up language based on organizational standards
- Reviewing document structure

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Review the requirements of the document
- Select the appropriate Mark-up language based on organizational standards
- Review document structure

1.1. Reviewing Document Requirements

The first step in building a simple website is to clarify why the website is needed and what functions it needs to perform.

You need to understand the purpose of the website, the audience you are trying to attract and the design elements that will be needed to meet these requirements. This information is included in a requirements (or design) document.

Web site development includes steps to ensure that the site content meets the client's needs. An important final check is then done after development, before presentation to the client, to find and resolve any last minute problems.

A web site is the public face of an organisation; its quality is seen to reflect the quality of the organisation itself. The reputation of an organisation can easily be damaged by a thing as seemingly-trivial as a broken link, spelling mistakes or inconsistent text. The consistency test, as part of that final check, is usually the final opportunity to ensure client needs are met.

The standards against which a web site might be assessed are usually found in project documentation such as:

- The project brief
- Style guide
- Technical specifications.

Project documentation is developed in conjunction with the client and it outlines what the client wants and the product should exist on completion. You should use project documentation as the basis for designing your quality assurance checklist, while also including test criteria to ensure that the web site complies with relevant web standards, such as those pertaining to HTML, cascading style sheets (CSS), and the standards recommended for accessibility.

1.1.1 Project brief

The project brief is used to work out, define and clarify the client's needs, and it is devised once a contract is in place to develop a web site. Once the project brief is agreed to, the final product must conform to it. To identify client needs the brief will answer questions such as:

- What is the purpose of the site?
- How is the 'brand' or image of the organisation conveyed?
- What is the user profile for the site?

Page 7 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

- What are the needs of the site’s users or customers?
- What are the deadlines or milestones for developing the site?
- What is the environment of the web site users likely to be?

The project brief is also the starting point for a final check of the site.

1.1.2 Style guide

The style guide describes the desired ‘look and feel’ of the web site and lists the elements needed to consistently produce the desired effect.

The project manager or team will have consulted with the client about the vision they have for the web site. Often a web site’s design or look will be linked to the corporate style of the organisation. For example, a bank that targets young and single customers might project a ‘cool’ youthful image and want an internet site with a funky look and feel, but will still use corporate colours and logos. The bank’s intranet site, which would be primarily used by staff, on the other hand, might be more formal and plain, but also retain the corporate colours and logos.

The style guide will set out requirements for design elements such as:

- colours, fonts and page layout
- the placement of logos or other significant graphics
- the style of multimedia and graphics.

Attributes of text-based content that might be assessed against client requirements include:

- font type, size and colour for body text and headings
- colour of text hyperlinks
- colour of visited link
- colour of active links
- leading or line height
- kerning
- text justification.

1.1.3 Technical specifications

Technical specifications describe the technology the web site requires or should work on. The quality assurance process must ensure that the web site functions correctly on each combination of the technology specified. It may list requirements such as those outlined in Table 1 to follow.

Table 1.1: Examples of requirements for technical specifications

Client requirements	Server requirements	Security model for system
<ul style="list-style-type: none"> • Preferred web browsers • Operating systems • Memory • Internet connection speeds (min) • Preferred resolution • Plug-ins 	<ul style="list-style-type: none"> • Software • Hardware • Backup • Administration • Redundancy and recovery 	<ul style="list-style-type: none"> • User profiles • Roles • Permission lists • Operating system security
Programming tools	Programming languages	
<ul style="list-style-type: none"> • Microsoft Visual Studio • Sun Microsystems Java Studio 	<ul style="list-style-type: none"> • Cold fusion • Visual basic • Cascading style sheets • JavaScript 	

1.2. Selecting Markup Language Based on Organizational Standards

This section will look at the different markup languages available for website development and cover things to consider in the design of a website.

What is markup language?

A markup language is a combination of text and information describing the text. This extra information explains how the text should be displayed on a page. Markup languages for the web use tags to tell a browser how to display text on a page.

1.2.1 Markup Languages for the Web

A. HTML (HyperText Mark-up Language)

HTML is a very simple language used to describe the logical structure and layout of a web document. It describes which parts of the text the web browser should emphasize, which text should be considered body text and which text should be headings.

B. XML (Extensible Mark-up Language)

This very versatile metalanguage (a language that describes another language) is used to describe what data is rather than just the way it looks. This means it is a semantic language rather than just a presentation language (like HTML).

XML lets the coder create their own tags to describe the data. XML is a complement to HTML. It can be used to describe the data, while HTML can be used to format and display the data. Languages for handheld devices such as mobile phones are based on XML.

C. XHTML (Extensible HyperText Markup Language)

XHTML is a combination of HTML and XML. XHTML has begun to replace HTML. It is a stricter and cleaner version of HTML. In fact the World Wide Web Consortium defines XHTML as the latest version of HTML and it is almost identical to HTML 4.01.

D. DHTML (Dynamic HTML)

Not really a language in its own right, DHTML is actually a combination of HTML (or XHTML), JavaScript and Cascading Style Sheets. It is used to create web pages with dynamic content such as animation, pop-up windows and drop down menus. A DHTML web page can produce a response to a user's action (such as a mouse click) without having to communicate with the server to have the page resent to the user's browser.

E. WML (Wireless Mark-up Language)

Used for handheld devices such as mobile phones and PDAs (Personal Digital Assistants).

Page 10 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

F. SGML (Standard Generalized Mark-up Language)

SGML is a metalanguage – perhaps ‘the’ metalanguage, as all web mark-up languages are based on it. HTML, XML and others are actually simplified applications of SGML, which is very difficult to code with.

G. Other mark-up languages

There are many other specialised mark-up languages in use—each has been specifically designed for a particular use. Some examples include:

- **MathML**—an application of XML for representing mathematical symbols and formulae
- **DocBook**—for technical documentation
- **SVG** (Scalable Vector Graphics)—an application of XML for representing two-dimensional vector graphics
- **Open eBook**—standard e-book format based on the XML format
- **XBRL** (eXtensible Business Reporting Language)—an emerging XML-based standard to define and exchange business and financial performance information
- **MusicXML**—an open, XML-based music notation file format
- **RSS**—a family of XML file formats for Web syndication used by (among other things) news websites and weblogs.

1.2.2 Choosing your mark-up language

With so many mark-up languages to choose from, how do you know which is the right one for your web document? As with any project you undertake, you should start by looking at what is needed and the options available to you.

A. *Website purpose*

The key element in choosing an appropriate mark-up language will be the purpose of the website.

If you are building a single ‘brochure-style’ website that displays straightforward information including graphics, some Flash animation and maybe some sound, you could use straightforward HTML. Most smaller websites are constructed this way.

If however, your site contains financial or scientific information, you may consider employing MathML in your site. Writing formulae (for example fractions and math equations) will be faster in a specialised XML-friendly application such as ‘MathType’ (from Design Science: www.dessci.com) than trying to code in HTML.

Page 11 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

If your site relies heavily on graphic design and animated menus and buttons you may consider using DHTML to liven up the user experience with interactive elements.

The flexibility of XML allows developers to create their own unique tags and classes of information. This is one of the reasons that XML in various forms is used in the creation of many larger dynamic websites.

Remember that your mark-up language will work in tandem with the programming language you choose (such as JavaScript, PHP, etc.) This aspect of web design is not covered in this learning pack.

B. Stakeholders

Stakeholders are people who will be affected by your website and can influence it but who may not be directly involved with doing the work. You must take the needs of stakeholders into account when you are planning a website and choosing a mark-up language.

If you are employed by an organisation to build a web page the stakeholders may include:

- Management
- Marketing
- Technical support staff
- Legal advisors
- Audience of website users

Some or all of these people will help determine the specifications and appropriate mark-up language by describing the kind of functions the website needs to perform. Access to technical resources may also mean access to people with additional programming skills as well—can you use these people in your web project?

Even if you are building a personal website just to show holiday photos to your distant relatives, there are still stakeholders to consider—mainly your relatives. What are their needs?

Who are the stakeholders in your website?

C. Audience needs

Think about the audience of your web site. There are a range of basic questions you should ask when planning a new site—these may include the following:

- Who are your website users?
- What do your users want to do on your website (eg purchase goods, make bookings, find information, etc.)?
- How comfortable are they with using the web?

Page 12 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

- What kind of computer will your user have?
- How will accessibility for vision-impaired users be addressed?
- What about accessibility for users with slow modem connections?
- Are there any business advantages in using features such as animation and sound?
- Will the site need to be accessed using a range of devices (for example handheld devices or mobile phones)?
- If your site requires the latest plug-ins (eg Flash) how will the user find them?

Determining how to meet your audience’s needs and expectations will help to define the specifications and type of mark-up language you will need to use on your site.

D. Standards

You will need to comply with certain web standards if you want your site to be accessed by the broadest possible range of users. Remember that you users will have a range of computer platforms, operating systems and web browsers. The manufacturers of these different systems will have made efforts to comply with agreed web standards.

Standards for the web are developed by the **World Wide Web Consortium** (W3C)—an international group including input from member organisations, full-time staff, and the public. W3C’s mission is: ‘To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web’. The W3C website (www.w3.org) is the hub for standards information and includes technical guidelines, educational information and code-checking tools called ‘validators’ that allow developers to ensure their sites meet required web standards.

You may also want to look at examples of current ‘best practice’ in web design. Take note of websites that you like or are similar in function to the site you intend to build—think about the technologies and mark-up they employ and their useability. Also take a look at sites showcasing recent designs such as:

- Webby Awards: www.webbyawards.com
- Cool Home Pages: www.coolhomepages.com
- NetGuide Australian Web Awards: www.netguide.com.au

Page 13 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

E. Project constraints

Your choice of mark-up language will inevitably be affected by the resources you have at your disposal. You will need assess and balance the amount of money, resources, time etc. that you have available for your project. Starting up an XML project from scratch will require more resources and planning than building a simple HTML site. However if you build in HTML because of initial time constraints, but the site really requires XML and database integration, you may have wasted your effort and need to rebuild your site anyway.

This is why it is important to fully assess you stakeholders’ and users’ needs before you start.

F. Available technology and skills

In a similar vein you need to assess what technology and skills you have available to you. You may personally be a competent HTML programmer but have no knowledge of MathML or RSS implementation. This does not necessarily make HTML the right choice for your project.

With a thorough analysis of the site requirements you may have identified that MathML and RSS would greatly benefit the site. Use this to argue for more resources from within your organisation or of you are working on your own—seek advice from programmers who are familiar with these technologies and employ them if you can.

If you want to learn the required skills yourself and have the time, start a small project to implement these skills. Take a course or make use of the extensive range of online tutorials and references available free on the web. Software that assists in implementing different mark-up languages (such as Dreamweaver: www.macromedia.com) often have tutorials built in to help you get started.

1.3. Reviewing Document Structure

Once you have made a decision about the markup language you will use to create a website, you need to consider how the website will be designed. There are two aspects of website design: the folder structure for the website and the storyboard for the website.

1.3.1 Folder Structure

Before starting out on a website project, you should create a folder structure to contain all the files for your website. It is essential that all your web page files, graphics and other files are collected in the same place right from the start of your project.

Page 14 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Minimum best practice for website development is that you have a 'root' folder to store your pages and a separate folder under the 'root' folder, usually called images or graphics, where you store all the graphics you use on the website. This minimum structure allows you to move the website to a different location (for example your USB storage device or a web server) without causing the links on your pages to stop working.

The larger the website the more complex the folder structure will be. Often there will be folders to store related pages, style sheet files and other files such as animations.

1.3.2 Website Storyboard

It is best practice to create website storyboard before you start creating individual web pages. A web storyboard is an outline of the pages required for the site (also called a website structure) and the layout, content, navigation and design elements of the pages (also called page structure).

Each page in your site should have a descriptive file name (see example storyboard below) and the home page should be called index.html or default.html.

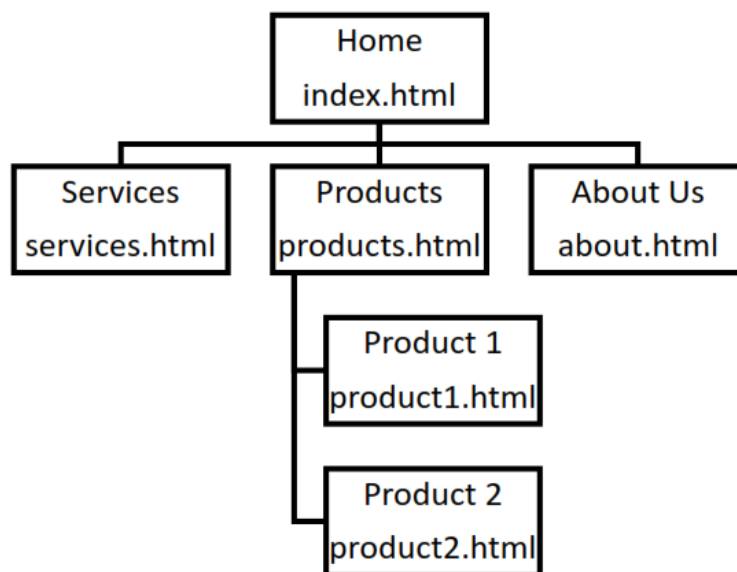


Figure 1.1 Simple Site Structure

Figure 1.2 below shows the layout, content, navigation and design elements of the pages. This was also created using the drawing tools in Microsoft Word.

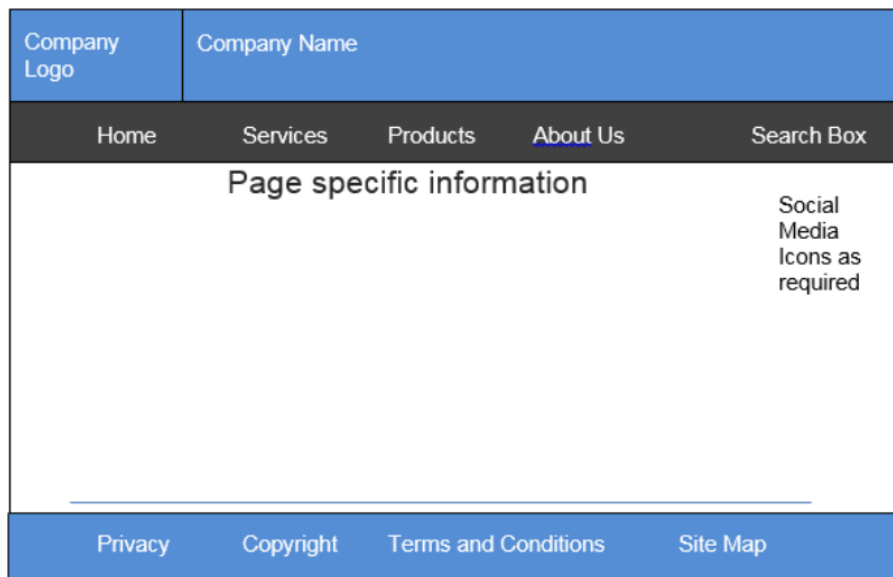


Figure 1.2 Web Page Layout

1.3.3 Absolute and relative addressing

When you create a link to a web page, or an image, or any file on the web, you could specify the complete address of the file, such as:

```
<a href="http://www.example.com/users/jbrown/pics.htm">
```

This is an **absolute** address because the file can be located from anywhere on the Web. You would use this form of addressing when the link is to a site somewhere outside of the server you are using (i.e. it is on someone else's website).

If the page you wish to link to is located in the same folder as the originating page, you would use **relative** addressing. For example, if you were the user (jbrown) in the example above, and you wanted to link from your own "home.htm" page to your "pics.htm" file in the same folder, a shorter way to refer to that HTML file would be:

```
<a href="pics.htm">
```

This is a relative address because it works as long as the new page is in the same folder on the same server relative to the originating page. It is not complete enough to be used from anywhere else on the Web.

Another common form of relative addressing is linking to an image:

```

```


This address says, "in the folder where you are, go into the folder called "images", and find the file called "picture_01.gif".

How do you go up a folder level? Take a look at the following example of a website folder structure:

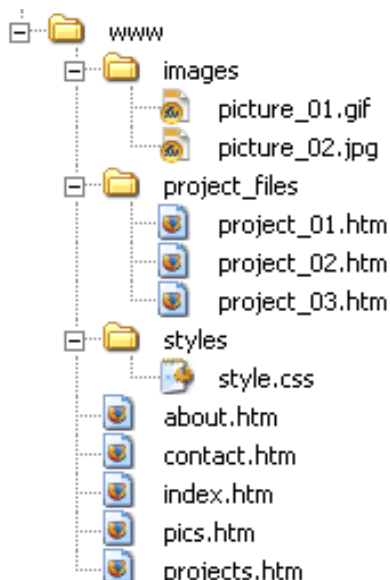


Figure 1.2: Folders within a website folder.

If you want to link from "project_01.htm" to "contact.htm", the link would look like this:

```
<a href="../contact.htm">
```

This address says "starting where you are, go up one level in the directories (that's the "../"), and find the file "contact.htm".

If you wanted link from "project_01.htm" in the "project_files" folder to a picture in the "images" folder, the link would look like this:

```

```

This link says "go up one level and then look in the "images" folder for the file "picture_01.gif"

In addition, to direct the browser to start at the **top** level of the server's directory structure **regardless of where the page you are linking from may be**, use "/" as shown in this example:

```
<a href="/project_files/project_01.htm">
```

This address says

- start at the top level of this website (that's the "/" symbol) and refers to the "www" folder.

Page 17 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

- go into the folder called "project_files"
- find the file called "project_01.htm"

Note that this reference works from any page on this server (but not on the web), because it starts by returning to the top level (the folder called "www" in the example above). This is a useful technique on websites where there are many files and folders and where HTML files may be moved around (for example, archiving news items)

1.3.4 Keeping your web site together

File and folder organisation is crucial to building a website. Your website must have a logical structure that does not change when you move files from your own computer to the web server where your files will be available on the web. The main reason is that if a file moves in relation to another file that contains a link to it, that link will break.

Creating Folders

Take a look at this sample of a website folder structure (it was created in Windows XP but the principle of folder structures will remain the same across all operating systems):

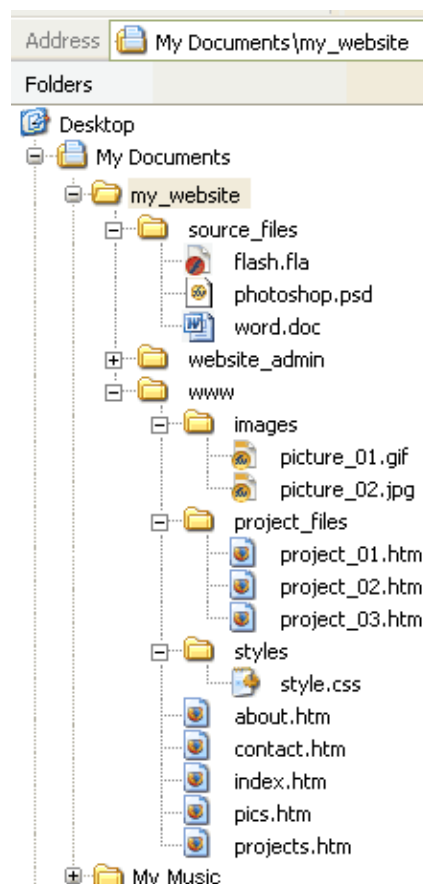


Figure 1.3: Illustration of folder structure for a simple website.

Page 18 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

When building a web site, it is very important to think about keeping your own pages, images, and other files as a unit. You must collect all of those files together into a single folder. This is known as the "root" folder or "root directory". In the example above the website root folder "www" is contained inside a bigger project folder called "my_website".

Within the website root folder "www", there may be other folders containing the images and media files. The root folder will contain only the files you need your viewer to see on your website. On the web server where the files will be made available on the internet, there will be no "my_website" folder. The web server will contain only the files in the "www" folder.

You can control where these files are located in relation to each other. These links to your own files will be relative links. You should be able to pick up your entire root folder ("www") full of files and perhaps other folders, move it to another server and all the links within the site would still work. None of the links can refer to the name of the server where the files live.

You could test this by saving your root folder onto a USB flash memory stick or burning it to a CD and taking it to another computer - once you open the home page in a browser, all the other links should still work.

Make sure you collect all of your files together into a single root folder ("www") and organise them first **before starting to make links between them**. If you link to an image and then move it into a different folder, that link will break.

If you are pointing to someone else's file on another website, you have to indicate the complete, absolute address of the file, since it most likely resides on another server.

Source files

Remember that your root folder ("www" in the example above) should contain only the files that need to be made available on the website.

Your "source" files should be saved outside your website folder and do not need to be transferred to the web server. In the example above the project folder "my_website" contains not only the website folder "www" but also a folder for "source_files" and a folder for "website_admin".

Source files can include:

- Administration files (contracts, emails etc)
- Original Word documents (source documents for information)
- Original graphics files (Photoshop files, camera files etc.)

Page 19 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Using graphics as an example, digital camera files are often over 2MB in size. You will need to reduce these significantly to fit on your web pages. The "optimised" files go in the "images" folder before you link to them - the originals do not need to be saved in your website folder - so you save them in the "source_files" folder. Similarly if you are creating graphics in Photoshop or Flash, the working files are usually quite large - you will use only the exported files in your site.

1.3.5 Naming your files

There are many different types of computers and servers used across the web. You need to make sure that your file names will be read the same on all systems. To do this there are a few simple rules which you must follow when creating files and folders for websites:

No spaces: Never use spaces in your file names - HTML will not read the code correctly. For example, a file called "my pictures.htm" may be changed to "my%20pictures.htm" ("%20" is code for a blank space) and your links may not work!

Use underscores or remove spaces: Call your file "my_pictures.htm" or "mypictures.htm"

Use lowercase: Most web servers are case-sensitive and it is too easy to get case-sensitive errors (Mypage.htm VS. MyPage.htm). Also lowercase names are much easier to remember. This also applies to your tags - **start making your tags lowercase now** because it will save you a lot of problems if you go on to learn other web mark-up languages (such as XHTML).

No special characters: HTML uses characters like &, >, #, !, /, etc. as part of it's code and each can signify something in other mark-up languages. Use basic numbers and letters for your filenames.

Less than 32 characters: Again this is an older convention (comes from earlier versions of Apple and Windows operating systems) but to ensure compatibility on all computers it is still a good rule to stick to.

Page 20 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Self Check - 1

Directions: Answer these questions:

1. The first step in building a simple website is to clarify _____ and _____.
2. To building a simple website you need to understand _____, _____ and _____.
3. Write a brief description of what a markup language is.
4. Write at least 3 Markup Languages for the Web
5. XHTML is a combination of _____ and _____.
6. Factors that affect your choice of mark-up language may be:
7. Give a brief explanation of each of the following words:
 - a) Hypertext
 - b) Mark-up
 - c) Language
8. What function do "tags" perform and how are they written? Give one example of a tag and what it does.
9. You have been asked to create a website for a new Italian restaurant chain. When you meet the marketing manager she tells you:

"We don't want any pizzas flying across the screen, or anything like that! We just want a simple website that says 'quality'. We want a photo, addresses, phone numbers and a menu that changes once a week. That's it."

Which of the following mark-up languages would you choose to build the site in and why?

 - A. HTML
 - B. SGML
 - C. XHTML
 - D. DHTML

Operation Sheet - 1

Operation Title: Creating Website Folder Structure

Purpose: Before starting out on a website project, you should create a folder structure to contain all the files for your website. It is essential that all your web page files, graphics and other files are collected in the same place right from the start of your project.

Required tools and equipment: MS Windows Folder Utility

Procedures:

1. Start your Windows OS and open Documents
2. Create a new 'root' folder to store your pages and give it a name of your website(e.g. my_website), no space is required use underscore instead.
3. Create a separate folder under the 'root' folder name it *images*, where you store all the graphics you use on the website.
4. Create another separate folder under the 'root' folder name it *styles*, where you store all the styles you use in the website.
5. Create another separate folder under the 'root' folder name it *scripts*, where you store all the *scripts* you use in the website.
6. Create additional separate folder under the 'root' folder name it *pages*, where you store all the *pages* you use in the website.
7. Create one additional separate folder under the 'root' folder name it *documents*, where you store all the *documents* you use on the website.

you can create additional folders based on the website requirements, those you created above, specially from 3-5, are the common or basic folders for a websites.

Page 22 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Lap Test - 1

Name: _____ Date: _____

Time Started: _____ Time Finished: _____

Instructions: Given necessary templates, workshop, tools and materials you are required to perform the following tasks.

Pixel Digital Products

Pixel Digital Products is a manufacturer and distributor of digital cameras, equipment and accessories.

The web site will be a place for customers to see Pixels products, look for frequently asked questions and obtain product user manuals. The company would like the first page to display only their logo, but would like to incorporate company information within the site.

The company colors are reflected in their logo:



They have 3 sample products: DC100 (a simple digital camera), DC250 (their next step up in digital cameras) and DC500 (their best digital camera)

The site does not need to incorporate animated design features, but the code used to create the site must be well formed and adhere to the latest standards.

Read the above scenario and perform the following tasks based on the scenario.

Task 1: Decide which markup language you will use to build a website for this client.

Task 2: Create Pixel Digital Products Folder Structure

- On your computer, create the minimum best practice folder structure for the Pixel website. Call your 'root' folder Pixel.

Task 3: Create A Requirements Document

- Complete the following steps to create a Requirements document for Pixel:
 1. Create a new word processing document called Pixel_Website.
 2. Include a Title page with the Company Name and the company logo
 3. Create a Table of Contents.
 4. Include the following headings and content:

Page 23 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

- **Overview** – put in your own words what you understand the website is meant to do, who might use the website and what markup language you have chosen and why
- **Folder structure** – show the folder structure you created in **Task 2**. You can use a screen shot for this.

Task 4: Create the Pixel Digital Products storyboard

- Review the scenario and add your storyboard for the Pixel website to the document created in **Task 3**.

Unit Two: Create Document Structure

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Creating basic elements of document to create the required web page.
- Depicting document structure of markup sections
- Writing simple markup language

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Create and assign the basic elements of the document
- Markup sections of the document to depict the structure
- Write simple markup language

2.1. Introduction to HTML

2.1.1. History of HTML

The "idea" for the world wide web began in 1945 with an article by Vannevar Bush called "How we may think" where he described an imaginary computer-aided hypertext network he called the "Memex". It would take over forty years for the idea to become reality.

HTML (HyperText Mark-up Language) was largely developed by Tim Berners-Lee in his work for CERN (European Organization for Nuclear Research) from around 1989. The first websites were launched in 1991 and the first graphical browser "Mosaic" was launched in 1993.

The huge growth in the web belies the very simple idea behind it. The idea of linking documents together was a radical one that we now take entirely for granted. Other important ideas included the concept that anyone could contribute to the network of documents and that it should be easy and free to use.

2.1.2. HTML Overview

HTML stands for **H**ypertext **M**arkup **L**anguage, and it is the most widely used language to write Web Pages.

- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

In simple terms, a **Web page** (or **HTML document**) is a plain text file that has been encoded using **Hypertext Markup Language (HTML)** so that it appears nicely formatted in a Web browser. Here's what HTML means, word-by-word:

- **Hypertext**: text that you click to jump from document to document. This is a reference to the ability of Web pages to link to one another
- **Markup**: tags that apply layout and formatting conventions to plain text. Literally, the plain text is "marked up" with the tags.
- **Language**: a reference to the fact that HTML is considered a programming language.

Note: HTML is an interpreted programming language. That means the program is distributed in human-readable format to users, and the program in which it is opened takes care of running it. The HTML code for Web pages resides in files. Each time your Web browser opens a Web page, it processes the HTML code within the file.

Page 26 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

2.1.3. HTML Document

Creating an HTML document is very similar to creating any other file. Although there are many web page building programs available, you can create a web page using just a simple text editor such as **Notepad** or **WordPad** (or **SimpleText** or **TextEdit** on a Mac).

You identify your file as being HTML by saving it with a special extension on the end of the file name - **'htm'** or **'html'**. Both of these extensions tell a browser to begin interpreting HTML. For example you could save your file as: **'index.htm'** or **'home.html'**.

Creating a web folder

Before starting out on a HTML project, you should start by creating a "root" folder - this is the folder that will contain all your files for your website. It is essential that all your web files and graphics are collected in the same place right from the start of your project. See the section at the end of this reading, "Keeping your website together" for more information.

Rules of HTML layout

HTML is simple and follows a few standards. HTML tags define all content (text and images). Tags consist of text surrounded by a less-than < and a greater-than > sign. For example, the tag indicates that text should be bold.

Most tags in HTML are also completed with a similar tag with a slash in it to specify an end to the formatting. This is known as *opening* and *closing* a tag.

 is an opening tag

 is a closing tag

For example, to emphasise some text, you could use the following HTML code:

```

this text is not bold <b>this text is bold</b> this text is not bold

```

To be more correct you could use the tag to bold your text. The code is shown below - the page would look exactly the same.

```

this text is not bold <strong>this text is bold</strong> this text is not bold

```

Note: Although HTML tags are not case-sensitive it's a **good idea to get in the habit of using lowercase**. This is because XHTML only recognises lowercase. Also get used to closing tags – even ones that don't need to be closed for HTML such as
. XHTML requires all tags to be closed. XHTML will eventually replace HTML as the major mark-up language on the web. To find out more about getting ready for XHTML go to the W3 Schools website (www.w3schools.com). Look for the "XHTML Tutorial and find the section on "Differences between XHTML and HTML".

2.1.4. Choosing an HTML Version

Different versions of HTML use different tags for some types of content, although they more similar than different overall, especially at the beginner level. Here's a quick comparison of the HTML versions you may encounter:

- **HTML4** A very stable, universally accepted code set, which is also fairly forgiving of small coding errors. Using HTML4 codes is desirable when compatibility with all browsers is important.
- **XHTML** A strict, standards-based implementation of HTML4 created with XML (eXtensible Markup Language). XHTML coding uses the same codes as HTML4, so it is compatible with the same browsers as HTML4. (See the sidebar about XML on the next page for more information.)
- **HTML5** A revised code set that builds upon HTML4 to add new capabilities. HTML5 offers many dramatic improvements in the areas of application handling and multimedia, but a lot of those features are beyond the scope of this book. In terms of basic coding, which is what this book teaches, the biggest difference is that there are new specific codes for different types of content that were previously handled with more general codes. For example, HTML5 has `<audio>` and `<video>` tags for inserting multimedia content, whereas HTML4 inserts all types of multimedia content via a generic `<embed>` tag.

2.1.5. Minimum System Requirements

There are no minimum system requirements for developing HTML; you can do it in any text editing program with any type of computer and any operating system. That's the beauty of HTML! We use Notepad as the text editor, but you can use any Text Editor that you like.

For testing your work, you will need an **HTML5-Compliant Web browser** application. The latest versions of Google Chrome and Firefox (both freely available online) will work fine for this, as will Internet Explorer 9 or higher.

Page 28 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

2.2. Creating Basic Elements of Document

2.2.1. Basic HTML Document

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

```
<!DOCTYPE html>
<html>
  <head>
    <title> Web and Database Level I </title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>Document content goes here.....</p>
  </body>
</html>
```

Let's save it in an HTML file **test.html** using Notepad text editor. Finally open it using a web browser like Internet Explorer or Google Chrome, or Firefox etc. It must show the following output:

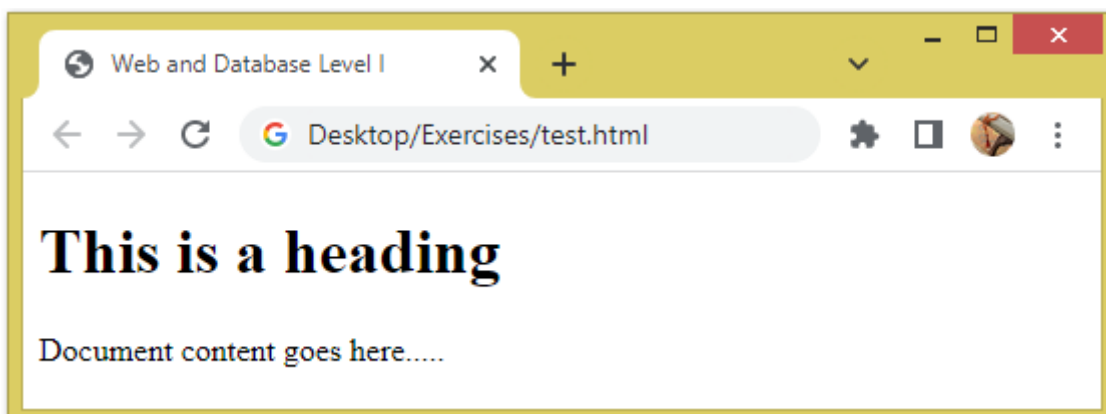


Figure 2-1: Web and Database Level I displayed on the title bar of the browser

1.3.6 HTML Tags

HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

Above example of HTML document uses the following tags:

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

To learn HTML, you will need to study various tags and understand how they behave, while formatting a textual document. Learning HTML is simple as users have to learn the usage of different tags in order to format the text or images to make a beautiful webpage. World Wide Web Consortium (W3C) recommends to use lowercase tags starting from HTML4.

1.3.7 HTML Basic Tags

A. Heading Tags

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

B. Paragraph Tag

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

C. Line Break Tag

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

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

D. Horizontal Lines

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

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

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

E. Preserve Formatting

Sometimes, you want your text to follow the exact format of how it is written in the HTML document. In these cases, you can use the preformatted tag `<pre>`.

Any text between the opening `<pre>` tag and the closing `</pre>` tag will preserve the formatting of the source document.

F. Nonbreaking Spaces

Suppose you want to use the phrase "12 Angry Men." Here, you would not want a browser to split the "12, Angry" and "Men" across two lines:

In cases, where you do not want the client browser to break text, you should use a nonbreaking space entity ` `; instead of a normal space. For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code:

```
<p>An example of this technique appears in the movie "12&nbsp;Angry&nbsp;Men."
</p>
```

G. HTML Comments

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

HTML comments are placed in between `<!-- ... -->` **tags**. So, any content placed within `<!-- ... -->` **tags** will be treated as comment and will be completely ignored by the browser. Single line comments:

```
<!-- Document Header Starts -->
```

HTML supports multi-line comments as well:

```
<!--
This is a multiline comment and it can
span through as many as lines you like.
-->
```

Page 31 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Commenting Script Code

Though you will learn JavaScript with HTML, but here you must make a note that if you are using Java Script or VB Script in your HTML code then it is recommended to put that script code inside proper HTML comments so that old browsers can work properly. Example:

```
<script>
<!--
  document.write("Hello World!")
//-->
</script>
```

Commenting Style Sheets

Though you will learn using style sheets with HTML in a separate tutorial, but here you must make a note that if you are using Cascading Style Sheet (CSS) in your HTML code then it is recommended to put that style sheet code inside proper HTML comments so that old browsers can work properly. Example:

```
<style>
<!--
.example {
  border:1px solid #4a7d49;
}
//-->
</style>
```

1.3.8 HTML Elements

An **HTML element** is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash as shown below with few tags:

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

So here <p>....</p> is an HTML element, <h1>...</h1> is another HTML element. There are some HTML elements which don't need to be closed, such as <img.../>, <hr /> and
 elements. These are known as **void elements (empty elements)**.

HTML documents consists of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.

A. HTML Tag vs. Element

An HTML element is defined by a *starting tag*. If the element contains other content, it ends with a *closing tag*.

For example, `<p>` is starting tag of a paragraph and `</p>` is closing tag of the same paragraph but `<p>This is paragraph</p>` is a paragraph element.

B. Nested HTML Elements

It is very much allowed to keep one HTML element inside another HTML element:

Example

```
<h1>This is <i>italic</i> heading</h1>
```

```
<p>This is <u>underlined</u> paragraph</p>
```

This will display the following result:

This is *italic* heading

This is underlined paragraph

1.3.9 HTML Attributes

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

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

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

Attribute names and *attribute values* are *case-insensitive*. However, the World Wide Web Consortium (W3C) *recommends lowercase attributes/attribute values* in their HTML4 recommendation.

A. Core Attributes

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

- Id
- Title
- Class
- Style

Page 33 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

a) The Id Attribute

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

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

b) The title Attribute

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

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

```
<h3 title="Hello HTML!">Titled Heading Tag Example</h3>
```

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

c) The class Attribute

The class attribute is used to associate an element with a style sheet, and specifies the class of element. You will learn more about the use of the class attribute when you will learn Cascading Style Sheet (CSS). So for now you can avoid it. The value of the attribute may also be a space-separated list of class names. For example:

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

d) The style Attribute

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

```
<p style="font-family:arial; color:#FF0000;">Some text...</p>
```

B. Internationalization Attributes

There are three internationalization attributes, which are available for most (although not all) XHTML elements.

- dir
- lang
- xml:lang

a) The *dir* Attribute

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

Value	Meaning
ltr	Left to right (the default value)
rtl	Right to left (for languages such as Hebrew or Arabic that are read right to left)

`<html dir="rtl">`

When *dir* attribute is used within the `<html>` tag, it determines how text will be presented within the entire document. When used within another tag, it controls the text's direction for just the content of that tag.

b) The *lang* Attribute

The *lang* attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML. This attribute has been replaced by the **xml:lang** attribute in new XHTML documents.

The values of the *lang* attribute are ISO-639 standard two-character language codes. Check [HTML Language Codes: ISO 639](#) for a complete list of language codes.

`<html lang="en">`

c) The *xml:lang* Attribute

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

C. Generic Attributes

Table 2.1: Attributes that are readily usable with many of the HTML tags.

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

2.3. Depicting Document Structure of Markup Sections

2.3.1. HTML Document Structure

All HTML documents share identical underlying structure, a kind of backbone onto which you build your unique page content. Most HTML tags come in pairs which define the content within them. HTML refers to these as *container tags*. An HTML document's basic structure is really just a series of large containers, inside of which you define the *two main sections* of your page: the *document head* and the *document body*.

A typical HTML document will have the following structure:

```

Document declaration tag
<html>
  <head>
    Document header related tags
  </head>
  <body>
    Document body related tags
  </body>
</html>

```

We will study all the header and body tags in subsequent sections, but for now let's see what is document declaration tag.

The <!DOCTYPE> Declaration

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

```
<!DOCTYPE html>
```

There are many other declaration types which can be used in HTML document depending on what version of HTML is being used.

2.3.2. HTML Header Section

This section will give a little more detail about header part which is represented by HTML <head> tag. The <head> tag is a container of various important tags like

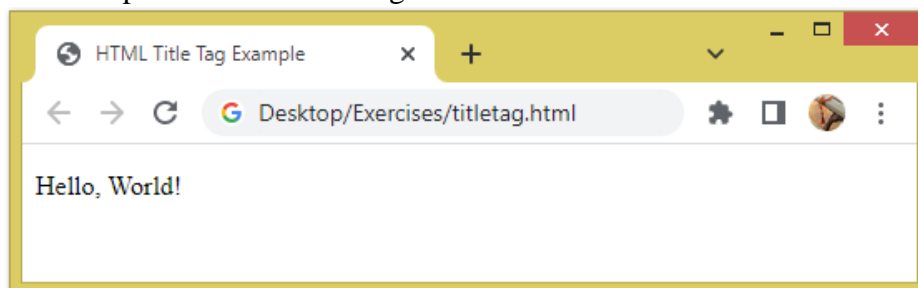
- A. <title>,
- B. <base>,
- C. <link>,
- D. <style>,
- E. <script>,
- F. <meta>, and
- G. <noscript> tags.

A. The HTML <title> Tag

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

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Title Tag Example</title>
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```

This will produce the following result:



B. The HTML <base> Tag

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

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

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

This will produce the following result:

<http://www.tutorialspoint.com/images/logo.png> and
<http://www.tutorialspoint.com/home/html/index.htm>

Page 37 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

C. The HTML <link> Tag

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

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

D. The HTML <style> Tag

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

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

E. The HTML <script> Tag

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

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

F. HTML <meta> Tag

A *document's head section* often contains descriptive information about the document, referred to as *metadata*. Using the *<meta> tag* and *its various attributes*, you can define such document properties as *the author, the expiration date, document keywords, and descriptions*. When search engines that support metadata read your document, they can use this information to index it in order to return your page when someone does a search on subjects matching the keywords you have defined.

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

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

Adding Meta Tags to Your Documents

You can add metadata to your web pages by placing *<meta>* tags inside the header of the document which is represented by *<head>* and *</head>* tags.

Table 2.2: A meta tag can have following attributes in addition to core attributes:

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

Because the *<meta> tag* is an empty tag, To make the *<meta>* tag both XHTML-compliant and still recognizable to browsers that don't yet support XHTML, insert a space and forward slash at the end of the tag.

Caution: If you repeat yourself by using the same or similar keywords, for example “stamp, stamps, stamp collecting,” some search engines may view this as a spamming tactic and rank your page low, or not at all.

Tips

- The object is not to supply every conceivable keyword you can think of but to tailor your keywords to the specific information contained in the document. Keywords can be single words as well as two- or three-word phrases.
- Work your keywords into your document titles and body text. The first word in your document title should be referenced early in your list of keywords, too, so you probably shouldn't start page titles with words like “The.” Any keyword that appears in the text of your document shouldn't be repeated more than seven times in that page.

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

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Meta Tag Example</title>
    <meta name="keywords" content="HTML, Meta Tags, Metadata" />
    <meta name="description" content="Learning about Meta Tags." />
    <meta name="revised" content="Tutorialspoint, 3/7/2014" />
    <meta http-equiv="refresh" content="30" />
    <meta http-equiv="refresh" content="30; url=http://www.tutorialspoint.com" />
    <meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15 23:59:59 GMT;" />
    <meta ame="author" content="Mahnaz Mohtashim" />
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="Content-Type" content="text/html; charset=Big5" />
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```

Specifying Keywords

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

```
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
```

Document Description

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

```
<meta name="description" content="Learning about Meta Tags." />
```

Document Revision Date

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

```
<meta name="revised" content="Tutorialspoint, 3/7/2014" />
```


Document Refreshing

A <meta> tag can be used to specify a duration after which your web page will keep refreshing automatically.

```
<meta http-equiv="refresh" content="30" />
```

Page Redirection

You can use <meta> tag to redirect your page to any other webpage. You can also specify a duration if you want to redirect the page after a certain number of seconds.

```
<meta http-equiv="refresh" content="30; url=http://www.tutorialspoint.com" />
```

Setting Cookies

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

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

```
<meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15  
23:59:59 GMT;" />
```

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

Setting Author Name

You can set an author name in a web page using meta tag. See an example below:

```
<meta ame="author" content="Mahnaz Mohtashim" />
```

Specify Character Set

You can use <meta> tag to specify character set used within the webpage. By default, Web servers and Web browsers use ISO-8859-1 (Latin1) encoding to process Web pages.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

To serve the static page with traditional Chinese characters, the webpage must contain a <meta> tag to set Big5 encoding:

```
<meta http-equiv="Content-Type" content="text/html; charset=Big5" />
```

G. HTML <noscript> Tag

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

JavaScript Example:

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

VBScript Example:

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

Default Scripting Language

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

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

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

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

2.3.3. HTML Body Section Text Formatting

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

A. Bold Text

Anything that appears within ... element, is displayed in bold as shown below:

```
<p>The following word uses a <b>bold</b> typeface.</p>
```

B. Italic Text

Anything that appears within <i>...</i> element is displayed in italicized as shown below:

```
<p>The following word uses a <i>italicized</i> typeface.</p>
```

Page 42 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

C. Underlined Text

Anything that appears within `<u>...</u>` element, is displayed with underline as shown below:

`<p>The following word uses a <u>underlined</u> typeface.</p>`

D. Strike Text

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

`<p>The following word uses a <strike>strikethrough</strike> typeface.</p>`

E. Monospaced Font

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

`<p>The following word uses a <tt>monospaced</tt> typeface.</p>`

F. Superscript Text

The content of a `^{...}` element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a character's height above the other characters.

`<p>The following word uses a ^{superscript} typeface.</p>`

G. Subscript Text

The content of a `_{...}` element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a character's height beneath the other characters.

`<p>The following word uses a _{subscript} typeface.</p>`

H. Inserted Text

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

`<p>I want to drink cola <ins>wine</ins></p>`

I. Deleted Text

Anything that appears within `...` element, is displayed as deleted text.

`<p>I want to drink cola <ins>wine</ins></p>`

J. Larger Text

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

`<p>The following word uses a <big>big</big> typeface.</p>`

Page 43 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

K. Smaller Text

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

`<p>The following word uses a <small>small</small> typeface.</p>`

```

<!DOCTYPE html>
<html>
  <head>
    <title>Bold Text Example</title>
  </head>
  <body>
    <p>The following word uses a <b>bold</b> typeface.</p>
    <p>The following word uses a <i>italicized</i> typeface.</p>
    <p>The following word uses a <u>underlined</u> typeface.</p>
    <p>The following word uses a <strike>strikethrough</strike> typeface.</p>
    <p>The following word uses a <tt>monospaced</tt> typeface.</p>
    <p>The following word uses a <sup>superscript</sup> typeface.</p>
    <p>The following word uses a <sub>subscript</sub> typeface.</p>
    <p>I want to drink <del>wine</del> <ins>Fanta</ins></p>
    <p>I want to drink <del>wine</del> <ins>Fanta</ins></p>
    <p>The following word uses a <big>big</big> typeface.</p>
    <p>The following word uses a <small>small</small> typeface.</p>
  </body>
</html>

```

2.3.4. HTML Body Section Phrase Tags

The phrase tags have been desicolgned for specific purposes, though they are displayed in a similar way as other basic tags like ``, `<i>`, `<pre>`, and `<tt>`, you have seen in previous chapter. This chapter will take you through all the important phrase tags, so let's start seeing them one by one.

A. Emphasized Text

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

`<p>The following word uses a emphasized typeface.</p>`

Page 44 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

B. Marked Text

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

`<p>The following word has been <mark>marked</mark> with yellow</p>`

C. Strong Text

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

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

D. Text Abbreviation

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

`<p>My best friend's name is <abbr title="Abhishek">Abhy</abbr>.</p>`

E. Acronym Element

The `<acronym>` element allows you to indicate that the text between `<acronym>` and `</acronym>` tags is an acronym. At present, the major browsers do not change the appearance of the content of the `<acronym>` element.

`<p>This chapter covers marking up text in <acronym>XHTML</acronym>.</p>`

F. Text Direction

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

`<p><bdo dir="rtl">This text will go right to left.</bdo></p>`

G. Special Terms

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

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

`<p>The following word is a <dfn>special</dfn> term.</p>`

H. Quoting Text

When you want to quote a passage from another source, you should put it in between `<blockquote>...</blockquote>` tags. Text inside a `<blockquote>` element is usually indented from the left and right edges of the surrounding text, and sometimes uses an italicized font.

`<blockquote>XHTML 1.0 is the W3C's first Recommendation for XHTML, following on from earlier work on HTML 4.01, HTML 4.0, HTML 3.2 and HTML 2.0.</blockquote>`

Page 45 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

I. Short Quotations

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

`<p>Amit is in Spain, <q>I think I am wrong</q>.</p>`

J. Text Citations

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

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

`<p>This HTML tutorial is derived from <cite>W3 Standard for HTML</cite>.</p>`

K. Computer Code

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

`<p>Regular text. <code>This is code.</code> Regular text.</p>`

L. Keyboard Text

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

`<p>Regular text. <code>This is inside kbd element</code> Regular text.</p>`

M. Programming Variables

This element is usually used in conjunction with the `<pre>` and `<code>` elements to indicate that the content of that element is a variable.

`<p><code>document.write("<var>user-name</var>")</code></p>`

N. Program Output

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

`<p>Result produced by the program is <code>Hello World!</code></p>`

O. Address Text

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

`<address>388A, Road No 22, Jubilee Hills - Hyderabad</address>`

2.3.5. HTML Body Section Fonts

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

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

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

A. Set Font Size

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

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

B. Relative Font Size

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

```
<font size="-1">Font size="-1"</font><br />
<font size="+1">Font size="+1"</font><br />
<font size="+2">Font size="+2"</font><br />
<font size="+3">Font size="+3"</font><br />
<font size="+4">Font size="+4"</font>
```

C. Setting Font Face

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

```
<font face="Times New Roman" size="5">Times New Roman</font><br />
<font face="Verdana" size="5">Verdana</font><br />
<font face="Comic sans MS" size="5">Comic Sans MS</font><br />
<font face="Bedrock" size="5">Bedrock</font><br />
```


Specify alternate font faces

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

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

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

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

D. Setting Font Color

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

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

```
<font color="red">This text is red</font>
```

E. The <basefont> Element:

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

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

```
<basefont face="arial, verdana, sans-serif" size="2" color="#ff0000">
```

```
<p>This is the page's default font.</p>
```

```
<h2>Example of the &lt;basefont&gt; Element</h2>
```

```
<p><font size="+2" color="darkgray">
```

```
This is darkgray text with two sizes larger
```

```
</font></p>
```

```
<p><font face="courier" size="-1" color="#000000">
```

```
It is a courier font, a size smaller and black in color.
```

```
</font></p>
```


2.3.6. HTML Body Section Backgrounds

By default, your webpage background is white in color. You may not like it, but no worries. HTML provides you following two good ways to decorate your webpage background.

- HTML Background with Colors
- HTML Background with Images

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

A. HTML Background with Colors

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

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

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

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

```
< body bgcolor="lime" >
```

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

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

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

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

B. HTML Background with Images

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

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

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

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

```
<body background="/images/html.gif">
```

Patterned & Transparent Backgrounds

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

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

Page 49 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

2.3.7. HTML Body Section Colors

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

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

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

HTML Color Coding Methods

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

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














Now we will see these coloring schemes one by one.

A. HTML Colors - Color Names

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

W3C Standard 16 Colors

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

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










`<body text="blue" bgcolor="green">`

B. HTML Colors - Hex Codes

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

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

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


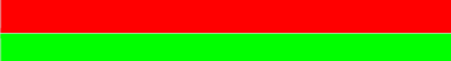




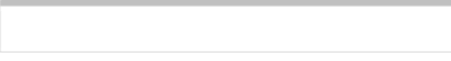
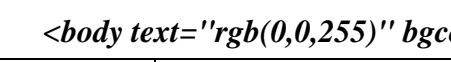
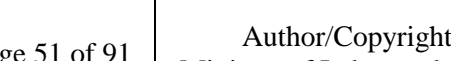
Color	Color HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

```
<body text="#0000FF" bgcolor="#00FF00">
```

C. HTML Colors - RGB Values

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

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

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

```
<body text="rgb(0,0,255)" bgcolor="rgb(0,255,0)">
```

D. Browser Safe Colors

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

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

99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF

CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF
FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

2.4. Writing Simple Markup Language

2.4.1. Structuring an HTML Document

The simple document template that you are about to build can be used again and again as the starting point for every page you create. All HTML documents share this identical underlying structure, a kind of backbone onto which you build your unique page content. As you learned in the previous task, most HTML tags come in pairs which define the content within them. HTML refers to these as container tags. An HTML document's basic structure is really just a series of large containers, inside of which you define the two main sections of your page: the document head and the document body.

```
<!DOCTYPE html>
<html>
<head>
  <title> Web and Database Level I </title>
</head>
<body>

</body>
</html>
```

Save your document. You can give it a name like *blank.html* and then use it each time you want to start a new document by opening it, making changes, and resaving the file with a different name.

Indenting the tags for the document title has no impact on the way the code is rendered by a browser. However, it greatly improves the readability of your code by others, including yourself. The head section defines information about the document that doesn't get displayed in the browser window.

2.4.2. Controlling the Document Background

You can specify a document's *background color* or *background image* using two different attributes of the *<body> tag*; *b bgcolor* and *background*. *Background colors* simply fill the *entire document*. *Background images* are *tiled* by the *browser*, meaning they are repeated left to right, top to bottom, filling up the visible space of the browser window. The default body *text color* is controlled with the *text attribute* of the *<body> tag*.

- To define a background color for a document, set the *bgcolor attribute* equal to a hexadecimal color value or predefined color name

```
<body bgcolor="#0033FF" >
```

- To specify a background image, set the *background attribute* equal to the pathname of the image file on your web server., as shown here:

```
<body background="images/bgstone.jpg">
```

Page 54 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

2.4.3. Setting Document Margins

You can control the document margin with four nonstandard attributes of the <body> tag. Two of the attributes were introduced by Microsoft Internet Explorer: *leftmargin* and *topmargin*; the other two by Netscape Navigator: *marginwidth* and *marginheight*. When defined together, you're guaranteed margin control, not only in these two major browsers but also in their competitors.

- To define the margins of your document, set each attribute equal to a numeric value (representing pixels). The value you specify for *leftmargin* and *marginwidth* set the width of both your left and right margins. Your *topmargin* and *marginheight* values set the width of both the top and bottom margins.

```
<body leftmargin="100" topmargin="50" marginwidth="100"
marginheight="50">
```

Setting your margins to zero allows your design to run to the edges of the browser window.

If being printer-friendly is an issue for your document, understand that the reason some Web pages don't print nicely is because there's content running out to the edges of the screen, which corresponds to where the printer rollers grab the paper. If you define sufficiently wide margins, there will be plenty of room for the rollers to grab without interfering with your page content.

2.4.4. Working with Source Code in the Browser

All major Web browsers allow you to view the source code of documents you view — an extremely useful feature. For example, imagine you're surfing the Internet and you come across a page you're really impressed with. To see how it was built, just view the source HTML code. Each browser has slightly different commands and it supplies slightly different options. Here's how you can view source code using Google Chrome and Microsoft Internet Explorer.

- In Google Chrome:
Right-Click on the Page ⇨ Select View Page Source
- In Internet Explorer:
Right-Click on the Page ⇨ Select View Source

While it's fine to copy source code to examine and learn from, do not plagiarize another developer's HTML. You could run into potential legal issues with the site's owner.

Because Notepad is a text editor, you can make changes to the code right there. Of course, if you're viewing a Web page on the Internet, changes you make to the code won't have an effect on the actual Web site.

2.4.5. Inserting Character Entities

There are about 100 keys on your keyboard, but with all those choices, how do you type something obscure like the copyright symbol (©)? In a word processor, you insert a symbol from some menu or dialog box. In HTML, these symbols are referred to as character entities or special characters. Instead of tags, character entities are rendered numerically, beginning

Page 55 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

with an ampersand (&) and pound sign (#) and ending with a semicolon. This task shows you how to render a number of the more common character entities.

- Type `©` to display the copyright symbol:
`<p> Copyright © 2003 </p>`
- Type `®` to produce the registered symbol:
`<p>W3C ®</p>`
- Type `™` to produce the trademark symbol:
`<p>Alpha-Gizmo™</p>`
- Enter `¼` to produce the fraction one-quarter:
`<p>¼ teaspoon salt</p>`
- Enter `½` to produce the fraction one-half:
`<p>½ teaspoon sugar</p>`
- Enter `¾` to produce the fraction three-quarters:
`<p>¾ cup of honey</p>`
- Enter `¢` to produce the cent symbol:
`<p>10¢</p>`
- Enter `£` to produce the British Pound symbol:
`<p>£125,000</p>`
- Enter `¥` to produce the Japanese Yen symbol:
`<p>¥500,000</p>`
- Enter `€` to produce the European Union's Euro symbol:
`<p>€700</p>`

Character entities are also referred to as *special characters*.

Most character entities have an English-language equivalent. For example, the copyright symbol can also be written as `©` and the registered sign as `®`.

2.4.6. Creating HTML Lists

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

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

Page 56 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

A. Creating an Ordered List

If you use a word processor to make a numbered list of items, all you have to do is click a button and start typing. Creating them in HTML is almost as easy: Use the `` (ordered list) and `` (list item) tags. Both are container tags with opening and closing forms. As the name implies, you use ordered lists to render information of a procedural nature — for example, the items in this task. Example

```
<h1>Honest Shampoo Instructions</h1>
<font face="Verdana, Arial, Helvetica, sans-serif" size="2">
  <ol>
    <li>Apply to wet hair</li>
    <li>Massage gently into hair and scalp</li>
    <li>Rinse thoroughly</li>
    <li>Repeat steps 1 - 3 to ensure you turn through a bottle of this stuff a week.</li>
  </ol>
</font>
```

It must show the following output:

Honest Shampoo Instructions

1. Apply to wet hair
2. Massage gently into hair and scalp
3. Rinse thoroughly
4. Repeat steps 1 - 3 to ensure you burn through a bottle of this stuff a week.

Browsers indent list items by default, not because we've done so in the code. Indenting your code just makes it easier to read.

Even though browsers allow you to omit them, always use closing `` tags. They produce cleaner, more readable code. And, of course, XHTML requires them. Bulleted lists (also called unordered lists) aren't radically different from ordered lists in their construction.

Modifying Ordered List Styles

By default, an ordered list renders items with Arabic numerals: 1, 2, 3, and so on. You can modify the style of list items by defining the type attribute of the `` tag. The type attribute for the `` tag accepts five possible values.

- To create an uppercase alphabetical list, set the type attribute equal to "A":
`<ol type="A">`
- To create a lowercase alphabetical list, set the type attribute equal to "a":
`<ol type="a">`
- To create an uppercase Roman numeral list, set the type attribute equal to "I":
`<ol type="I">`
- To create a lowercase Roman numeral list, set the type attribute equal to "i":
`<ol type="i">`
- To create an Arabic numeral list, set the type attribute equal to "1":
`<ol type="1">`

Page 57 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Modifying an Ordered List's Starting Character

The attribute you define to specify the starting number or character in an ordered list is named *start*. This attribute allows you to maintain an unbroken ordering sequence even if you have to separate lists with paragraph text. You specify the value of an individual list item using the value attribute.

```
<p>To shampoo, follow these steps:</p>
<ol>
  <li>Apply to wet hair</li>
  <li>Massage gently into hair and scalp</li>
  <li>Rinse thoroughly</li>
</ol>

<p>Then, to condition proceed by:</p>
<ol start="4">
  <li>Wringing excess water from hair</li>
  <li>Apply conditioner and massage gently into hair from roots to ends</li>
  <li>Rinse thoroughly with warm water</li>
</ol>
```

While the values of the start and value attributes are always defined with an integer, the corresponding list item character may not be numerical. For example, if the ordered list's type attribute equals A (creating an uppercase A, B, C list), setting the start or value attribute equal to 3 begins the list with "C." In an uppercase Roman numeral list (type="I"), the list would begin with III, etc.

The above example must show the following output:

Honest Shampoo Instructions

To shampoo, follow these steps:

1. Apply to wet hair
2. Massage gently into hair and scalp
3. Rinse thoroughly

Then, to condition proceed by:

4. Wringing excess water from hair
5. Apply conditioner and massage gently into hair from roots to ends
6. Rinse thoroughly with warm water

B. Creating an Unordered List

What word processing calls a bulleted list, HTML refers to as an unordered list. You create these using the `` tag and the same `` tag that ordered lists use.

```
<h1>Points to Remember</h1>
<ul>
  <li>Don't run with scissors</li>
  <li>Don't play with your food</li>
  <li>Don't forget to wash your hands</li>
</ul>
```

It must show the following output:

Points to Remember

- Don't run with scissors
- Don't play with your food
- Don't forget to wash your hands

Just like ordered lists, there's no limit to the number of list items you can have.

As with the ordered lists, browsers indent list items by default, not because we've done so in the code. Indenting your code makes it easier to read.

The HTML standard contains a directory and menu list, created with `<dir>` and `<menu>` tags. Like ordered and unordered lists, they used `` tags for their list items. The directory list was supposed to create multicolumn lists, and the menu list was meant for single columns. Unfortunately, no browser ever attempted to render these lists. Using these tags simply creates an unordered list.

Always use closing `` tags. They make for cleaner, more readable code, and XHTML requires them.

The default bullet style for unordered lists is typically a small filled-in disc. To see how to modify bullet styles, see next Part.

Modifying Bullet Styles

Just like the `` tag, the `` tag accepts the `type` attribute. In this case, the `type` attribute governs the style of the bullet that precedes each list item. The possible values for the `type` attribute are `disc`, `square`, and `circle`.

- To create square bullets, set the `type` attribute equal to `square`:
`<ul type="square">`
- To create circular bullets, set the `type` attribute equal to `circle`:
`<ul type="circle">`
- To create disc bullets, set the `type` attribute equal to `disc`:
`<ul type="disc">`

Page 59 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

C. Nesting Lists

Nesting simply means to place elements inside other elements. When you nest lists, you insert a new ordered or unordered list between list items in an existing list. The existing list is called the parent list and the second, nested list is called the child list. You can, in turn, nest a third list within the second, a fourth within the third, and so on. By nesting lists in this fashion, each list becomes a sub list of the parent list item above it. This technique is ideal for creating formal outlines.

```

<ol type="I">
  <li>Main Idea</li>
  <ol type="A">
    <li>Subordinate Idea</li>
    <ol type="1">
      <li>Supporting Detail</li>
      <ol type="a">
        <li>Example</li>
        <li>Example</li>
      </ol>
      <li>Supporting Detail</li>
    </ol>
  </ol>
</ol>

<ul>
  <li>Type defaults to disc</li>
  <ul>
    <li>First nest defaults to circle</li>
    <ul>
      <li>Third nest defaults to square...</li>
      <ul>
        <li>and continues as square until you set a <tt>type</tt>
          attribute</li>
      </ul>
    </ul>
  </ul>
</ul>

```

It must show the following output:

-
- I. Main Idea
 - A. Subordinate Idea
 - 1. Supporting Detail
 - a. Example
 - b. Example
 - 2. Supporting Detail
 - Type defaults to disc
 - First nest defaults to circle
 - Third nest defaults to square...
 - and continues as square until you set a type attribute

D. Creating Definition Lists

Definition lists are slightly different than the previous list types you've encountered. The list items in a definition list consist of two parts: a term and a description. Browsers render definition lists by placing the term on one line and indenting the definition underneath it, creating what's called a hanging indent. You use three pairs of tags to create a definition list: `<dl>` and `</dl>` tags define where the list begins and ends, `<dt>` and `</dt>` tags define the term, and `<dd>` and `</dd>` tags define the term's definition.

```
<h1>Lists in HTML</h1>
<dl>
  <dt>The Ordered List</dt>
  <dd>Created using the OL element. This list should contain information
  where order should be emphasized.</dd>
  <dt>The Unordered List</dt>
  <dd>Created using the UL element. This list should be used to express a
  series of significant points</dd>
  <dt>The Definition List</dt>
  <dd>Create using the DL element. This list should be used to define a list
  of terms.</dd>
</dl>
```

It must show the following output:

Lists in HTML

The Ordered List

Created using the OL element. This list should contain information where order should be emphasized.

The Unordered List

Created using the UL element. This list should be used to express a series of significant points

The Definition List

Create using the DL element. This list should be used to define a list of terms.

2.4.7. HTML Images

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

A. Insert Image

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

``

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

Page 61 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

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

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

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

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

B. Set Image Location

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

```
<body>
  <p>Simple Image Insert</p>
  
</body>
```

C. Set Image Width/Height

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

```
<body>
  <p>Setting image width and height</p>
  
</body>
```

D. Set Image Border

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

```
<body>
  <p>Setting image Border</p>
  
</body>
```

Page 62 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

E. Set Image Alignment

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

```
<body>
  <p>Setting image Alignment</p>
  
</body>
```

2.4.8. HTML Embed Multimedia

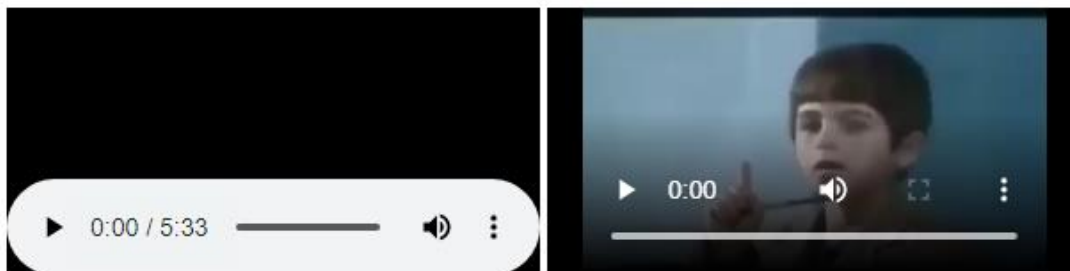
Sometimes you need to add music or video into your web page. The easiest way to add video or sound to your web site is to include the special HTML tag called **<embed>**. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports **<embed>** tag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

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

Here is a simple example to play an embedded midi file:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML embed Tag</title>
  </head>
  <body>
    <embed src="DikiDike.mp3" >
      <noembed></noembed>
    </embed>
    <embed src="tokofitoko.mp4">
      <noembed></noembed>
    </embed>
  </body>
</html>
```

This will produce the following result:



You can put any media file in **src attribute**. You can try it yourself by giving various types of files.

Page 63 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

A. The <embed> Tag Attributes

Following is the list of important attributes which can be used with <embed> tag.

Attribute	Description
align	Determines how to align the object. It can be set to either center, left or right.
autostart	This boolean attribute indicates if the media should start automatically. You can set it either true or false.
loop	Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false)
playcount	Specifies the number of times to play the sound. This is alternate option for loop if you are using IE.
hidden	Specifies if the multimedia object should be shown on the page. A false value means no and true values means yes.
width	Width of the object in pixels
height	Height of the object in pixels
name	A name used to reference the object.
src	URL of the object to be embedded.
volume	Controls volume of the sound. Can be from 0 (off) to 100 (full volume).

B. Supported Video Types

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

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

C. Background Audio

You can use HTML <bgsound> tag to play a soundtrack in the background of your webpage. This tag is supported by Internet Explorer only and most of the other browsers ignore this tag. It downloads and plays an audio file when the host document is first downloaded by the user and displayed. The background sound file also will replay whenever the user refreshes the browser.

This tag is having only two attributes *loop* and *src*. Both these attributes have same meaning as explained above.

```
<bgsound src="/html/yourfile.mid">
  <noembed></noembed>
</bgsound>
```

This will produce the blank screen. This tag does not display any component and remains hidden.

Internet Explorer can also handle only three different sound format files: wav, the native format for PCs; au, the native format for most Unix workstations; and MIDI, a universal music-encoding scheme.

Page 64 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

D. HTML Object Tag

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

Here are a few examples:

Example - 1

You can embed an HTML document in an HTML document itself as follows:

```
<object data="data/test.htm" type="text/html" width="300" height="200">
  alt : <a href="data/test.htm">test.htm</a>
</object>
```

Here **alt attribute** will come into picture if browser does not support *object* tag.

Example - 2

You can embed a PDF document in an HTML document as follows:

```
<object data="data/test.pdf" type="application/pdf" width="300" height="200">
  alt : <a href="data/test.pdf">test.htm</a>
</object>
```

Example - 3

You can specify some parameters related to the document with the **<param>** tag. Here is an example to embed a wav file:

```
<object data="data/test.wav" type="audio/x-wav" width="200" height="20">
  <param name="src" value="data/test.wav">
  <param name="autoplay" value="false">
  <param name="autoStart" value="0">
  alt : <a href="data/test.wav">test.wav</a>
</object>
```

Example - 4

You can add a java applet into HTML document as follows:

```
<object classid="clsid:8ad9c840-044e-11d1-b3e9-00805f499d93" width="200"
height="200">
  <param name="code" value="applet.class">
</object>
```

The **classid** attribute identifies which version of Java Plug-in to use. You can use the optional *codebase* attribute to specify if and how to download the JRE.

Page 65 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

2.4.9. HTML Tables

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

The HTML tables are created using the `<table>` tag in which the `<tr>` tag is used to create table rows and `<td>` tag is used to create data cells. *Example*

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML Tables</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td>Row 1, Column 1</td>
        <td>Row 1, Column 2</td>
      </tr>
      <tr>
        <td>Row 2, Column 1</td>
        <td>Row 2, Column 2</td>
      </tr>
    </table>
  </body>
</html>

```

This will produce the following result:

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

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

A. Table Heading

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

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Salary</th>
  </tr>
  <tr>
    <td>Ramesh Raman</td>
    <td>5000</td>
  </tr>
  <tr>
    <td>Shabbir Hussein</td>
    <td>7000</td>
  </tr>
</table>
```

This will produce the following result:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

B. Cellpadding and Cellspacing Attributes

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

```
<table border="1" cellpadding="5" cellspacing="5">
  <tr>
    <th>Name</th>
    <th>Salary</th>
  </tr>
  <tr>
    <td>Ramesh Raman</td>
    <td>5000</td>
  </tr>
  <tr>
    <td>Shabbir Hussein</td>
    <td>7000</td>
  </tr>
</table>
```

This will produce the following result:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

C. Colspan and Rowspan Attributes

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

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

This will produce the following result:

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

D. Tables Backgrounds

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

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

```
<table border="1" bordercolor="red" bgcolor="yellow">
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr> <td rowspan="2">Row 1 Cell 1</td> <td>Row 1 Cell 2</td>
    <td>Row 1 Cell 3</td> </tr>
  <tr> <td>Row 2 Cell 2</td> <td>Row 2 Cell 3</td> </tr>
  <tr>
    <td colspan="3">Row 3 Cell 1</td>
  </tr>
</table>
```

This will produce the following result:

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

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

```
<table border="1" bordercolor="red" background="/images/pattern.png">
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
  <tr> <td rowspan="2">Row 1 Cell 1</td> <td>Row 1 Cell 2</td>
    <td>Row 1 Cell 3</td> </tr>
  <tr> <td>Row 2 Cell 2</td> <td>Row 2 Cell 3</td> </tr>
  <tr>
    <td colspan="3">Row 3 Cell 1</td>
  </tr>
</table>
```

This will produce the following result

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

E. Table Height and Width

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

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

This will produce the following result:

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

F. Table Caption

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

```
<table border="1" width="100%">
  <caption>This is the caption</caption>
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
```

This will produce the following result:

This is the caption	
Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

G. Table Header, Body, and Footer

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

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

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

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

```
<table border="1" width="100%">
  <thead>
    <tr>
      <td colspan="4">This is the head of the table</td>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="4">This is the foot of the table</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>Cell 1</td>
      <td>Cell 2</td>
      <td>Cell 3</td>
      <td>Cell 4</td>
    </tr>
    <tr>
      <td>Cell 11</td>
      <td>Cell 12</td>
      <td>Cell 13</td>
      <td>Cell 14</td>
    </tr>
  </tbody>
</table>
```

This will produce the following result:

This is the head of the table			
Cell 1	Cell 2	Cell 3	Cell 4
Cell 11	Cell 12	Cell 13	Cell 14
This is the foot of the table			

Self Check - 2

Directions: Answer these questions:

1. What is HTML?
2. What are HTML tags?
3. What tags describe an entire web page?
4. What tags enclose the visible content of a web page?
5. What tag is used for images?
6. Give an example of an HTML Element
7. What is an Attribute?
8. Between which tags does the main content of a web page go in an HTML document:
 - A. <html> </html>
 - B. <body> </body>
 - C. <title> </title>
 - D. <head> </head>
9. Which would be the best file name for the homepage of a website?
 - A. homepage.html
 - B. home.txt
 - C. index.html
 - D. welcome.html
10. Which is the correct HTML to attach an image?
 - A.
 - B.
 - C. <image src="images/logo.gif">
 - D.
 - E.
11. When writing HTML tags, how do you spell the words "colour", "centre" and "align"?
12. Describe the difference between the following tags: and . What do they mean and what do they do?
13. What are the HTML tags that describe the following?
 - Table
 - Table Row
 - Table Cells
14. You want to include the following kinds of images on your website:
 - a photograph
 - a black and white cartoon line drawing
 - a simple pie chart
 What file formats would you use for each?

Operation Sheet - 2.1

Operation Title: Structuring an HTML Document

Purpose: The simple document template that you are about to build can be used again and again as the starting point for every page you create.

Required tools and equipment: Text Editor and Browser

Procedures:

8. Open your text editor and begin a new blank document.
9. Type `<!DOCTYPE html>`, this tag defines the document type and HTML version.
10. Hit Enter and type the tag `<html>`, this tag begins the document's primary container.
11. This opening `<html>` tag requires a closing tag, so hit Enter twice to move down a few lines and then enter the closing tag `</html>`. Your document appear like this:


```
<!DOCTYPE html>
<html>

</html>
```
12. Place your cursor on the line between the opening and closing tags. Type the tag `<head>`, which defines the head section of the document.
13. Hit Enter twice and then type `</head>`. Your document should now resemble below.


```
<head>

</head>
```
14. To create the document title, which appears in the title bar of the browser window, enter `<title>` and `</title>` between the head tags of your document, as shown below.


```
<head>
  <title>Web and Database Level I</title>
</head>
```
15. The last element to add to your document template is the body section. Between the closing `</head>` and the closing `</html>` tags, enter opening and closing body tags.


```
<!DOCTYPE html>
<html>
<head>
  <title> Web and Database Level I </title>
</head>
<body>

</body>
</html>
```
16. Save your document. You can give it a name like `blank.html` and then use it each time you want to start a new document by opening it, making changes, and resaving the file with a different name.

Page 73 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Operation Sheet - 3.2

Operation Title: Text Formatting

Purpose: learn the usage of different tags in order to format the text to make a beautiful webpage.

Required tools and equipment: Text Editor and Browser

Working with Headings

The following series of tags create document headings akin to those in newspapers and magazines. There are six levels of headings, ranging from a heading 1 (the largest) to a heading 6 (the smallest).

1. To format a word or phrase as a heading, place an opening heading tag in front of it, as shown:

`<h1>`This is a Heading 1

`<h2>`This is a Heading 2

`<h3>`This is a Heading 3

`<h4>`This is a Heading 4

`<h5>`This is a Heading 5

`<h6>`This is a Heading 6

2. Place a corresponding closing heading tag after the word or phrase, as shown:

`<h1>`This is a Heading 1`</h1>`

`<h2>`This is a Heading 2`</h2>`

`<h3>`This is a Heading 3`</h3>`

`<h4>`This is a Heading 4`</h4>`

`<h5>`This is a Heading 5`</h5>`

`<h6>`This is a Heading 6`</h6>`

3. The heading tag's only allowable attribute is align. It's possible values are left, right, and center. To align a heading, insert the alignment attribute within the heading tag, as shown here:

`<h1 align>`

4. Set your align attribute equal to the desired alignment. Here we use the center alignment.

`<h1 align="center">`Heading 1 - Centered`</h1>`

Page 74 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Operation Sheet - 4.3

Operation Title: Working with Paragraphs

Purpose: learn how to format basic paragraph text

Required tools and equipment: Text Editor and Browser

1. To indicate the beginning of a paragraph, enter an opening `<p>` tag in the `<body>` section of your code, as shown:

```
<p>
```

2. To mark the end of a paragraph, place the closing `</p>` tag at the end of your paragraph, as shown:

```
<html>
```

```
<head>
```

```
<title>Defining and Aligning Paragraphs</title>
```

```
</head>
```

```
<body>
```

```
<p>HTML only recognizes single spaces between characters. Other than a single tap on the space bar, HTML has little regard for how you type things. What it does have regard for is tags.</p>
```

```
</body>
```

```
</html>
```

3. To align a paragraph, add the `align` attribute to the paragraph tag:

```
<p align>
```

4. Set the `align` attribute equal to `left`, `right`, `center`, or `justified`, as shown:

```
<p align="right">HTML only recognizes single spaces between characters. Other than a single tap on the space bar, HTML has little regard for how you type things. What it does have regard for is tags.</p>
```

Lap Test - 2

Name: _____ Date: _____

Time Started: _____ Time Finished: _____

Instructions: Given necessary templates, workshop, tools and materials you are required to perform the following tasks.

Task 1: Create The Pixel Website Using HTML

Using your Pixel storyboard from **Lap Test - 1**, create the Pixel website using HTML. Incorporate the various elements you practiced in these TTLM. Use an internal style sheet for all colors, fonts etc. As a minimum you should have the following pages:

- A startup page with the company logo only, linking to a Home page
- A Home page that has information about the company
- A Products page (use named hyperlinks to link to each product)
- A Frequently Asked Questions page
- A page where people can download user manuals
- Use a list to provide navigation on each page and an external style sheet for colors, layout and fonts
- Include appropriate accessibility options

Unit Three: Validate Documents

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Validating markup language document
- Validating markup language document in different browsers
- Validating simple markup language document

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Validate markup language document against requirements
- Validate markup language document in different browsers
- Validate simple markup language document

3.1. Validating Markup Language Document

A website must meet the particular requirements of the client. These specifications usually aim to ensure consistency of organisational identity and may outline procedures to be followed in the development process.

Web development also requires a variety of tests to ensure or "validate" that a web site will function as expected in a number of computing environments and that the site is error-free. There are a number of tests that can be performed to validate mark-up documents and web sites:

- Functional test
- Compatibility test
- Performance test
- Regression test

In practice these four stages may occur simultaneously or in separate, systematic steps. This will depend on both the complexity of the product you are testing and the number of team members involved in testing. The bigger the team, the more clearly defined the stages of progress need to be documented.

3.1.1. Functional testing

Functional testing refers to:

- Confirming that the site meets client specifications.
- Testing for errors or mistakes in mark-up and in content

A. Meeting client specifications

A set of specification for your website should be agreed upon between you (or your organization) and the client. Keep these in mind and check that your work complies as you develop your mark up language documents.

Specifications could include:

- The style or formatting of elements such as text
- Style guidelines to fit in with organization 'branding'
- The positions of elements on the page
- The structure of the page
- Web browsers that the web page must work on
- Operating systems the web page must work with
- Size limits for files

Read the project documents carefully in order to identify specifications and communicate with your supervisor or with the client to verify expectations.

Analysing these requirements will assist with setting up your testing stages.

Page 78 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

B. Validation tools

You need to "validate" your mark-up documents (such as CSS, HTML and XHTML) to ensure that they are correct and error-free. The easiest and most thorough way to check that your mark-up code is correct is to use one of the free validation tools available on the internet. These will check your mark-up document and return a list of errors found so that you can correct them.

Some online validation tools are:

- World Wide Web Consortium (W3C) HTML, CSS and XHTML validators: validator.w3.org
- Also see the "Validators" page at W3C: www.w3.org
- HTML validator from the Web Design Group: htmlhelp.org/tools/validator/

You can also download a number of standalone validation tools from the Tucows site: www.tucows.com. Search for the term 'Validator'.

Note: Some applications for creating mark-up (such as Dreamweaver) also have their own built-in mark-up language validators. Beware that even though they may be able to check your site, they may not be 100% standards-compliant.

C. Common validation errors

A few common HTML/XHTML errors that you might encounter are:

- **No DTD:** You should make the first line of your HTML a 'document type definition' (DTD) so that the validator knows what version and type of mark-up language you are using—for example `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`
- **Missing alt tags** - all graphics in your website should include "alt" tags to provide a text alternative to users with vision impairment
- **Improper nesting of elements** - mark-up tags need to be opened and closed in the correct order - i.e. the order they were opened in. For example `<p>` will be followed by `</p>` (not `</p>`)
- **Not closing tags** - missing quote marks, brackets or colons
- **Using spaces in document titles** - use an underscore "_" instead
- **Missing title** - HTML/XHTML documents must have a "title" element in the head of the document
- **Using upper case tags** - XHTML is case sensitive (use `<body>` not `<BODY>`)
- **Missing quotes** - attributes should have quotation marks surrounding them. For example ``

On finding these errors you will need to address them and ideally re-validate your document until you have eliminated all errors.

Page 79 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

D. XML validation

XML stands for eXtensible Markup Language. XML is a mark-up language but is different in structure and intent than HTML.

XML was designed to describe data and to focus on what data is. HTML was designed to display data and to focus on how data looks.

XML documents will need to be validated either against a Document Type Definition (DTD) or against an appropriate XML Schema. For more information about getting started with XML and validating XML documents, take a look at the W3 Schools website: www.w3schools.com. Start with 'Learn XML' under 'Tutorials' specifically take a look at the section in the tutorial called 'XML Validation'. For in-depth technical discussions of XML, visit the World Wide Web Consortium (W3C): www.w3.org. W3C is the international standards body overseeing development of most web mark-up languages.

QA Tracking

All of these testing phases are part of the larger process of Quality Assurance (QA). Ensuring that a website or multimedia product is error-free and works to specification is crucial before the product can be released.

It is essential; that QA processes are well documented. As you can see from the testing phases mentioned above, QA will often mean that the whole product will be tested many times in multiple computing environments. The more people involved in testing and the more complex the site or product, the greater reliance will be placed on QA documentation.

QA documentation

QA documentation allows testers and developers to accurately ensure that all concerns have been addressed and avoids unnecessary and time-consuming double-checking. Also good documentation at all production stages can help ensure bugs and errors are caught early and fixed well before production is almost complete. Documentation can also provide essential evidence in the face of disputes between clients and developers.

Take a look at this sample Quality Assurance form. (This is also available for download from the online "Reading" section of this learning pack). This form is a general QA document that covers three areas:

- Design
- Editorial
- Functional

Note that there may be a separate procedure for each of these three areas of quality assurance.

For example, bug tracking during the development lifecycle of a multimedia product may be handled by a custom-made database. The database should allow progress to be tracked along a timeline - ensuring that all errors are re-checked and fixed before sign-off.

Page 80 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

3.2. Validating Markup Language Document in Different Browsers

3.2.1. Compatibility testing

Compatibility testing looks at a product to ensure that it will perform as expected in a range of computing environments. For web developers this often means testing in a range of:

- A. Browsers
- B. Operating Systems (OS)
- C. Display Resolutions

Preview your web documents in different computing environments (browser, platform, OS etc.) regularly throughout the production process to catch errors early. Before delivering your product to the client you will need to perform comprehensive compatibility testing.

Compatibility testing may show up "Bugs" in the site - unexpected behaviour in certain computing environments. This is not the same as checking for simple errors in code - covered in the 'Functional test' section of this Reading.

A. *Browsers*

Different browsers can interpret mark-up code in different ways. It's important to be confident that your mark-up documents will display correctly in all the browsers specified by the client.

Browser usage

At time of writing the most common browsers in use are:

- Internet Explorer - 87%
- Firefox - 8%
- Netscape - <2%
- Safari (Mac) - <2%
- Opera = <1%

There are a range of other specialised web browsers. For current information on browser usage, go to the Market Share website: marketshare.hitslink.com. You can also read more at W3 Schools: www.w3schools.com.

Page 81 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Appearance across browsers.

Here is a sample of a simple HTML page shown side-by-side in two different browser windows (Firefox Version 1.5 and Internet Explorer Version 6 - both on Windows XP).

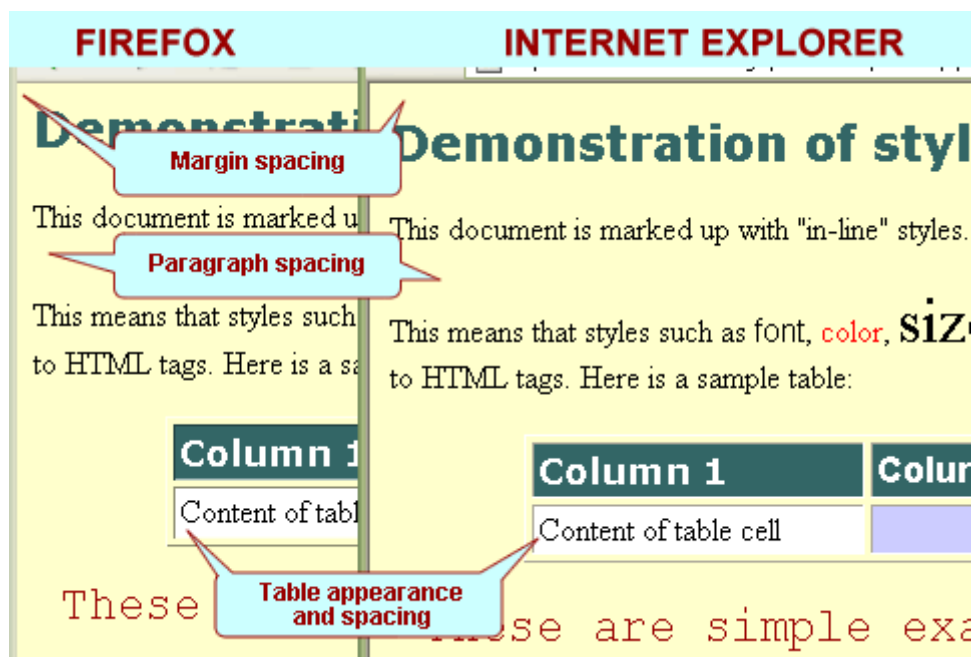


Figure 3.1 Side-by-side comparison of same HTML page in two browsers

Both browsers display the page clearly but there are small differences, even in this very basic mark-up document. The most noticeable differences include:

- Page margins - IE places the H1 text 2 pixels right and 7 pixels down from where Firefox places the same text
- Paragraphs - paragraphs are spaced further apart in IE than Firefox
- Tables- table borders display slightly differently and table cells are more widely spaced in IE.

For a professional web designer, these small differences can make or break a site design. There are many strategies that designers use in their mark-up to ensure consistent appearance across browsers (and platforms) using Cascading Style Sheets (CSS) and occasionally specifying different CSS to be delivered to different browsers.

For in-depth discussions on using CSS, go to the Front Page website: css-discuss.incutio.com. This is a CSS discussion Wiki - maintained and contributed to by a range of web developers.

For specific and technical information on cross-browser bugs and CSS fixes, take a look at the Position Is Everything site: www.positioniseverything.net

While the example shown above is quite simple, the same principles apply to all aspects of creating mark-up documents. You need to check your mark-up across browsers during development

B. Operating Systems

The two main operating systems in use on personal computers are Windows and MacOS (Macintosh Operating System - now Unix-based). Unix and variations of Linux are largely used on servers rather than personal computers. At time of writing some of the most common operating systems are:

- Windows XP - 80%
- Windows 2000 - 9%
- Windows 98 and NT - 4%
- Mac (all versions) - 4%
- Unix/Linux - 0.5%

For current information on operating system usage, go to the Market Share website: marketshare.hitslink.com. You can also read more at W3 Schools: www.w3schools.com.

Once again the way that each of these operating systems interprets your mark-up will vary. You need to test your product in a range of operating systems.

C. Display resolutions

A challenge for web and multimedia developers is to anticipate the screen resolution of their user's computers. Computer resolution is measured in pixels.

At time of writing, the general trends for monitor settings are as follows (width x height in pixels - percentage of all computer users):

- 640 x 480 - 2%
- 800 x 600 - 20%
- 1024 x 768 - 55%
- Higher - 17%
- Unknown - 6%

You should allow for variations in these "screen real estate" settings when designing your multimedia or web-based products. Keep in mind the following points:

- For most non-technical computer users, their monitor resolution will be the setting that came with the computer when they bought it
- Vision-impaired users may adjust their resolution to make everything bigger (e.g. 640 x480)
- Tech-savvy users (and people with good eyesight!) often set their monitors to higher resolutions to fit more stuff on screen.
- Larger LCD monitors also allow more screen space and therefore much higher resolution than older CRT monitors.

Page 83 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

D. Standards

Some clients will specify that their website must comply with web standards. Even if your client doesn't, there are good reasons for writing web documents using standards compliant mark-up code. One reason is that you will save time trying to get your pages to display the same way in different browsers. Various standards-compliant browsers will display your standards compliant pages in mostly the same way.

By complying with web standards your web pages will be more accessible. They will be easier and cheaper to maintain and update and their size will be smaller, reducing download time and required bandwidth. Your web site will reach a larger audience because pages will be compatible with a variety of browsers, platforms and devices. In addition, search engines such as Google (www.google.com.au) will also find and index your site more easily.

E. DTD

Document Type Definition (DTD). This shows which international standard SGML and XML documents should conform to and informs how browsers, search engines etc. should handle the document. All SGML and XML documents (including HTML and XHTML) should include a **Doctype** reference to a DTD. Doctype is short for 'Document Type Declaration' - here is an example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd" />
```

This says that the standard that applies to this document is 'HTML 4.0 Transitional' and then gives the URL where this standard can be found.

F. W3C

The recommendations from the World Wide Web Consortium (W3C): www.w3.org are widely accepted as the international standards for web mark-up languages.

The W3C website contains vast resources for web developers. Much of the discussion of evolving technologies and standards is highly technical. However there are also significant tools for general web development such as validators for a variety of mark-up languages.

3.2.2. Performance testing

Performance testing is required for larger and more complex websites. It is also essential for software and application developers. Performance testing may focus on:

- Server capacity
- Network capacity and connection speed
- Database functions
- Scalability

Performance testing will require a set of standards or benchmarks against which the site or product will be tested.

Page 84 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

3.2.3. Regression testing

Regression testing is the cycle of testing that continues throughout a product's development. Specifically a 'regression bug' is a fault that may be introduced after programming code or mark-up has been altered to fix another problem.

In practice, regression testing means that products are tested repeatedly through different development stages for two reasons:

- to check that identified problems are fixed and remain fixed
- to make sure that 'fixes' does not introduce new errors.

Regression bugs are a common problem in multimedia and web development.

Page 85 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

3.3. Validating Simple Markup Language Document

3.3.1. Accessibility

Complying with web standards and appropriate design. By complying with web standards your site will be more accessible. File sizes on the site will most likely be smaller, that way reducing download time and required bandwidth and the site will be easier and cheaper to maintain and update. By being compatible with a variety of browsers, platforms and devices, the site will also reach a larger audience. It may also be compatible with Internet enabled mobile phones and PDAs. Google and other search engines will find your site more easily, if you have followed web standards.

When web sites are not appropriately designed, people with disabilities including those affecting their vision, motor skills or cognition can encounter problems using them. For example, many visually impaired people use a screen reader to interpret web pages and these often encounter problems if pages are not written using standard HTML.

Simple design elements like always providing a meaningful ‘ALT tag’ for images will make your site more accessible and increase your audience.

Further information

Details of the requirements of the *Commonwealth Disability Discrimination Act 1992*, can be found in the advisory notes for that Act provided by the Australian Human Rights and Equal Opportunities Commission at:
http://www.hreoc.gov.au/disability_rights/standards/www_3/www_3.html

The Act states that all web sites must adhere to web content accessibility guidelines devised by the World Wide Web consortium (W3C), which can be found at:
<http://www.w3.org/TR/WCAG20/>

3.3.2. Using a quality assurance checklist

The quality assurance process demands meticulous planning and execution. Every single detail on each web page is checked for errors and consistency. The best way to ensure you don’t miss anything is to prepare a checklist.

Project documents can help identify the assessment criteria for your list. Useful criteria are set out quite explicitly in the style guide and technical specifications described above. The project brief can also reveal useful information. For example, the corporate image the organisation wants to project can help you judge the appropriateness of the graphics.

Criteria for a checklist should include standards pertaining to HTML, XHTML and cascading style sheets (CSS), and principles of accessibility. Recommendations from bodies such as the World Wide Web Consortium (W3C), mentioned above, and the International Organisation for Standardisation (ISO) are widely accepted as web standards that can be included in a checklist.

The checklist can be recorded in a database or spreadsheet. It should include space for recording the recommended action or remedy to be taken, the name of the staff member to whom the problem is referred, and progress in resolving the problem.

Page 86 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Self Check - 3

Directions: Answer these questions:

1. Name three web page specifications that a client may ask you to satisfy:
2. Is the following statement True or False
 - All web browsers interpret mark-up code in the same way.
A. True **B. False**
3. Name four reasons for writing your mark-up documents to conform to standards
4. What is a validator?
5. Where can you find a validator? Which markup languages are these validators for?
6. Describe three common mark-up code errors that you need to watch out for:
7. What is a "regression bug"?

Operation Sheet - 5

Operation Title: Use a Validation Tool

Purpose: You need to "validate" your mark-up documents (such as CSS, HTML and XHTML) to ensure that they are correct and error-free.

Required tools and equipment: Text Editor, Browser and Internet Connection

Step 1:

Create a table in a spreadsheet or word processing document based on the table below (the first row has been filled in as an example).

Validation error	How I'm going to fix it!
(example: missing </p> tag at end of paragraph)	(insert code as required)

Step 2:

Choose a validation tool from the list below:

World Wide Web Consortium (W3C) HTML and XHTML validators: validator.w3.org

HTML validator from the Web Design Group: htmlhelp.org/tools/validator/

Alternatively, you can download a number of standalone validation tools from the Tucows site: www.tucows.com. Search for the term "Validator".

Step 3:

Use the validation tool to validate your mark-up document. Follow the instructions for the validation tool that you chose.

Use the table you created to record the errors and what you will do to fix them.

Lap Test - 3

Name: _____ Date: _____

Time Started: _____ Time Finished: _____

Instructions: Given necessary templates, workshop, tools and materials you are required to perform the following tasks.

Task 1: Validate The Pixel Website you created on Lap Test - 2, using World Wide Web Consortium (W3C) HTML and XHTML validators: validator.w3.org

Task 2: Validate The Pixel Website you created on Lap Test - 2, using HTML validator from the Web Design Group: htmlhelp.org/tools/validator/

Reference

- Robert G. Fuller and Laurie Ann Ulrich. HTML in 10 Simple Steps or Less, Wiley Publishing, Inc.
- Tutorials Point Pvt. Ltd, HTML 5 Tutorial, 2012, tutorialspoint.com

WEB Address

- <https://web1.muirfield-h.schools.nsw.edu.au/technology/resources/vetit/>

Page 90 of 91	Author/Copyright Ministry of Labor and Skills	Creating A Simple Markup Language Document	Version - 1
			September, 2022

Developers Profile

No	Name	Level	Field of Study	Organization/ Institution	Mobile number	E-mail
1	Abel G/Egziabher	A	Computer Science	MOLS	0911776728	Ab.smart99@gmail.com
2	Endalew Kassa	A	IT	Debremarkos PTC	0913305454	crouchkecho@gmail.com
3	Frew Atkilt	A	Network & Information Security	Bishoftu PTC	0911787374	Frew_at@gmail.com
4	Getnet Alemu	B	IT	Nefasmewucha PTC	0922550906	Getnetalemu783@gmail.com
5	Remedan Mohammed	A	ICT	Harar PTC	0913478937	remedanm77@gmail.com